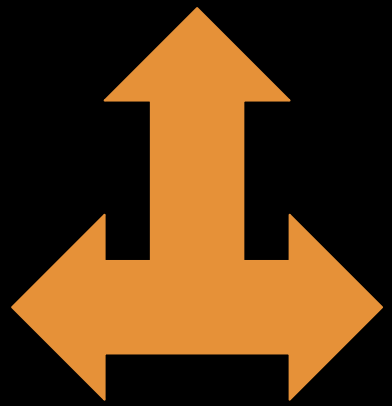
The Bug Hunter's Methodology v4.01

Recon

TBHM v4



Recon

Application Analysis

About Me

- Father, husband, hacker, gamer, sometimes streamer.
- 28th on BC leaderboard
- Twitter | YouTube | Twitch

- Currently Playing:

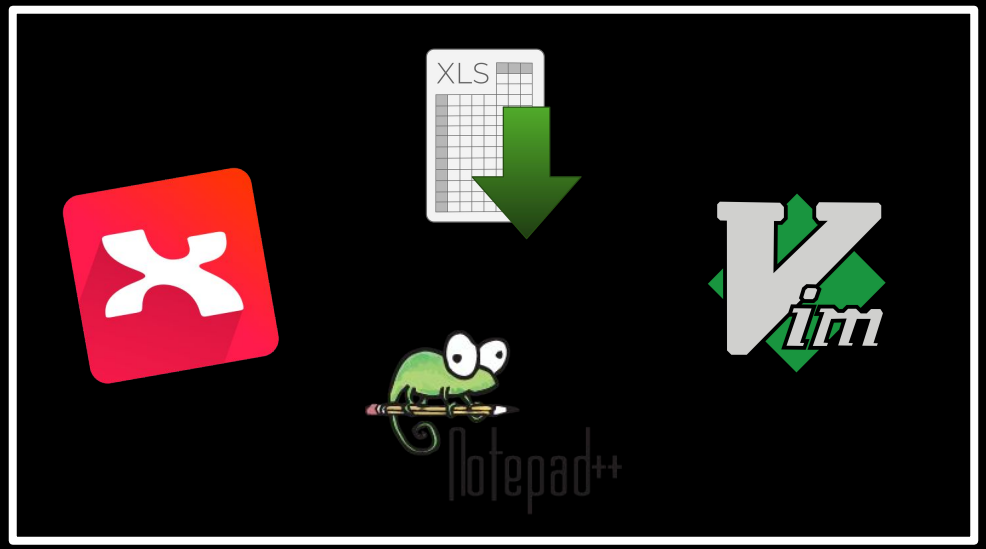
DESTINY 2

CALL OF DUTY WARZONE

TOM CLANCY'S THE DIVISION 2

PATH OF EXILE

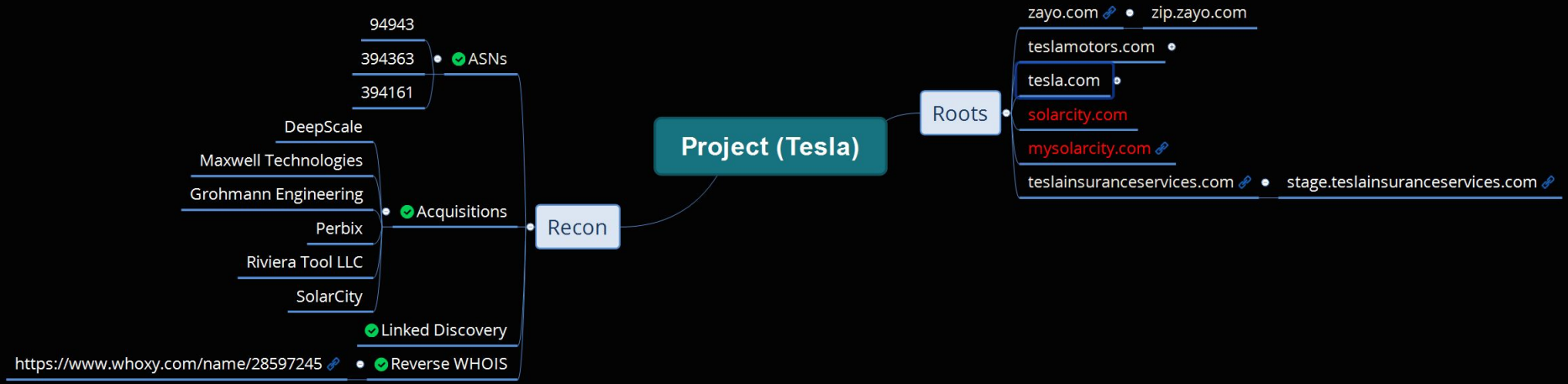
Project Tracking

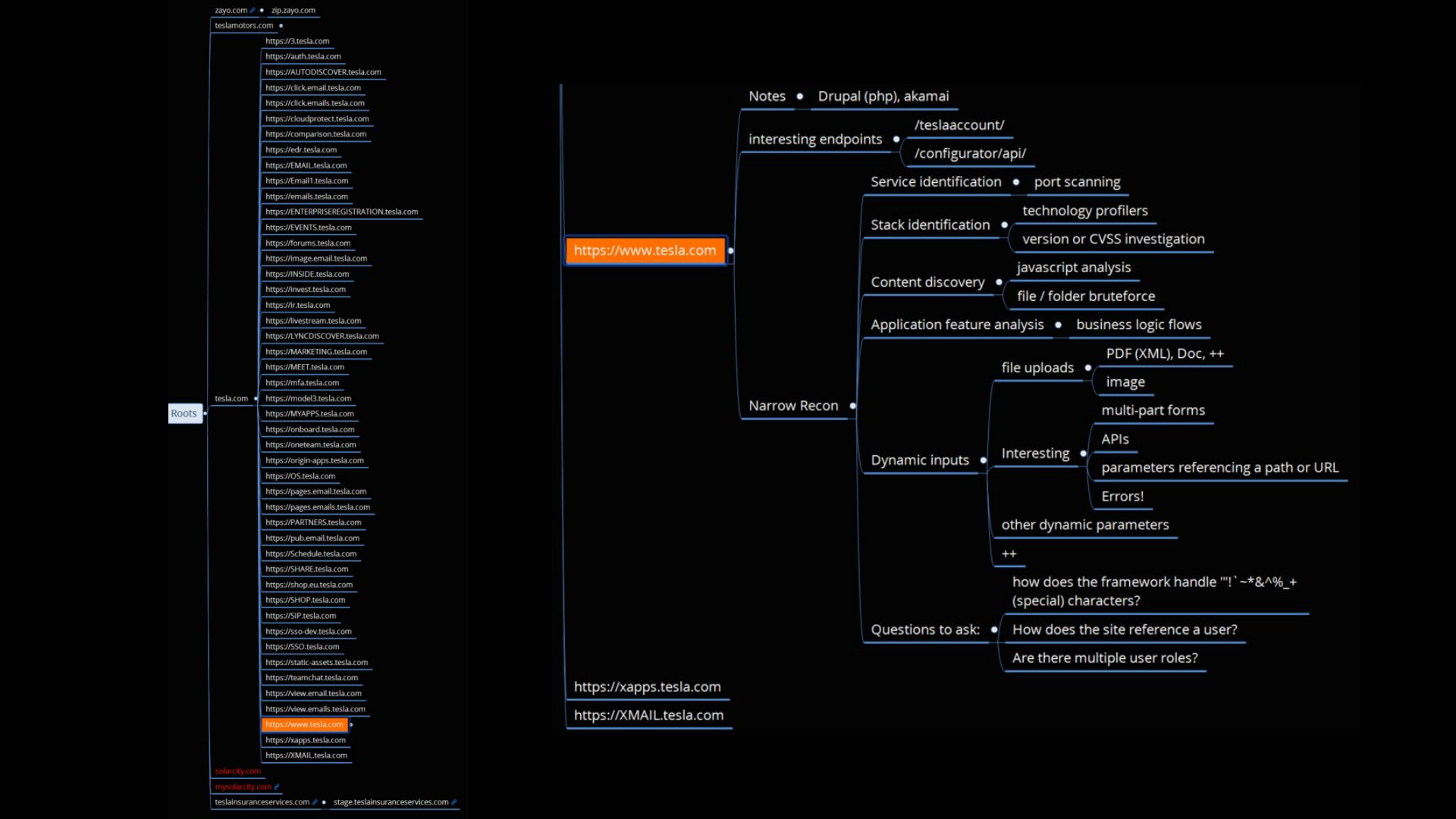


In many parts of the workshop we will need to keep track of site-hierarchy, tools output, interesting notes, etc.

I use mindmaps with Xmind but the same effect can be achieved through a lot of different programs.

Mindmaps allow me to visualize large scope bug hunting targets and also allow me to break up methodology for in-depth bug hunting as well.


Project (Tesla)
Roots
zayo.com
zip.zayo.com
teslamotors.com
tesla.com
solarcity.com
mysolarcity.com
teslainsuranceservices.com
stage.teslainsuranceservices.com
Recon
ASNs
94943
394363
394161
Acquisitions
DeepScale
Maxwell Technologies
Grohmann Engineering
Perbix
Riviera Tool LLC
SolarCity
Linked Discovery
Reverse WHOIS
https://www.whoxy.com/name/28597245


Roots
zayo.com
zip.zayo.com
teslamotors.com
tesla.com
https://3.tesla.com
https://auth.tesla.com
https://AUTODISCOVER.tesla.com
https://click.email.tesla.com
https://click.emails.tesla.com
https://cloudprotect.tesla.com
https://comparison.tesla.com
https://edr.tesla.com
https://EMAIL.tesla.com
https://Email1.tesla.com
https://emails.tesla.com
https://ENTERPRISEREGISTRATION.tesla.com
https://EVENTS.tesla.com
https://forums.tesla.com
https://image.email.tesla.com
https://INSIDE.tesla.com
https://invest.tesla.com
https://ir.tesla.com
https://livestream.tesla.com
https://LYNCDISCOVER.tesla.com
https://MARKETING.tesla.com
https://MEET.tesla.com
https://mfa.tesla.com
https://model3.tesla.com
https://MYAPPS.tesla.com
https://onboard.tesla.com
https://oneteam.tesla.com
https://origin-apps.tesla.com
https://OS.tesla.com
https://pages.email.tesla.com
https://pages.emails.tesla.com
https://PARTNERS.tesla.com
https://pub.email.tesla.com
https://Schedule.tesla.com
https://SHARE.tesla.com
https://shop.eu.tesla.com
https://SHOP.tesla.com
https://SIP.tesla.com
https://sso-dev.tesla.com
https://SSO.tesla.com
https://static-assets.tesla.com
https://teamchat.tesla.com
https://view.email.tesla.com
https://view.emails.tesla.com
https://www.tesla.com
https://xapps.tesla.com
https://XMAIL.tesla.com
solarcity.com
mysolarcity.com
teslainsuranceservices.com
stage.teslainsuranceservices.com
https://www.tesla.com
Notes
Drupal (php), akamai
interesting endpoints
/teslaaccount/
/configurator/api/
Narrow Recon
Service identification
port scanning
Stack identification
technology profilers
version or CVSS investigation
Content discovery
javascript analysis
file / folder bruteforce
Application feature analysis
business logic flows
Dynamic inputs
file uploads
PDF (XML), Doc, ++
image
Interesting
multi-part forms
APIs
parameters referencing a path or URL
Errors!
other dynamic parameters
++
Questions to ask:
how does the framework handle '"!`~*&^%_+ (special) characters?
How does the site reference a user?
Are there multiple user roles?
https://xapps.tesla.com
https://XMAIL.tesla.com

Mission

Wide Recon is the art of discovering as many assets related to a target as possible.
Make sure your scope permits testing these sites.

Scope Domains

Acquisitions

ASN Enumeration

Other

Vulns:
Subdomain Takeover
Buckets
Github leaks

Automation/Helper:
Interlace
Screenshotting
Frameworks

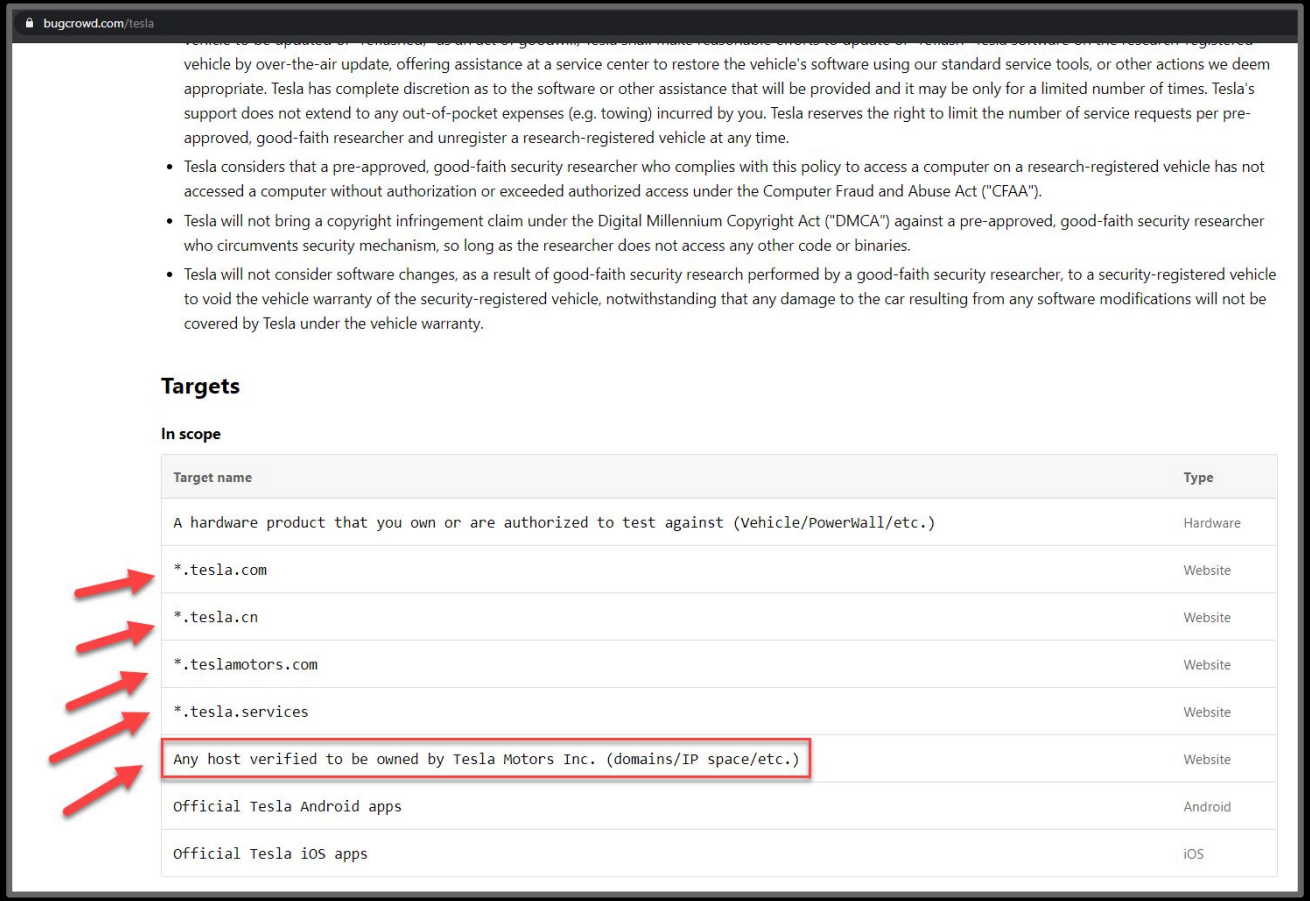
Reverse WHOIS

Subdomain Enumeration

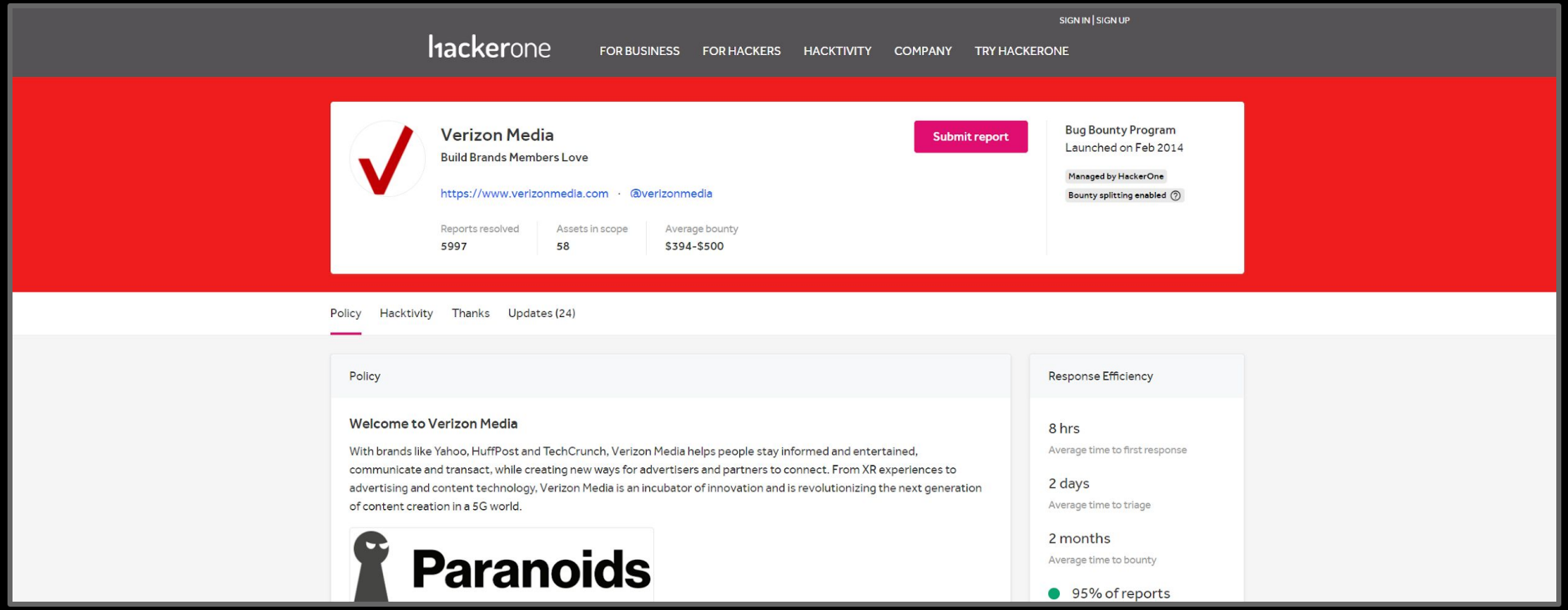
Port Analysis

Finding Seeds/Roots

Scope Domains (Bugcrowd)

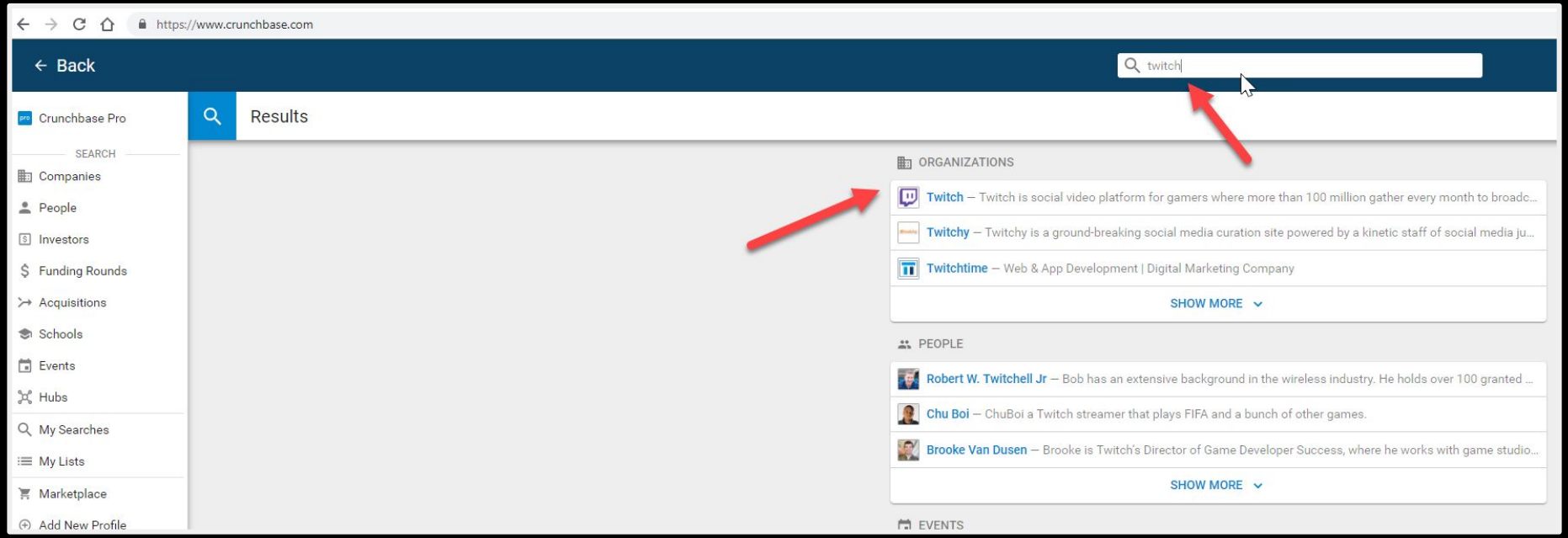


Scope Domains (HackerOne)



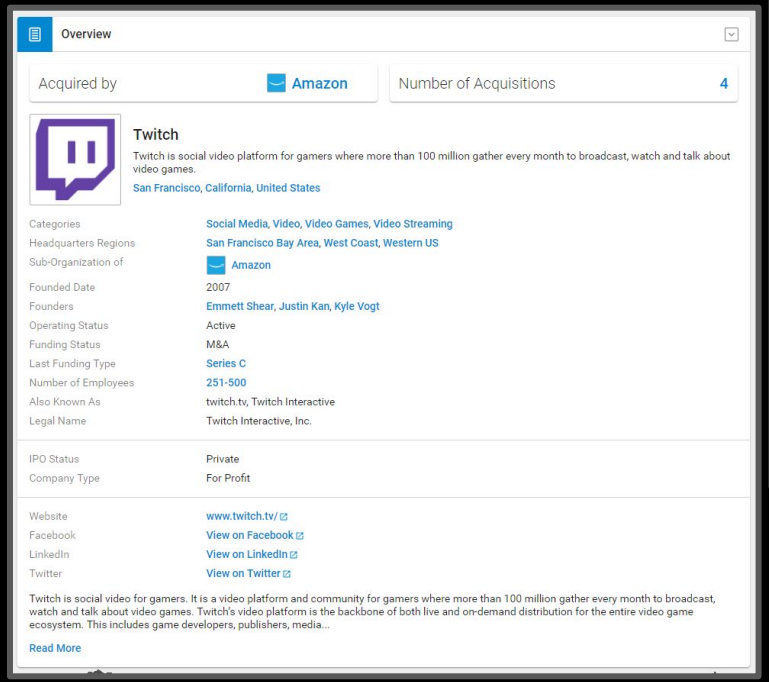
If you've found a vulnerability that affects an asset belonging to Verizon Media, but is not included as in scope on any of the Verizon Media programs, please report it to this program.

