Insecurity through Censorship: Vulnerabilities Caused by The Great Firewall

Insecurity through Censorship: Vulnerabilities Caused by The Great Firewall - Shubham Shah, assetnote.io.

- Published: date not stated
- Original: <https://www.assetnote.io/resources/research/insecurity-through-censorship-vulnerabilities-caused-by-the-great-firewall>
- Preserved from: https://www.assetnote.io/resources/research/insecurity-through-censorship-vulnerabilities-caused-by-the-great-firewall (live) on 2026-08-09
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

> *The testing tool to identify if your domain is vulnerable to this attack is **located at the end of the blog post**.*

At Assetnote, we are contstantly resolving millions of DNS as part of the operation of our Attack Surface Management platform. When performing DNS resolutions at the scale that we do across the number of diverse customer attack surfaces that we are monitoring we start to notice some really interesting behavior in the wild.

Around two years ago, we had a new customer onboard with us who had a large presence in China. We started seeing an incredible number of subdomains that were resolving to what seemed like random IP addresses.

These subdomains did not seem "real", in the sense that they did not look like legitimate infrastructure that this company had set up. In addition to this, these records did not consistently resolve to the same IP address.

While we thought this was bizarre behavior at the time, we focus heavily on ensuring that the customers of our Attack Surface Management platform only see assets that are "real" (meaning we automatically filter out wildcards and other records that are resolving that aren't legitimately owned by the company).

We came up with a plan to strategically filter out these subdomains for this customer, and we put it down to some sort of weird DNS misconfiguration that was isolated to this customer. At the time, we did not realize that we had actually come across China's systematic approach in poisoning or tampering with DNS queries.

After experiencing this across a few customers, we investigated this issue further and understood that China's tampering of DNS queries is based on patterns within the subdomain.

In 2022, it was reported that the total number of Chinese domains had reached 33.8 million. This does not account for all of the other TLDs that also route their DNS resolutions through Chinese infrastructure (i.e. AlibabaDNS, Cloudflare China, AWS Route53 China).

The attack vectors we have found in this blog post affect any domains that are being routed through Chinese infrastructure. It's hard to quantify the exact number of affected zones, but we estimate the number would be in the tens of millions.

After the discovery of this issue, we reviewed notes with Eric (todayisnew), who had independently found this issue and had spent a lot of time investigating it. Eric provided an additional exploitation vector that doesn't rely on the ability to claim a Fastly domain. You can skip to the attack vectors at the bottom of this blog post if you're not interested in the analysis.

We have also created an interactive tool to test if your domain is affected by this DNS poisoning issue, which can be found here.

Unreliable DNS Resolvers

Initially, we considered this inconsistency as an outlier or side-effect of unreliable DNS resolvers. “Unreliable” DNS resolvers where the resolver would not respond consistently to our requests, returning different records or no records to the same request. We considered if we were experiencing server-side load balancing of DNS records with methods such as DNS load balancing, or IP load balancing, however for these customers we concluded they were not using any of these features and we were experiencing genuinely “unreliable” DNS resolvers. We suspected this unreliability may come from other server-side load balancing algorithms, packet injection or packet tampering.

Note that unreliable DNS resolvers here refers to the authoritative name server for a DNS record, the server responsible for definitive answers about a given domain, in contrast to recursive resolvers that relay queries to authoritative DNS servers for you.

When our platform identifies a domain that resides on an unreliable authoritative DNS server, we typically perform further analysis to ensure that the subdomains are real. In this case, we found an interesting pattern, that all domains belonging to the .cn TLD, were being marked as unreliable.

Later, we realized that this poisoning was happening for other TLDs as well, as long as the nameservers that the domain used were located in China. This poisoning was not limited to domains with the .cn TLD.

