Discover via OSINT

Once we have established our goals for the project (e.g. compromise anything associated with our the "Lizard Blue" target organization), then we next need to map out the target as well as possible to ensure we are able to discover weakness in the targets Internet facing infrastructure and leverage those weaknesses to meet our goals.

Frequently, information systems with considerable attack surfaces will have subdomains pointing to them to help people find and interact with the services, hence discovery of an organization's domain names and subdomain names is critical before we are able to test the information systems that would be considered in-scope for the engagement.

Once we discover all of the subdomains, aka Fully Qualified Domain Names (FQDNs), associated with the target organization, we can then start to drill down further into this attack surface to meet the goals that we have previously established for the engagement.

We can also then test to see if any of these FQDNs are still pointed to on-demand IT services which are no longer being used by the organization and hence could be claimed by an attacker who wishes to leverage the trust of the target organization's brand name in association with attacking another information system. This failure to clean out old and no longer needed DNS entries which left pointing (aka dangling) to on-demand IT services (aka Cloud Services) is commonly referenced to as a "subdomain takeover" and has become a popular finding in recent years due to "Cloud Sprawl" or the poorly controlled expansion of an organization's expanse into on-demand IT services.

Discovering Root Domains

Frequently finding the root domain name for a target organization is as easy as using google to search for the company's name and looking at the results. While this may be helpful to get us started down the discovery path towards an entry point into the target organization, frequently we have to dig slightly deeper to find our way into the target. Really great ways to find root domain names include:

1. Searching CrunchBase.com for the company's name and looking at all of the other company's which have previously been acquired by the target organization. Then through further research, also determining what root domain names are associated with those acquisitions, which frequently may also be considered in-scope for the engagement.
2. Leveraging similarities in domain name whois information to find additional root domain names. "Domlink" is an excellent tool to automate this process, if you have a WHOXY.com API key. Otherwise, you can frequently leverage "Domain Dossier" to find unique selectors (e.g. email addresses) and then sometimes leverage search engines (e.g. Google) to find other domains associated with those selectors.
3. Finding IP ranges which belong to your target organization (e.g. ipv4info.com), and then leveraging information collected relating to TLS/SSL certificates (Censys.io) in use within that network range to discover other root domains used by the organization.

Subdomains

Once we determine the root domain names for a company and all of it's subsidiaries and/or acquisitions, we will then want to find all available targets associated with that entity. Most employees working at a company will want to associate a more easy to work with name with an on-demand IT service, than an IP address and/or a very long URL, hence they will frequently create sub-domains for each information system and/or service that will be used regularly.

Amass - Basic Usage

Amass is a powerful tool written in Go which is built to quickly discover subdomains via querying multiple open source resources on the Internet which contain data relating to subdomains. The basic usage of Amass uses the -d flag to find subdomains using reverse DNS lookup and name alterations:

Terminal

```
ubuntu@ip-10-0-1-215:~$ sudo su -
root@ip-10-0-1-215:~# cd /shared/
root@ip-10-0-1-215:/shared# cnoio_amass -d lizardblue.com

...

oasis.lizardblue.com
lizardblue.com
hash.lizardblue.com
ixhash.lizardblue.com
www.lizardblue.com
```

OSINT with Multiple Data Sources

Amass leverages many publicly accessible repositories of data to collect information about the target domain. These repositories enable amass to quickly support Open-Source Intelligence (OSINT) gathering. OSINT is when data is gathered from publicly available sources to be used to gain intelligence against a target. The work "open" frequently means overt in this context, as the information required to meet our goals is readily available from these public sources, we just need to go collect it.

Amass currently collects information from the following sources which archive websites which were previously hosted publicly on Internet:

Source	Type	Reference
Archive Today	ARCHIVE	http://archive.is
Open UK Arc	ARCHIVE	http://www.webarchive.org.uk/wayback/archive
Wayback Arc	ARCHIVE	http://web.archive.org/web
etc.		

Amass currently also collects information from the following sources by scraping websites for the relevant information:

Source	Type	Reference
Censys	SCRAPE	https://www.censys.io
Exalead	SCRAPE	http://www.exalead.com
FindSubDmns	SCRAPE	https://findsubdomains.com
etc.		

Amass also currently collects information from the sources which collect data about server certificates on the Internet:

Source	Type	Reference
CertSpotter	CERT	https://certspotter.com
crt.sh	CERT	https://crt.sh
Entrust	CERT	https://ctsearch.entrust.com
etc.		

