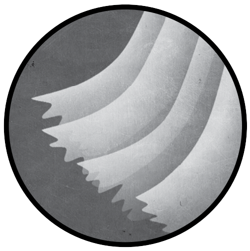
PART II

TRY HARDER

You're unlikely to discover something new without a lot of practice on old stuff.
Richard P. Feynman

4 HEALTHY STALKING



Our bouncing servers are silently humming in a datacenter somewhere in Europe. Our attacking infrastructure is eagerly awaiting our first order. Before we unleash the plethora of attack tools that routinely flood the infosec Twitter timeline, let's take a couple of minutes to understand how Gretsch Politico actually works. What is their business model? Which products and services do they provide? This kind of information will give us a direction to go in and help us narrow down attack targets. Drawing tangible goals may very well be our first challenge. Their main website (*www.gretschpolitico.com/*) does not exactly help: it is a boiling, bubbling soup of fuzzy marketing keywords that only make sense to the initiated. We'll start, then, with benign public-facing information.

Understanding Gretsch Politico

In an effort to better understand this industry, let's dig up every PowerPoint deck and PDF presentation that bears a reference to "Gretsch Politico" (GP). SlideShare (*https://www.slideshare.net/*) proves to be an invaluable ally in this quest. Many people simply forget to delete their presentations after a talk, or default them to "public access," giving us a plethora of information to begin our quest for understanding (see Figure 4-1).

Figure 4-1: Some Gretsch Politico slides

SlideShare is but one example of services hosting documents, so we next scour the web looking for resources uploaded to the most popular sharing platforms: Scribd, Google Drive, DocumentCloud, you name it. The following search terms will narrow down your results in most search engines:

```
# Lookup public Google Drive documents
site:docs.google.com "Gretsch politico"

# Search for documents on documentcloud.org
site:documentcloud.org "Gretsch politico"

# Documents uploaded to Scribd
site:scribd.com "gretschpolitico.com"

# Public power point presentations
intext:"Gretsch politico" filetype:pptx

# Public PDF documents
intext:"Gretsch politico" filetype:pdf

# Docx documents on GP's website
intext:"Gretsch politico" filetype:docx
```

Google may be your default search engine, but you may find you achieve better results in others, like Yandex, Baidu, Bing, and so on, since Google tends to observe copyright infringement and moderates its search output.

Another great source of information about a company's business is meta-search engines. Websites like Yippy and Biznar aggregate information from a variety of general and specialized search engines, giving a nice overview of the company's recent activity.

NOTE *The compilation of resources available at* https://osintframework.com/ *is a goldmine for any open source intelligence operator. You can easily lose yourself exploring and cross-referencing results between the hundreds of reconnaissance tools and apps listed there.*

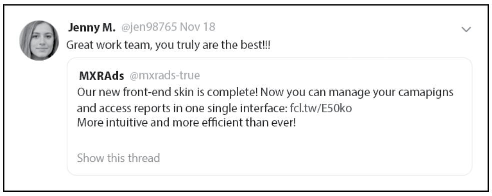
From my initial search, many interesting documents pop out, from campaign fund reports mentioning GP to marketing pitches for campaign directors. Manually skimming through this data makes it clear that GP's core service is building voter profiles based on multiple data inputs. These voter profiles are then studied and fed into an algorithm that decides which pitch is most suitable to lock in a voter.

Finding Hidden Relationships

GP's algorithms mash the data, that much is clear, but where does the data come from? To understand GP, we need to understand its closest partners. Whatever company or medium is delivering all this data must be working closely with GP. Multiple documents hint to the existence of at least two main channels:

- **Data brokers or data management platforms**: Companies that sell data gathered from telecom companies, credit card issuers, online stores, local businesses, and many more sources.
- **Research studies and surveys**: It seems that GP reaches out to the population somehow to send out questionnaires and collect opinions.

Although GP's main website barely mentions advertising as a way to reach the public, PDF documents abound with references to a particular advertising platform with tremendous reach, both on social and traditional media websites. We could not find a straight link to this advertising platform, but thanks to these selfsame social media websites they are so fond of, we dig out the retweet shown in Figure 4-2 from Jenny, VP of marketing at GP according to her Twitter profile.