Analysis

When debugging this DNS issue, we wanted to understand where exactly we were experiencing the inconsistency. Was it with our recursive resolvers or with the authoritative resolver?

```
dig +trace 5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn

; <<>> DiG 9.10.6 <<>> +trace 5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn
;; global options: +cmd
.                       515969  IN      NS      a.root-servers.net.
.                       515969  IN      NS      b.root-servers.net.
.                       515969  IN      NS      c.root-servers.net.
.                       515969  IN      NS      d.root-servers.net.
.                       515969  IN      NS      e.root-servers.net.
.                       515969  IN      NS      f.root-servers.net.
.                       515969  IN      NS      g.root-servers.net.
.                       515969  IN      NS      h.root-servers.net.
.                       515969  IN      NS      i.root-servers.net.
.                       515969  IN      NS      j.root-servers.net.
.                       515969  IN      NS      k.root-servers.net.
.                       515969  IN      NS      l.root-servers.net.
.                       515969  IN      NS      m.root-servers.net.
;; Received 525 bytes from 1.1.1.1#53(1.1.1.1) in 14 ms

sg.                     172800  IN      NS      dsany3.sgnic.sg.
sg.                     172800  IN      NS      dsany.sgnic.sg.
sg.                     172800  IN      NS      ns4.apnic.net.
sg.                     172800  IN      NS      pch.sgzones.sg.
sg.                     172800  IN      NS      dsany2.sgnic.sg.

REDACTED.vn             3600 IN     NS      ns1.alibabadns.com.
REDACTED.vn             3600 IN     NS      ns2.alibabadns.com.
;; Received 655 bytes from 185.159.197.170#53(dsany2.sgnic.sg) in 129 ms

5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn 180 IN A 47.88.58.234
;; Received 122 bytes from 140.205.103.194#53(ns2.alibabadns.com) in 482 ms
```

We can see that walking the resolution tree, the domain above delegates to nameservers on alibabadns.com. Across multiple resolutions, we were observing that the IP address was unstable

```
assetnote@agent-1:~$ for i in $(seq 1 10); do dig 5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpage
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. 170 IN A 108.160.166.9
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. 224 IN A 162.125.32.12
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. 194 IN A 108.160.162.109
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. 134 IN A 116.89.243.8
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
;5-builtin.webproxy.idc-lorien21swww-web5245.bl.wwwytcos-linkpages.e.REDACTED.vn. IN A
```

Diving deeper, we tried to create a minimal reproducible example, and managed to distill it down to specifically any query including webproxy.id. Later we would find out there were a number of “keywords” that would be intercepted.

```
assetnote@agent-1:~$ for i in $(seq 1 10); do dig webproxy.id.REDACTED.vn @ns2.alibabadns.com | egrep 'IN\s+A'; dor
;webproxy.id.REDACTED.vn.       IN      A
webproxy.id.REDACTED.vn. 104    IN      A       128.242.245.43
;webproxy.id.REDACTED.vn.       IN      A
webproxy.id.REDACTED.vn. 231    IN      A       65.49.26.98
;webproxy.id.REDACTED.vn.       IN      A
webproxy.id.REDACTED.vn. 203    IN      A       199.59.149.237
;webproxy.id.REDACTED.vn.       IN      A
webproxy.id.REDACTED.vn. 107    IN      A       199.59.148.9
;webproxy.id.REDACTED.vn.       IN      A
;webproxy.id.REDACTED.vn.       IN      A
webproxy.id.REDACTED.vn. 133    IN      A       108.160.165.173
;webproxy.id.REDACTED.vn.       IN      A
webproxy.id.REDACTED.vn. 98     IN      A       182.50.139.56
;webproxy.id.REDACTED.vn.       IN      A
webproxy.id.REDACTED.vn. 171    IN      A       173.208.182.68
;webproxy.id.REDACTED.vn.       IN      A
;webproxy.id.REDACTED.vn.       IN      A
```

However, it turns out that the domain doesn’t even need to exist for us to receive invalid responses (please don’t buy this domain to prove us wrong). We would expect for this domain that the server would always return NXDOMAIN or REFUSED, however it sometimes does return a

response. Additionally, the domain looked up didn't have to be .cn as we initially expected, it only had to contain a keyword and be responded to by an authoritative nameserver hosted in China (we suspected).

```
assetnote@agent-1:~$ for i in $(seq 1 10); do dig webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com @ns2.ali
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 80 IN A 108.160.167.148
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 72 IN A 199.59.149.244
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 221 IN A 199.59.148.96
```

