Crawl/Walk/Run

Introduction to DNSDB®

Part 2: Walk - Overcoming Cloudflare’s Reverse Proxy Service and finding related domains at scale
Learn from Farsight Security CEO Dr. Paul Vixie, as he teaches the fundamental investigative techniques and methodologies of how to use DNSDB to combat common cyberattacks. Starting with a single IOC (indicator of compromise), Dr. Vixie will guide us on how to pivot through domain infrastructure to build intelligence of associated malicious activity.

In this session of the Crawl/Walk/Run series, we'll get more specific about use cases for investigators. You'll find our abbreviated version with just the highlights below, or you can tune in for the full webinar here.

In this guide, we’ll be covering:

- **Part I: Overcome Cloudflare’s Reverse Proxy Service.**
  
  We’ll uncover a starting clue to further investigate. Using DNSDB as a time machine, we’ll investigate a domain hiding behind Cloudflare to uncover name servers and IP addresses of the actual backend site.

- **Part II: Find Related Domains At Scale.**
  
  Using what we found, we’ll replicate the process to widen our search and then leverage Cloudflare as a common factor to narrow our results.
Part I

Goal: Overcome Cloudflare’s Reverse Proxy Service

Background

Cloudflare’s job is to publish (not host) some of the Internet’s content. While a popular service, it protects and maintains unaccountability among users. Rather than seeing where a site is actually hosted, an investigator sees just the IP addresses of Cloudflare’s reverse proxy servers. Cloudflare protected domains will also typically use Cloudflare’s nameservers.

We want to pierce that veil and get to the actual backend servers, not just see Cloudflare’s front end servers.

Why Investigate?

Why should we devote effort into trying to find out where a bad guy lives?

1. Getting a bad guy’s site blacklisted (or a bad guy arrested) requires discovery of the actual bad guy backed site or sites.
   a. Cloudflare (along with Domain Whois privacy/proxy services and GDPR-redactions) currently helps bad guys avoid detection.

2. Unrelated sites get interleaved on shared Cloudflare reverse proxy servers:
   a. Makes it hard to find related bad sites (bad and good sites go onto the same IP).
   b. Makes it risky to block the bad sites by IP address (collateral damage will inevitably arise due to unrelated 3rd part sites getting blocked).

3. Cloudflare replaces a site’s real name servers with its own.
   a. This frustrates attempts to identify additional related bad domains sharing the same dedicated name servers.

Can we overcome that architecture?

The example we’re using today is a site that claims to be selling oxycodone, a schedule II narcotic, without requiring a prescription.

Let’s start with the easiest place—a search engine.
Percocet 10mg for Sale Online | Buy Percocet online no ...

Buy Percocet 10mg Online (Acetaminophen) No Prescription Required with Overnight delivery order now at Eir Pharmacy. PERCOCET 10MG (Acetaminophen) quantity. Add to Wishlist. SKU: percocet-10mg-for-sale Category: Pain Relief Tags: Buy Acetaminophen 10mg, Buy Oxycodone 10mg, Buy Percocet 10mg, Buy Percocets 10mg, Buy Percocets 30 Online, Buy ...
Checking Whois

$ whois eirmed360.com
Domain Name: EIRMED360.COM
Registry Domain ID: 2278976800_DOMAIN.COM-VSN
Registrar WHOIS Server: whois.ilovewww.com
Registrar URL: http://www.ilovewww.com
Updated Date: 2019-09-17T23:52:42Z
Creation Date: 2018-06-25T16:31:46Z
Registry Expiry Date: 2020-06-25T16:31:46Z
Registrar: Shinjiru Technology Sdn Bhd
Registrar IANA ID: 1741
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Name Server: OSWALD.NS.CLOUDFLARE.COM
Name Server: VERA.NS.CLOUDFLARE.COM
DNSSEC: unsigned

https://www.shinjiru.com/terms-conditions/ also says “The laws of Malaysia apply.”

While not entirely helpful, we do get more information on the real whois server, the registrar name, and the Cloudflare name servers.

Investigating the Shinjiru page, we can glean clues into the type of audience it caters to.

We can deduce that they are advertising to criminals.
Is the Site Leveraging Cloudflare?