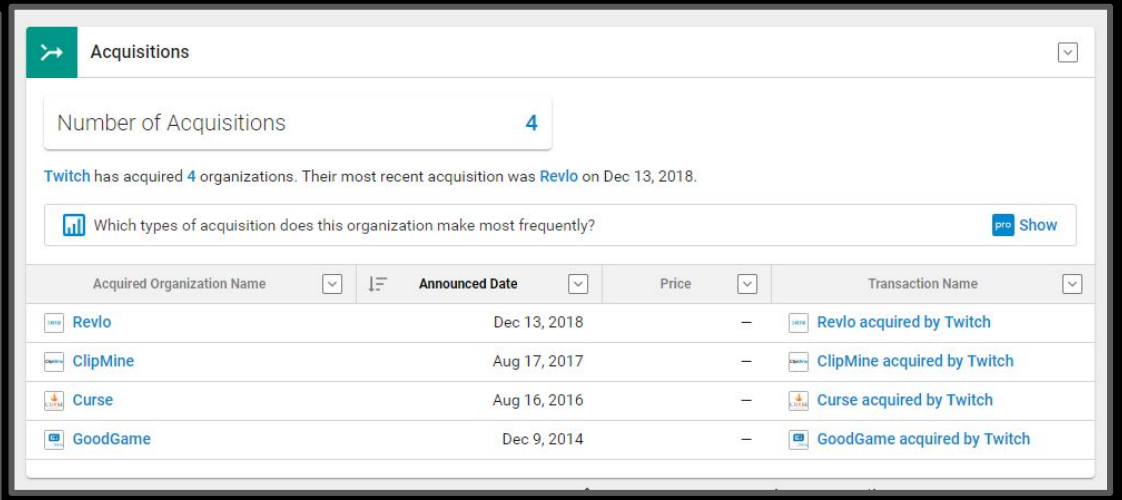
Acquisitions (Crunchbase)



We want to continue to gather seed/root domains. Acquisitions are often a new way to expand our available assets if they are in scope. We can investigate a company's acquisition on sites like https://crunchbase.com, wikipedia, and Google.

Acquisitions (Crunchbase)





Here we can possibly drill down into old domains related to Revlo, ClipMine, Curse, and GoodGame.

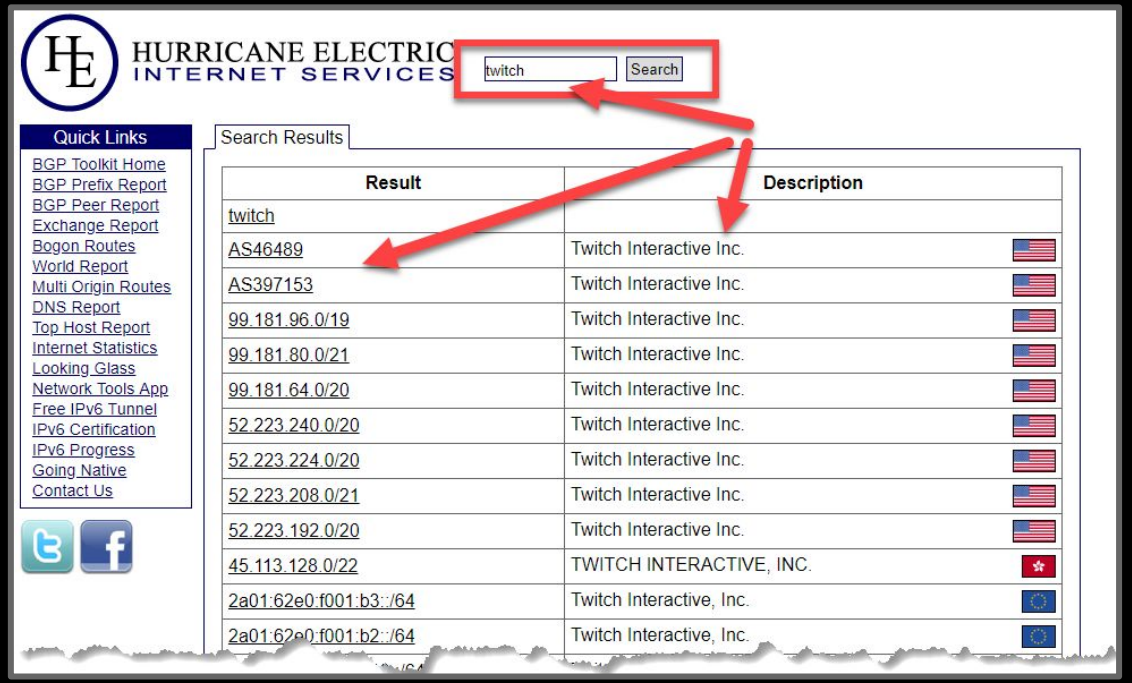
Remember to do some Googling on these acquisitions to see if they are still owned by the parent company. Many times acquisitions will split back out or get sold to another company.

ASN Enumeration (bgp.he.net)

Autonomous System Numbers are given to large enough networks. These ASN's will help us track down some semblance of an entity's IT infrastructure. The most reliable way to get these is manually through Hurricane Electric's free-form search:

- http://bgp.he.net

Because of the advent of cloud infrastructure, ASNs aren't always a complete picture of a network. Rogue assets could exist on cloud environments like AWS and Azure. Here we can see several IP ranges.



ASN Enumeration (cmd line)

```
root@j3ssie ▶ /tmp/demo # echo 'tesla' | metabigor net --org -v
[0000]  INFO Metabigor beta v1.1 by @j3ssiejjj
[0000]  INFO Store log file to: /tmp/metabigor.log
[0000]  INFO [*] Starting get IP Info for Organization: tesla
[0000]  INFO Get data from: http://asnlookup.com/api/lookup?org=tesla
[0000]  INFO Get data from: https://bgp.he.net/search?search%5Bsearch%5D=tesla&commit=Search
[0001]  INFO 192.95.64.0/24
[0001]  INFO 199.120.48.0/24
[0001]  INFO 199.120.49.0/24
[0001]  INFO 199.66.10.0/24
[0001]  INFO 199.66.11.0/24
[0001]  INFO 199.66.9.0/24
[0001]  INFO 205.234.11.0/24
[0001]  INFO 209.133.79.0/24
[0001]  INFO 213.19.141.0/24
[0001]  INFO 8.21.14.0/24
[0001]  INFO 8.45.124.0/24
```

https://twitter.com/j3ssiejjj

Some automation is available to get ASNs. One such tool is the ‘net’ switch of “Metabigor” by j3ssiejjj which will fetch ASN data from a keyword from bgp.he.net and asnlookup.com

Another is “ASNLookup” by Yassine Aboukir which utilizes the maxmind.com dataset.

One problem with cmd line enumeration is that you could return records from another org on accident that contains the keyword ‘tesla’.

```
root@TBox4:~/tools/Asnlookup# python asnlookup.py -o tesla

 ___ ___ _ _ _    ___ ___ _ _ _ _ ___
|__| [__  |\ | |    |  | |  | |_/  |  | |__]
|  | ___] | \| |___ |__| |__| | \_ |__| |

        Author: Yassine Aboukir (@yassineaboukir)

[*] User's license key is valid!

[*] IP addresses owned by tesla are the following (IPv4 or IPv6):

[*] IPv4 addresses saved to:
/root/tools/Asnlookup/output/tesla_ipv4.txt

192.95.64.0/24
199.120.48.0/24
199.120.49.0/24
199.66.10.0/24
199.66.11.0/24
199.66.9.0/24
205.234.11.0/24
209.133.79.0/24
213.19.141.0/24
8.21.14.0/24
8.45.124.0/24
```

https://twitter.com/yassineaboukir

ASN Enumeration (with Amass)

For discovering more seed domains we want to scan the whole ASN with a port scanner and return any root domains we see in SSL certificates, etc.

We can do this with Amass intel

Amass is written by Jeff Foley and the Amass team.


```
amass 3.7.1 from Jeff Foley (caffix) installed
root@TBox4:~/tools/Asnlookup# amass intel -asn 46489
justin.tv
ttvnw.net
twitch.tv
twitchcon.com
socialcam.com
```

https://twitter.com/owaspamass https://twitter.com/jeff_foley

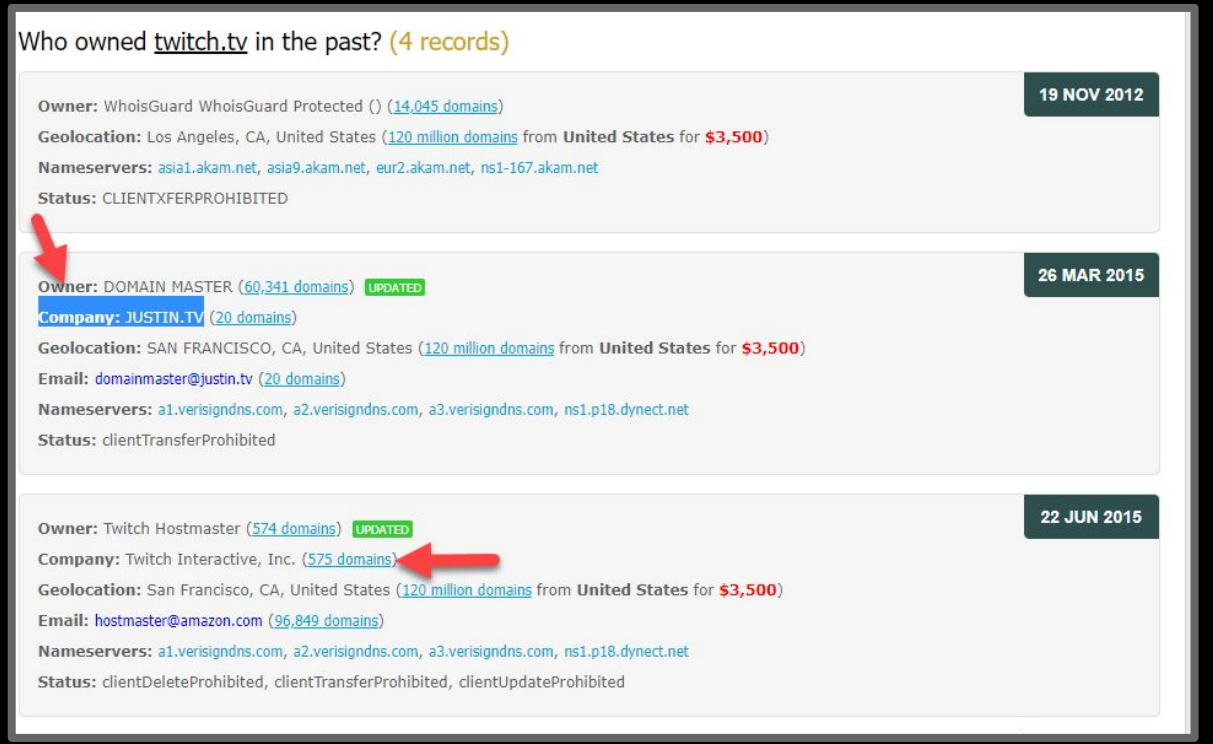
Reverse WHOIS (with Whoxy.com)

Every website has some registration info on file with the registrars. Two key pieces of data we can use are Organization name and any emails in the WHOIS data. To do this you need access to a large WHOIS database. WHOXY.com is one such database.

You can use whoxy.com in this fashion, after you register and your free API key:

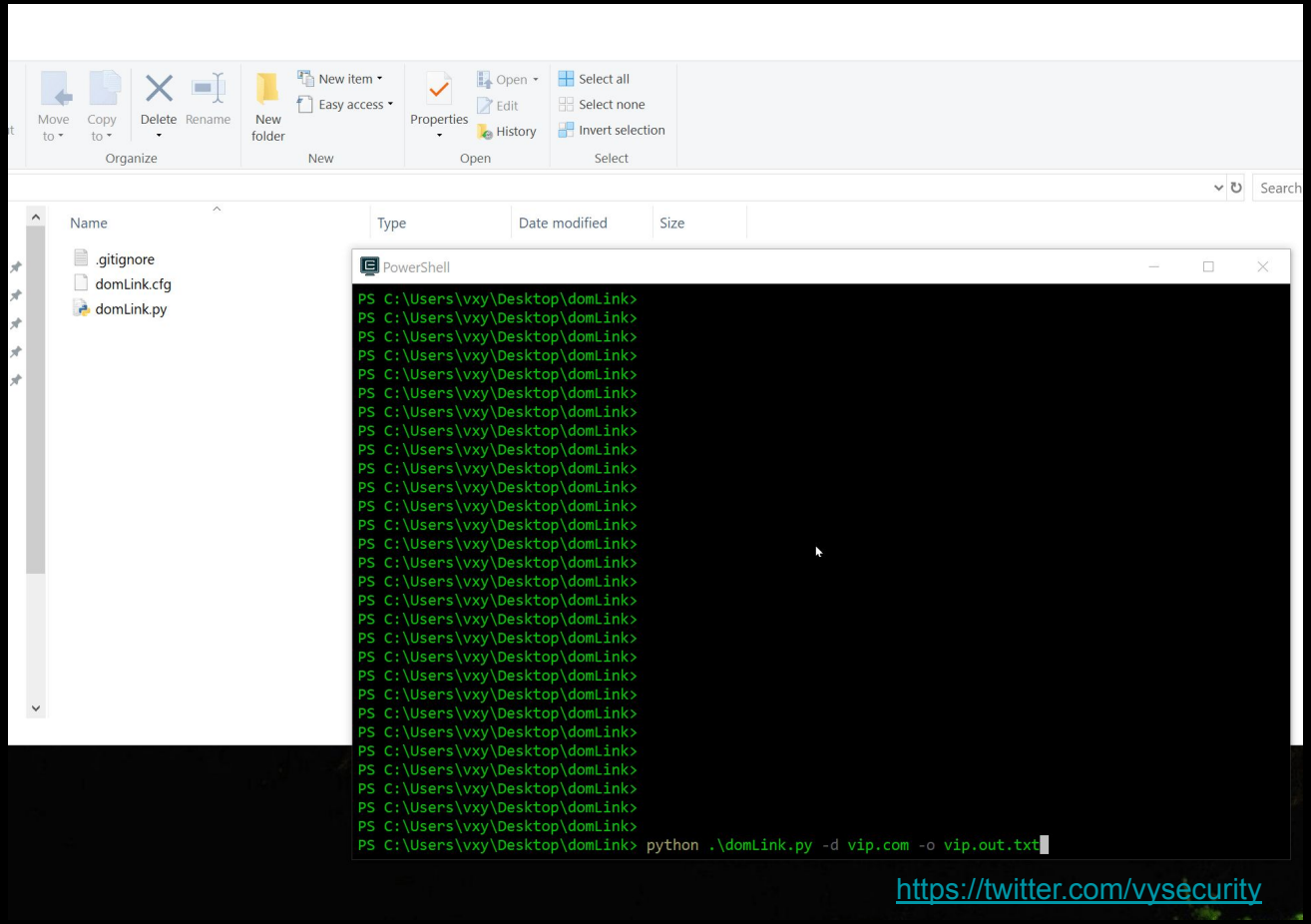
- http://api.whoxy.com/?key=APIkeyHERE&reverse=whois&name=Twitch+Hostmaster

Careful with reverse whois data as it is the least high fidelity source of new root/seed domains. It might include many parked domains or redirects to out of scope assets.



Reverse WHOIS (with DOMLink)

DOMLink is a tool written by Vincent Yiu (@vysecurity) which will recursively query the WHOXY WHOIS API. It will start by querying our targets WHOIS record, then analyze all the data and look for other records which contain the organization name or are registered to emails in the record. It does this recursively until it finds no more records of match.



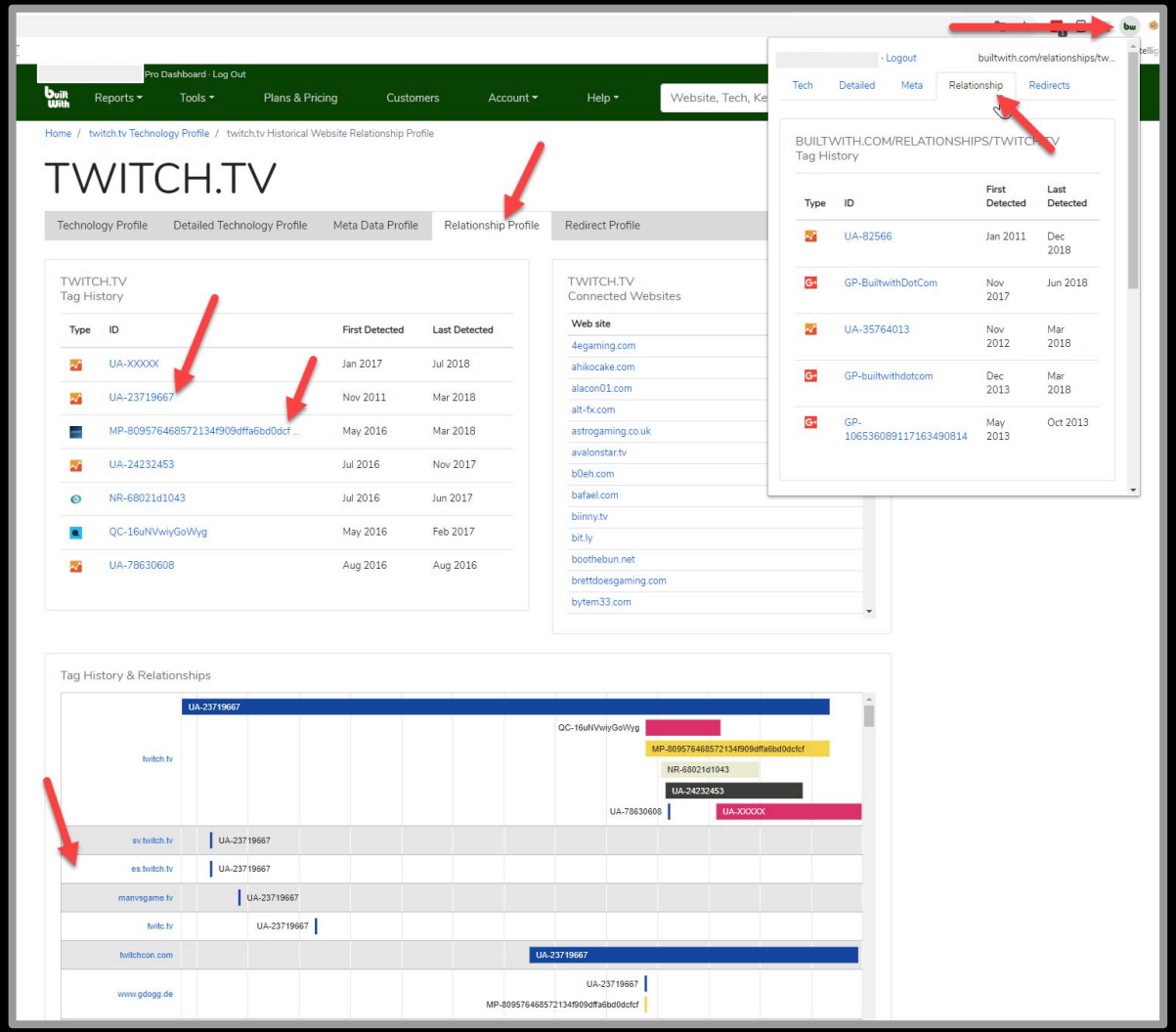
https://twitter.com/vysecurity

Ad/Analytics Relationships (builtwith.com)

You can also glean related domains and subdomains by looking at a target's ad/analytics tracker codes. Many sites use the same codes across all their domains. Google analytics and New Relic codes are the most common. We can look at these "relationships" via a site called BuiltWith. Builtwith also has a Chrome and Firefox extension to do this on the fly.

- https://builtwith.com/relationships/twitch.tv

BuiltWith is also a tool we'll use to profile the technology stack of a target in later slides.



Ad/Analytics Relationships (getrelationship.py)

Want to do it on the command line?

@M4ll0k has you covered!

https://raw.githubusercontent.com/m4ll0k/Bug-Bounty-Toolz/master/getrelationship.py

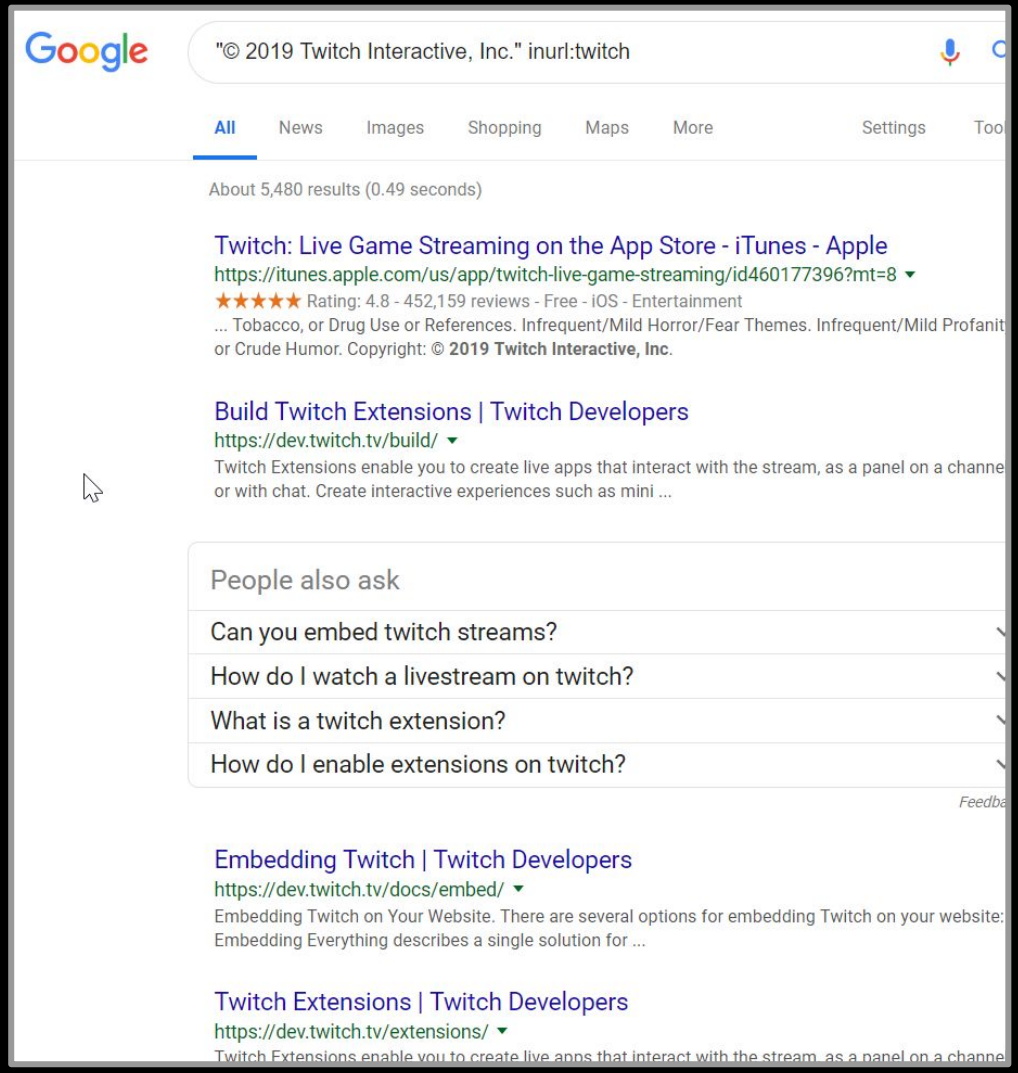
```
m4ll0k@m4ll0k Desktop % echo "uber.com"|python3 getrelationship.py
m.21north.in
999yese.com
amerencsr.com
a-pple.es
avs.auto
chewy.com
chromadex.com
daftaruber.com
deck.gl
driveuber.co.nz
durangodelivery.com
edrsilver.com
fastenal.com
internetubeatbr.com
ishutter.in
jackcooperlogistics.com
jacksonvillewatertaxi.com
jump.com
kitchensofubereats.com
laureate.net
lingnei.cn
lingnei.com
```

Google-Fu

You can Google the:

- Copyright text
- Terms of service text
- Privacy policy text

from a main target to glean related hosts on Google.