We also suspected that this was isolated to a single DNS resolver. However we soon found another customer exhibiting the same behavior. Except they instead used Cloudflare for their DNS which perplexed us even more.

```
assetnote@agent-1:~$ dig webproxy.id.32323232323232323232.REDACTED2.cn  @ns45.dns.cf-ns.com

; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> webproxy.id.32323232323232323232.REDACTED2.cn @ns45.dns.cf-
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35477
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;webproxy.id.32323232323232323232.REDACTED2.cn. IN A

;; ANSWER SECTION:
webproxy.id.32323232323232323232.REDACTED2.cn. 229 IN A 199.59.150.43

;; Query time: 204 msec
;; SERVER: 119.0.67.92#53(ns45.dns.cf-ns.com) (UDP)
;; WHEN: Mon Nov 20 00:31:52 UTC 2023
;; MSG SIZE  rcvd: 75
```

We could confirm that our keyword intercepting was in place, because weproxy.id would consistently return NXDOMAIN as expected, and similarly, an entirely non-existent domain with webproxy.id would be returning records

```
assetnote@agent-1:~$ for i in $(seq 1 10); do dig webproxy.id.REDACTED2.cn  @ns45.dns.cf-ns.com. | egrep 'IN\s+A'; d
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   69      IN      A       202.160.128.40
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   191     IN      A       31.13.95.48
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   203     IN      A       192.133.77.191
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   127     IN      A       67.228.102.32
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   95      IN      A       104.244.46.57
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   244     IN      A       103.214.168.106
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   113     IN      A       104.244.46.208
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   144     IN      A       103.39.76.66
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   78      IN      A       192.133.77.189
;webproxy.id.REDACTED2.cn.          IN      A
webproxy.id.REDACTED2.cn.   106     IN      A       59.188.250.54
assetnote@agent-1:~$ for i in $(seq 1 10); do dig weproxy.id.REDACTED2.cn  @ns45.dns.cf-ns.com. | egrep 'IN\s+A'; do
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
;weproxy.id.REDACTED2.cn.           IN      A
assetnote@agent-1:~$ for i in $(seq 1 10); do dig webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com @ns45.d
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 175 IN A 157.240.8.50
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 214 IN A 157.240.21.9
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 76 IN A 162.125.80.5
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 156 IN A 173.244.217.42
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 211 IN A 108.160.162.102
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
```

```
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 183 IN A 162.125.18.133
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 137 IN A 185.45.7.185
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 244 IN A 103.252.114.11
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 209 IN A 174.37.243.85
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 140 IN A 108.160.163.108
```

Coming back to our original question, was this a recursive resolver or authoritative resolver issue? Turns out the answer is neither! But something fishy is happening at the authoritative nameserver level. Attempting to resolve our entirely made up domain against Cloudflare's resolvers, we find that the expected REFUSED response for a non-existent domain (weproxy.id...) is replaced with a NOERROR. We suspect therefore, that any resolution crossing the Great Firewall of China with specific keywords would be poisoned.

```
assetnote@agent-1:~$ dig webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com @ns45.dns.cf-ns.com.

; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com @ns45
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29229
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN A

;; ANSWER SECTION:
webproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. 190 IN A 31.13.67.33

;; Query time: 156 msec
;; SERVER: 61.159.93.124#53(ns45.dns.cf-ns.com.) (UDP)
;; WHEN: Mon Nov 20 00:35:54 UTC 2023
;; MSG SIZE  rcvd: 86

assetnote@agent-1:~$ dig weproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com @ns45.dns.cf-ns.com.

; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> weproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com @ns45.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 876
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
; EDE: 20 (Not Authoritative)
;; QUESTION SECTION:
;weproxy.id.1-2-3-4-5-6-7-CERTAINLYAUNIQUEDOMAIN.com. IN       A

;; Query time: 180 msec
;; SERVER: 61.159.93.124#53(ns45.dns.cf-ns.com.) (UDP)
;; WHEN: Mon Nov 20 00:36:04 UTC 2023
;; MSG SIZE  rcvd: 86
```

Turns out there already is literature published on this very topic:

- https://www.catchpoint.com/blog/great-firewall-of-china
- https://www.crowdstrike.com/blog/cyber-kung-fu-great-firewall-art-dns-poisoning/

- https://www.usenix.org/system/files/sec21-hoang.pdf

Finding the Bad IP Addresses & Domains

Finding the bad IP addresses was a simple affair of resolving enough domains to sample what IP addresses were in the pool of returned domains. Turns out that there is a finite number of IPs that we could possibly return.

```
> for i in $(seq 1 10000); do echo $i.webproxy.id.doesnotexistkjbn1k2j4b1k2j3n.com; done > fakedomains.txt

> cat fakedomains.txt | onodns -w=false --cname=false -s ns2.alibabadns.com:53 --retries=0 --pps=500 2>&1 | tee resu

<snip>

> cat results.txt | grep -v '{' | cut -f3 -d' ' | sort | uniq -c | sort | wc -l
  592
```

We have attached a list of the known poisoned IPs to the end of this post, and the corresponding whois information we pulled from these IPs. Although research from Hoang et al., identifies over 1700 IP addresses, we suspect our list may be different due to the narrower scope of keywords we're resolving or more simply, a different list at that point in time was used.

