How I Identified 93k Domain-Frontable CloudFront Domains

FEBRUARY 23, 2018

A Quick Intro to CloudFront for Domain Fronting

If you haven't heard of domain fronting before, head over to Optiv to read a great introduction by Chris Patten and Tom Steele. Also worth checking out Vincent Yiu's post for an intro to discovering these domains.

CloudFront is one of the easiest CDNs to use for domain fronting, so we are going to focus on it here. These tools and techniques could easily be applied to other CDNs as well.

When using domain fronting in a C2 channel, it is best to use a popular, reputable domain to maintain the most legitimacy. There are some such CloudFront domains mentioned in Vincent Yiu's post (cdn.az.gov, media.tumblr.com, images.instagram.com, cdn.zendesk.com and cdn.atlassian.com), and it's relatively easy to randomly browse around and look for CloudFront references in server responses to discover more.

Before I dive in to discovering more domains, let's take a look at how we can confirm that a domain can be used for domain fronting. We are going to make a request for a resource on CloudFront, then change the target domain while leaving the host header the same. If the tested target domain can be used for domain fronting, we will receive the resource from the domain specified in the host header. We could host our own file on an EC2 server and then setup CloudFront, but let's just borrow a random file on Amazon's site for easy testing.

https://images-na.ssl-images-amazon.com/images/I/01rgQ3jqo7L.css
contains the single string:
#octane-aui-example-id{color:red}

Now let's use curl to specify a host header of "images-na.ssl-images-amazon.com" while changing the target domain. We can test with cdn.atlassian.com.

```
curl -s -H "Host: images-na.ssl-images-amazon.com" -H "Connection: close"
"https://cdn.atlassian.com/images/I/01rgQ3jqo7L.css"
```

We still get the same content back! (You can also drop the host header to prove the point that 01rgQ3jqo7L.css definitely isn't on Atlassian's site—you'll get a 404 instead.)

So that's our definitive test for confirming that a domain can be used for domain fronting via CloudFront. We could use this curl command on a long list of domains and see which are frontable, but that will be relatively slow. Especially if we want to check a big list—like the Alexa top million domains...

Method #1

Instead of large scans, we can use some things we know about CloudFront and domain fronting to come up with a way to find frontable domains offline.

1. CloudFront domains are going to ultimately resolve to a CloudFront edge server IP address.
2. CloudFront edge server IP addresses.
3. We only want to front through HTTPS (otherwise the all-important host header will be visible). Most CloudFront implementations are going to support HTTPS anyways.

Now all we need is a list of domains with IPs that support HTTPS connections... Thankfully censys.io has us covered! Here is a list of domains in the Alexa top million that support TLS on port 443 (I used the alexa-results.csv.lz4 file). CloudFront will always support TLS so that works.

Now we just need to compare the IP of each domain with the CloudFront edge server ranges. I wrote a quick Python script to do just that. It's not the most optimized thing, so it took about 5 minutes to run for me.

Not bad, I got 4540 domains!

Note: These aren't guaranteed to all work for domain fronting, and if you read on, I get to confirming which ones work at the end. (Spoiler, it's more than 99%...)

Method #2

Okay, so the top million gives you a lot of solid domain name choices. But if we could have more... We're going to get more!

This time we're going to key off of a different indicator of CloudFront: CNAME records! When you set up a domain to use CloudFront, you need to add a CNAME record to redirect DNS queries to CloudFront (apparently unless you're using Amazon's Route 53 DNS, but we'll ignore this).

We're going to use a monster of a data set to find all of these—Rapid7's Forward DNS which "contains the responses to DNS requests for all forward DNS names known by Rapid7's Project Sonar". We'll only need the 'A' record version here, which is about 14 GB compressed. And 137 GB uncompressed... Fun!

Parsing is easy, but well, it takes a sec to grep through 1.47 billion lines.

```
grep -E "\"cname\",\"value\":\".*cloudfront.net\"}$" 2018-02-16-1518768001-fdns_a.json | cut -d'"' -f8 >> CFdomains.txt
```

In short: if it's a CNAME record that ends in cloudfront.net, grab the domain name.

No doubt this would be much faster in some other language, but I don't need to run it everyday, so ¯\_(ツ)_/¯

This method returned 94256 domains!