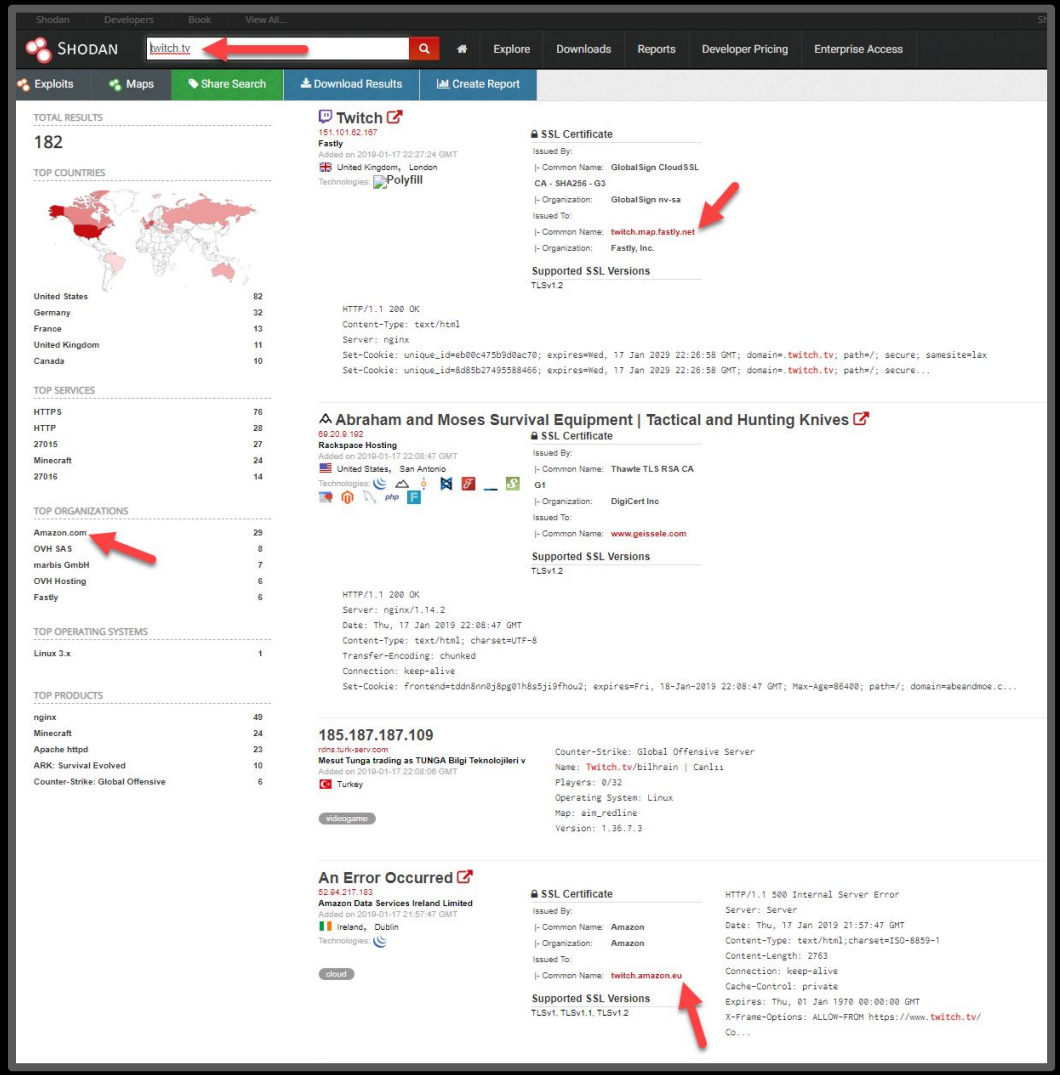
Shodan

Shodan is a tool that continuously spiders infrastructure on the internet. It is much more verbose than regular spiders. It captures response data, cert data, stack profiling data, and more. It requires registration.

Example:

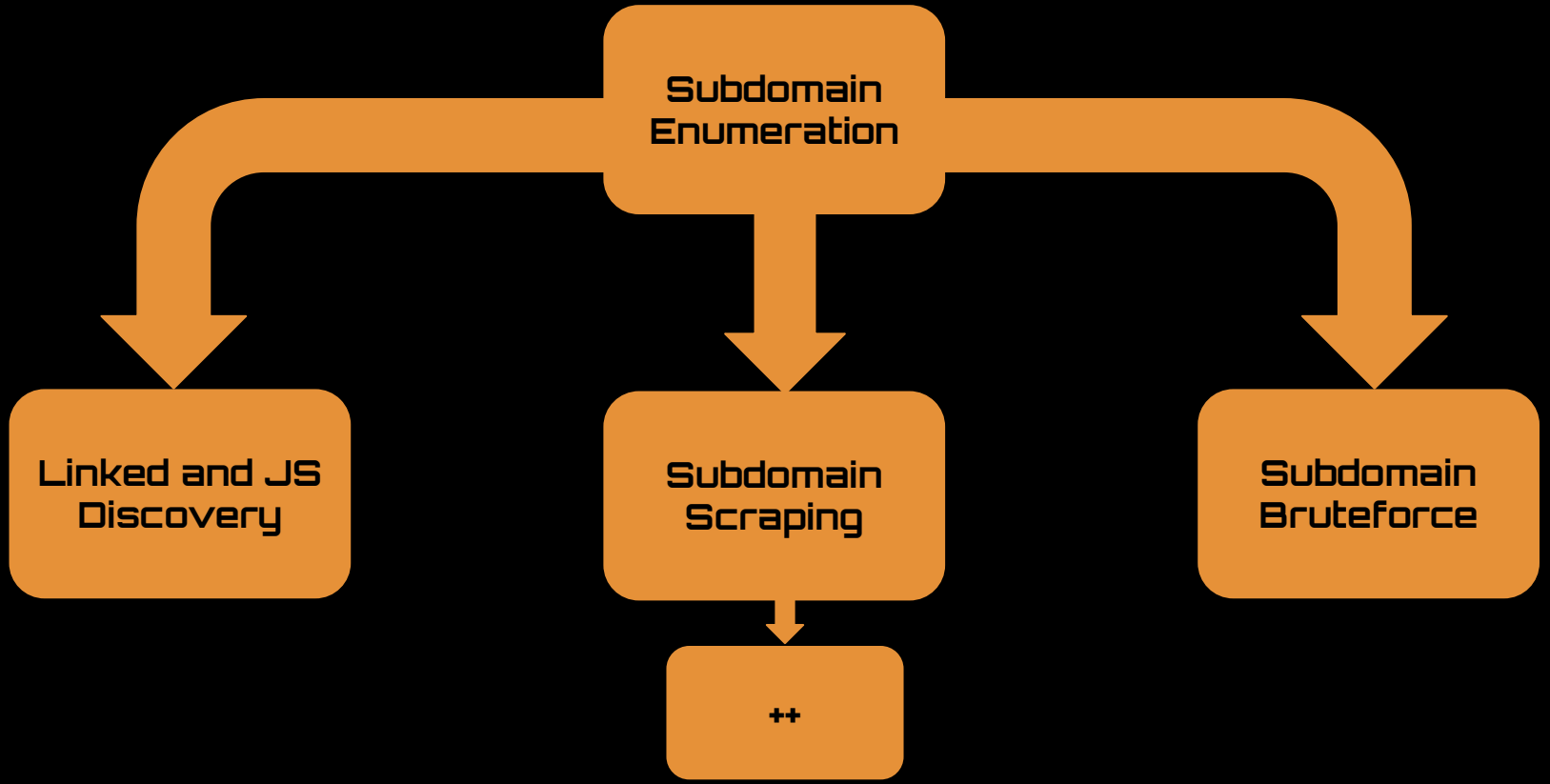
https://www.shodan.io/search?query=twitch.tv

We can glean a valuable question: is twitch.amazon.eu relevant to our testing?



Finding Subdomains

Subdomain Enumeration




Linked and JS
Discovery

Linked Discovery (with Burp Suite Pro)

Another way to to widen our scope is to examine all the links of our main target. We can do this using Burp Suite Pro.

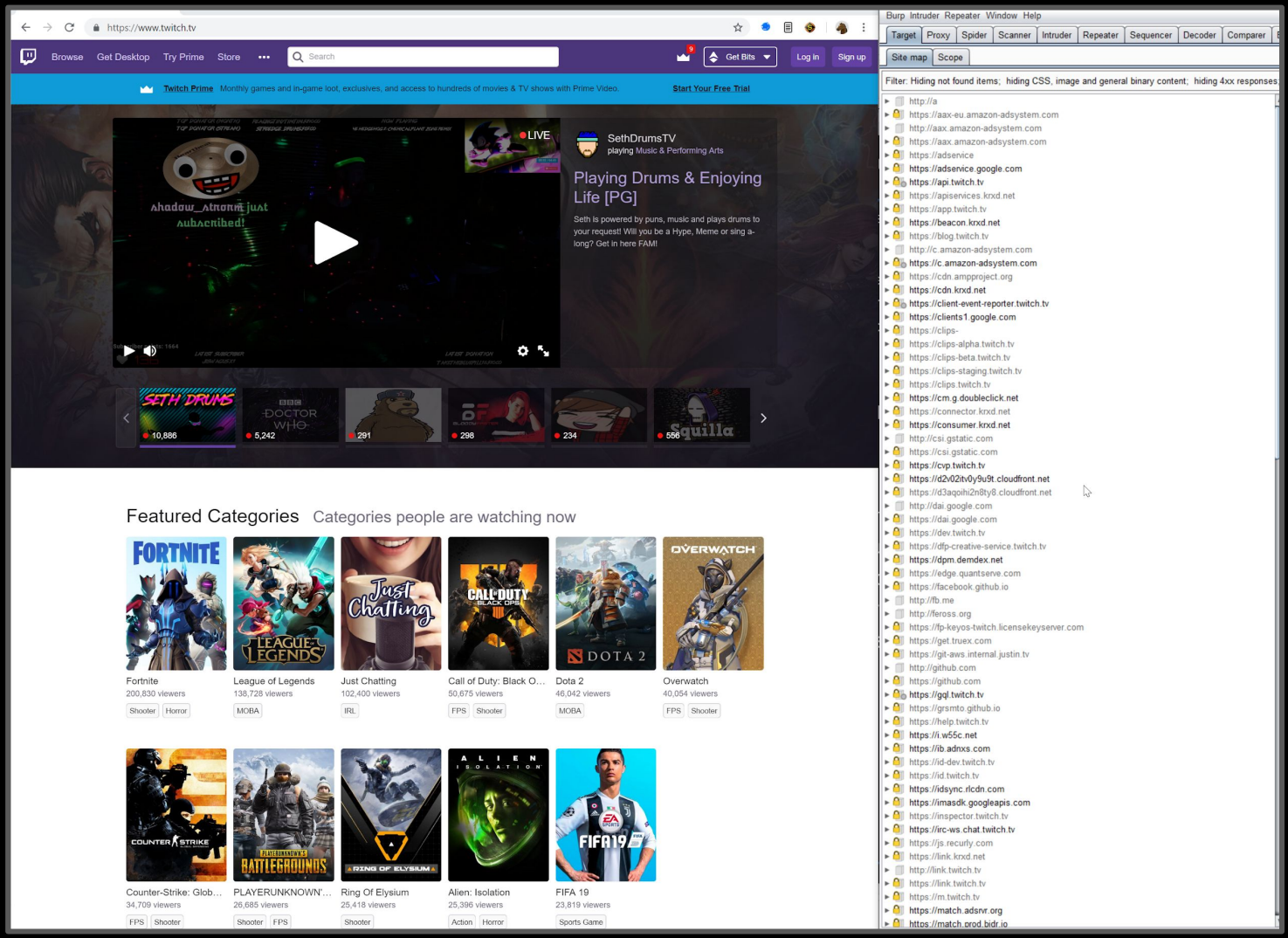
We can visit a seed/root and recursively spider all the links for a term with regex, examining those links... and their links, and so on... until we have found all sites that could be in our scope.

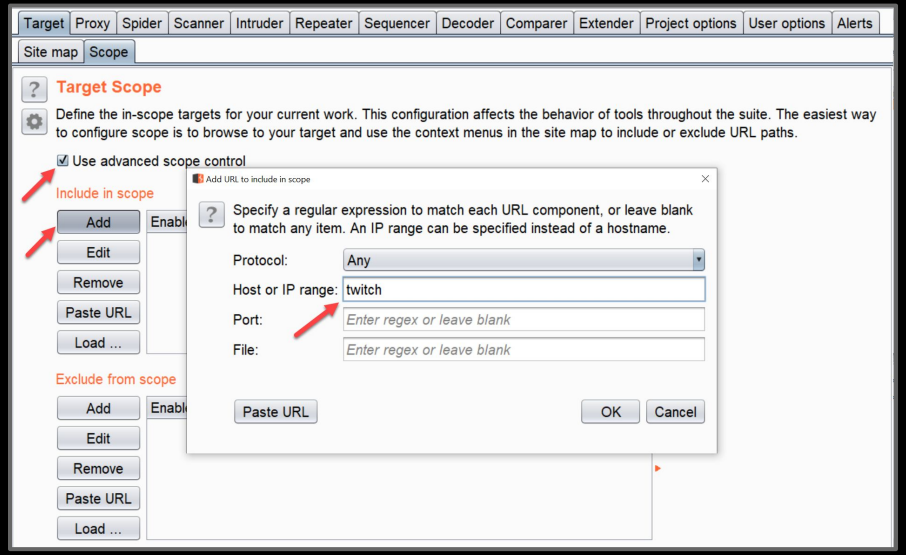
This is a hybrid technique that will find both roots/seeds and subdomains.

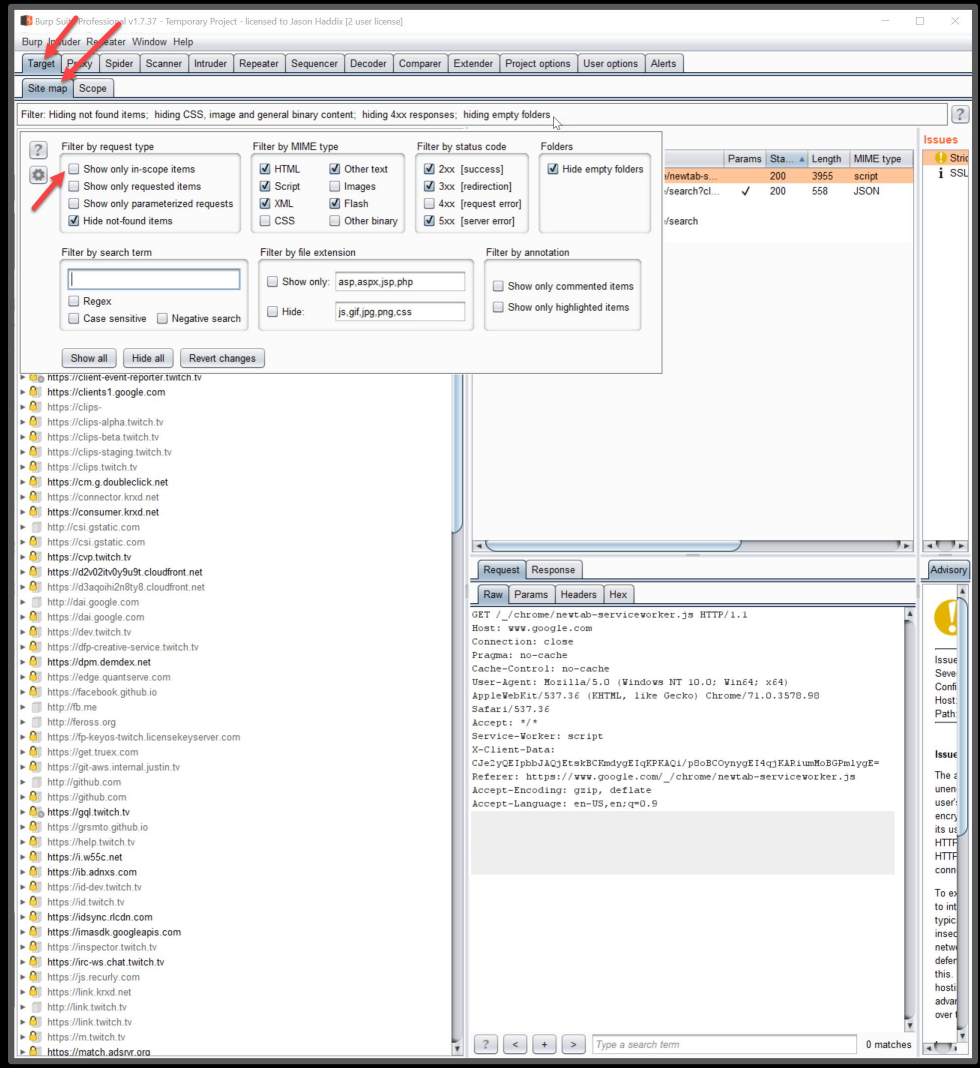
1. Turn off passive scanning
2. Set forms auto to submit (if you're feeling frisky)
3. Set scope to advanced control and use "keyword" of target name (not a normal FQDN)
4. Walk+browse main site, then spider all hosts recursively!
5. Profit

Linked Discovery (with Burp Suite Pro)

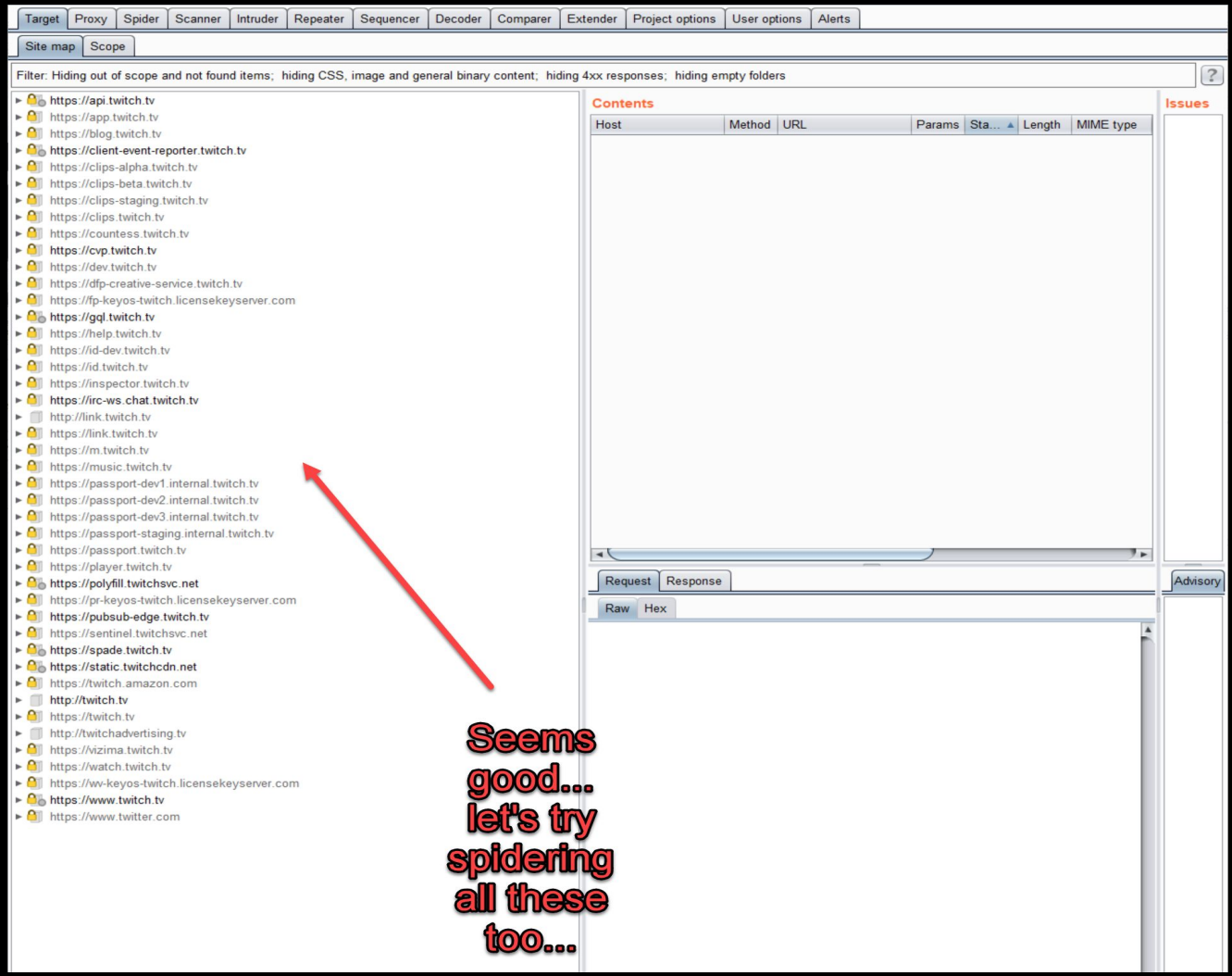
Burp after requesting one site:




Target
Proxy
Spider
Scanner
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
User options
Alerts
Site map
Scope
Target Scope
Define the in-scope targets for your current work. This configuration affects the behavior of tools throughout the suite. The easiest way to configure scope is to browse to your target and use the context menus in the site map to include or exclude URL paths.
Use advanced scope control
Include in scope
Add
Edit
Remove
Paste URL
Load ...
Exclude from scope
Add
Edit
Remove
Paste URL
Load ...
Add URL to include in scope
Specify a regular expression to match each URL component, or leave blank to match any item. An IP range can be specified instead of a hostname.
Protocol:
Any
Host or IP range:
twitch
Port:
Enter regex or leave blank
File:
Enter regex or leave blank
Paste URL
OK
Cancel