Conversely, bad keywords could be identified with a reverse DNS lookup on these IPs and seeing what nonsensical domains are pointing to IPs that are known to belong to a single host.

Further work was done by Eric, who compiled a list of known keywords from his own research which we have attached to the end of this post.

Weaponizing via Fastly

Knowing that we can include a specific string inside the subdomain that will lead to a resolution to a set of IP addresses that are responding to HTTP requests, we dove into the possibilities of how an attacker could potentially use this to their advantage.

The first and most powerful technique that was discovered related to Fastly. A number of these IP addresses returned from China's poisoning were found to be pointing to Fastly. When you visit these poisoned subdomains in your browser, you would get an error message from Fastly stating that there are no such CDN profiles with that domain configured:

```
Fastly error: unknown domain 46999-info727672.drupal-helpcenter-pre-www-chameleon-pre-pin.hgfe.latindc.ssh.pc54
```

The default behavior of Fastly's CDN is that any one customer can add a domain to their CDN profile. If the domain is claimed by another customer, it cannot be used on another Fastly account.

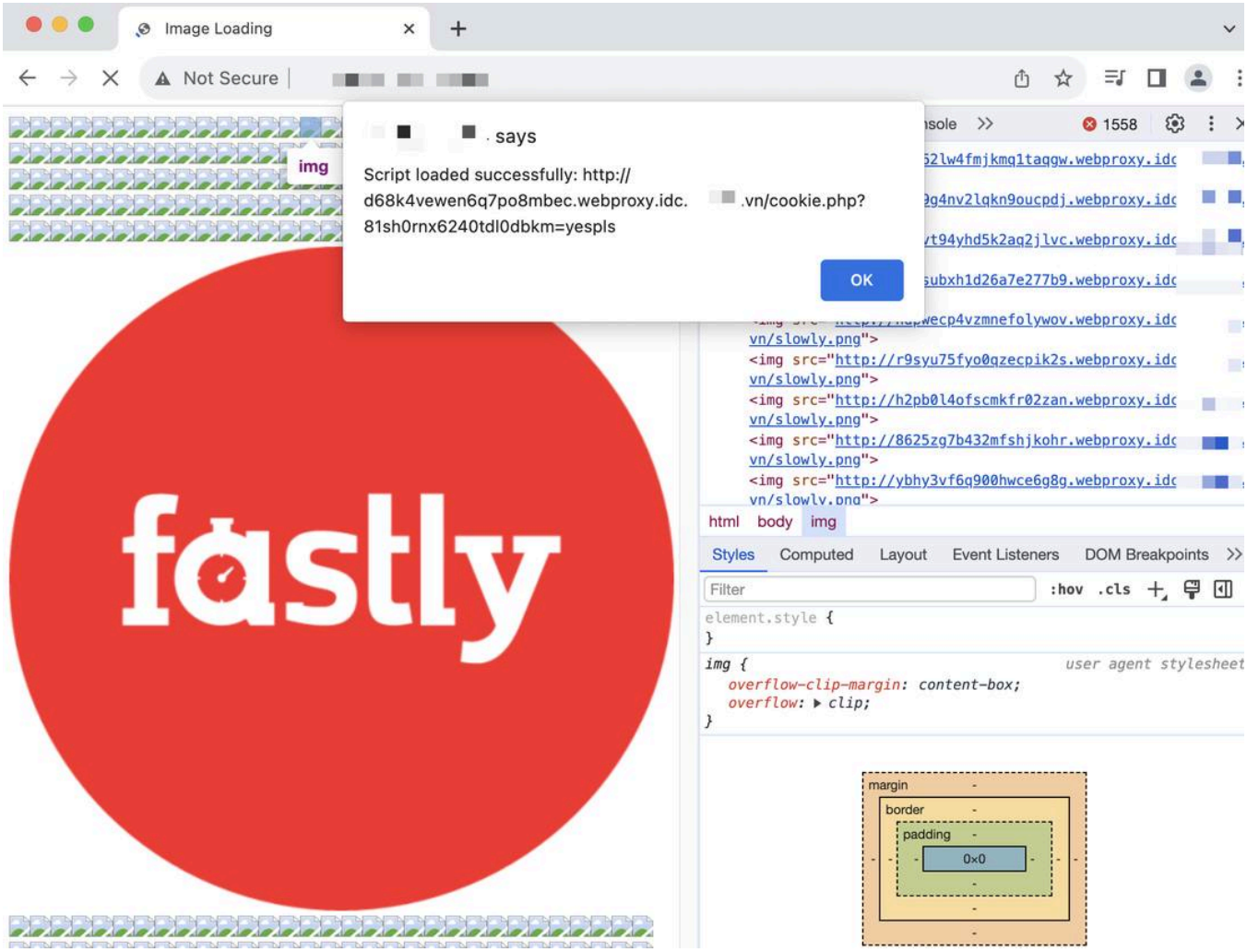
This was the only prerequisite for being able to exploit the poisoning, the domain being targeted should not have been added to Fastly by another customer.