Confirming Domain Fronting

So far, everything I've done has been "offline" (if you ignore downloading data sets). To make sure the data I've collected is valid, it'd be nice to verify that these domains can indeed be used for domain fronting. I put the curl command from above into a loop to run through all of the collected domains and check them.

```
cat suspected.txt | while read -r line; do curl -s -k -m 2 --connect-timeout 1 -H "Host: images-na.ssl-images-amazon.com" -H "Connection: close" https://"$line"/images/I/01rgQ3jqo7L.css | grep -q "#octane-aui-example-id{color:red}" && echo $line >> confirmed.txt; done
```

I combined the two lists I gathered, removed duplicates, and then ran them through this loop. Let's look at the numbers separately first though.

From the first set of 4540 domains identified from the Alexa top million, 4508 proved to work for domain fronting. That's over 99%!

From the second set of 94256 domains identified from CNAME records in Rapid7's scans, 88994 worked for domain fronting. That's 94.4%.

Clearly the first method had a significantly higher accuracy than the second. This makes sense. Popular (top million) websites are only going to resolve to a CloudFront IP address if they are using CloudFront, otherwise that website isn't going to work. The less used websites of the second list are much more likely to have out of date DNS records (CNAME records that point to dead CloudFront distributions).

Conclusions

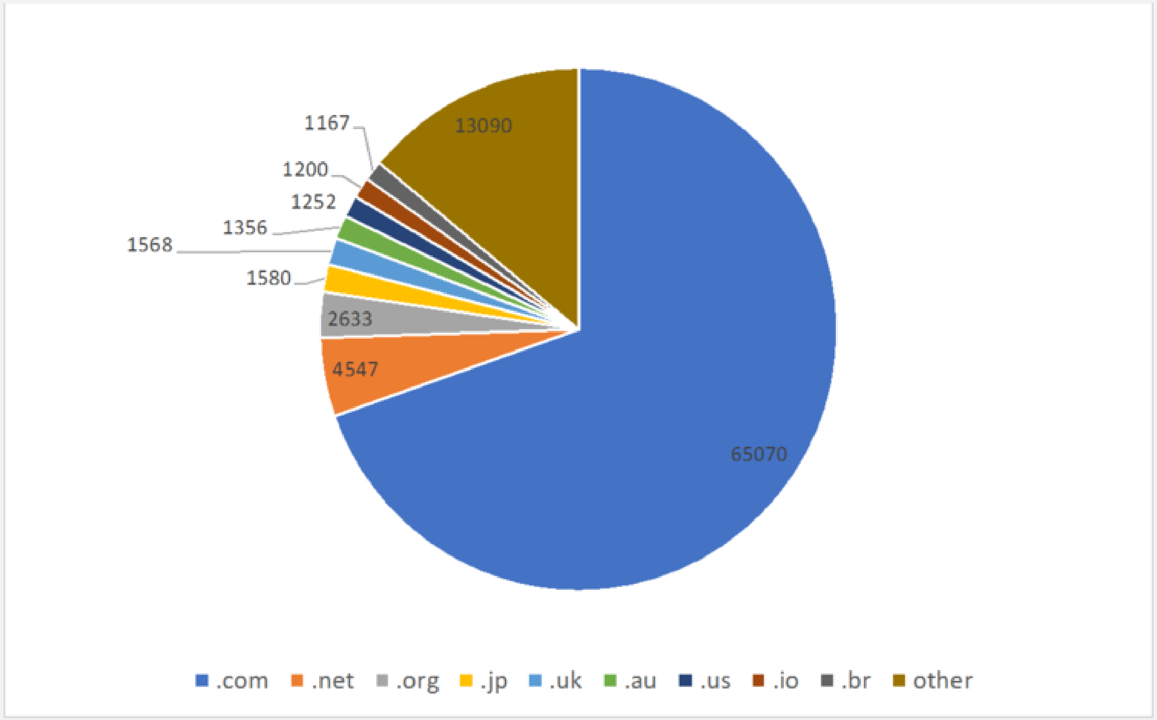
The final results!

I identified 93463 domains that work for domain fronting through CloudFront.