Jenny M. @jen98765 Nov 18
Great work team, you truly are the best!!!

MXRAds @mxrads-true
Our new front-end skin is complete! Now you can manage your camapigns and access reports in one single interface: fcl.tw/E50ko
More intuitive and more efficient than ever!

Show this thread

Figure 4-2: A revealing GP retweet

The link in the tweet innocuously points to an online advertising agency: MXR Ads. They deliver ads on all kinds of websites, charge per thousand impressions (CPM), and go quietly about their business of increasing the internet's load time.

Short of this excited tweet by Jenny of GP, there is not a single visible link between the two companies; there's barely even a backlink on Google. So what's the connection? We quickly solve this mystery by consulting the legal records of the two companies on *https://opencorporates.com/*, a database of companies worldwide, and an excellent resource for digging out old companies' filings, shareholders lists, related entities, and so on. It turns out that MXR Ads and Gretsch Politico share most of the same directors and officers—hell, they even shared the same address a couple of years back.

This kind of intertwined connection can be very profitable for both companies: MXR Ads gathers raw data about people's engagement with a type of product or brand. They know, for example, that the person bearing the cookie 83bdfd57a5e likes guns and hunting. They transfer this raw data to Gretsch Politico, who analyzes it and groups it into a data segment of similar profiles labeled "people who like guns." GP can then design creatives and videos to convince the population labeled "people who like guns" that their right to gun ownership is threatened unless they vote for the right candidate. GP's client, who is running for office in some capacity, is pleased and starts dreaming about champagne bubble baths at the Capitol, while GP pushes these ads on every media platform with a functioning website. Of course, MXR Ads receives its share of creatives to distribute on its network as well, thus completing the self-feeding ouroboros of profit and desperation. Chilling.

From this close connection we can reasonably suspect that pwning either MXR Ads or GP could prove fatal to *both* companies. Their sharing of data implies some link or connection that we can exploit to bounce from one to the other. Our potential attack surface just expanded.

Now that we have a first, though very speculative, knowledge of the company's modus operandi, we can set out to answer some interesting questions:

- How precise are these data segments? Are they casting a large net targeting, say, all 18- to 50-year-olds, or can they drill down to a person's most intimate habits?
- Who are GP's clients? Not the pretty ponies they advertise on their slides, like health organizations trying to spread vaccines, but the ugly toads they bury in their databases.
- And finally, what do these creatives and ads look like? It might seem like a trivial question, but since they're supposedly customized to each target population, it is hard to have any level of transparency and accountability.

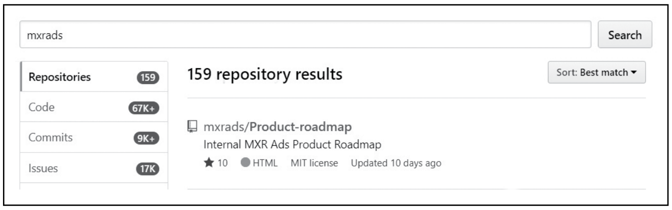
NOTE *Zeynep Tufekci has a great TED talk called "We're building a dystopia just to make people click on ads," about the dystopian reality encouraged by online ads.*

In the next few chapters we'll attempt to answer these questions. The agenda is pretty ambitious, so I hope you are as excited as I am to dive into this strange world of data harvesting and deceit.

Scouring Github

A recurrent leitmotif in almost every presentation of Gretsch Politico and MXR Ads' methodology is their investment in research and design and their proprietary machine learning algorithms. Such technology-oriented companies will likely have some source code published on public repositories for various purposes, such as minor contributions to the open source world used as bait to fish for talent, partial documentation of some API, code samples, and so on. We might just find some material that contains an overlooked password or sensitive link to their management platform. Fingers crossed!

Searching public repositories on GitHub is rather easy; you don't even need to register a free account. Simply proceed to look for keywords like "Gretsch Politico" and "MXR Ads." We search for MXR Ads' repository, shown in Figure 4-3.