Burp Suite Professional v1.7.37 - Temporary Project - licensed to Jason Haddix [2 user license]
Burp Intruder Repeater Window Help
Target
Proxy
Spider
Scanner
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
User options
Alerts
Site map
Scope
Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders
Filter by request type
Show only in-scope items
Show only requested items
Show only parameterized requests
Hide not-found items
Filter by MIME type
HTML
Script
XML
CSS
Other text
Images
Flash
Other binary
Filter by status code
2xx [success]
3xx [redirection]
4xx [request error]
5xx [server error]
Folders
Hide empty folders
Filter by search term
Regex
Case sensitive
Negative search
Filter by file extension
Show only:
asp,aspx,jsp,php
Hide:
js,gif,jpg,png,css
Filter by annotation
Show only commented items
Show only highlighted items
Show all
Hide all
Revert changes
Params
Length
MIME type
Issues
Request
Response
Raw
Params
Headers
Hex
GET /_/chrome/newtab-serviceworker.js HTTP/1.1
Host: www.google.com
Connection: close
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98
Safari/537.36
Accept: */*
Service-Worker: script
X-Client-Data:
CJe2yQEIpbbJAQjEtskBCKmdygEIqKPKAQi/pSoBCOynygEI4qjKARiumMoBGPmlygE=
Referer: https://www.google.com/_/chrome/newtab-serviceworker.js
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Type a search term
0 matches
Advisory

Linked Discovery (with Burp Suite Pro)

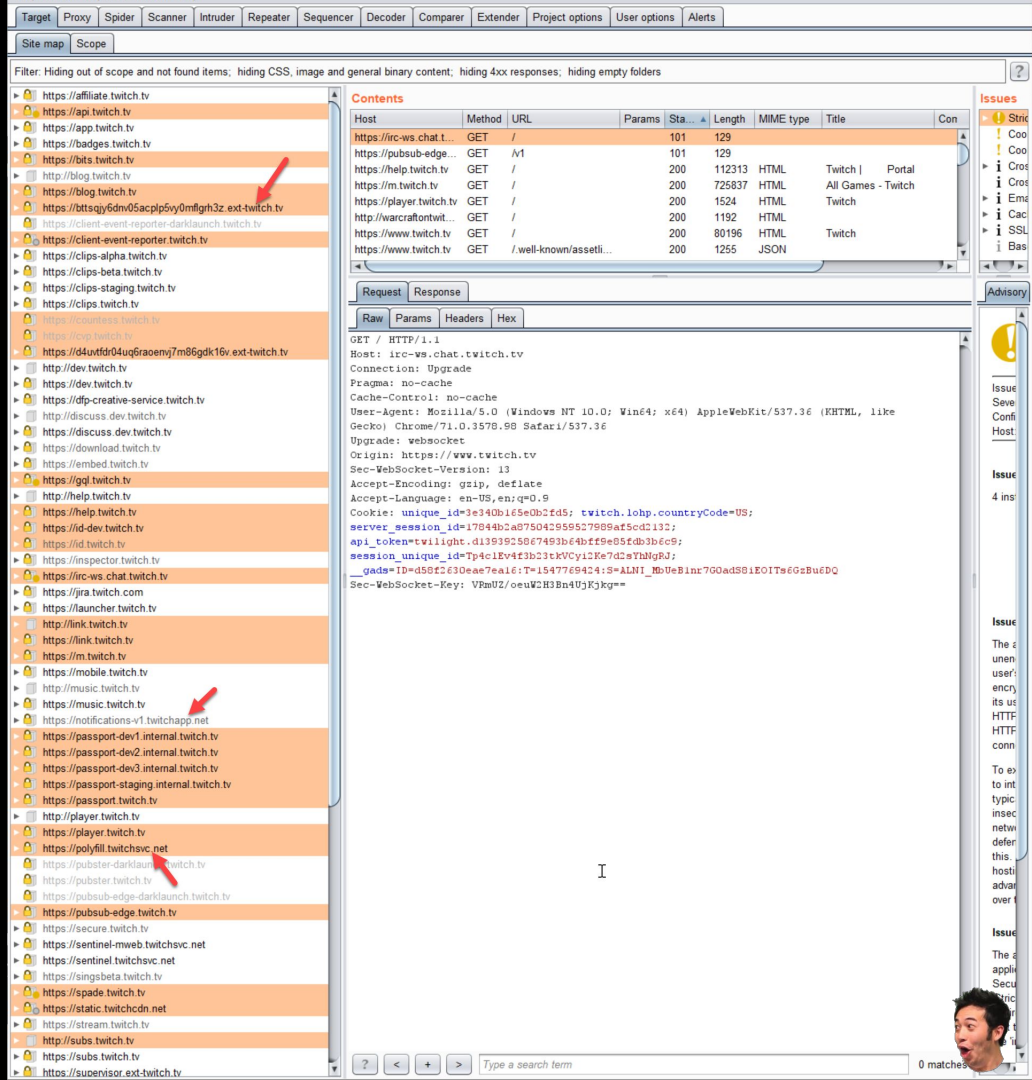


Linked Discovery (with Burp Suite Pro)

After the 1st spider run we've now discovered a ton of linked URLs that belong to our project.

Not only subdomains, but **NEW** seeds/roots (twtchapp.net, ext-twitch.tv, twitchsvc.net).

We can also now spider these new hosts and repeat until we have Burp Spider fatigue.

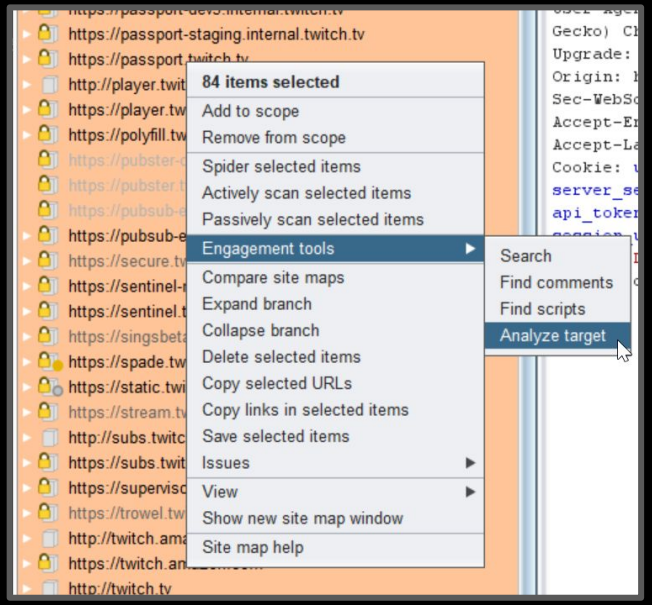


Linked Discovery (with Burp Suite Pro)

Now that we have this data, how do we export it?

Clumsily =(

1) Select all hosts in the site tree
2) In PRO ONLY right click the selected hosts
3) Go to “Engagement Tools” -> “Analyze target”
4) Save report as an html file
5) Copy the hosts from the “Target” section



Target analysis

Targets

- https://affiliate.twitch.tv/
- https://api.twitch.tv/
- https://app.twitch.tv/
- https://badges.twitch.tv/
- https://bits.twitch.tv/
- http://blog.twitch.tv/
- https://blog.twitch.tv/
- https://bttsqjy6dnv05acplp5vy0mflgrh3z.ext-twitch.tv/
- https://client-event-reporter-darklaunch.twitch.tv/
- https://client-event-reporter.twitch.tv/
- https://clips-alpha.twitch.tv/
- https://clips-beta.twitch.tv/
- https://clips-staging.twitch.tv/
- https://clips.twitch.tv/
- https://countess.twitch.tv/
- https://cvp.twitch.tv/
- https://d4uvtfdr04uq6raoenvj7m86gdk16v.ext-twitch.tv/
- http://dev.twitch.tv/
- https://dev.twitch.tv/
- https://dfp-creative-service.twitch.tv/
- http://discuss.dev.twitch.tv/
- https://discuss.dev.twitch.tv/
- https://download.twitch.tv/
- https://embed.twitch.tv/
- https://gql.twitch.tv/
- http://help.twitch.tv/
- https://help.twitch.tv/
- https://id-dev.twitch.tv/
- https://id.twitch.tv/
- https://inspector.twitch.tv/
- https://irc-ws.chat.twitch.tv/
- https://jira.twitch.com/
- https://launcher.twitch.tv/
- http://link.twitch.tv/
- https://link.twitch.tv/
- https://m.twitch.tv/
- https://mobile.twitch.tv/
- http://music.twitch.tv/
- https://music.twitch.tv/
- https://notifications-v1.twitchapp.net/
- https://passport-dev1.internal.twitch.tv/
- https://passport-dev2.internal.twitch.tv/
- https://passport-dev3.internal.twitch.tv/
- https://passport-staging.internal.twitch.tv/
- https://passport.twitch.tv/
- http://player.twitch.tv/

Linked Discovery

(with GoSpider or hakrawler)

Linked discovery really just counts on using a spider recursively.

One of the most extensible spiders for general automation is GoSpider written by j3ssiejjj which can be used for many things and supports parsing js very well.

In addition hakrawler by hakluke has many parsing strategies that interest bug hunters.

```
root@lBox4:~/tools# gospider -s https://www.twitch.tv
[robots] - https://www.twitch.tv/
[robots] - https://www.twitch.tv/directory
[robots] - https://www.twitch.tv/directory/all
[robots] - https://www.twitch.tv/directory/*
[robots] - https://www.twitch.tv/videos/week
[robots] - https://www.twitch.tv/.well-known/assetlinks.json
[robots] - https://www.twitch.tv/admin/*
[robots] - https://www.twitch.tv/email-unsubscribe/
[robots] - https://www.twitch.tv/login$
[robots] - https://www.twitch.tv/message/*
[robots] - https://www.twitch.tv/signup$
[robots] - https://www.twitch.tv/user/*
[subdomains] - api.twitch.tv
[subdomains] - passport.twitch.tv
[subdomains] - gql.twitch.tv
[subdomains] - cvp.twitch.tv
[subdomains] - irc-ws.chat.twitch.tv
[subdomains] - pubsub-edge.twitch.tv
[subdomains] - spade.twitch.tv
[subdomains] - www.twitch.tv
[url] - [code-200] - https://www.twitch.tv
[javascript] - https://polyfill.twitchsvc.net/
.prototype.includes,default,fetch,Intl.~locale
[javascript] - https://static.twitchcdn.net/co
[javascript] - https://static.twitchcdn.net/as
[javascript] - https://static.twitchcdn.net/as
[url] - [code-200] - https://www.twitch.tv/.we
[subdomains] - m.twitch.tv
[url] - [code-200] - https://www.twitch.tv/
[url] - [code-200] - https://www.twitch.tv/dir
[url] - [code-200] - https://www.twitch.tv/mes
```

```
HAKRAWLER
Crafted with <3 by hakluke
[robots] http://bugcrowd.com/*?preview
[sitemap] https://bugcrowd.com/
[sitemap] https://bugcrowd.com/contact/
[sitemap] https://bugcrowd.com/faq/
[sitemap] https://bugcrowd.com/leaderboard/
[sitemap] https://bugcrowd.com/list-of-bug-bounty-programs/
[sitemap] https://bugcrowd.com/press/
[sitemap] https://bugcrowd.com/pricing/
[sitemap] https://bugcrowd.com/privacy/
[sitemap] https://bugcrowd.com/terms/
[sitemap] https://bugcrowd.com/resources/responsible-disclosure-program/
[sitemap] https://bugcrowd.com/resources/why-care-about-web-security/
[sitemap] https://bugcrowd.com/resources/what-is-a-bug-bounty/
[sitemap] https://bugcrowd.com/stories/movember/
[sitemap] https://bugcrowd.com/stories/riskio/
[sitemap] https://bugcrowd.com/stories/tagged/
[sitemap] https://bugcrowd.com/tour/
[sitemap] https://bugcrowd.com/tour/platform/
[sitemap] https://bugcrowd.com/tour/crowd/
[sitemap] https://bugcrowd.com/customers/programs/new
[sitemap] https://bugcrowd.com/portal/
[sitemap] https://bugcrowd.com/portal/user/sign_in/
[sitemap] https://bugcrowd.com/portal/user/sign_up/
[url] https://bugcrowd.com/user/sign_in
[subdomain] bugcrowd.com
[url] https://tracker.bugcrowd.com/user/sign_in
[subdomain] tracker.bugcrowd.com
[url] https://www.bugcrowd.com/
[subdomain] www.bugcrowd.com
[url] https://www.bugcrowd.com/products/how-it-works/
[url] https://www.bugcrowd.com/products/how-it-works/the-bugcrowd-difference/
[url] https://www.bugcrowd.com/products/platform/
[url] https://www.bugcrowd.com/products/platform/integrations/
[url] https://www.bugcrowd.com/products/platform/vulnerability-rating-taxonomy/
[url] https://www.bugcrowd.com/products/attack-surface-management/
[url] https://www.bugcrowd.com/products/bug-bounty/
[url] https://www.bugcrowd.com/products/vulnerability-disclosure/
[url] https://www.bugcrowd.com/products/next-gen-pen-test/
[url] https://www.bugcrowd.com/products/bug-bash/
[url] https://www.bugcrowd.com/resources/reports/priority-one-report
```

Subdomain Enumeration (with SubDomainizer)

Subdomainizer by Neeraj Edwards is a tool with three purposes in analyzing javascript. It will take a page and run

1. Find subdomains referenced in js files
2. Find cloud services referenced in js files
3. Use the Shannon Entropy formula to find potentially sensitive items in js files

It will take a single page and scan for js files to analyze.

```
key":"BGzteaQYOqrTAPN8EuuowBVG67pHwyZo879XZkC7cUV2QP4qQf-92Pmm9tyOuriJdiKnMDDRi28F5HQK6uSk0vM"
Key="6Ld65QcTAAAAAMBbAE8dkJq4Wi4CsJy7flvKhYqX"
```

If just looking for subdomains subscraper by Cillian-Collins might be better because it has recursion.

```
Finding Subdomains and cloud data of given domain in all Javascript files...
Searching completed...
Got all the important data.


~~~~~~~~~~~~~~~~~~~~~~~~~RESULTS~~~~~~~~~~~~~~~~~~~~~~~~~


Some cloud services url's found. They might be interesting, Here are the URLs:

d2v02itv0y9u9t.cloudfront.net
d3aqoihi2n8ty8.cloudfront.net

Successfully got all the subdomains...

Total Subdomains: 35
static.twitchcdn.net
api.twitch.tv
app.twitch.tv
blog.twitch.tv
canary.twitch.tv
irc-ws.chat.twitch.tv
client-event-reporter.twitch.tv
client-event-reporter-darklaunch.twitch.tv
clips.twitch.tv
cvp.twitch.tv
cvp-test.twitch.tv
dev.twitch.tv
glass.twitch.tv
gql.twitch.tv
help.twitch.tv
id.twitch.tv
id-dev.twitch.tv
passport-dev1.internal.twitch.tv
passport-dev2.internal.twitch.tv
passport-dev3.internal.twitch.tv
passport-staging.internal.twitch.tv
music.twitch.tv
passport.twitch.tv
player.twitch.tv
player-core-web.twitch.tv
pubster.twitch.tv
pubster-darklaunch.twitch.tv
pubsub-edge.twitch.tv
pubsub-edge-darklaunch.twitch.tv
rc.twitch.tv
release.twitch.tv
spade.twitch.tv
tmi.twitch.tv
vizima.twitch.tv
www.twitch.tv
```

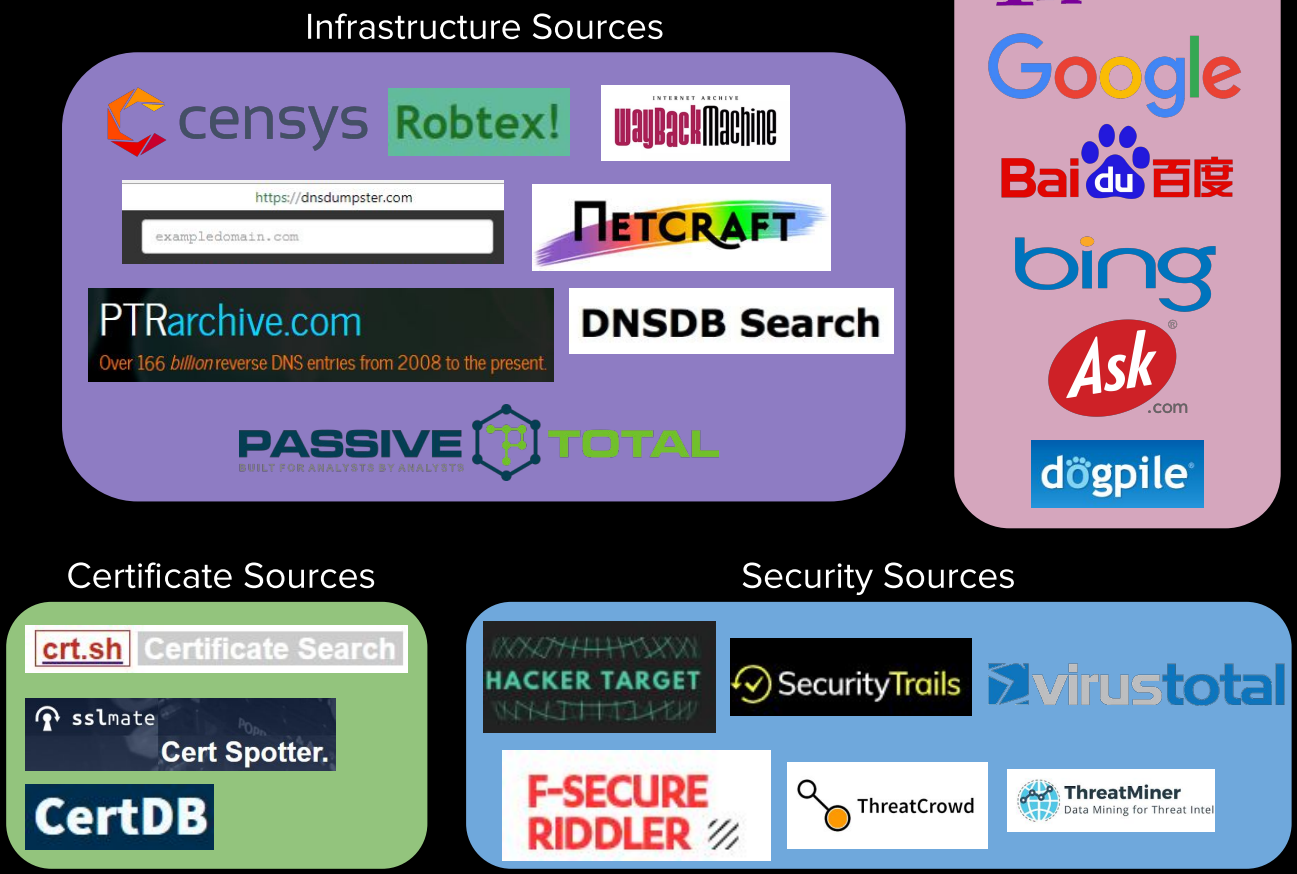
Subdomain Scraping

Subdomain Scraping Sources

The next set of tools scrape domain information from all sorts of projects that expose databases of URLs or domains.

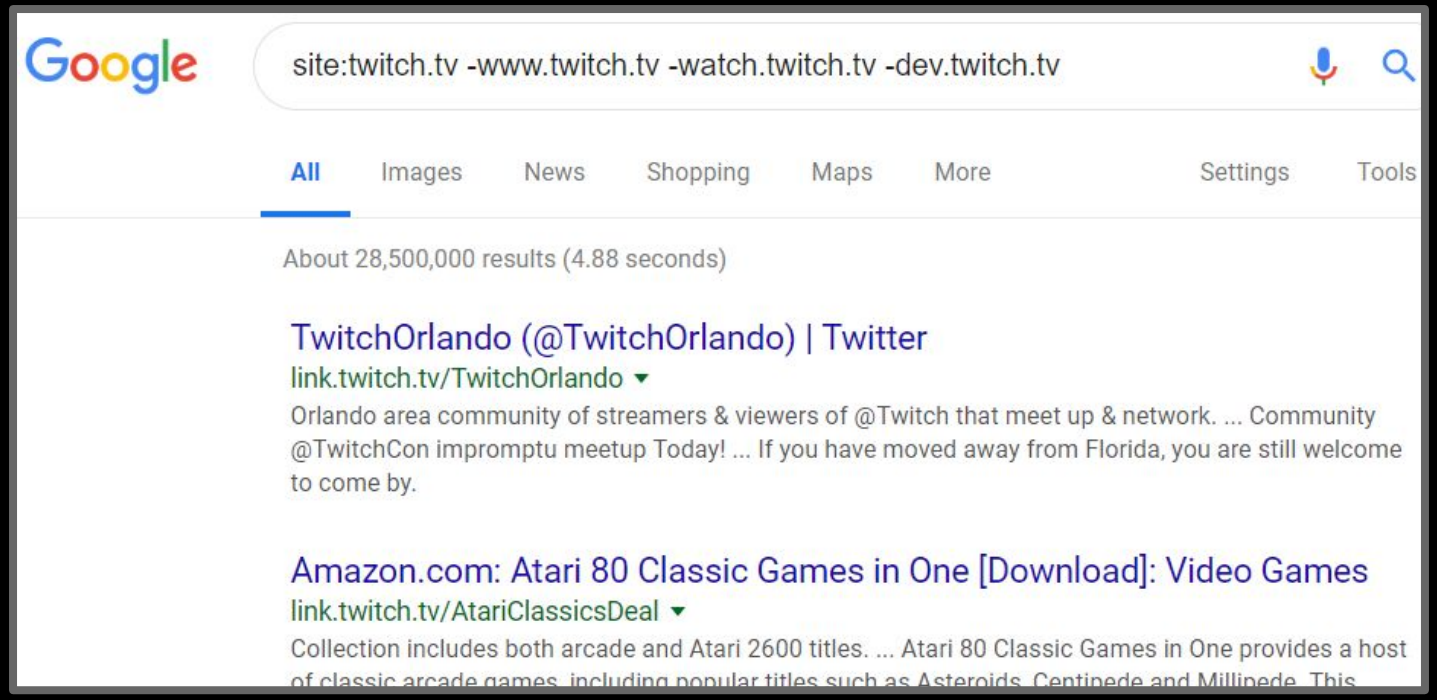
New sources are coming out all the time so the tools must evolve constantly.

This is only a small list of sources. Many more exist.



Subdomain Scraping Example (Google)

1. site:twitch.tv -www.twitch.tv
2. site:twitch.tv -www.twitch.tv -watch.twitch.tv
3. site:twitch.tv -www.twitch.tv -watch.twitch.tv -dev.twitch.tv
4. ...



Subdomain Scraping (Amass)

For scraping subdomain data there are two industry leading tools at the moment; Amass and Subfinder. They parse all the “sources” referenced in the previous slide, and more.

Amass has the most sources, extensible output, bruteforcing, permutation scanning, and a ton of other modes to do additional analysis of attack surfaces.