When we check a generic recursive resolver for the eirmed360 domain, we see:

```bash
$ dig +trace eirmed360.com +nodnssec +nocrypto
[...]
;; Received 188 bytes from 192.12.94.30#53(e.gtld-servers.net) in 40 ms

www.eirmed360.com. 300 IN A 104.31.80.99 Cloudflare IPs
www.eirmed360.com. 300 IN A 104.31.81.99
;; Received 78 bytes from 173.245.59.218#53(oswald.ns.cloudflare.com) in 21 ms
```

Before we can do productive things with DNSDB, we need to overcome the Cloudflare reverse proxy service and figure out what the system's actually using for name servers (and then where the site is actually hosted). Then, we can leverage passive DNS.

Uncovering Backend Servers

Can we also uncover the REAL backend server? In this case, yes.

If we can find the right authoritative name server, that means we can find the real:

- SOA (State of Authority) record.
- Name servers.
- IP address servicing the site that's being reverse proxied by Cloudflare.

With this information, a law enforcement agency can get suitable paperwork to investigate.

But how can we find the RIGHT actual authoritative name server? Perhaps the domain owner was operationally careless before moving the site behind Cloudflare.

Using DNSDB Scout®

We can use DNSDB Scout® as a time machine.
We see a 3-day period where eirmed360 was hosted on ns1[dot]ipage[dot]com and ns2[dot]ipage[dot]com. These name servers may have more information for us.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>TTL</th>
<th>Domain</th>
<th>Type</th>
<th>RRS</th>
<th>RData</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-09-08</td>
<td>21:45</td>
<td>48</td>
<td>eirmed360.com</td>
<td>NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019-09-07</td>
<td>16:55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019-09-14</td>
<td>14:05</td>
<td>114</td>
<td>eirmed360.com</td>
<td>NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019-09-17</td>
<td>23:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019-02-07</td>
<td>06:40</td>
<td>12</td>
<td>eirmed360.com</td>
<td>NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019-02-27</td>
<td>17:02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2020-05-11</td>
<td>16:02</td>
<td>121</td>
<td>eirmed360.com</td>
<td>NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2020-05-12</td>
<td>07:51</td>
<td>8508</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2020-01-20</td>
<td>19:29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 to 10 of 10 Results

(available as a browser extension and web application)
Real Authoritative Name Server

Now, we go back to the dig utility. We're telling dig to not look where the DNS is telling it to look. Instead, we're asking to direct the query to a specific name server.

```
$ dig eirmed360.com soa @ns1.ipage.com +short
ns1.ipage.com. dnsadmin.ipage.com. 2019091608 10800 3600
604800 3600  the "real" SOA details

$ dig eirmed360.com ns @ns1.ipage.com +short
ns1.ipage.com.  the "real" name servers
ns2.ipage.com. 

$ dig eirmed360.com a @ns1.ipage.com +short
66.96.147.111  the "real" domain IP

$ dig www.eirmed360.com a @ns1.ipage.com +short
66.96.147.111  the web site's on that same IP
```

We find the real SOA, the real name servers, and the real domain IP. To ensure we've got the right IP address, we repeat the query with "www."

Double Check the Website Returned by the IP

Because any authoritative DNS server can answer any DNS query it receives, we need to make sure that this DNS server we've discovered is really pointing at the bad page.

- Edit your /etc/hosts file to point the domain of interest at the IP address you've just uncovered for it.
- Now, try visiting that website through Firefox.
  - Alert: Assume direct visits to a site are logged by the bad guys and noticed by the site's operator.
  - Caution: It can be dangerous to visit known problematic sites.
  - Reminder: When you're done testing, remove the bypass/override entry from your system.
  - Debugging: Some browsers may disregard in /etc/hosts.

Double Check the Website IP Using “curl --resolve”

Curl (https://curlhaxx.se/) will normally resolve the domain you’re trying to access conventionally, but you can override the normal resolution explicitly in the curl command.

```
$ curl https://www.eirmed360.com --insecure --resolve \n'www.eirmed360.com:443:66.96.147.11
```

Advantages
- No need to futz around with /etc/hosts.
- Reduced risk of contact with problematic content.

Disadvantages
- Page content is “raw” HTML/isn't rendered for graphical display.
- If you want to try to fly under the radar, you probably want to tweak the default user agent sent by curl.
Part I: Key Takeaways

We’ve found a starting clue and began to investigate it:

- Using DNSDB as a time machine, we’ve investigated a domain hiding behind Cloudflare.
- With identified name servers and IP address of the actual backend site, authorities can investigate the site, potentially identifying the perpetrator and prosecuting them (or other actions).

What else can we do?