159 repository results

mxrads/Product-roadmap
Internal MXR Ads Product Roadmap
★ 10 ● HTML MIT license Updated 10 days ago

Figure 4-3: The MXR Ads GitHub repository

A single company with 159 public repositories? That seems like a lot. After a cursory inspection, it's clear only half a dozen of these repos actually belong to either MXR Ads or one of their employees. The rest are simply forks (copied repositories) that happen to mention MXR Ads—for instance, in ad-blocking lists. These forked repositories provide little to no value, so we'll focus on those half a dozen original repos. Luckily, GitHub offers some patterns to weed out unwanted output. Using the two search prefixes `org:` and `repo:`, we can limit the scope of the results to the handful of accounts and repositories we decide are relevant.

We start looking for hardcoded secrets, like SQL passwords, AWS access keys, Google Cloud private keys, API tokens, and test accounts on the company's advertising platform. Basically, we want anything that might grant us our first beloved access.

We enter these queries in the GitHub search and see what we get:

```
# Sample of GitHub queries

org:mxrAds  password
org:mxrAds  aws_secret_access_key
org:mxrAds  aws_key
```

```
org:mxrAds  BEGIN RSA PRIVATE KEY
org:mxrAds  BEGIN OPENSSH PRIVATE KEY
org:mxrAds  secret_key
org:mxrAds  hooks.slack.com/services
org:mxrAds  sshpass -p
org:mxrAds  sq0csp
org:mxrAds  apps.googleusercontent.com
org:mxrAds  extension:pem key
```

The annoying limitation of GitHub's search API is that it filters out special characters. When we search for "aws_secret_access_key," GitHub will return any piece of code matching any of the four individual words (aws, secret, access, or key). This is probably the only time I sincerely miss regular expressions.

NOTE *The GitHub alternative Bitbucket does not provide a similar search bar. They even specifically instruct search engines to skip over URLs containing code changes (known as commits). Not to worry:* Yandex.ru *has the nasty habit of disregarding these rules and will gladly show you every master tree and commit history on Bitbucket public repos using something like* site:bitbucket.org inurl:master.

Keep in mind that this phase of the recon is not only about blindly grabbing dangling passwords; it's also about discovering URLs, API endpoints, and acquainting ourselves with the technological preferences of the two companies. Every team has some dogma about which framework to use and which language to work with. This information might later help us adjust our payloads.

Unfortunately, preliminary GitHub search queries did not return anything worthy, so we bring out the big guns and bypass GitHub limitations altogether. Since we're only targeting a few dozen repositories, we'll download the entire repositories to disk to unleash the full wrath of good ol' grep!

We'll start with the very interesting list of hundreds of regex (regular expression) patterns defined in `shhgit`, a tool specifically designed to look for secrets in GitHub, from regular passwords to API tokens (*https://github.com/eth0izzle/shhgit/*). The tool itself is also very interesting for defenders, as it flags sensitive data pushed to GitHub by listening for webhook events—a *webhook* is a call to a URL following a given event. In this case, GitHub sends a POST request to a predefined web page every time a regex matches a string in the code submitted.

We rework the list of patterns to make it grep friendly; you can find this list in *secret_regex_patterns.txt* at *https://www.hacklikeapornstar.com/secret_regex_patterns.txt*. Then we download all repos:

```
root@Point1:~/# while read p; do \
git clone www.github.com/MXRads/$p
done <list_repos.txt
```

And start the search party:

```
root@Point1:~/# curl -vs https://gist.github.com/HackLikeAPornstar/
ff2eabaa8e007850acc158ea3495e95f > regex_patterns.txt

root@Point1:~/# egrep -Ri -f regex_patterns.txt *
```

This quick-and-dirty command will search through each file in the downloaded repositories. However, since we are dealing with Git repositories, egrep will omit previous versions of the code that are compressed and hidden away in Git's internal file system structure (*.git* folder). These old file versions are of course the most valuable assets! Think about all the credentials pushed by mistake or hardcoded in the early phases of a project. The famous line "It's just a temporary fix" has never been more fatal than in a versioned repository.