By registering a Fastly account, creating a CDN profile, and then adding `*.domain.tld` to the configuration, it was possible to serve all requests for any subdomain of that domain, when the requests were being served by Fastly.

But how do we practically exploit this in the wild? How do we take advantage of exploiting the origin of our target in the browser? The subdomains are clearly resolving to different IPs and are constantly rotating.

The first step is pointing the Fastly CDN profile to your server's IP as its origin server, so that all requests will be routed to your server for `*.domain.tld`. After doing so, host an image on your server, which will aid us in our exploitation process.

Since the subdomain poisoning patterns are predictable, we can leverage some client side code to generate a list of random subdomains that will lead to poisoned responses. While we do not know which subdomain will eventually point to Fastly, we can simply load up an image from all of the random subdomains until we win, leveraging event handlers to notify us about the successful poison.



Show Fastly Proof-of-Concept Code

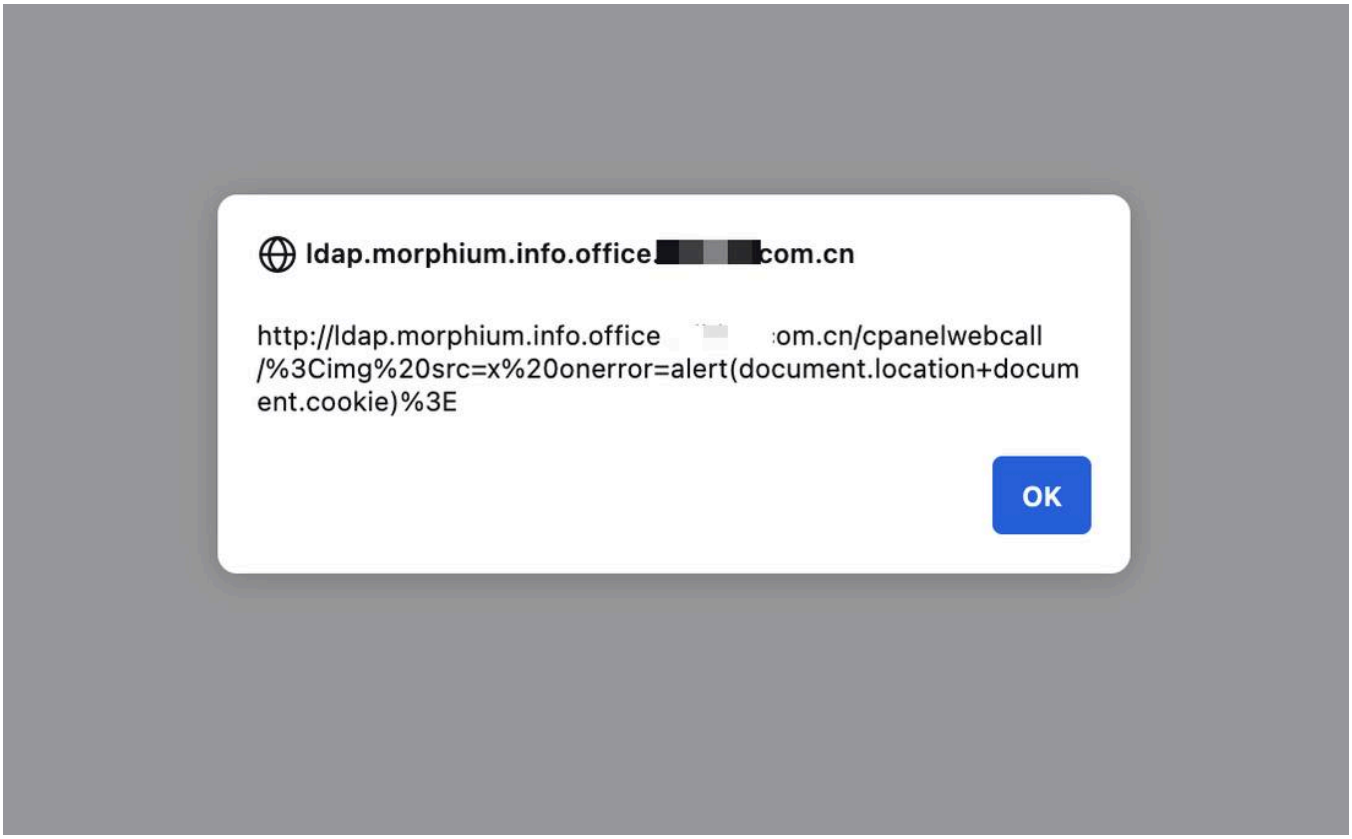
Weaponizing via cPanel XSS

Earlier last year, we released some research on a cross-site scripting issue affecting cPanel.

When speaking with Eric about the chinese DNS poisoning issue, he shared a proof of concept that was more universal than our Fastly proof-of-concept but had a slightly lower impact as you cannot capture HTTPOnly cookies.

Eric let us know that he had spent over 100 hours on this issue from a research perspective, coming to similar conclusions to us that it affects any domain that is routed through China. He was keen for this issue to be discussed more widely and with his permission we are including the proof-of-concept he created to demonstrate the risks of DNS poisoning from China.

This proof-of-concept relies on the fact that one of the IPs within the pool of IPs returned through the poisoning will be running a vulnerable version of cPanel, leading to XSS. It will iterate through generated subdomains until the XSS fires.



We have simplified Eric's proof-of-concept and our version can be found below:

Show cPanel Proof-of-Concept Code

What is the impact?

As this vulnerability is effectively the same as a subdomain takeover, the same impacts that affect typical subdomain takeover vulnerabilities apply. There were two separate exploitation vectors described in this blog post and they both have slightly different impacts.