If we want to see how Amass found the subdomains we can use the -v flag. We can also use the -ip argument to see what IP address the subdomains resolve to:

Terminal

```
root@ip-10-0-1-215:/shared# cnoio_amass -v -ip -d lizardblue.com

[HackerTargt] hash.lizardblue.com,54.192.30.107,54.192.30.85,54.192.30.115,54.192.30.113
[Forward DNS] lizardblue.com,52.219.104.203
[CertSpotter] ixhash.lizardblue.com,52.72.167.156,107.21.109.176,34.236.195.157,54.161.132.134,54.225.138.64,18.211.119.3
[Google]      test.lizardblue.com,18.223.82.226

Amass v2.3.2 Jeff Foley (@jeff_foley)
-------------------------------------------------------------------------------
4 names discovered - scrape: 2, dns: 1, cert: 1
-------------------------------------------------------------------------------
ASN: 16509 - AMAZON-02, US
52.219.104.0/24 1 Subdomain Name(s)
18.220.0.0/14 1 Subdomain Name(s)
54.192.28.0/22 4 Subdomain Name(s)
ASN: 14618 - AMAZON-AES, US
34.224.0.0/12 1 Subdomain Name(s)
54.160.0.0/14 1 Subdomain Name(s)
54.224.0.0/15 1 Subdomain Name(s)
18.208.0.0/13 1 Subdomain Name(s)
52.72.0.0/15 1 Subdomain Name(s)
107.21.64.0/18 1 Subdomain Name(s)
```

Other commonly used options with amass include "-brute" to brute force guess subdomains and "-min-for-recursive" to try to recursively find subdomains:

Terminal

```
root@ip-10-0-1-215:/shared# cnoio_amass -v -ip -brute -w /shared/lists/namelist.txt -min-for-recursive 3 -d lizardblue.com

[CertSpotter] hash.lizardblue.com,54.192.30.113,54.192.30.115,54.192.30.85,54.192.30.107
[Forward DNS] lizardblue.com,52.219.96.245
[CertSpotter] www.lizardblue.com,3.22.102.178,18.191.53.168,3.16.106.119
[CertSpotter] ixhash.lizardblue.com,3.214.248.113,34.232.187.71,18.211.0.65,52.201.190.253,3.222.45.192,34.200.64.3

Amass v2.3.2 Jeff Foley (@jeff_foley)
-------------------------------------------------------------------------------
4 names discovered - dns: 1, cert: 3
-------------------------------------------------------------------------------
ASN: 16509 - AMAZON-02, US
54.192.28.0/22 4 Subdomain Name(s)
52.219.96.0/24 1 Subdomain Name(s)
3.20.0.0/14 1 Subdomain Name(s)
18.191.0.0/16 1 Subdomain Name(s)
3.16.0.0/14 1 Subdomain Name(s)
ASN: 14618 - AMAZON-AES, US
3.208.0.0/12 2 Subdomain Name(s)
34.224.0.0/12 1 Subdomain Name(s)
18.208.0.0/13 1 Subdomain Name(s)
52.200.0.0/13 1 Subdomain Name(s)
34.192.0.0/12 1 Subdomain Name(s)
```

Chaining Tools via File Output

We can output the results to a text file with the -o flag. This is useful because we can save this information for later and feed this output into other tools/scripts.

We are going to use docker's "-v" switch to share the "/shared" directory between the host environment and the container's environment, hence anything we store in the "/shared" directory will still be accessible even after the docker container has finished running:

Terminal

```
root@ip-10-0-1-215:/shared# cnoio_amass -o /shared/results_subdomains_amass.txt -d lizardblue.com
ixhash.lizardblue.com
hash.lizardblue.com
www.lizardblue.com
test.lizardblue.com
```

We can now view the content of the text file with the cat command, even though the docker container using amass is no longer running.

Terminal

```
root@ip-10-0-1-215:/shared# cat /shared/results_subdomains_amass.txt
ixhash.lizardblue.com
hash.lizardblue.com
www.lizardblue.com
test.lizardblue.com
```

Recon via DNS Bruteforce

The brute force approach allows us to find subdomains not otherwise discoverable through other methods. It works by checking for subdomains using a wordlist. Although Amass has options for this, we feel like from our experiences that Gobuster is a faster and more reliable tool created specifically for this purpose.