The git command provides the necessary tools we'll use to walk down the commit memory lane: `git rev-list`, `git log`, `git revert`, and the most relevant to us, `git grep`. Unlike the regular grep, `git grep` expects a commit ID, which we provide using `git rev-list`. Chaining the two commands using xargs (extended arguments), we can retrieve all commit IDs (all changes ever made to the repo) and search each one for interesting patterns using `git grep`:

```
root@Point1:~/# git rev-list --all | xargs git grep "BEGIN [EC|RSA|DSA|OPENSSH] PRIVATE KEY"
```

We could also have automated this search using a bash loop or completely relied on a tool like GitLeaks (*https://github.com/zricethezav/gitleaks/*) or truffleHog (*https://github.com/dxa4481/truffleHog/*) that takes care of sifting through all commit files.

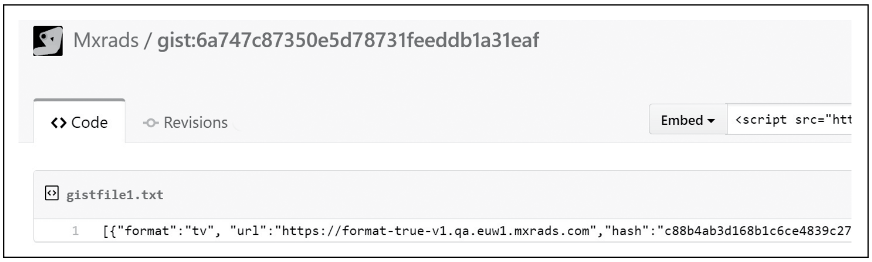
After a couple of hours of twisting that public source code in every fashion possible, one thing becomes clear: there seems to be no hardcoded credentials anywhere. Not even a fake dummy test or test account to boost our enthusiasm. Either MXR Ads and GP are good at concealment or we are just not that lucky. No matter, we'll move on!

One feature of GitHub that most people tend to overlook is the ability to share snippets of code on *gist.github.co*, a service also provided by *https://pastebin.com/*. These two websites, and others such as *codepen.io*, often contain pieces of code, database extracts, buckets, configuration files, and anything that developers want to exchange in a hurry. We'll scrape some results from these sites using some search engine commands:

```
# Documents on gist.github.com
site:gist.github.com "mxrads.com"

# Documents on pastebin
site:pastebin.com "mxrads.com"

# Documents on justepasteit
site:justpasteit.com "mxrads.com"
```

```
# Documents on pastefs
site:pastefs.com "mxrads.com"

# Documents on codepen
site:codepen.io "mxrads.com"
```

One search yields the result shown in Figure 4-4.



```
[{"format":"tv", "url":"https://format-true-v1.qa.euw1.mxrads.com","hash":"c88b4ab3d168b1c6ce4839c27
```

Figure 4-4: A snippet of an MXR Ads log file

This seems to be an extract of a log file just hanging in a public Gist, available for everyone to see. Isn't that just lovely? Sadly, no critical information is immediately available, but we do get these unique URLs:

- *format-true-v1.qa.euw1.mxrads.com*
- *dash-v3-beta.gretschpolitico.com*
- *www.surveysandstats.com/9df6c8db758b35fa0f1d73. . .*

We test these in a browser. The first link times out, and the second one redirects to a Google authentication page (see Figure 4-5).

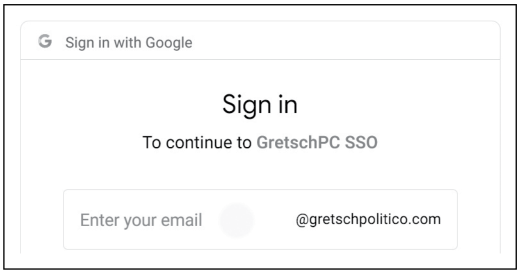


Figure 4-5: Gretsch Politico sign-in link found in the log file snippet

Gretsch Politico evidently subscribes to Google Workspace (formerly G Suite) apps to manage their corporate emails and likely their user directory and internal documents. We'll keep that in mind for later when we start scavenging for data.