```
root@Test2:~/tools/amass# amass -d twitch.tv
twitch.tv
passport-external.aws.twitch.tv
gql.twitch.tv
pubsub-edge.twitch.tv
pubsub-edge.chat.twitch.tv
passport.twitch.tv
www.twitch.tv
m.twitch.tv
irc-ws-edge.chat.twitch.tv
irc-ws.chat.twitch.tv
app.twitch.tv
download.twitch.tv
discuss.dev.twitch.tv
invite.twitch.tv
join.twitch.tv
edgecast.hls.twitch.tv has a static DNS wildcard
blog.twitch.tv
polls.twitch.tv
th.blog.twitch.tv
link.twitch.tv
servers.twitch.tv
cis.blog.twitch.tv
graphql.prod.us-west2.twitch.tv
it.blog.twitch.tv
rc.twitch.tv
de.blog.twitch.tv
tr.blog.twitch.tv
release.twitch.tv
nl.blog.twitch.tv
canary.twitch.tv
ccu.event-engineering.twitch.tv
pong.prod.us-west2.twitch.tv
jp.blog.twitch.tv
event-panel.event-engineering.twitch.tv
assets.help.twitch.tv
uploads-regional.twitch.tv
websub-test-proxy.twitch.tv
```

Subdomain Scraping (Amass)

Amass also correlates these scraped domains to ASNs and lists what network ranges they appeared in.

Useful.

If a new ASN is discovered you can feed it back to amass intel.

```
Average DNS names processed: 49/sec

OWASP Amass v2.9.0                                https://github.com/OWASP/Amass
-------------------------------------------------------------------------------
439 names discovered - alt: 77, dns: 3, cert: 105, archive: 24, scrape: 218, api: 12
-------------------------------------------------------------------------------
ASN: 54113 - FASTLY - Fastly, US
        151.101.64.0/22         2    Subdomain Name(s)
        151.101.0.0/22          2    Subdomain Name(s)
        151.101.188.0/22        52   Subdomain Name(s)
        151.101.40.0/22         1    Subdomain Name(s)
        151.101.244.0/22        1    Subdomain Name(s)
        151.101.192.0/22        2    Subdomain Name(s)
        151.101.128.0/22        2    Subdomain Name(s)
ASN: 16509 - AMAZON-02 - Amazon.com, Inc., US
        52.24.0.0/14            35   Subdomain Name(s)
        52.84.48.0/23           4    Subdomain Name(s)
        13.35.8.0/23            4    Subdomain Name(s)
        34.208.0.0/12           15   Subdomain Name(s)
        35.160.0.0/13           16   Subdomain Name(s)
        54.254.128.0/17         1    Subdomain Name(s)
        54.186.0.0/15           9    Subdomain Name(s)
        52.18.0.0/15            2    Subdomain Name(s)
        52.36.0.0/14            6    Subdomain Name(s)
        35.178.0.0/15           1    Subdomain Name(s)
        2600:9000:202d::/48     54   Subdomain Name(s)
        52.40.0.0/14            5    Subdomain Name(s)
        54.192.144.0/22         12   Subdomain Name(s)
        52.220.0.0/15           1    Subdomain Name(s)
        54.171.0.0/16           2    Subdomain Name(s)
        3.8.0.0/14              1    Subdomain Name(s)
        54.192.12.0/22          55   Subdomain Name(s)
        2600:9000:2001::/48     26   Subdomain Name(s)
        54.148.0.0/15           10   Subdomain Name(s)
        2600:9000:204b::/48     8    Subdomain Name(s)
        52.10.0.0/15            3    Subdomain Name(s)
        2600:9000:201d::/48     8    Subdomain Name(s)
        52.84.44.0/22           1    Subdomain Name(s)
        52.88.0.0/15            7    Subdomain Name(s)
        2600:9000:2145::/48     8    Subdomain Name(s)
        184.169.128.0/17        1    Subdomain Name(s)
        52.208.0.0/13           3    Subdomain Name(s)
        52.32.0.0/14            11   Subdomain Name(s)
        13.35.124.0/22          44   Subdomain Name(s)
        50.112.128.0/19         3    Subdomain Name(s)
        52.52.0.0/15            3    Subdomain Name(s)
        54.215.128.0/18         2    Subdomain Name(s)
        54.68.0.0/15            2    Subdomain Name(s)
        54.191.0.0/16           3    Subdomain Name(s)
        54.70.0.0/15            2    Subdomain Name(s)
        54.200.0.0/15           6    Subdomain Name(s)
        50.18.0.0/18            1    Subdomain Name(s)
ASN: 0 - Private Networks
        10.0.0.0/8              66   Subdomain Name(s)
ASN: 46489 - JUSTINTV - Twitch Interactive Inc., US
        52.223.240.0/20         21   Subdomain Name(s)
        192.16.64.0/21          32   Subdomain Name(s)
        99.181.64.0/20          2    Subdomain Name(s)
        52.223.224.0/20         27   Subdomain Name(s)
        45.113.128.0/22         10   Subdomain Name(s)
        192.108.239.0/24        4    Subdomain Name(s)
        52.223.208.0/21         28   Subdomain Name(s)
        199.9.248.0/21          164  Subdomain Name(s)
        185.42.204.0/22         15   Subdomain Name(s)
```

```
ASN: 46489 - JUSTINTV - Twitch Interactive Inc., US
        52.223.240.0/20         21   Subdomain Name(s)
        192.16.64.0/21          32   Subdomain Name(s)
        99.181.64.0/20          2    Subdomain Name(s)
        52.223.224.0/20         27   Subdomain Name(s)
        45.113.128.0/22         10   Subdomain Name(s)
        192.108.239.0/24        4    Subdomain Name(s)
        52.223.208.0/21         28   Subdomain Name(s)
        199.9.248.0/21          164  Subdomain Name(s)
        185.42.204.0/22         15   Subdomain Name(s)
        23.160.0.0/24           8    Subdomain Name(s)
        52.223.192.0/20         11   Subdomain Name(s)
ASN: 40341 - Q9-AS-CAL2 - Q9 Networks Inc., CA
        162.219.8.0/21          1    Subdomain Name(s)
ASN: 14618 - AMAZON-AES - Amazon.com, Inc., US
        54.84.0.0/15            1    Subdomain Name(s)
        52.72.0.0/15            1    Subdomain Name(s)
        52.0.0.0/15             71   Subdomain Name(s)
        2600:1f18::/33          1    Subdomain Name(s)
        52.2.0.0/15             2    Subdomain Name(s)
        18.204.0.0/14           3    Subdomain Name(s)
        52.200.0.0/13           4    Subdomain Name(s)
        52.4.0.0/14             137  Subdomain Name(s)
        52.20.0.0/14            1    Subdomain Name(s)
ASN: 395224 - BITLY-AS - Bitly Inc, US
        67.199.248.0/24         4    Subdomain Name(s)
ASN: 15169 - GOOGLE - Google LLC, US
        35.185.0.0/19           1    Subdomain Name(s)
ASN: 22606 - EXACT-7 - ExactTarget, Inc., US
        13.111.18.0/24          5    Subdomain Name(s)
        13.111.19.0/24          1    Subdomain Name(s)
        13.111.20.0/24          1    Subdomain Name(s)
        13.111.97.0/24          1    Subdomain Name(s)
ASN: 11377 - SENDGRID - SendGrid, Inc., US
        167.89.64.0/19          1    Subdomain Name(s)
ASN: 38895 - AMAZON-AS-AP Amazon.com Tech Telecom, JP
        2600:9000:20c7::/48     8    Subdomain Name(s)
```

Subdomain Scraping (Subfinder v2)

Subfinder is another best in breed tool originally written by ice3man and Michael Skelton.

It is now maintained by a larger group, called projectdiscovery.io

It incorporates multiple sources, has extensible output, and more.

```
root@b0x:~# subfinder -d hackerone.com -v

                _     __ _         _
____  _| |__ / _(_)_ _  __| |___ _ _
(_-< || | '_ \  _| | ' \/ _  / -_) '_|
/__/\_,_|_.__/_| |_|_||_\__,_\___|_| v2

                projectdiscovery.io

[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[WRN] By using subfinder, you also agree to the terms of the APIs used.

[INF] Enumerating subdomains for hackerone.com
[sitedossier] www.hackerone.com
[virustotal] api.hackerone.com
[virustotal] support.hackerone.com
[virustotal] docs.hackerone.com
[virustotal] mta-sts.hackerone.com
[virustotal] mta-sts.forwarding.hackerone.com
[virustotal] a.ns.hackerone.com
[virustotal] b.ns.hackerone.com
[virustotal] links.hackerone.com
[virustotal] info.hackerone.com
[archiveis] hackerone.com
[securitytrails] email.gh-mail.hackerone.com
[securitytrails] mta-sts.managed.hackerone.com
[securitytrails] web-seo-content-for-business.theflyingkick.websitedesignresource.api.hackerone.com
[passivetotal] cf-ssl5349-protected-cover-photos-user-content.hackerone.com
[passivetotal] o1.email.hackerone.com
[passivetotal] go.hackerone.com
[passivetotal] cf-ssl5349-protected-profile-photos-user-content.hackerone.com
[passivetotal] o3.email.hackerone.com
[passivetotal] profile-photos-user-content.hackerone.com
[passivetotal] cf-ssl41462-protected-profile-photos-user-content.hackerone.com
[passivetotal] cover-photos-user-content.hackerone.com
[passivetotal] staging.hackerone.com
[passivetotal] cf-ssl41462-protected-cover-photos-user-content.hackerone.com
```

Subdomain Scraping (github-subdomains.py)

```
root@TBox4:~/tools/github-search# python3 github-subdomains.py -t "                                " -d twitch.tv > twitch.tv
root@TBox4:~/tools/github-search# cat twitch.tv |wc -l
1828
root@TBox4:~/tools/github-search# cat twitch.tv | grep -v ".tmi"
player.twitch.tv
www.twitch.tv
api.twitch.tv
blog.twitch.tv
passport.twitch.tv
m.twitch.tv
rechat.twitch.tv
go.twitch.tv
clips.twitch.tv
dev.twitch.tv
irc.chat.twitch.tv
id.twitch.tv
link.twitch.tv
discuss.dev.twitch.tv
usher.twitch.tv
secure.twitch.tv
live.twitch.tv
irc-ws.chat.twitch.tv
```

A new addition to my subdomain enumeration is scraping Github. Github-subdomains.py is a script written by Gwendal Le Coguic as part of his EPIC Github enumeration repo called “github- search” It will query the Github API for references to a root and pull out subdomains.

Note: the Github API returns somewhat random results and is rate limited. In my automation I run 5 iterations of this script. Four of them with a six second sleep in between them, and the last with a 10 second sleep, to get some consistency.

Subdomain Scraping (shosubgo)

Another valuable bespoke scraping technique is gathering subdomains from Shodan.

Shosubgo is a Go script written by inc0gbyt3 which is effective and reliable for this method.

```
root@TBox4:~/tools/shosubgo# go run main.go -d twitch.tv -s [redacted]
*.canary.twitch.tv
*.rc.twitch.tv
*.release.twitch.tv
api-a.chat.twitch.tv
api.globetrotter.external.twitch.tv
app.twitch.tv
assets.help.twitch.tv
badges.twitch.tv
ccu.event-engineering.twitch.tv
chatdepot.twitch.tv
click.twitch.tv
click-staging.twitch.tv
clips-canary.twitch.tv
clock.event-engineering.twitch.tv
costream.twitch.tv
download-staging.sings.twitch.tv
es-es.twitch.tv
event-panel.event-engineering.twitch.tv
gap-v4-cubic.event-engineering.twitch.tv
go.twitch.tv
id.twitch.tv
id-canary.twitch.tv
im.twitch.tv
ingest.twitch.tv
irc-ws.chat.twitch.tv
irc.chat.twitch.tv
ja.twitch.tv
lv.twitch.tv
metadata.twitch.tv
moonlight.twitch.tv
owl.internal.us-west-2.twitch.tv
passport.twitch.tv
pong.twitch.tv
production.helix-origin.twitch.tv
pubsub-edge.twitch.tv
pubsub-edge.chat.twitch.tv
spectre.twitch.tv
staging.helix-origin.twitch.tv
staging.passport.twitch.tv
stg.moonlight.twitch.tv
tmi.twitch.tv
ttv-redirect.m7g.twitch.tv
us-west-2.uploads-regional.twitch.tv
vpn-anycast-2.twitch.tv
```

Subdomain Scraping (Cloud Ranges)

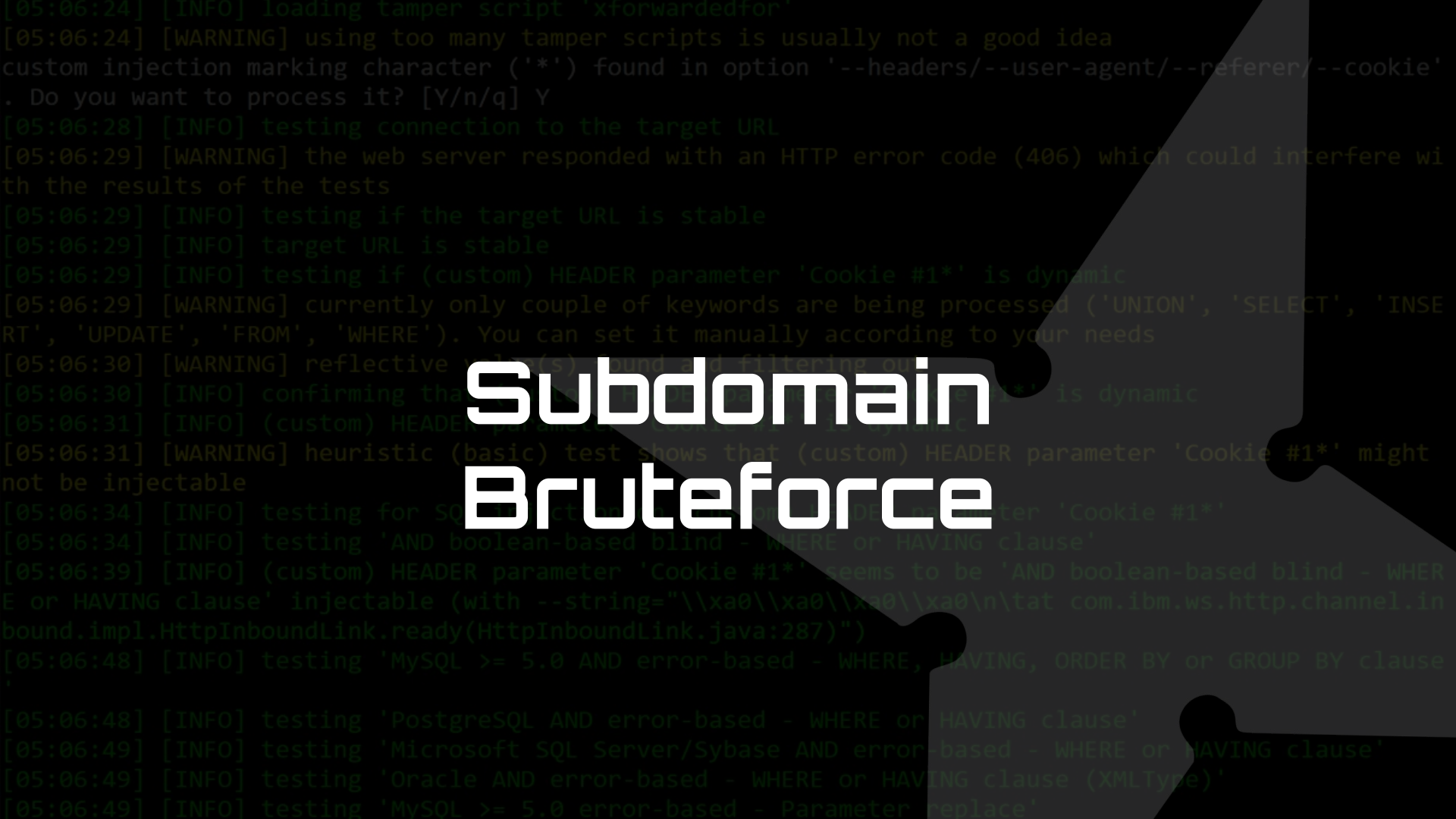
A highly valuable technique is to monitor whole cloud ranges of AWS, GCP, and Azure for SSL sites, and parse their certificates to match your target.

Doing this is cumbersome on your own but possible with something like masscan. Daehee Park outlines it here.

Luckily Sam Erb did a wonderful defcon talk on this and created a service which scans every week.

Some bash scripting required ;)

```
root@TBox4:~/tools/sslScrape# curl 'https://tls.bufferover.run/dns?q=.twitch.tv' 2>/dev/null | jq .Results
*.canary.twitch.tv
*.chat.twitch.tv
*.dev.us-west2.twitch.tv
*.go.twitch.tv
*.m.twitch.tv
*.rc.twitch.tv
*.release.twitch.tv
*.tv.twitch.tv
*.twitch.tv
[
]
app.twitch.tv
canary-extensions-worker.sings.twitch.tv
canary-extensions.sings.twitch.tv
click-staging.twitch.tv
click.marketing.twitch.tv
click.twitch.tv
cloud.marketing.twitch.tv
countess.twitch.tv
dev-extensions-worker.sings.twitch.tv
dev-extensions.sings.twitch.tv
dev.twitch.tv
dev.us-west2.twitch.tv
discuss.dev.twitch.tv
dmca.twitch.tv
download.twitch.tv
extensions-worker.sings.twitch.tv
extensions.sings.twitch.tv
i18n.m7g.twitch.tv
id.twitch.tv
image.marketing.twitch.tv
invite.twitch.tv
join.twitch.tv
lvs-prod.twitch.tv
m.twitch.tv
metadata.twitch.tv
pages.marketing.twitch.tv
passport.twitch.tv
polls.twitch.tv
pong-staging.twitch.tv
```


Subdomain
Bruteforce

Subdomain Bruting

At this point we move into guessing for live subdomains.

If we try and resolve

thistotallydoesntexist.company.com we will *usually* not get a record.

So we can use a large list of common subdomain names and just try and resolve them analyzing if they succeed.

The problem in this method is that only using one DNS server to do this will take forever. Some tools have come out that are both threaded and use multiple DNS resolvers simultaneously. This speeds up this process significantly. Massdns by @blechschmidt pioneered this idea.

Amass (8 revolvers by default) does this with the -rf flag.

Subdomain Bruting (Amass)

Amass offers bruteforcing via the “enum” tool using the “brute” switch.

- amass enum -brute -d twitch.tv -src

It has a built in list but you can specify your own lists.

You can also specify any number of resolvers

- amass enum -brute -d twitch.tv -rf resolvers.txt -w bruteforce.list

Doing this with amass also gives us the opportunity to resolve the found domains.

I haven’t checked out aisdnsbrute yet but i’ve heard it’s fast.

Subdomain Bruting (shuffleDNS)

```
root@b0x:~# shuffledns -d hackerone.com -w words.txt -r resolvers-excellent.txt

                __        _______      __
  ___ / /  __ __/ _/ _/ /__  ___/ /__ ___
 (_-</ _ \/ // / _/ _/ / -_)/ _  / _ \(_-<
/___/_//_/\_,_/_//_//_/\__/ \_,_/_//_/___/ v1