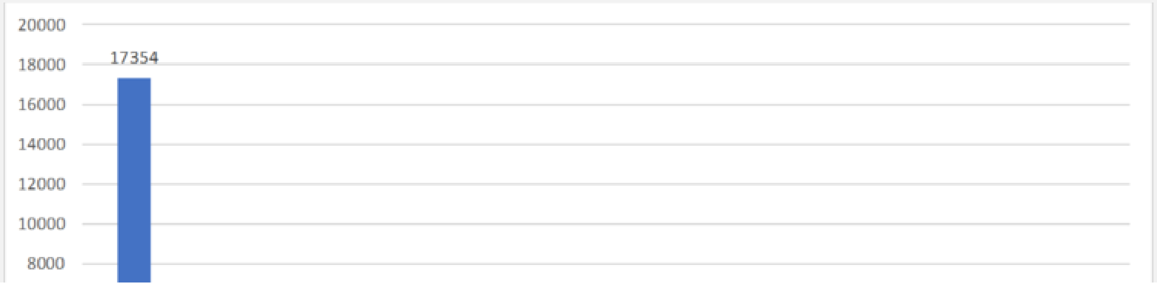
The full list can be found here: https://github.com/peewpw/DomainFrontDiscover/blob/master/ConfirmedFrontableDomains.txt

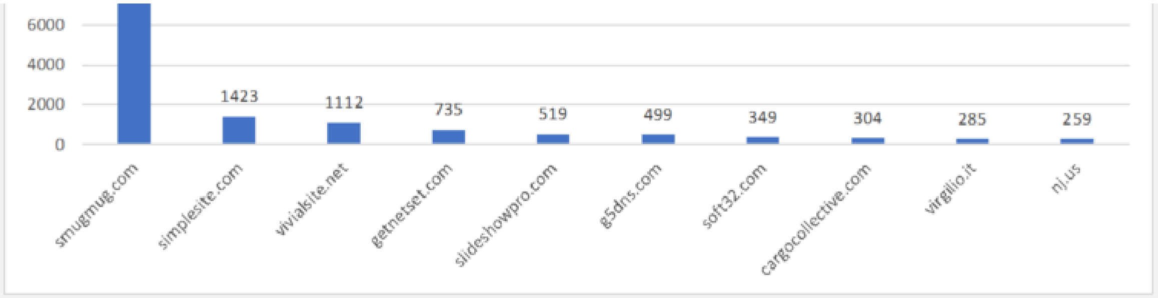
Let's take a stab at visualizing the domains we have here:

TLDS

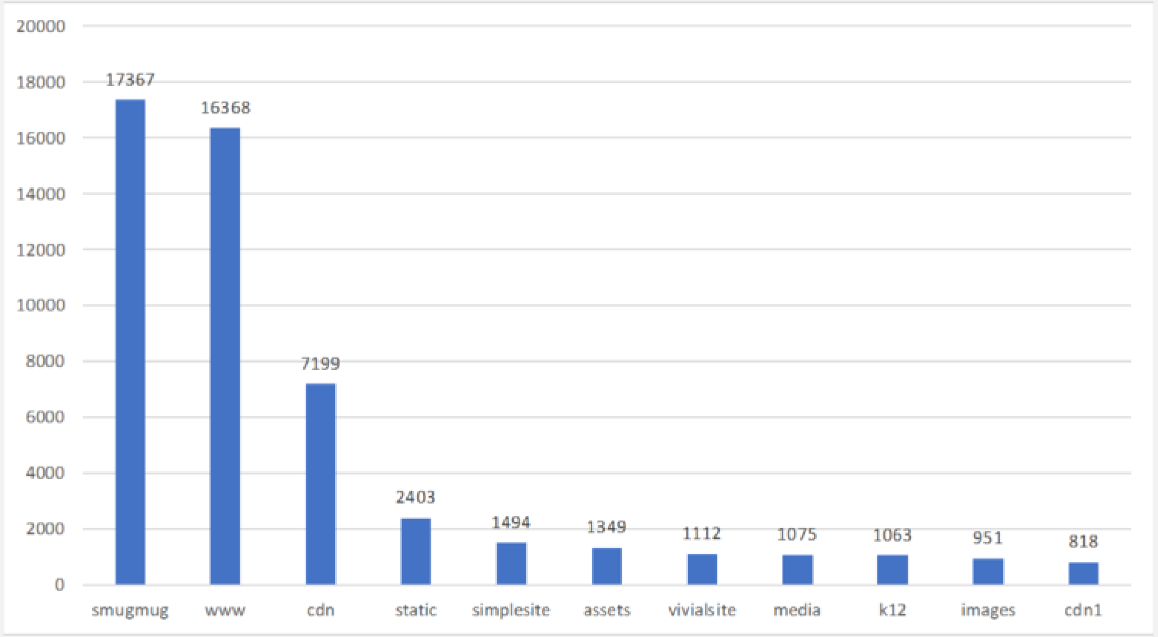


TOP DOMAINS





TOP DOMAIN PARTS (TLDS REMOVED)