The third URL, pointing to Figure 4-6, is promising.

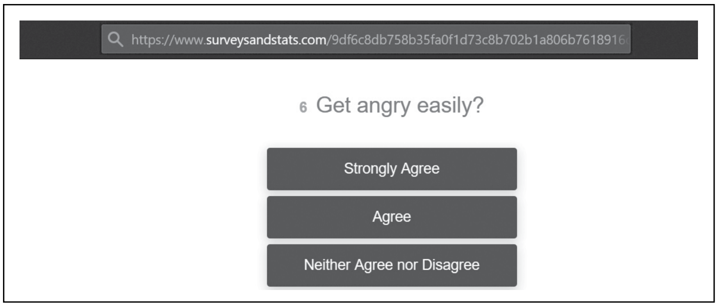

https://www.surveysandstats.com/9df6c8db758b35fa0f1d73c8b702b1a806b761891
6 Get angry easily?
Strongly Agree
Agree
Neither Agree nor Disagree

Figure 4-6: Link to an MXR Ad survey found in the log file snippet

This must be one of these surveys MXR Ads uses to gather seemingly harmless information about people. Attempting to pwn MXR Ads or Gretsch Politico through one of their pernicious forms is quite tempting, but we are still in the midst of our reconnaissance work, so let's just note this for a later attempt.

Pulling Web Domains

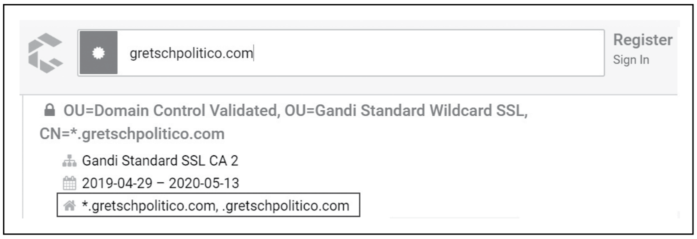
Passive reconnaissance did not yield us many entry points so far. I believe it's time we seriously started digging up all the domains and subdomains related to MXR Ads and Gretsch Politico. I am sure we can find so much more than the three measly websites on a forgotten Gist paste. Hopefully we'll land on a forlorn website with a sneaky vulnerability welcoming us inside.

We'll begin our search by first checking certificate logs for subdomains.

From Certificates

Censys (*https://censys.io/*) is a tool that routinely scans certificate logs to ingest all newly issued TLS certificates, and it's number one on any pentester's domain discovery tool list. Upon their issuance by a certificate authority, certificates are pushed to a central repository called a *certificate log*. This repository keeps a binary tree of all certificates, where each node is the hash of its child nodes, thus guaranteeing the integrity of the entire chain. It's roughly the same principle followed by the Bitcoin blockchain. In theory, all issued TLS certificates should be publicly published to detect domain spoofing, typo-squatting, homograph attacks, and other mischievous ways to deceive and redirect users.

We can search these certificate logs to eke out any new registrations matching certain criteria, like MXR Ads. The ugly side of this beautiful canvas is that all domains and subdomain names are openly accessible online. Secret applications with little security hiding behind obscure domains are therefore easily exposed. Tools like Censys and *Crt.sh* explore these certificate logs and help speed up subdomain enumeration by at least an order of magnitude—a cruel reminder that even the sweetest grapes can hide the most bitter seeds. In Figure 4-7 we use Censys to search for subdomains of *gretschpolitico.com.*



OU=Domain Control Validated, OU=Gandi Standard Wildcard SSL, CN=*.gretschpolitico.com
Gandi Standard SSL CA 2
2019-04-29 – 2020-05-13
*.gretschpolitico.com, .gretschpolitico.com

Figure 4-7: Looking for subdomains with Censys

So much for transparency. It seems that GP did not bother registering subdomain certificates and has instead opted for a wildcard certificate: a generic certificate that matches any subdomain. One certificate to rule them all. Whether this is a brilliant security move or pure laziness, the fact is, we're no further than the top domain. We try other top-level domains in Censys—*gretschpolitico.io*, *mxrads.tech*, *mxrads.com*, *gretschpolitico.news*, and so forth—but come up equally empty-handed. Our list of domains grew by a whopping big fat zero. . . but do not despair! We have other tricks up our collective sleeves.