                projectdiscovery.io

[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[INF] Started generating bruteforce permutation
[INF] Generating permutations took 46.728µs
[INF] Creating temporary massdns output file: /tmp/shuffledns474707985/bpbe1p67dd9u0me2rqs0
[INF] Executing massdns on hackerone.com
[INF] Massdns execution took 165.709341ms
[INF] Parsing output and removing wildcards
[INF] Finished enumeration, started writing output
support.hackerone.com
b.ns.hackerone.com
api.hackerone.com
www.hackerone.com
docs.hackerone.com
mta-sts.managed.hackerone.com
mta-sts.forwarding.hackerone.com
mta-sts.hackerone.com
a.ns.hackerone.com
[INF] Finished resolving. Hack the Planet!
```

If you like separating the work to different tools or prefer the massdns core you can use shuffleDNS by the ProjectDiscovery team.

Subdomain Bruting Lists

A multi resolver, threaded subdomain bruter is only as good as it's wordlist.

There are two trains of thought here:

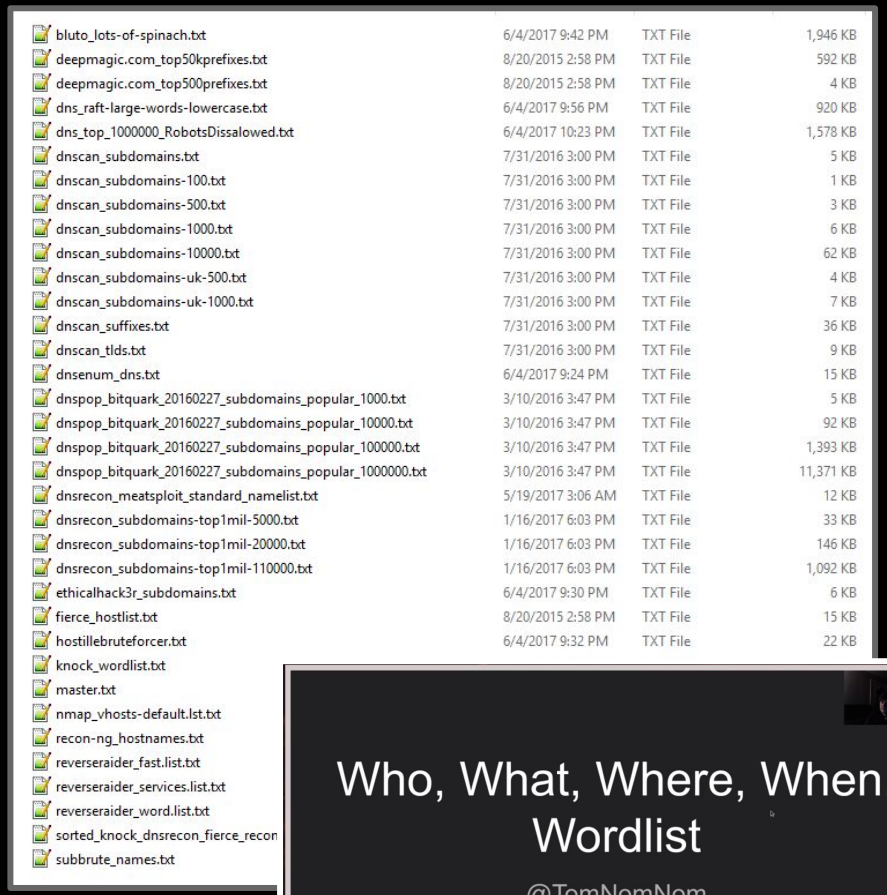
- Tailored wordlists
- Massive wordlists

Both have advantages.

My all.txt file is still what i use on a regular basis. It combines 7 years of DNS bruteforce lists into one.

But you can make customized wordlists with something like what TomNomNom talked on yesterday.

I also think there was a tool for this recently...



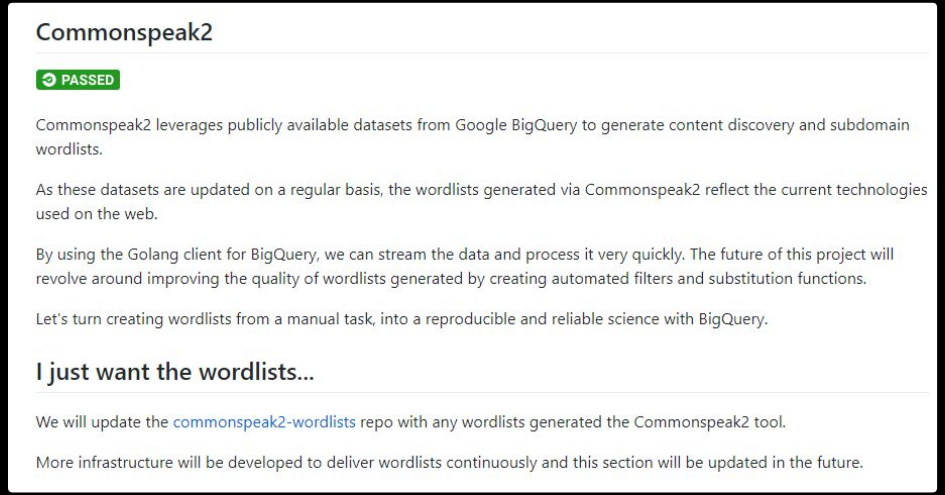
bluto_lots-of-spinach.txt	6/4/2017 9:42 PM	TXT File	1,946 KB
deepmagic.com_top50kprefixes.txt	8/20/2015 2:58 PM	TXT File	592 KB
deepmagic.com_top500prefixes.txt	8/20/2015 2:58 PM	TXT File	4 KB
dns_raft-large-words-lowercase.txt	6/4/2017 9:56 PM	TXT File	920 KB
dns_top_1000000_RobotsDissalowed.txt	6/4/2017 10:23 PM	TXT File	1,578 KB
dnscan_subdomains.txt	7/31/2016 3:00 PM	TXT File	5 KB
dnscan_subdomains-100.txt	7/31/2016 3:00 PM	TXT File	1 KB
dnscan_subdomains-500.txt	7/31/2016 3:00 PM	TXT File	3 KB
dnscan_subdomains-1000.txt	7/31/2016 3:00 PM	TXT File	6 KB
dnscan_subdomains-10000.txt	7/31/2016 3:00 PM	TXT File	62 KB
dnscan_subdomains-uk-500.txt	7/31/2016 3:00 PM	TXT File	4 KB
dnscan_subdomains-uk-1000.txt	7/31/2016 3:00 PM	TXT File	7 KB
dnscan_suffixes.txt	7/31/2016 3:00 PM	TXT File	36 KB
dnscan_tlds.txt	7/31/2016 3:00 PM	TXT File	9 KB
dnsenum_dns.txt	6/4/2017 9:24 PM	TXT File	15 KB
dnspop_bitquark_20160227_subdomains_popular_1000.txt	3/10/2016 3:47 PM	TXT File	5 KB
dnspop_bitquark_20160227_subdomains_popular_10000.txt	3/10/2016 3:47 PM	TXT File	92 KB
dnspop_bitquark_20160227_subdomains_popular_100000.txt	3/10/2016 3:47 PM	TXT File	1,393 KB
dnspop_bitquark_20160227_subdomains_popular_1000000.txt	3/10/2016 3:47 PM	TXT File	11,371 KB
dnsrecon_meatsploit_standard_namelist.txt	5/19/2017 3:06 AM	TXT File	12 KB
dnsrecon_subdomains-top1mil-5000.txt	1/16/2017 6:03 PM	TXT File	33 KB
dnsrecon_subdomains-top1mil-20000.txt	1/16/2017 6:03 PM	TXT File	146 KB
dnsrecon_subdomains-top1mil-110000.txt	1/16/2017 6:03 PM	TXT File	1,092 KB
ethicalhack3r_subdomains.txt	6/4/2017 9:30 PM	TXT File	6 KB
fierce_hostlist.txt	8/20/2015 2:58 PM	TXT File	15 KB
hostillebruteforcer.txt	6/4/2017 9:32 PM	TXT File	22 KB
knock_wordlist.txt			
master.txt			
nmap_vhosts-default.lst.txt			
recon-ng_hostnames.txt			
reverseraider_fast.list.txt			
reverseraider_services.list.txt			
reverseraider_word.list.txt			
sorted_knock_dnsrecon_fierce_recon			
subbrute_names.txt			

Subdomain Bruting Lists

New lists for subdomain bruteforce are relatively the same nowadays, but the 1st team to really iterate on this is the AssetNote team with their commonspeak data collection.

The all.txt file includes commonspeak v1 data but there is also a second version of commonspeak data out:

- https://github.com/assetnote/commonspeak2



Alteration Scanning

When bruteforcing or gathering subdomains via scraping you may come across a naming pattern in these subdomains.

Even though you may not have found it yet, there may be other targets that conform to naming conventions.

In addition, sometimes targets are not explicitly protected across naming conventions.

The first tool to attempt to recognize these patterns and bruteforce for some of them was altdns written by Naffy and Shubs.

Now amass contains logic to check for these “permutations”. Amass includes this analysis in a default run Some personal experience cited on the next page.

dev.company.com

dev1.company.com

dev2.company.com

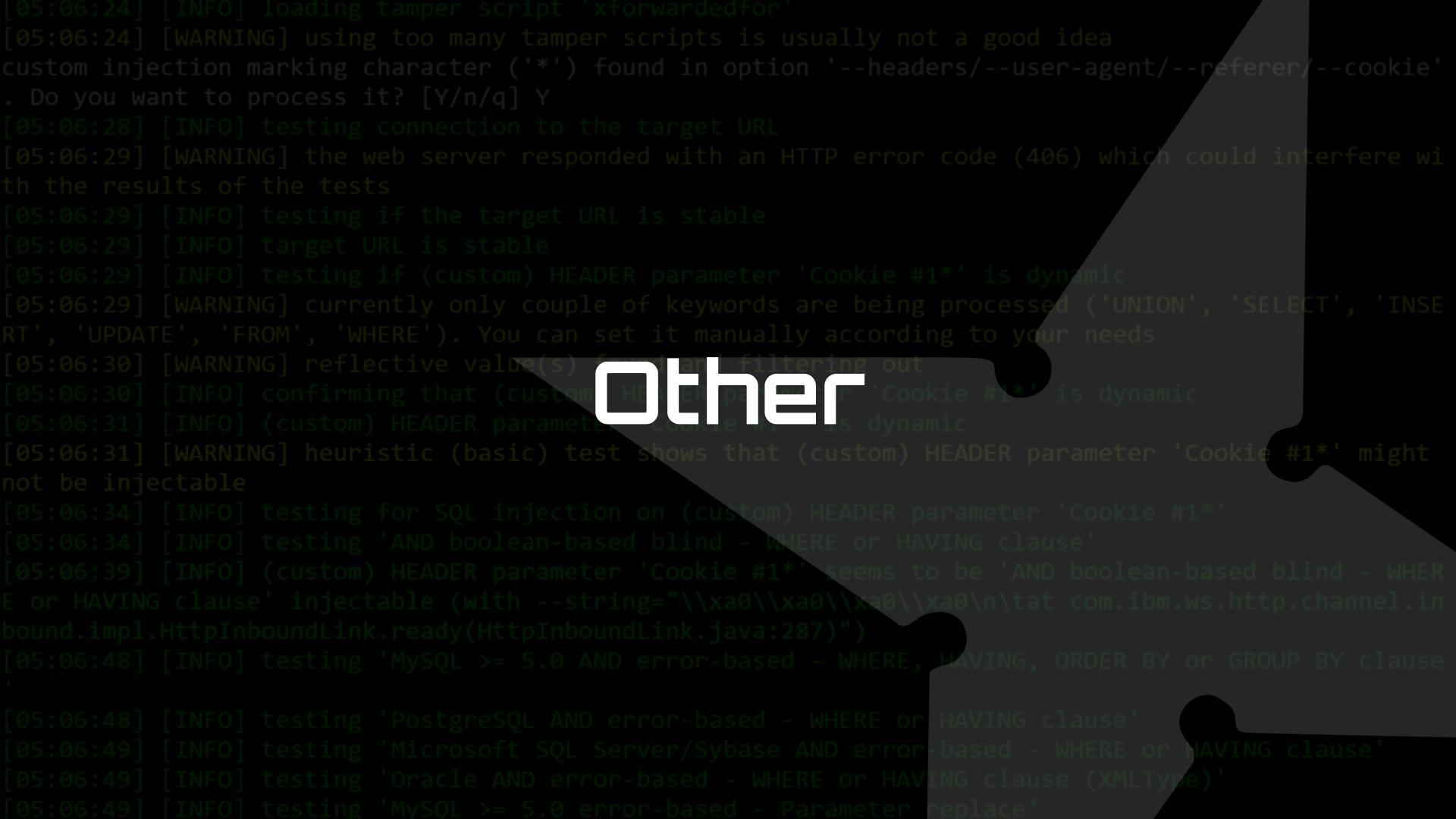
dev-1.company.com

Dev-2.company.com

Dev.1.company.com

Dev.2.company.com

Alteration Scanning (WAF Bypass)


Other

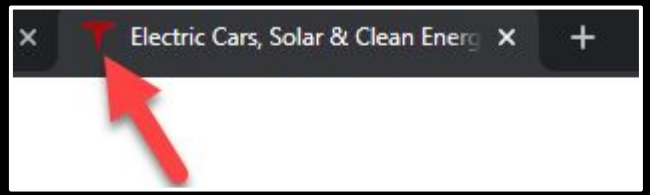
Favicon Analysis (favfreak)

A more fringe technique of discovering a brands assets is taking their favicon and hashing it. You can then take this hash and search shodan for it. You can also scan IP ranges and cloud blocks to find assets with the same hash.

In addition you can make hashes of commonly used admin portals or framework logins. This method is useful when an org has modified URL paths and maybe your scanners are only looking for a specific path. If the org has changed it all you won't find it, but org's rarely change the favicon for the frameworks. Check out FavFreak by Devansh Batham (0xAsm0d3us)

```
shodan search org:"Target"
http.favicon.hash:116323821 --fields
ip str,port --separator " " | awk '{print
$1":"$2}'
```

```
levi@Asm0d3us-Hackb0x:~/Desktop/proc$ cat urls.txt | python3 favfreak.py -o output

 _____ _____ _ _ _____ _____ _____ _____ _ _
|  ___|  ___| \ / |  ___|  ___|  ___|  ___| | |
|  ___|  ___| \ / |  ___|  ___| |_  |  ___| | |
|_|   |_____|_| \_|_____|_|   |_____|_|   |_|_|

                - Coded with <3 by Devansh Batham

[ERR] Not Fetched 'http://accounts-                    com'
[INFO]Fetched 'http://apiproxy-movi
[ERR] Not Fetched 'http://abossmoni
[ERR] Not Fetched 'https://apiproxy
[ERR] Not Fetched 'https://android-
[ERR] Not Fetched 'https://appcdn.p
[ERR] Not Fetched 'https://accounts                    .com'
[ERR] Not Fetched 'http://apiproxy.
[INFO]Fetched 'https://credit-card.
[INFO]Fetched 'https://apiproxy-mov
```




```
[FingerPrint Based Detection Results] -

[spring-boot] 116323821 - count : 2
[big-IP] 878647854 - count : 2
[slack-instance] 99395752 - count : 1
```

Port Analysis (masscan)

Most hacker education would have you use nmap here, but masscan by Robert Graham is much faster for general “finding-open-ports-on-TCP”. Chaining masscan’s output to then be nmap’ed can save a lot of time.

Masscan achieves this speed with a re-written TCP/IP stack, true multi-threading, and is written in C.

Sample syntax for scanning a list of IPs:

- masscan -p1-65535 -iL $ipFile --max-rate 1800 -oG $outPutFile.log

A full syntax guide of masscan (authored by Daniel Miessler) can be found here: https://danielmiessler.com/study/masscan/

```
root@Test2:~# host twitch.tv
twitch.tv has address 151.101.194.167
twitch.tv has address 151.101.2.167
twitch.tv has address 151.101.130.167
twitch.tv has address 151.101.66.167
twitch.tv mail is handled by 30 alt2.aspmx.l.google.com.
twitch.tv mail is handled by 50 aspmx3.googlemail.com.
twitch.tv mail is handled by 10 aspmx.l.google.com.
twitch.tv mail is handled by 40 aspmx2.googlemail.com.
twitch.tv mail is handled by 20 alt1.aspmx.l.google.com.
root@Test2:~# masscan -p1-65535 151.101.194.167 --max-rate 1800

Starting masscan 1.0.3 (http://bit.ly/14GZzcT) at 2019-01-18 06:34:05 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Discovered open port 80/tcp on 151.101.194.167
Discovered open port 443/tcp on 151.101.194.167
```

Port Analysis (dnmasscan)

One limitation of masscan is that it only scans IP addresses. Y

you can write you own simple converter script or you can use something like [dnmasscan](#) by [@rastating](#)

```
root:~# dnmasscan example.txt dns.log -p80,443 -oG masscan.log
[*] Resolving domains...
[*] Saved resolved addresses to dns.log
[*] Launching masscan...
------------------------

Starting masscan 1.0.6 (http://bit.ly/14GZzcT) at 2019-09-17 22:56:48 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 6 hosts [2 ports/host]
```

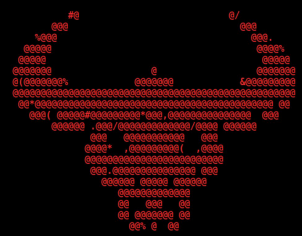
```
root:~# cat dns.log
github.com
==========
140.82.118.4

rastating.github.io
===================
185.199.111.153
185.199.109.153
185.199.108.153
185.199.110.153

google.com
==========
172.217.169.14
```

```
root:~# cat masscan.log
# Masscan 1.0.6 scan initiated Tue Sep 17 22:56:48 2019
# Ports scanned: TCP(1;80-80,) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Timestamp: 1568761008   Host: 185.199.111.153 ()        Ports: 80/open/tcp//http//
Timestamp: 1568761008   Host: 185.199.109.153 ()        Ports: 443/open/tcp//https//
Timestamp: 1568761008   Host: 185.199.110.153 ()        Ports: 443/open/tcp//https//
Timestamp: 1568761008   Host: 185.199.108.153 ()        Ports: 443/open/tcp//https//
Timestamp: 1568761008   Host: 185.199.111.153 ()        Ports: 443/open/tcp//https//
Timestamp: 1568761008   Host: 172.217.169.14 () Ports: 443/open/tcp//https//
Timestamp: 1568761008   Host: 172.217.169.14 () Ports: 80/open/tcp//http//
Timestamp: 1568761008   Host: 185.199.108.153 ()        Ports: 80/open/tcp//http//
Timestamp: 1568761008   Host: 140.82.118.4 ()   Ports: 80/open/tcp//http//
Timestamp: 1568761008   Host: 140.82.118.4 ()   Ports: 443/open/tcp//https//
Timestamp: 1568761008   Host: 185.199.109.153 ()        Ports: 80/open/tcp//http//
Timestamp: 1568761008   Host: 185.199.110.153 ()        Ports: 80/open/tcp//http//
# Masscan done at Tue Sep 17 22:56:59 2019
```

Service Scanning (brutespray)

When we get this service/port information we can feed it to nmap to get a OG outputfile.

We can then scan the remote administration protocols for default passwords with a tool called Burtespray by @x90skysn3k which takes the nmap OG file format.



BRUTESPRAY

```
Loading File: \

Welcome to interactive mode!

WARNING: Leaving an option blank will leave it empty and refer to default

Available services to brute-force:
Service: ftp on port 21 with 9 hosts
Service: smtp on port 25 with 8 hosts
Service: smtp on port 587 with 1 hosts
Service: ssh on port 22 with 8 hosts
Service: telnet on port 23 with 1 hosts
Service: mysql on port 3306 with 1 hosts

Enter services you want to brute - default all (ssh,ftp,etc): ftp,ssh,telnet
Enter the number of parallel threads (default is 2): 5
Enter the number of parallel hosts to scan per service (default is 1): 10
Would you like to specify a wordlist? (y/n): y
Enter a userlist you would like to use:
Enter a passlist you would like to use: /usr/share/wordlists/
/usr/share/wordlists/dirb            /usr/share/wordlists/fern-wifi       /usr/share/wordlists/sqlmap.txt
/usr/share/wordlists/dirbuster       /usr/share/wordlists/metasploit      /usr/share/wordlists/wfuzz
/usr/share/wordlists/dnsmap.txt      /usr/share/wordlists/nmap.lst
/usr/share/wordlists/fasttrack.txt   /usr/share/wordlists/rockyou.txt.gz
Enter a passlist you would like to use: /usr/share/wordlists/metasploit/password.lst
Would to specify a single username or password (y/n): y
Enter a username: admin

Starting to brute, please make sure to use the right amount of threads(-t) and parallel hosts(-T)...  \

Brute-Forcing...
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>
```

Github Dorking (manual)

Many organizations quickly grow in their engineering teams. Sooner or later a new developer, intern, contractor, or other staff will leak source code online, usually through a public Github.com repo that they mistakenly thought they had set private.

Enjoy my quick github dork collection:

https://gist.github.com/jhaddix/1fb7ab2409ab579178d2a79959909b33

*** Helps if your console supports clickable hyperlinks*

The repo mentioned earlier by Gwendal Le Coguic called “github-search” has some automated github tools for this as well.

Also check out @th3g3ntelman’s full module on Github and Sensitive data Exposure.

```
root@Test2:~# bash Gdorkslinks.sh twitch.tv

************ Github Dork Links (must be logged in) ******************
  password
https://github.com/search?q=%22twitch.tv%22+password&type=Code
https://github.com/search?q=%22twitch%22+password&type=Code
 npmrc _auth
https://github.com/search?q=%22twitch.tv%22+npmrc%20_auth&type=Code
https://github.com/search?q=%22twitch%22+npmrc%20_auth&type=Code
 dockercfg
https://github.com/search?q=%22twitch.tv%22+dockercfg&type=Code
https://github.com/search?q=%22twitch%22+dockercfg&type=Code
  pem private
https://github.com/search?q=%22twitch.tv%22+pem%20private&type=Code
https://github.com/search?q=%22twitch%22+extension:pem%20private&type=Code
  id_rsa
https://github.com/search?q=%22twitch.tv%22+id_rsa&type=Code
https://github.com/search?q=%22twitch%22+id_rsa&type=Code
 aws_access_key_id
https://github.com/search?q=%22twitch.tv%22+aws_access_key_id&type=Code
https://github.com/search?q=%22twitch%22+aws_access_key_id&type=Code
  s3cfg
https://github.com/search?q=%22twitch.tv%22+s3cfg&type=Code
https://github.com/search?q=%22twitch%22+s3cfg&type=Code
 htpasswd
https://github.com/search?q=%22twitch.tv%22+htpasswd&type=Code
https://github.com/search?q=%22twitch%22+htpasswd&type=Code
 git-credentials
https://github.com/search?q=%22twitch.tv%22+git-credentials&type=Code
https://github.com/search?q=%22twitch%22+git-credentials&type=Code
 bashrc password
https://github.com/search?q=%22twitch.tv%22+bashrc%20password&type=Code
https://github.com/search?q=%22twitch%22+bashrc%20password&type=Code
 sshd_config
https://github.com/search?q=%22twitch.tv%22+sshd_config&type=Code
```

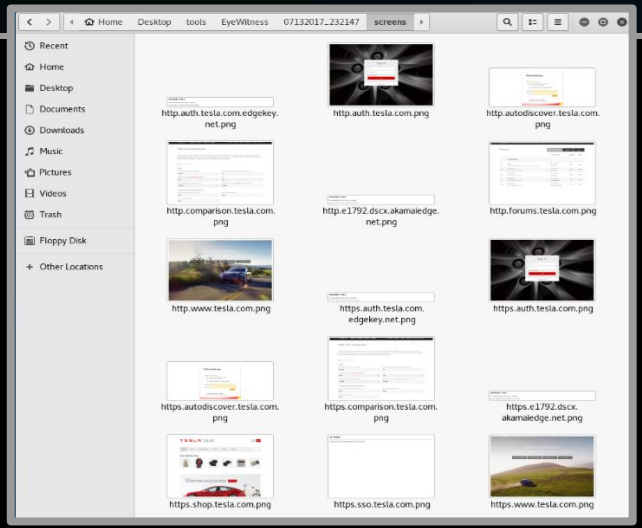
Screenshotting (Eyewitness, Aquatone, httpscreenshot)

At this point we have a lot of attack surface. We can feed possible domains to a tool and attempt to screenshot the results. This will allow us to “eye-ball” things that might be interesting.

There are many tools for this. Aquatone is a wider recon framework that does this, HTTPscreenshot, and Eyewitness. I use Eyewitness because it will prepend both the http and https protocol for each domain we have observed. I’m not highly tied to this tool though, find one that works for you.

Also, check out WitnessMe

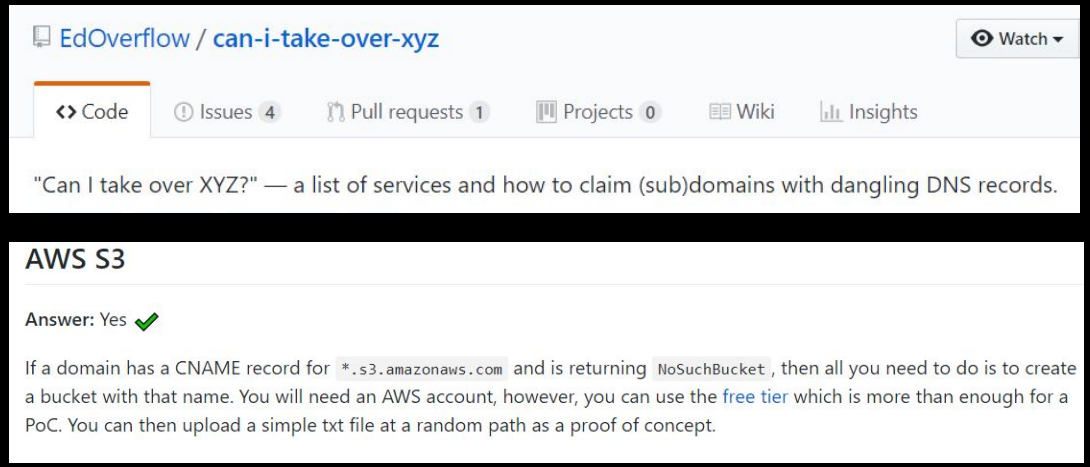
```
################################################################################
#                              EyeWitness                                      #
################################################################################