Part II: Find Related Domains At Scale

Goal: Find related domains at scale

- Going from one IP address to the FQDNs using that IP is a very common pivot.
  - It doesn’t tend to work well for Cloudflare reverse proxy IP addresses. But if we get the real backend site’s IP, many times we can find related domains that share that backend IP.
- Check the encompassing IP address range (in case there are related bad sites on “nearby” IP addresses).
- Check all the netblocks announced by the ASN (autonomous system) for additional hosts of interest.
- Every step of that journey, the volume of passive DNS results multiplies. That’s why it can be key to use command line tools that can you script.

Command line techniques include:

- Learning how to simplify and condense the results you receive.
- Automating the identification of potentially relevant additional sites.

Dredging for Related Domains

![Image of Dredging for Related Domains](image-url)
What Else is on That IP?

Eirdmed360[dot]com was using 66.96.147.111—what else is on that IP?

1. Check DNSDB.
   a. If you’re new to DNSDB, get and install DNSDB from [https://github.com/dnsdb/dnsdbq](https://github.com/dnsdb/dnsdbq).
   b. DNSDBq will let us get up to a million results per query (and a total of four million results if we make additional queries using the new DNSDBq offset feature).
   c. See “Getting More Results from DNSDB Using the New -O (Offset) Option,” [https://www.farsightsecurity.com/blog/txt-record/offset-20190924/](https://www.farsightsecurity.com/blog/txt-record/offset-20190924/).

2. Get and install jq, a Swiss Army knife like command line tool for manipulating JSON output.
   a. See [https://stedolan.github.io/jq](https://stedolan.github.io/jq).


Finding Potentially Related Sites on a Single IP

Using 66.96.147.111

get up to a million results from the last 7 days, saving the results in JSON Lines format
$ dnsdbq -i 66.96.147.111 -10 -A7d -j > 66.96.147.111.jsonl
(note that -10 is "dash ell zero")

now let's get just the RRnames from that output, uniquifying them, and reversing them
$ jq -r ".rrname" < 66.96.147.111.jsonl | sort -u | reverse-domain-names | sort > 66.96.147.111-unique-rrnames-only.txt
$ wc -l 66.96.147.111-unique-rrnames-only.txt
28260

less 66.96.147.111-unique-rrnames-only.txt

[...]
com.eirmed360.medscoopvila.www
com.eirmed360.quest4chem
com.eirmed360.speedgreen-dispensary

We’ve added several instructions here.

- Uncap the limit (10k answers vs 1M answers).
- Leave the output in raw JSON.
- Limit to things seen in the last 7 days.

Now, let's get just the RRnames from that output, uniquifying them, and reversing them.

We find over 28000 unique names that have pointed at this IP address within the past 7 days. Having done enough of these investigations, we know that “quest4chem” is something worth noting.
Let's take “quest4chem” over to Google and see if we can find anything related.

The quest4chem site also appears to be offering Schedule II controlled substances for sale.

Checking Domain Whois

Whois for this domain looks very familiar:

```
$ whois quest4chem.com
Domain Name: QUEST4CHEM.COM
Registry Domain ID: 2348547581_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.ilovewww.com
Registrar URL: http://www.ilovewww.com
Updated Date: 2020-02-07T09:36:14Z
Creation Date: 2018-12-31T14:02:13Z
Registry Expiry Date: 2020-12-31T14:02:13Z
Registrar: Shinjiru Technology Sdn Bhd
Registrar IANA ID: 1741
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Name Server: OSWALD.NS.CLOUDFLARE.COM
Name Server: VERA.NS.CLOUDFLARE.COM
DNSSEC: unsigned
```

We see the same thin Whois server, registrar, and Cloudflare name servers.
Real Name Servers

What do the real name servers look like for this one?

- The public name servers for the domain are Cloudflare, as shown above.
- If we check DNSDB using dnsdbq, what former name servers do we see? Can we do the name server “time machine” trick for this domain too? Yes!

```bash
$ dnsdbq -r quest4chem.com/ns -s -k first
[...]
;; zone times: 2020-01-03 17:02:32 .. 2020-02-06 17:02:37
;; count: 35; bailiwick: com.
[...]
```

Forcing dig to use the real authoritative name server, we found:

```bash
$ dig eirmed360.com soa @ns1.ipage.com +short
ns1.ipage.com. dnsadmin.ipage.com. 2019091608 10800 3600 604800 3600  □ the "real" SOA details

$ dig eirmed360.com ns @ns1.ipage.com +short
ns1.ipage.com. □ the "real" name servers
ns2.ipage.com. □

$ dig eirmed360.com a @ns1.ipage.com +short
66.96.147.111 □ the "real" domain IP

$ dig www.eirmed360.com a @ns1.ipage.com +short
66.96.147.111 □ the web site's on that same IP
```

Again, be sure to validate that the site that’s really on this IP is the same as the site that’s being offered via Cloudflare.