NOTE *Of course, wildcard certificates present another security problem: they are a brazen single point of failure. Should we stumble upon the private key while roaming the company's network, we could intercept the communication flow of all applications using that same parent domain.*

By Harvesting the Internet

If certificates are not the way to gather subdomains, then maybe the internet can lend us a helping hand. Sublist3r is a great and easy-to-use tool that harvests subdomains from various sources: search engines, PassiveDNS, even VirusTotal. First, we fetch the tool from the official repository and install requirements:

```
root@Point1:~/# git clone https://github.com/aboul3la/Sublist3r
root@Point1:sub/# python -m pip install -r requirements.txt
```

Then we proceed to search subdomains, as shown in Listing 4-1.

```
root@Point1:~/# python sublist3r.py -d gretschpolitico.com
[-] Enumerating subdomains now for gretschpolitico.com
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
--snip--
[-] Searching now in ThreatCrowd..
[-] Searching now in PassiveDNS..

[-] Total Unique Subdomains Found: 12
```

```
dashboard.gretschpolitico.com
m.gretschpolitico.com
--snip--
```

Listing 4-1: Enumerating domains with sublist3r

We've found 12 subdomains, so that's encouraging. I bet we'd have even more luck with *mxrads.com*. They are, after all, a media company. However, it can get boring to use the same tools and methods repeatedly. For the *mrxads.com* domain, let's use a different tool to perform a classic brute-force attack using well-known subdomain keywords like *staging.mxrads.com*, *help .mxrads.com*, *dev.mxrads.com*, and so on. There are a few tools we can choose from for the job.

Amass (*https://github.com/OWASP/Amass/*) from the OWASP project is written in Golang and cleverly uses goroutines to parallelize the load of DNS queries. Whereas most other Python tools rely on the system's DNS resolver to retrieve domains by calling functions like `socket.gethostname()`, Amass crafts DNS queries from scratch and sends them to various DNS servers, thus avoiding the bottleneck caused by using the same local resolver. However, Amass is bloated with so many other colorful features, like visualizations and 3D graphs, that it may feel like wielding a ten-pound hammer to scratch an itch on your back. Tempting, but there are lighter alternatives.

A less mediatized yet very powerful tool that I highly recommend is Fernmelder (*https://github.com/stealth/fernmelder/*). It's written in C, is barely a few hundred lines of code, and is probably the most efficient DNS brute-forcer I have tried lately. Fernmelder takes two inputs: a list of candidate DNS names and the IPs of DNS resolvers to use. This is what we'll use.

First, we create our list of possible DNS names using some `awk` magic applied to a public subdomain wordlist. Daniel Miessler's SecLists is a good start for instance: *https://github.com/danielmiessler/SecLists/.*

```
root@Point1:~/# awk '{print $1".mxrads.com"}' top-10000.txt > sub_mxrads.txt
root@Point1:~/# head sub_mxrads.txt
test.mxrads.com
demo.mxrads.com
video.mxrads.com
--snip--
```

Listing 4-2: Creating a list of potential MXR Ads subdomains

This gives us a few thousand potential subdomain candidates to try. As for the second input, you can borrow the DNS resolvers found at the Fernmelder repo, shown in Listing 4-3.

```
root@Point1:~/# git clone https://github.com/stealth/fernmelder
root@Point1:~fern/# make

root@Point1:~fer/# cat sub_mxr.txt | ./fernmelder -4 -N 1.1.1.1 \
-N 8.8.8.8 \
-N 64.6.64.6 \
-N 77.88.8.8 \
-N 74.82.42.42 \
-N 1.0.0.1 \
-N 8.8.4.4 \
-N 9.9.9.10 \
-N 64.6.65.6 \
-N 77.88.8.1 \
-A
```

Listing 4-3: Resolving our subdomain candidates to see which are real

Be careful adding new resolvers, as some servers tend to play dirty and will return a default IP when resolving a nonexistent domain rather than the standard `NXDOMAIN` reply. The `-A` option at the end of the command hides any unsuccessful domain resolution attempts.