SOME OF MY PERSONAL FAVORITES...

login.gov
duo.com

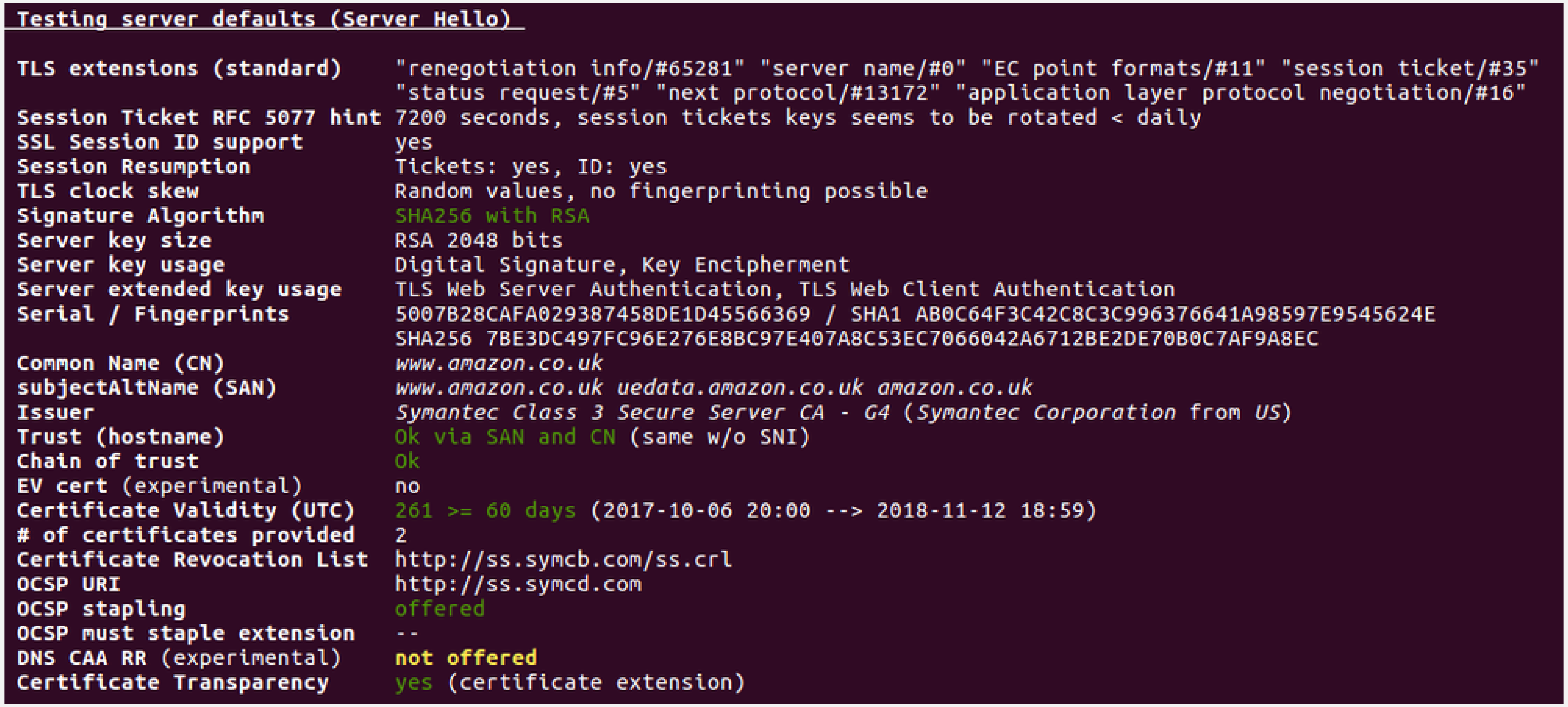
www.secureworks.com
status.symantec.com
dnc.org
fitbit.com
zillow.com
ethereum.org
refer.walmart.com
www.netgear.com
view.yahoo.com
today.msnbc.com
www.amazon.co.uk
46 fbi.gov domains (regional offices)
101 nasa.gov domains

UPDATE - Choosing a Domain for C2 Channels

Vincent Yiu pointed out to me that even though all of these domains "work", many are not desirable for C2 channels due to SSL issues. When choosing a domain, you'll want to make sure that the SSL certificate is trusted, and that it isn't going to expire during the time you are using it.