BruteForce with gobuster

We can verify that gobuster is installed correctly like this:

Terminal

```
root@ip-10-0-1-215:/shared# cnoio_gobuster

...

Gobuster v1.4.1              OJ Reeves (@TheColonial)
=====================================================
2 errors occurred:

* [!] WordList (-w): Must be specified
* [!] Url/Domain (-u): Must be specified
```

We can check out the list that gobuster will use when trying to discover new subdomains:

Terminal

```
root@ip-10-0-1-215:/shared# head /shared/lists/subdomains-top1mil-5000.txt
www
mail
ftp
...

root@ip-10-0-1-215:/shared# tail /shared/lists/subdomains-top1mil-5000.txt
inc
xmas
tulip
...
```

We can now use gobuster like this to look for subdomains under lizardblue.com and save it to file called "results_subdomains_gobuster.txt":

Terminal

```
root@ip-10-0-1-215:/shared# cnoio_gobuster -t 100 -i -m dns -w /shared/lists/subdomains-top1mil-5000.txt -u lizardblue.com -o /shared/raw_results_subdomains_gobuster.txt

Gobuster v1.4.1              OJ Reeves (@TheColonial)
=====================================================
=====================================================
[+] Mode         : dns
[+] Url/Domain   : lizardblue.com
[+] Threads      : 100
[+] Wordlist     : /shared/gobuster_wordlist.txt
[+] Output file  : /shared/gobuster_found_subdomains.txt
=====================================================
Found: test.lizardblue.com [18.223.82.226]
Found: images.lizardblue.com [52.219.104.4]
Found: www.lizardblue.com [3.15.63.120, 3.22.104.145, 3.23.211.130]
Found: cdn.lizardblue.com [52.216.232.123]
Found: staff.lizardblue.com [52.219.100.176]
Found: cdn2.lizardblue.com [52.219.101.131]
Found: WWW.lizardblue.com [3.23.211.130, 3.15.63.120, 3.22.104.145]
Found: oasis.lizardblue.com [52.219.104.40]
Found: ixhash.lizardblue.com [34.236.195.157, 52.72.167.156, 54.161.132.134, 54.225.138.64, 107.21.109.176, 18.211.119.3]
=====================================================

root@ip-10-0-1-215:/shared# cat /shared/raw_results_subdomains_gobuster.txt
Found: test.lizardblue.com [18.223.82.226]
Found: images.lizardblue.com [52.219.104.4]
Found: www.lizardblue.com [3.15.63.120, 3.22.104.145, 3.23.211.130]
Found: cdn.lizardblue.com [52.216.232.123]
Found: staff.lizardblue.com [52.219.100.176]
Found: cdn2.lizardblue.com [52.219.101.131]
Found: WWW.lizardblue.com [3.23.211.130, 3.15.63.120, 3.22.104.145]
Found: oasis.lizardblue.com [52.219.104.40]
Found: ixhash.lizardblue.com [34.236.195.157, 52.72.167.156, 54.161.132.134, 54.225.138.64, 107.21.109.176, 18.211.119.3]
```

GoBuster Flag Breakdown

The flags we used, and what they mean:

Flags	Description
-u	Which domain to look for subdomains under.
-w	Path to the wordlist used for brute forcing.
-m	Which mode to use, either dir or dns (default: dir).
-o	Specify a file name to write the output to.
-i	Show all IP addresses for the result.
-t	Number of threads to run (default: 10).

Parsing the Output

Several commands are very useful when we want to quickly parse output from various tools:

Tool	Description	Common Usage	Usage Explained
cat	print file contents to standard output (e.g. the screen)	cat /shared/raw_results_subdomains_gobuster.txt	print the text file to the screen
cut	remove sections from each line of files	cut -d "[" -f 2	break the line up into fields when it finds the (-d) delimiter (-f) only output to standard output (e.g. the screen) this field #
tr	translate or delete characters	tr ',' '\n'	find a comma (,) and replace it with a new line (\n)
awk	pattern scanning and processing	awk '{$1=$1};1'	removes white spaces and tabs before and after each line
sort	sort lines of text files	sort -u	sort the lines and (-u) unique them
shuf	generate random permutations	shuf	randomizes the lines order

To get a list of just the IP addresses, we chain these tools together:

Terminal

```
root@ip-10-0-1-215:/shared# cat /shared/raw_results_subdomains_gobuster.txt | cut -d "[" -f 2 | cut -d "]" -f 1 | tr ',' '\n' | awk '{$1=$1};1' | sort -u > /shared/results_ips_gobuste

root@ip-10-0-1-215:/shared# cat /shared/results_ips_gobuster.txt
18.223.82.226
18.224.230.124
52.14.111.235
...

root@ip-10-0-1-215:/shared# cat /shared/results_ips_gobuster.txt | shuf
```

To get a list of just the sub-domains:

Terminal

```
root@ip-10-0-1-215:/shared# cat /shared/raw_results_subdomains_gobuster.txt | cut -d " " -f 2 | tr '[:upper:]' '[:lower:]' | sort -u > /shared/results_subdomains_gobuster.txt

root@ip-10-0-1-215:/shared# cat /shared/results_subdomains_gobuster.txt
cdn2.lizardblue.com
cdn.lizardblue.com
oasis.lizardblue.com
staff.lizardblue.com
www.lizardblue.com
```

Recon via Web Crawling

Another effective way to discover subdomains and s3 bucket names associated with a target is via spidering the company's known websites to find links to other previously unknown but related domain names.

Web spiders search though a website's Internet accessible web pages to discover links to other content and websites. We can use wget as a make shift webspider to find domains and then use nsloolup to quick see if any of those domain names point to an s3 bucket. Let's walk-through this workflow together.

We can use the wget command with the following options:

Description	Option	Value
Turn on recursive retrieving... The default maximum depth is 5.	-r	
Enable spanning across hosts when doing recursive retrieving.	--span-hosts	
Wget will behave as a Web spider... but it will not save the pages	--spider	

Set domains to be followed.	-D	lizardblue.com
Log all messages to logfile... normally reported to standard error.	-o	spider.log
	[URL]	http://lizardblue.com

Putting it all together we get something like:

Terminal
```
root@ip-10-0-1-215:/shared# cd /shared/spider/

root@ip-10-0-1-215:/shared/spider# wget -r --span-hosts --spider -D lizardblue.com -o /shared/spider/spider.log https://www.lizardblue.com

root@ip-10-0-1-215:/shared/spider# ls -alF /shared/spider/
total 16
drwxr-xr-x 3 root root 4096 Jul  3 22:32 ./
drwx------ 8 root root 4096 Jul  3 22:29 ../
drwxr-xr-x 2 root root 4096 Jul  3 22:32 lizardblue.com/
-rw-r--r-- 1 root root 3607 Jul  3 22:32 spider.log

root@ip-10-0-1-215:/shared/spider# head /shared/spider/spider.log
Spider mode enabled. Check if remote file exists.
--2019-10-14 22:53:43--  https://www.lizardblue.com/
Resolving www.lizardblue.com (www.lizardblue.com)... 18.224.155.224, 18.224.164.47, 13.59.144.144
Connecting to www.lizardblue.com (www.lizardblue.com)|18.224.155.224|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/html]
Remote file exists and could contain links to other resources -- retrieving.

--2019-10-14 22:53:44--  https://www.lizardblue.com/
Reusing existing connection to www.lizardblue.com:443.



We can see from this output that whenever wget finds a new link that it creates a line within the log file similar to the following:

--2019-10-14 22:53:43--  https://www.lizardblue.com/
```

We can isolate all of the lines with a unique series of characters leveraging the grep command within Linux and escaping special characters using a backslash:

Terminal
```
root@ip-10-0-1-215:/shared/spider# echo $(date '+%Y')
2019

root@ip-10-0-1-215:/shared/spider# grep "\-\-$(date '+%Y')" /shared/spider/spider.log
--2019-07-03 22:32:02--  http://lizardblue.com/
--2019-07-03 22:32:02--  http://lizardblue.com/
--2019-07-03 22:32:02--  http://lizardblue.com/robots.txt
--2019-07-03 22:32:02--  http://cdn2.lizardblue.com/robots.txt
--2019-07-03 22:32:02--  http://lizardblue.com/styles.css
--2019-07-03 22:32:02--  http://lizardblue.com/reviews
--2019-07-03 22:32:02--  http://lizardblue.com/reviews
--2019-07-03 22:32:03--  http://cdn2.lizardblue.com/pricing.pdf
--2019-07-03 22:32:03--  http://cdn2.lizardblue.com/gecko.jpg
FINISHED --2019-07-03 22:32:03--
```

We can get rid of unwanted lines within this output using grep's "-v" option:

Description	Option	Value
Invert the sense of matching, to select non-matching lines. (-v is specified by POSIX.)	-v	