Finding Additional Patterns

What if the bad guy doesn’t conveniently provide “easy clues”?

- Sites can have characteristic names so you can use grep (or ripgrep) on the results for commonly seen name substrings.
  - Avoid assuming any site you find that way is good or bad until you’ve carefully checked it out.
    ```bash
    $ rg -N pharm 66.96.147.111-unique-rrnames-only.txt
    alnoor-pharma.com
eastwindpharm.com
infinitypharm-eg.net
    [etc]
    ```
- You can also look for patterns in the additional domains you find.
  - Example: Our identified sites had the pattern of having had both cloudflare.com and ipage.com name servers.
Domains That Have Both ipage.com NSs and CF NSs

1. Find the name servers for each delegation point discovered in a prefix of interest (this might be tens of thousands of domains, so confirm you’ve got enough DNSDB queries).

```
[...]
dnsdbp -r abcdrugcard.com/ns -A7d -j &gt;&gt; 66.96.147.111-nameservers.json

dnsdbp -r abcdsallthebestcds.com/ns -A7d -j &gt;&gt; 66.96.147.111-nameservers.json
[...]
```

2. Use ripgrep to extract just the domains that used ipage.com name servers.

```
rg -N ipage.com 66.96.147.111-nameservers.json | jq -r '.rrname' | sort &gt; ipage-hits.txt
```

3. Use ripgrep to extract just the domains that used cloudflare.com name servers.

```
rg -N cloudflare.com 66.96.147.111-nameservers.json | jq -r '.rrname' | sort &gt; cf-hits.txt
```

4. Find the domains that have been seen using both of those sort of name servers...

```
comm -12 ipage-hits.txt cf-hits.txt | sort -u
```

Sample Sites Using Both ipage.com and Cloudflare.com Name Servers

```
$ comm -12 ipage-hits.txt cf-hits.txt | sort -u
a34arc.com
algomhorianow.com.
alkhayyat-ice.com
artbyjunaid.com
artfairashop.com
automeridianltd.com.
bhagyshreeent.com.
[etc]
```

- Having found those sites, we can then check Whois to see if any of these domains were also registered via Shinijiru.
- If there are any sites that still are in-scope, you could then check to see if the site is “good” or “bad” (or has content with the same theme as the site we started with).

Going Wider

What about the netblock around 66.96.147.111?

```
$ whois 66.96.147.111
[...]
NetRange: 66.96.128.0 - 66.96.191.255
CIDR: 66.96.128.0/18 16,384 IPs, each of which may obviously host many FQDNS
[...]
OrgName: The Endurance International Group, Inc.
OrgId: EIG-12
Address: 10 Corporate Drive
Address: Suite 300
City: Burlington
StateProv: MA
PostalCode: 01803
[...]
OrgAbusePhone: +1-877-659-6181
OrgAbuseEmail: eig-abuse@endurance.com
[...]
```
We find that the IP address we’re looking at is part of “/18,” which indicates there are 16,384 IPs in that block. This is likely not a criminal organization. They may have a mixed bag of customers.

Next, we want to ask about the whole “/18” group. We had to run the query four times (each query is capped at 1 million results).

```bash
$ dnsdbq -i 66.96.128.0/18 -10 -A7d -j -S -k count > 66.96.128.0-0.jsonl
$ dnsdbq -i 66.96.128.0/18 -10 -A7d -j -S -k count -O 1000000 > 66.96.128.0-1.jsonl
$ dnsdbq -i 66.96.128.0/18 -10 -A7d -j -S -k count -O 2000000 > 66.96.128.0-2.jsonl
$ dnsdbq -i 66.96.128.0/18 -10 -A7d -j -S -k count -O 3000000 > 66.96.128.0-3.jsonl

$ cat 66.96.128.0-[0123].jsonl | jq -r ".rrname" | sort -u > 66.96.128.0-unique-rrnames-only.txt
$ wc -l 66.96.128.0-unique-rrnames-only.txt
3456628 66.96.128.0-unique-rrnames-only.txt

That's a LOT of unique RRnames!

That’s a lot of names for only ~16,000 IP addresses. This doesn’t look good for Endurance.

Narrowing In

- Reduce the FQDNs to just unique delegation points using 2nd-level-dom-large script.