Starting Web Requests (20 Hosts)
Attempting to screenshot http://auth.tesla.com
Attempting to screenshot https://auth.tesla.com
Attempting to screenshot http://auth.tesla.com.edgekey.net
Attempting to screenshot https://auth.tesla.com.edgekey.net
Attempting to screenshot http://autodiscover.tesla.com
Attempting to screenshot https://autodiscover.tesla.com
Attempting to screenshot http://comparison.tesla.com
Attempting to screenshot https://comparison.tesla.com
Attempting to screenshot http://e1792.dscx.akamaiedge.net
Attempting to screenshot https://e1792.dscx.akamaiedge.net
Attempting to screenshot http://forums.tesla.com
[*] Hit timeout limit when connecting to http://comparison.tesla.com, retrying
Attempting to screenshot https://forums.tesla.com
```



Subdomain takeover (can i take over xyz)

“Subdomain takeover vulnerabilities occur when a subdomain (subdomain.example.com) is pointing to a service (e.g. GitHub pages, Heroku, etc.) that has been removed or deleted. This allows an attacker to set up a page on the service that was being used and point their page to that subdomain. For example, if subdomain.example.com was pointing to a GitHub page and the user decided to delete their GitHub page, an attacker can now create a GitHub page, add a CNAME file containing subdomain.example.com, and claim subdomain.example.com.”

A great resource for subdomain takeover is Ed Overlow’s repo can-i-take-over-xyz

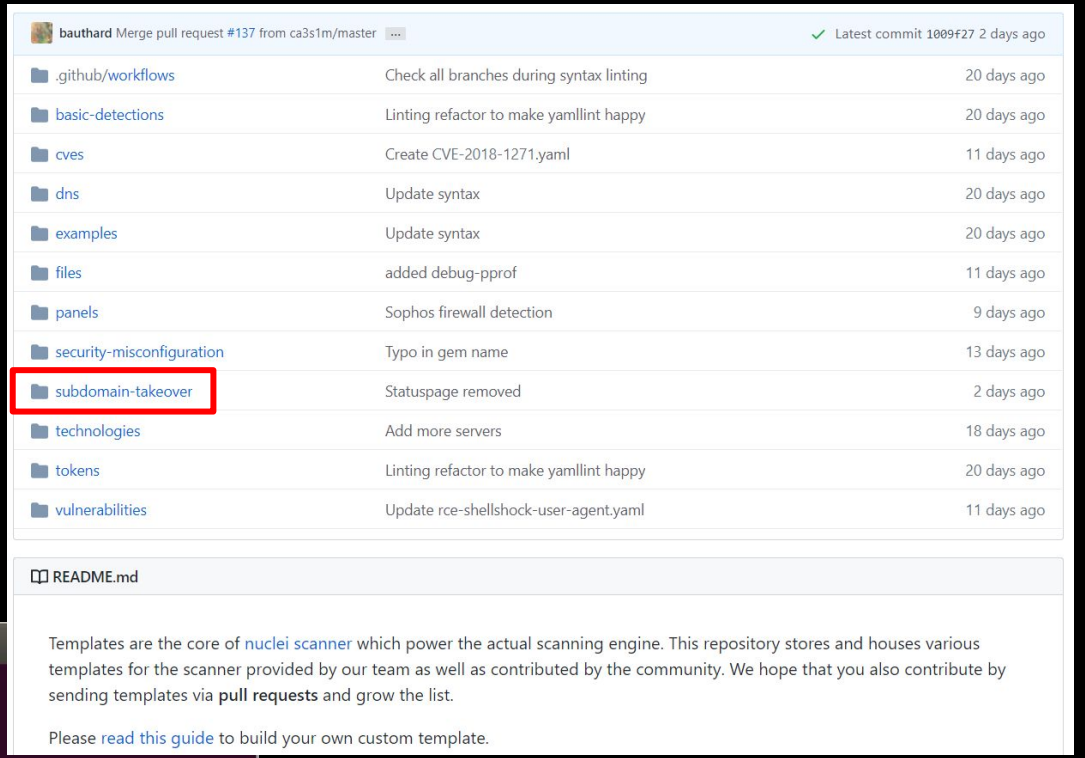


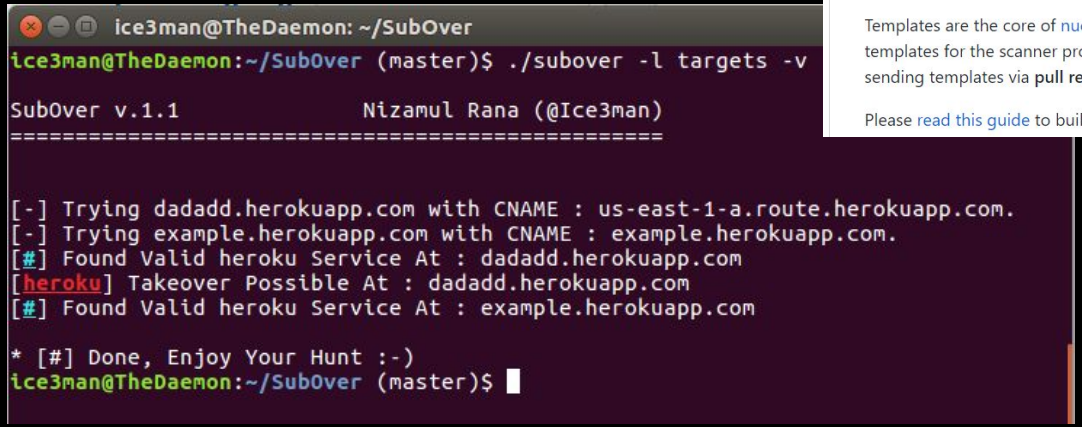
Subdomain takeover
(SubOver & nuclei)

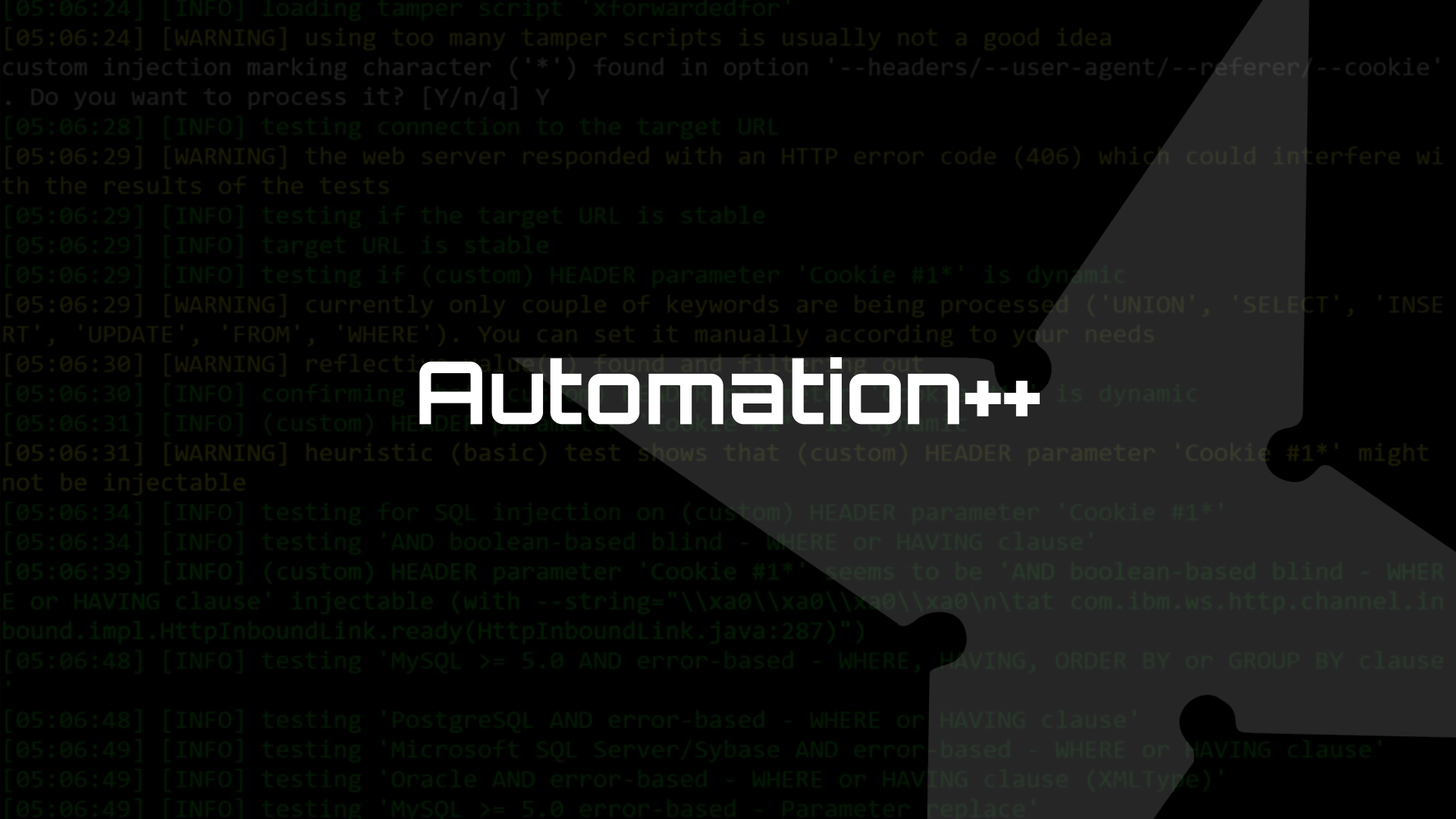
To find subdomain takeovers we can use a few tools.

SubOver is a discontinued stand alone tool by Ice3man and has since been incorporated to Project Discovery's nuclei scanner.

Nuclei is part of a larger scanning framework but boasts the most takeover checks of any tool i've seen.






Automation++

Extending tools (interlace)

Eventually you will want to make script or recon framework of you own. Quickly you will come up against some problems:

- Not all tools extend to take different sources of input
- Some lack threading.
- Not all can be distributed

You can rewrite a tool yourself to handle these issues but some help does exist here.

Interlace by Michael Skelton aka Codingo is an awesome tool than help glue together a recon framework.

Interlace can take these tools and add support for: CIDR input, Glob input, threading, proxying, queued commands, and more.

Hakluke wrote a great guide on it here.

If I were scanning 1000 hosts, that would take too long to write out, so I might have used some bash scripting instead:

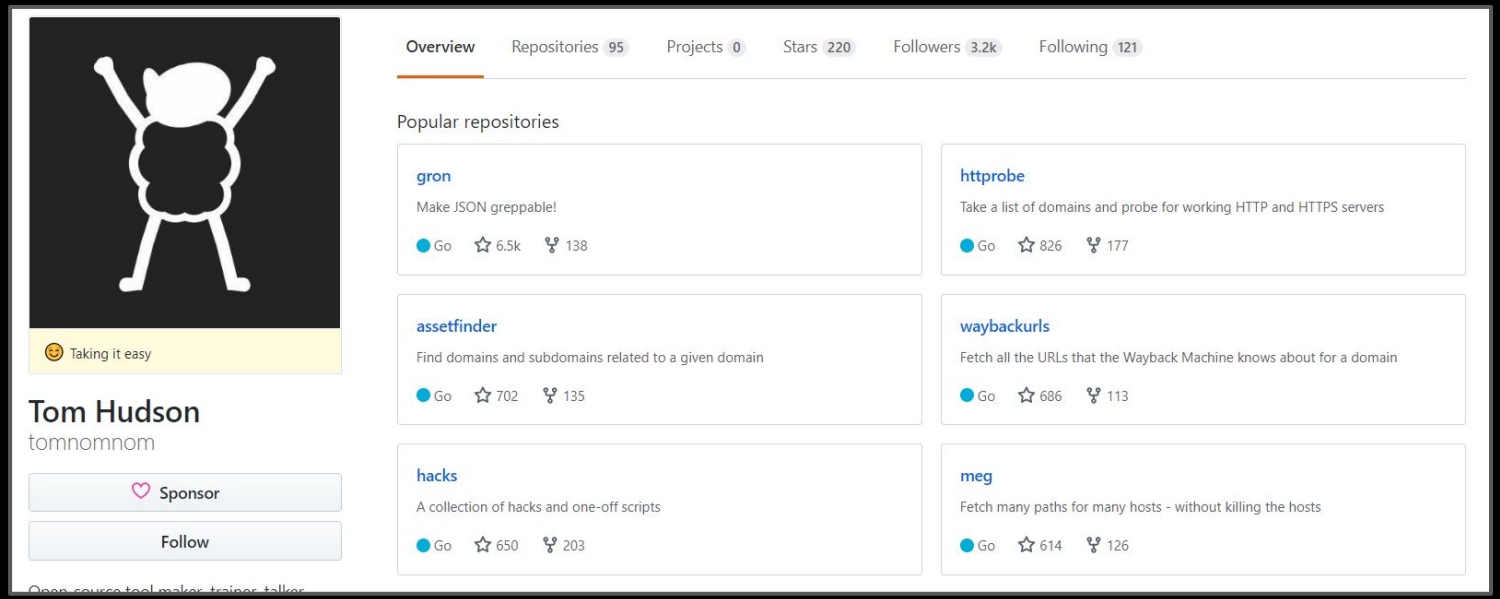
```
while read line; do
    nikto --host $line > ./$line-nikto.txt;
done < targets.txt
```

Problem? It's still single threaded and too slow. Especially if, like me, you are suffering from ridiculously slow internet speeds in Australia. This is a great example of a situation where Interlace excels:

```
luke$ interlace -tL ./targets.txt -threads 5 -c "nikto --host
_target_ > ./_target_-nikto.txt" -v
=============================================
Interlace v1.2 by Michael Skelton (@codingo_)
=============================================
[13:06:16] [VERBOSE] [nikto --host yahoo.com > ./yahoo.com-nikto.txt]
Added after processing
[13:06:16] [VERBOSE] [nikto --host google.com > ./google.com-
nikto.txt] Added after processing
[13:06:16] [VERBOSE] [nikto --host hackerone.com > ./hackerone.com-
nikto.txt] Added after processing
[13:06:16] [VERBOSE] [nikto --host bugcrowd.com > ./bugcrowd.com-
nikto.txt] Added after processing
[13:06:16] [THREAD] [nikto --host google.com > ./google.com-
nikto.txt] Added to Queue
[13:06:16] [THREAD] [nikto --host hackerone.com > ./hackerone.com-
nikto.txt] Added to Queue
[13:06:16] [THREAD] [nikto --host bugcrowd.com > ./bugcrowd.com-
nikto.txt] Added to Queue
[13:06:16] [THREAD] [nikto --host yahoo.com > ./yahoo.com-nikto.txt]
Added to Queue
```

Extending tools (anything TomNomNom writes)

Tomnomnom has an extensive repo of tools which are awesome. I highly suggest you check them all out.



Frameworks

It could be recon is not really your thing. That's all right.

Several hunters have open sourced their automation at this point and you can choose one that fits you and use it without worrying too much. I usually classify recon frameworks in rough tiers:

C-Tier: automation built around scripting up other tools in bash or python. Step based, no workflow. Few techniques. Little extensibility.

B-Tier: automation writing a few of their own modules. Some GUI or advanced workflow. Medium number of techniques. Runs point-in-time. Flat files.

A-Tier: (maybe) automation writing all their own modules. Has GUI. Runs iterativley. Manages data via db.

S-Tier: automation writing their own modules. Has GUI. Runs iterativley. Manages data via db. Scales across multiple boxes. Sends alerts to user. Uses novel techniques and iterates quickly. ML + AI.

Frameworks

Warning: My scale is mostly subjective and can be missing factors. It is also based off of my rough experience and gut feel.

Frameworks (C-Tier)

- https://github.com/AdmiralGaust/bountyRecon
- https://github.com/offhourscoding/recon
- https://github.com/Sambal0x/Recon-tools
- https://github.com/JoshuaMart/AutoRecon
- https://github.com/yourbuddy25/Hunter
- https://github.com/venom26/recon/blob/master/ultimate_recon.sh
- https://gist.github.com/dwisiswant0/5f647e3d406b5e984e6d69d3538968cd

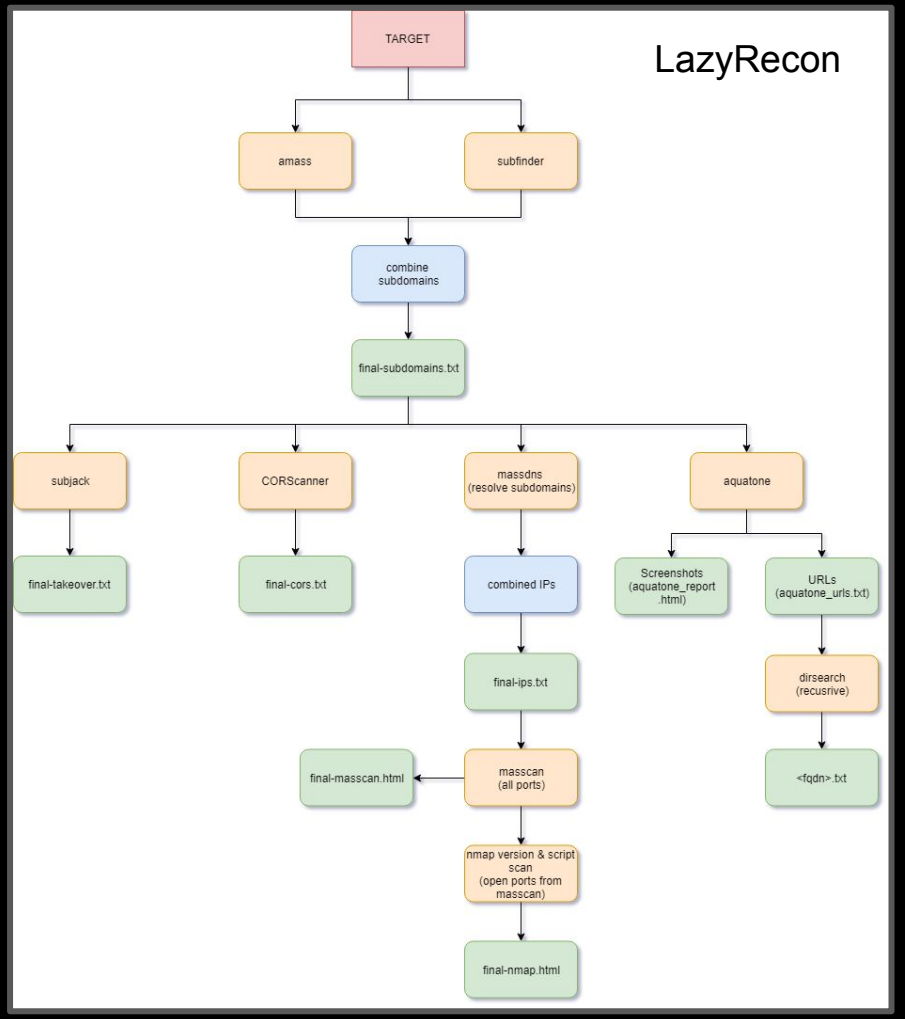
ultimate_recon.sh

135 lines (113 sloc) | 4.48 KB — Raw | Blame | History

```
#!/bin/bash

while getopts ":d:" input;do
        case "$input" in
                d) domain=${OPTARG}
                        ;;
                esac
        done
if [ -z "$domain" ]
        then
                echo "Please give a domain like \"-d domain.com\""
                exit 1
fi

sublist3r -d $domain -v -o op.txt
subfinder -d $domain -o op.txt
assetfinder --subs-only $domain | tee -a op.txt
amass enum -passive -d $doamin | tee -a op.txt
amass enum -active -d $domain -ip | tee -a amass_ips.txt
cat amass_ips.txt | awk '{print $1}' | tee -a op.txt
cat op.txt | sort -u | tee -a all.txt
echo -e "######Starting Bruteforce######\n"
altdns -i all.txt -o data_output -w ~/tools/recon/patterns.txt -r -s results_output.txt
mv results_output.txt dns_op.txt
cat dns_op.txt output.txt

cat output.txt | sort -u | tee -a all.txt
echo "Checking for alive subdomains"
cat all.txt | httprobe | tee -a alive2.txt
cat alive2.txt | sort -u | tee -a alive.txt