There are a lot of ways to check this. You could load the site in a browser, and view the certificate details there. Lots of these domains have redirects though, so you'd need to break out the developer tools to find the cert you care about. Nmap scripts can output some certificate information (--script ssl-cert) but I didn't find one that would actually test the chain of trust. I found the tool testssl.sh to be the most convenient way to do certificate verification. We can run it with the -S flag since all we care about is the certificate related information. Here's the output from a domain with a good certificate:

```
./testssl.sh -S www.amazon.co.uk
```



```
 Testing server defaults (Server Hello) 

 TLS extensions (standard)    "renegotiation info/#65281" "server name/#0" "EC point formats/#11" "session ticket/#35"
                              "status request/#5" "next protocol/#13172" "application layer protocol negotiation/#16"
 Session Ticket RFC 5077 hint 7200 seconds, session tickets keys seems to be rotated < daily
 SSL Session ID support       yes
 Session Resumption           Tickets: yes, ID: yes
 TLS clock skew               Random values, no fingerprinting possible
 Signature Algorithm          SHA256 with RSA
 Server key size              RSA 2048 bits
 Server key usage             Digital Signature, Key Encipherment
 Server extended key usage    TLS Web Server Authentication, TLS Web Client Authentication
 Serial / Fingerprints        5007B28CAFA029387458DE1D45566369 / SHA1 AB0C64F3C42C8C3C996376641A98597E9545624E
                              SHA256 7BE3DC497FC96E276E8BC97E407A8C53EC7066042A6712BE2DE70B0C7AF9A8EC
 Common Name (CN)             www.amazon.co.uk
 subjectAltName (SAN)         www.amazon.co.uk uedata.amazon.co.uk amazon.co.uk
 Issuer                       Symantec Class 3 Secure Server CA - G4 (Symantec Corporation from US)
 Trust (hostname)             Ok via SAN and CN (same w/o SNI)
 Chain of trust               Ok
 EV cert (experimental)       no
 Certificate Validity (UTC)   261 >= 60 days (2017-10-06 20:00 --> 2018-11-12 18:59)
 # of certificates provided   2
 Certificate Revocation List  http://ss.symcb.com/ss.crl
 OCSP URI                     http://ss.symcd.com
 OCSP stapling                offered
 OCSP must staple extension   --
 DNS CAA RR (experimental)    not offered
 Certificate Transparency     yes (certificate extension)
```

You can see here that the chain of trust is OK, the certificate is valid for 261 more days, and a lot more details. If you want to see some bad output, I highly recommend playing with the subdomains of badssl.com, listed here: https://badssl.com/dashboard/. In general though, as long as nothing is red in the testssl.sh output, you should be good to go!

Happy domain fronting!

NEXT

Writing .NET Executables for Pentesters - Part 2

Follow us on Twitter @peewpw @swizzlez_

```
 ______   ______   ______   __  __  __        ______   __  __  __
/_____/\ /_____/\ /_____/\ /_//_//_/\      /_____/\ /_//_//_/\
\:::_ \ \\::::_\/_\::::_\/_\:\\:\\:\ \     \:::_ \ \\:\\:\\:\ \
 \:(_) \ \\:\/___/\\:\/___/\\:\\:\\:\ \  ___\:(_) \ \\:\\:\\:\ \
  \: ___\/ \::___\/_\::___\/_\:\\:\\:\ \/__/\\: ___\/ \:\\:\\:\ \
   \ \ \    \:\____/\\:\____/\\:\\:\\:\ \::\ \\ \ \    \:\\:\\:\ \
    \_\/     \_____\/ \_____\/ \_______\/ \:_\/ \_\/     \_______\/
```