For example, we can isolate all of the lines with a unique series of characters leveraging the grep command within Linux and escaping special characters using a backslash:

Terminal
```
root@ip-10-0-1-215:/shared/spider# grep "\-\-$(date '+%Y')" /shared/spider/spider.log | grep -v "FINISHED"
--2019-07-03 22:32:02--  http://lizardblue.com/
--2019-07-03 22:32:02--  http://lizardblue.com/
--2019-07-03 22:32:02--  http://lizardblue.com/robots.txt
--2019-07-03 22:32:02--  http://cdn2.lizardblue.com/robots.txt
--2019-07-03 22:32:02--  http://lizardblue.com/styles.css
--2019-07-03 22:32:02--  http://lizardblue.com/reviews
--2019-07-03 22:32:02--  http://lizardblue.com/reviews
--2019-07-03 22:32:03--  http://cdn2.lizardblue.com/pricing.pdf
--2019-07-03 22:32:03--  http://cdn2.lizardblue.com/gecko.jpg
```

In this output we no longer see the last line containing the word "FINISHED".

Furthermore, we are really only interested with the domain names at the moment, so we can leverage the "cut" command to easily extract just the domain names from this output using the following options:

Description	Option	Value
use DELIM instead of TAB for field delimiter	-d	'/'
select only these fields; also print any line that contains no delimiter character	-f	3

For example:

Terminal
```
root@ip-10-0-1-215:/shared/spider# grep "\-\-$(date '+%Y')" /shared/spider/spider.log | grep -v "FINISHED" | cut -d '/' -f 3
lizardblue.com
lizardblue.com
lizardblue.com
cdn2.lizardblue.com
lizardblue.com
lizardblue.com
lizardblue.com
cdn2.lizardblue.com
cdn2.lizardblue.com
```

In this output we now only see the domains which the webspider found.

From here, we really want to get rid of duplicates, so we can leverage the "sort" command to do so with the following switch:

Description	Option	Value
unique the result	-u	

For example:

Terminal
```
root@ip-10-0-1-215:/shared/spider# grep "\-\-$(date '+%Y')" /shared/spider/spider.log | grep -v "FINISHED" | cut -d '/' -f 3 | sort -u
cdn2.lizardblue.com
lizardblue.com
```

In this output we now only see the de-duplicated domains which the webspider found.

Now that we have a list of the domain names that the web spider was able to find, we can redirect this output to a file using the greater than character:

Terminal
```
root@ip-10-0-1-215:/shared/spider# grep "\-\-$(date '+%Y')" /shared/spider/spider.log | grep -v "FINISHED" | cut -d '/' -f 3 | sort -u > /shared/results_subdomains_spider.txt

root@ip-10-0-1-215:/shared/spider# cat /shared/results_subdomains_spider.txt
cdn2.lizardblue.com
lizardblue.com
```

Burp Suite for S3 Buckets

A common tool used by red teams to spider websites is "Burp Suite" and within the Application Store for Burp, you can install the "Cloud Storage Tester" application which will highlight anytime it finds a reference to a s3 bucket by creating a finding within burp. This plugin can also test the permissions of s3 buckets if you create a secret within your AWS account to be used in conjunction with the application.

Combining the Results

We can combine the output from all of your recon activity (e.g. amass, gobuster, spidering, etc...) into a single file for future reference:

Terminal
```
root@ip-10-0-1-215:/shared/spider# cd /shared

root@ip-10-0-1-215:/shared# cat /shared/results_subdomains_*.txt | sort -u >> /shared/all_subdomains.txt

root@ip-10-0-1-59:/shared# cat all_subdomains.txt
cdn2.lizardblue.com
cdn.lizardblue.com
ixhash.lizardblue.com
lizardblue.com
oasis.lizardblue.com
staff.lizardblue.com
www.lizardblue.com
```

Exercise

The Plan:

Core Ops:

- Collect a list of subdomains

-- Leverage amass to discover sub-domains for the "lizardblue.com" root domain name.

-- Leverage gobuster to discover additional sub-domains for the "lizardblue.com" root domain name.

-- Crawl the lizardblue.com website using your own make shift web spider.

-- Combine the results into a useful list of unique domain names for future use.

Above & Beyond:

- Research an organization's acquisitions using CrunchBase.com for root domain names

References

Check out the following references for more information:

- amass - https://github.com/caffix/amass
- gobuster - https://github.com/OJ/gobuster
- explainshell - https://explainshell.com/