~/tools/massdns/bin/massdns -r ~/tools/massdns/lists/resolvers.txt -q -t A -o S -w massdns.raw all.txt
cat massdns.raw | grep -e ' A ' |  cut -d 'A' -f 2 | tr -d ' ' > massdns.txt
cat *.txt | sort -V | uniq > $IP_PATH/final-ips.txt
echo -e "${BLUE}[*] Check the list of IP addresses at $IP_PATH/final-ips.txt${RESET}"

echo "Starting Nuclei"
mkdir nuclei_op
nuclei -l alive.txt -t "/root/tools/nuclei-templates/cves/*.yaml" -c 60 -o nuclei_op/cves.txt
nuclei -l alive.txt -t "/root/tools/nuclei-templates/files/*.yaml" -c 60 -o nuclei_op/files.txt
nuclei -l alive.txt -t "/root/tools/nuclei-templates/panels/*.yaml" -c 60 -o nuclei_op/panels.txt
nuclei -l alive.txt -t "/root/tools/nuclei-templates/security-misconfiguration/*.yaml" -c 60 -o nuclei_op/security-misconfiguration.txt
nuclei -l alive.txt -t "/root/tools/nuclei-templates/technologies/*.yaml" -c 60 -o nuclei_op/technologies.txt
nuclei -l alive.txt -t "/root/tools/nuclei-templates/tokens/*.yaml" -c 60 -o nuclei_op/tokens.txt
nuclei -l alive.txt -t "/root/tools/nuclei-templates/vulnerabilities/*.yaml" -c 60 -o nuclei_op/vulnerabilities.txt
```

Frameworks (B-Tier)

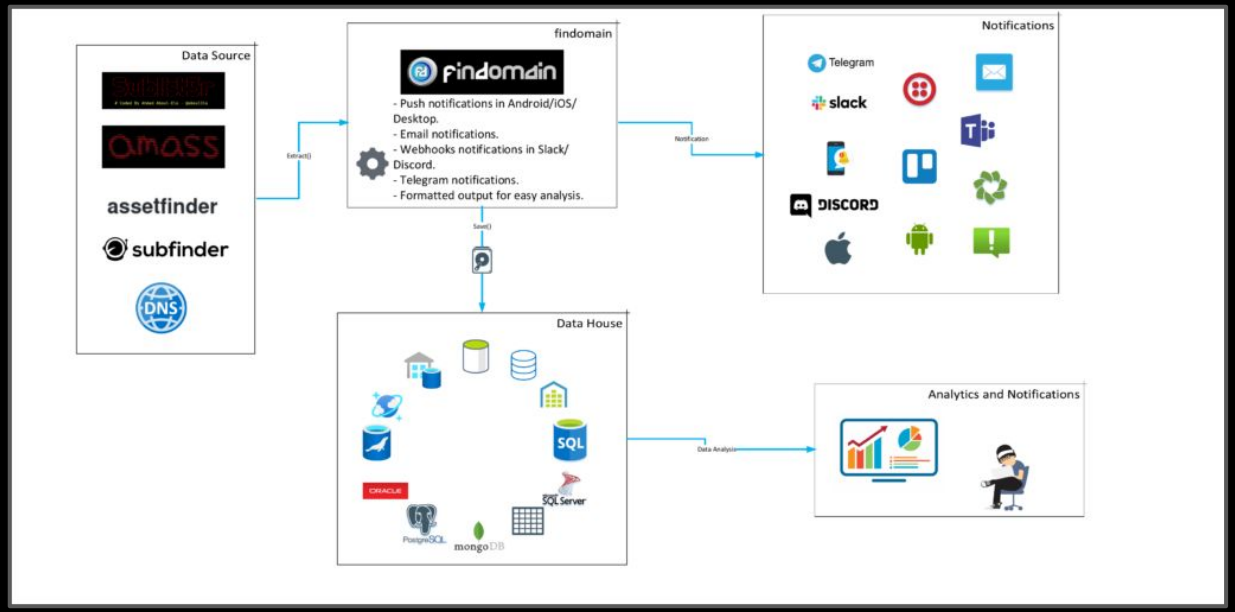
https://github.com/capt-meelo/LazyRecon	https://github.com/Screetsec/Sudomy
https://github.com/phspade/Automated-Scanner	https://github.com/devanshbatham/Gorecon
https://github.com/shmilylty/OneForAll	https://github.com/LordNeoStark/tugarecon
https://github.com/SolomonSklash/chomp-scan	https://github.com/s0md3v/photon
https://github.com/TypeError/domained	



Frameworks

(A-Tier)

https://github.com/Edu4rdSHL/findomain
https://github.com/SilverPoision/Rock-ON
https://github.com/epi052/recon-pipeline



Findomain+ alert: 7 new subdomains found for [redacted]

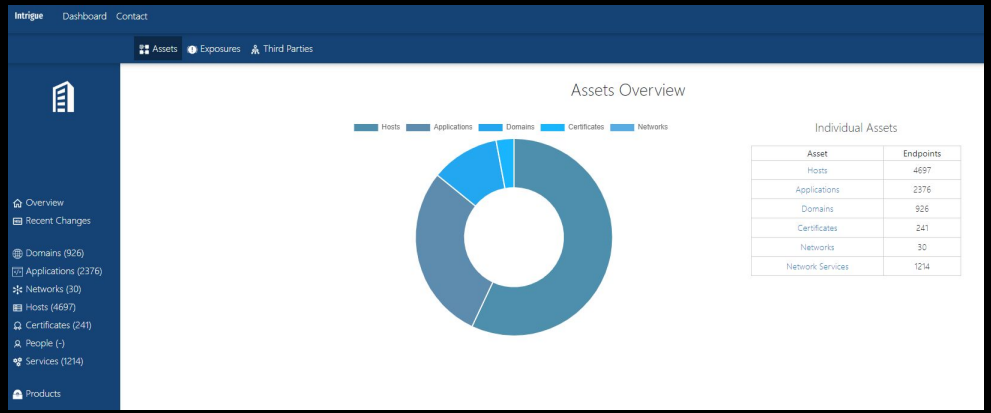
monitoring@findomain.app 6:36 AM (13 hours ago)

to [redacted]

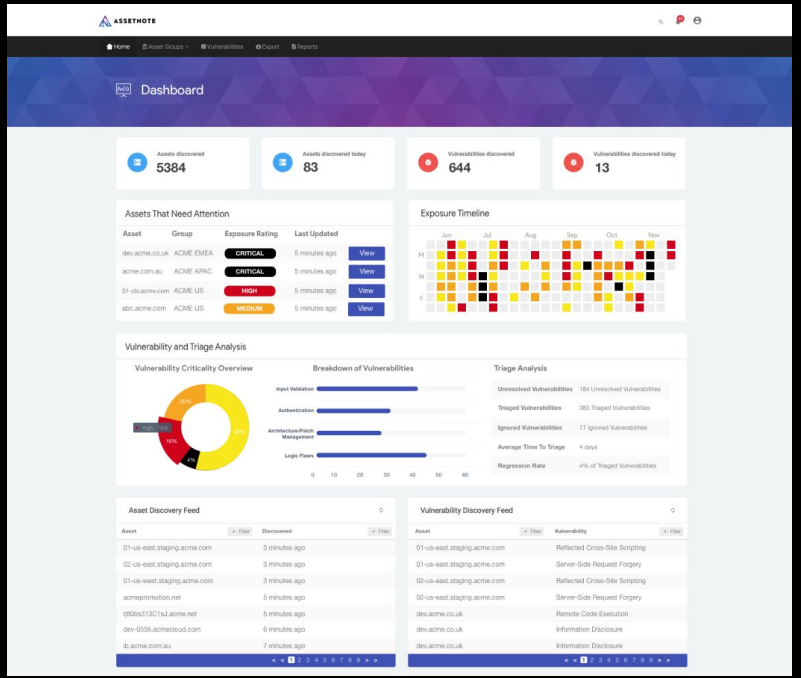
HOST: utildl1r3vdna.psl.labs[redacted],IP: 12.1[redacted],HTTP/S: INACTIVE,OPEN PORTS: NULL
HOST: apis.[redacted]IP: 2.1[redacted],HTTP/S: https://apis[redacted],OPEN PORTS: [80, 443, 3306, 5432]
HOST: sponsoreddata[redacted],IP: 2[redacted],HTTP/S: https://sponsoreddata[redacted],OPEN PORTS: [21, 22, 53, 80, 443]
HOST: utildl4r3vdna.psl.labs[redacted],IP: 12.[redacted],HTTP/S: INACTIVE,OPEN PORTS: NULL
HOST: ort-developer[redacted],IP: 23[redacted],HTTP/S: https://ort-developer[redacted],OPEN PORTS: [80, 443]
HOST: utildl3r3vdna.psl.labs[redacted],IP: 12.1[redacted],HTTP/S: INACTIVE,OPEN PORTS: NULL
HOST: utildl2r3vdna.psl.labs[redacted],IP: 12.1[redacted],HTTP/S: INACTIVE,OPEN PORTS: NULL

Frameworks (S-Tier)

Intrigue.io



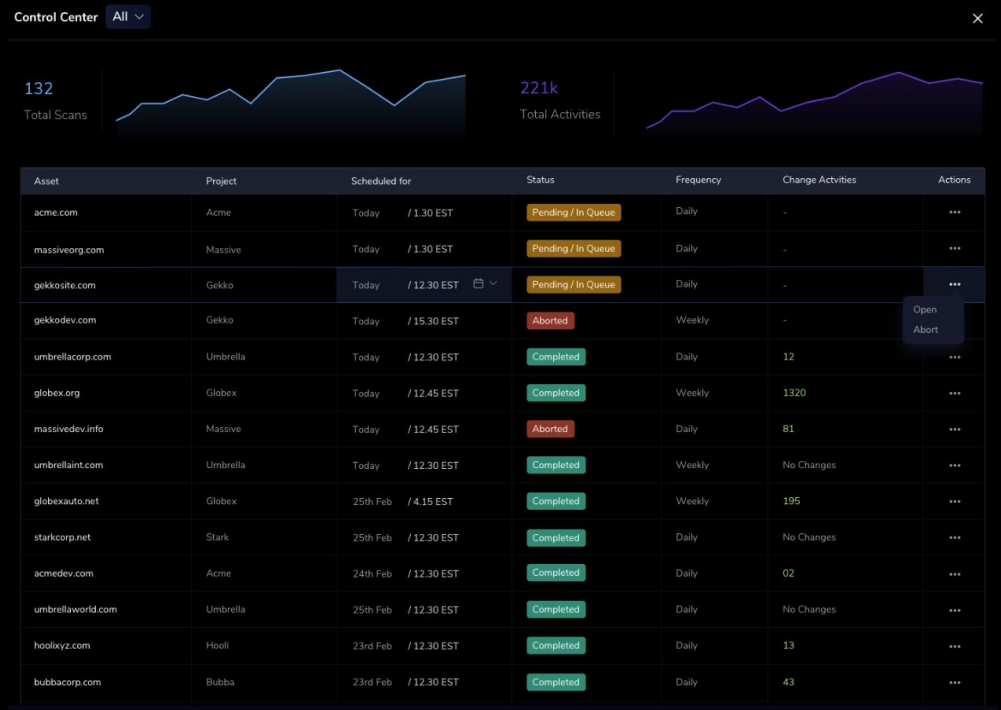
AssetNote



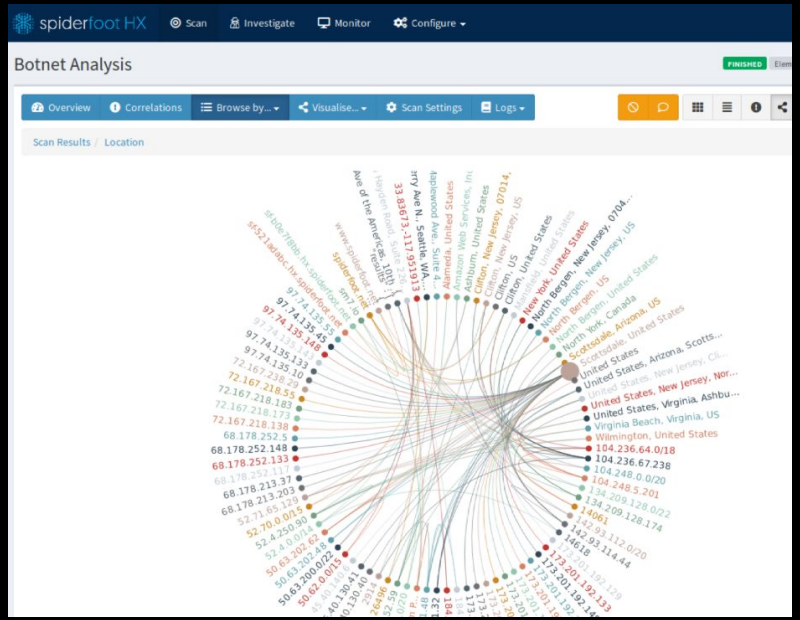
Frameworks (S-Tier)

Project Discovery Framework

(unreleased)

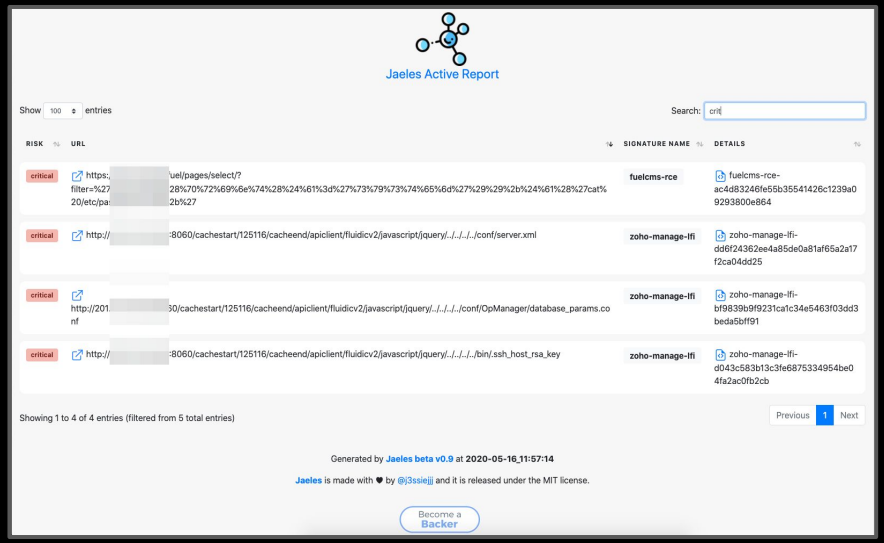


Spiderfoot



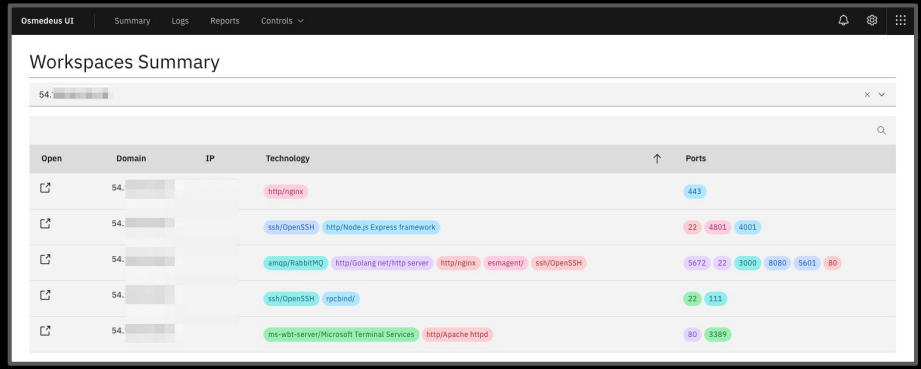
Frameworks (S-Tier)

Jaeles
(scanner)

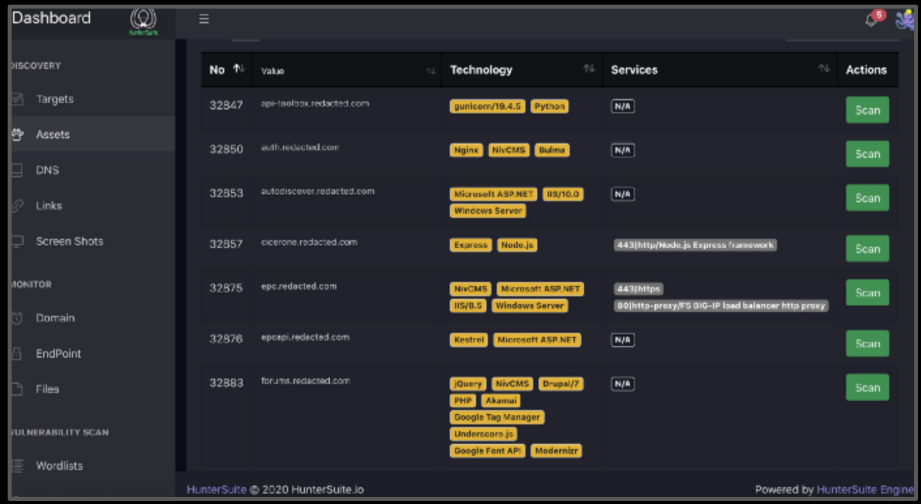


written by [j3ssiejjj](#) and team 

Osmedeus



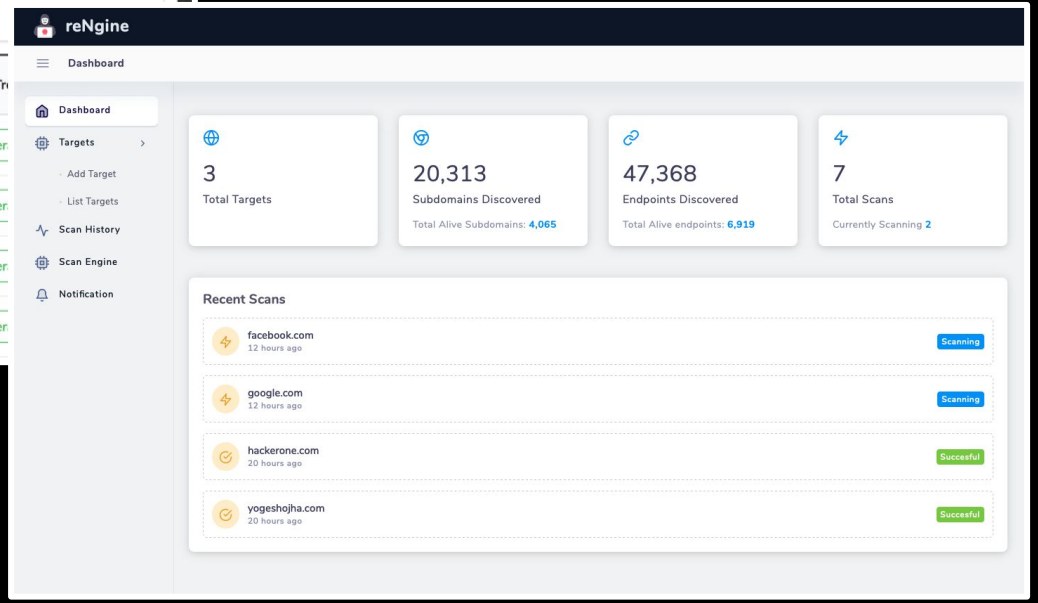
HunterSuite.io
(unreleased)



Frameworks (S-Tier)

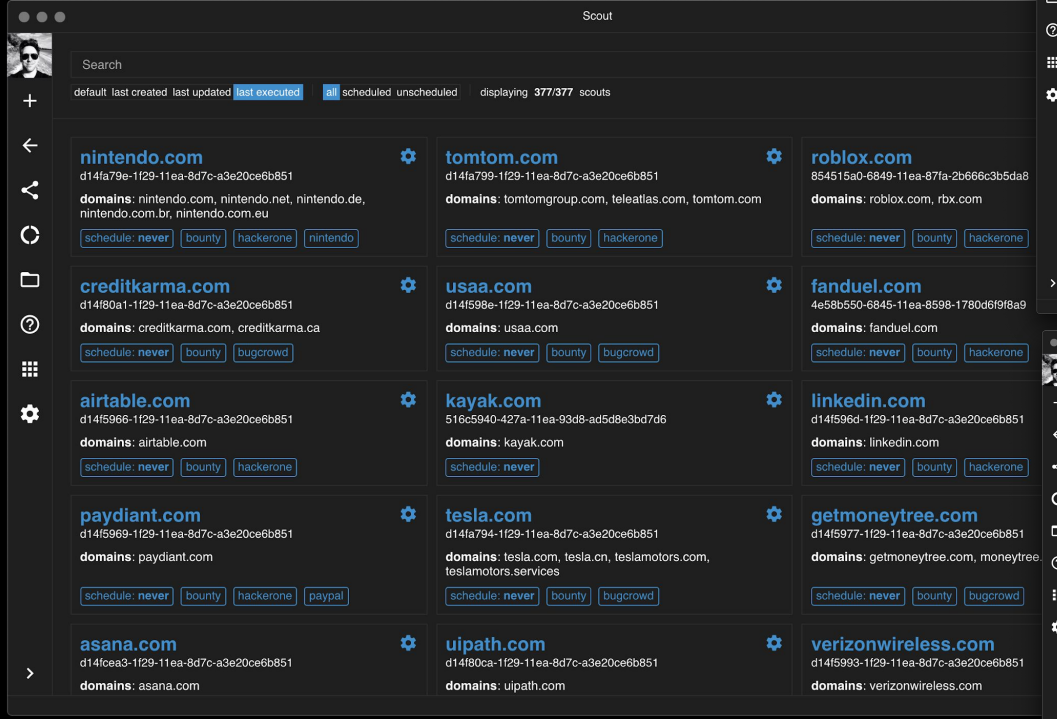
Bounty.offensiveai.com (paid) by @ghostlulz1337

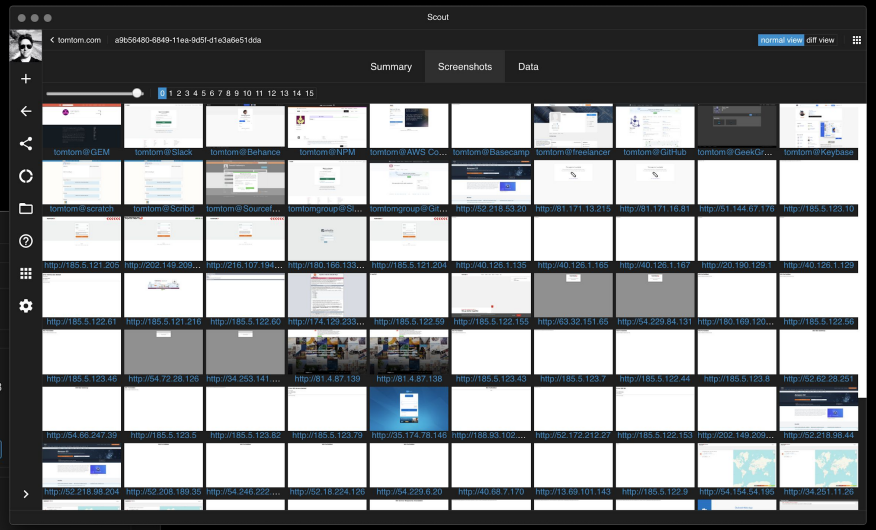
reNgine by yogeshojha

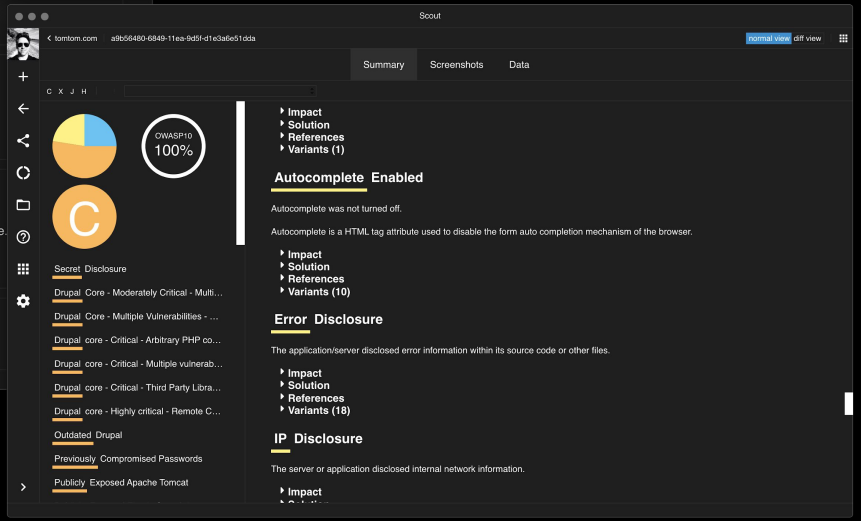


Frameworks (S-Tier)

Scout (paid) by the secapps.com team







Honorable Mention

Nuclei

1. CHOOSE TEMPLATES

```
root@b0x:~/templates/files# ls
adminer-sql.yml       codeception.yaml  go-expvar.yml         jkstatus.yml  put-rce.yml       server-status-localhost.yml  tomcat-status.yml
akamai-arls.yml       dot-env.yaml      go-prometheus.yml     lfi.yml       rails-config.yml  spring-env.yml
application-wadl.yml  git-core.yml      jboss-serverinfo.yml  phpinfo.yml   s3-xss.yml        ssrf-files.yml
```

2. RUN AGAINST LIST OF HOSTS

```
root@b0x:~/templates/files# ~/nuclei -t dot-env.yaml -l ~/targets.txt


                     __     _
   ____  __  _______/ /__  (_)
  / __ \/ / / / ___/ / _ \/ /
 / / / / /_/ / /__/ /  __/ /
/_/ /_/\__,_/\___/_/\___/_/   v1

                projectdiscovery.io

[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[INF] [dot-env] Loaded template Dot Env file (@Ice3man) [high]
[dot-env] http://vulnerable-site.io/.env
[dot-env] http://random-affected-site.net/.env
[dot-env] http://is-this-affected-too.com/.env
[dot-env] http://some-vulnerable-site.com/.env
[dot-env] http://so-many-hosts.com/.env
[dot-env] http://another-vulnerable-site.com/.env
```

The End