Results from Listing 4-3 start pouring in impressively fast. Of the thousand subdomains we tried resolving, a few dozen responded with valid IP addresses:

```
Subdomain           TTL Class Type    Rdata
electron.mxrads.net.   60  IN      A      18.189.47.103
cti.mxrads.net.        60  IN      A      18.189.39.101
maestro.mxrads.net.    42  IN      A      35.194.3.51
files.mxrads.net.      5   IN      A      205.251.246.98
staging3.mxrads.net.   60  IN      A      10.12.88.32
git.mxrads.net.        60  IN      A      54.241.52.191
errors.mxrads.net.     59  IN      A      54.241.134.189
jira.mxrads.net.       43  IN      A      54.232.12.89
--snip--
```

Watching these IP addresses roll on the screen is mesmerizing. Each entry is a door waiting to be subtly engineered or forcefully raided to grant us access. This is why this reconnaissance phase is so important: It affords us the luxury of choice, with over 100 domains belonging to both organizations!

NOTE *Check out AltDns, an interesting tool that leverages Markov chains to form predictable subdomain candidates:* https://github.com/infosec-au/altdns/.

Discovering the Web Infrastructure Used

The traditional approach to examining these sites would be to run WHOIS queries on these newly found domains, from which we can figure out the IP segment belonging to the company. With that we can scan for open ports in that range using Nmap or Masscan, hoping to land on an unauthenticated database or poorly protected Windows box. We try WHOIS queries on a few subdomains:

```
root@Point1:~/# whois 54.232.12.89
NetRange:       54.224.0.0 - 54.239.255.255
CIDR:           54.224.0.0/12
NetName:        AMAZON-2011L
OrgName:        Amazon Technologies Inc.
OrgId:          AT-88-Z
```

However, looking carefully at this list of IP addresses, we quickly realize that they have nothing to do with Gretsch Politico or MXR Ads. It turns out that most of the subdomains we collected are running on AWS Infrastructure. This is an important conclusion. Most internet resources on AWS, like load balancers, content distribution networks, S3 buckets, and so on, regularly rotate their IP addresses.

NOTE A content distribution network (CDN) *is a set of geographically distributed proxies that help decrease end-user latency and achieve high availability. They usually provide local caching, point users to the closest server, route packets through the fastest path, and other services. Cloudflare, Akamai, AWS CloudFront are some of the key players.*

That means that if we feed this list of IPs to Nmap and the port scan drags on longer than a couple of hours, the IP's addresses will have already been assigned to another customer and the results will no longer be relevant. Of course, companies can always attach a fixed IP to a server and directly expose their application, but that's like intentionally dropping an iron ball right on your little toe. Nobody is that masochistic.

Over the last decade, we hackers have gotten into the habit of only scanning IP addresses and skipping DNS resolution in order to gain a few seconds, but when dealing with a cloud provider, this could prove fatal. Instead, we should scan domain names; that way, the name resolution will be performed closer to the actual scan to guarantee its integrity.

That's what we will do next. We launch a fast Nmap scan on all the domain names we've gathered so far to look for open ports:

```
root@Point1:~/# nmap -F -sV -iL domains.txt -oA fast_results
```

We focus on the most common ports using `-F`, grab the component's version using `-sV`, and save the results in XML, RAW, and text formats with `-oA`. This scan may take a few minutes, so while waiting for it to finish, we turn our attention to the actual content of the hundreds of domains and websites we found belonging to MXR Ads and Gretsch Politico.

Resources

Leaked credentials happen more often than you think, as evidenced by this bug report of a researcher finding API tokens in a Starbucks-owned repo: *https://hackerone.com/reports/716292/*.

Read the quick tutorial at *https://juristr.com/blog/2013/04/git-explained/* if you're not familiar with Git's internals.