```perl
#!/usr/bin/perl
use strict;
use warnings;
use IO::Socket::SSL::PublicSuffix;

my $pslfile = '/usr/local/share/public_suffix_list.dat';
my $ps = IO::Socket::SSL::PublicSuffix->from_file($pslfile);

while (my $line = <STDIN>) {
    chomp($line);
    my $root_domain = $ps->public_suffix($line,1);
    printf( "%s\n", $root_domain );
}
```

```bash
$ 2nd-level-dom-large < 66.96.128.0-unique-rrnames-only.txt | sort -u > 66.96.128.0-unique-delegation-points.txt
$ wc -l 66.96.128.0-unique-delegation-points.txt
1212816 66.96.128.0-unique-delegation-points.txt  that helped!
```
Now just keep the domains rooted in .com (since this guy seems to like .com). That takes us down to 817001 domains—we're now down to 23.6% of the original 3,456,628.

Now query DNSDB for just the NS records for each of the remaining delegation points (NOTE: this is obviously a significant number of queries for this example!).

Look for domains that have previously had NS records both in ipage.com and cloudflare.com (as previously described for the single IP case).

Going REALLY Wide

Most providers have multiple prefixes that they use, and a bad guy may have some hosts on IPs from one netblock, and other hosts on other IPs from other netblocks.

So, what to do?

Answer:
  a. Brute force it—find ALL the prefixes the ASN uses*.
  b. Check DNSDB for each of those prefixes.
  c. Combine those results, then proceed as previously shown.

*How to find what ASN announces a given IP:

```bash
#!/bin/sh

origip=`echo $1`
revip=`echo $1 | sed 's/([0-9]*).([0-9]*).([0-9]*).([0-9]*)/4.3.2.1/'
listing=`host -w -t txt ${revip}.asn.routeviews.org 2>/dev/null | tail -1`
listing2=`echo $listing | awk '{print $4}' | sed 's/"//g'`
echo "${listing2} ${origip}"
```

$ ip2asn 66.96.147.111
29873 66.96.147.111
$ whois as29873

ASNNumber: 29873
ASName: BIZLAND-SD
OrgName: The Endurance International Group, Inc.
OrgId: EIG-12
Address: 10 Corporate Drive
Address: Suite 300
City: Burlington
StateProv: MA
PostalCode: 01803

We can then obtain the full list of 55 prefixes announced by AS29873 by checking bgp.he.net for that AS number.
Build a script to query DNSDB for each prefix

```bash
dnsdbq -i 143.95.160.0/23 -A7d -10 -j >> as29873.txt
dnsdbq -i 206.125.208.0/20 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.224.0/24 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.226.0/24 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.228.0/24 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.230.0/23 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.232.0/21 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.240.0/21 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.248.0/24 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.250.0/23 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.252.0/24 -A7d -10 -j >> as29873.txt
dnsdbq -i 207.148.254.0/23 -A7d -10 -j >> as29873.txt
dnsdbq -i 209.34.160.0/22 -A7d -10 -j >> as29873.txt
```

Part II: Key Takeaways

We’ve seen how to go from one bad site to another related site on the same IP. We’ve learned how to:

- Expand the radius of your search (include searching entire encompassing netblock).
- Find all prefixes a given operator may be announcing from a specific autonomous system (each of which can be searched in DNSDB).
- Reduce FQDNs to delegation points.
- Turn the use of Cloudflare from an anonymizing technique to a “finger point”/“attention attracting” factor.
About Farsight Security, Inc.

Farsight Security®, Inc. is the world’s largest provider of historical and realtime DNS intelligence solutions. We enable security teams to qualify, enrich and correlate all sources of threat data and ultimately save time when it is most critical - during an attack or investigation. Our solutions provide enterprise, government and security industry personnel and platforms with unmatched global visibility, context and response. Farsight Security is headquartered in San Mateo, California, USA.

Learn more about how we can empower your threat platform and security team with Farsight Security passive DNS solutions at www.farsightsecurity.com
or follow us on Twitter: @FarsightSecInc

+1-650-489-7919  Farsight Security®, Inc.177 Bovet Rd Ste 180 San Mateo, CA 94402 USA
info@farsightsecurity.com  www.farsightsecurity.com