With the Fastly takeover technique, you are able to listen to all traffic that gets routed to your origin. This is more impactful than just an XSS vulnerability, as it means that it will be possible to steal HTTPOnly cookies which are not made available to JavaScript.

Typically, this would be even more impactful if existing infrastructure was making calls to the subdomain that has been taken over, but due to the dynamic nature of these subdomains, the

chances of this are slim. The real value of this attack vector is still client-facing, attacking end users.

Unfortunately, this attack's prerequisite is that the domain must not be taken on Fastly already. This may prevent attackers from being able to exploit this issue successfully for some high profile domains that legitimately use Fastly.

Alternatively, the cPanel XSS vector is much more universal and works on any domain affected by China's poisoning. The only noticeable difference between the Fastly vector and this one is that it is not possible to steal HTTPOnly cookies.

For both attack vectors, it is not possible to steal cookies that have been marked as "Secure", as they will only be transmitted when the communication with the origin is via https. There are angles of using a self signed certificate, but it would require the user to explicitly ignore certificate warnings.

There may be a way to claim a TLS certificate if you are able to retry a subdomain enough times with the certificate provider until it hits your Fastly origin, but this wasn't immediately possible with our testing.

Of course, as with any cross-site scripting attack, it is possible to phish users and perform any other type of client side attack while you have the origin of `*.target.tld`. Depending on the configuration of the target's web applications, cookie stealing or cross-application interactions may be possible.

Theories Explaining This Behaviour

On the surface, this behavior is very odd and we began to wonder why this infrastructure operated this way. Based on our analysis of the keywords that trigger this behavior (see Appendix below), it seems that the most logical theory is that this is simply a censorship mechanism.

Our leading theory is that the Great Firewall inside China is manipulating DNS responses due to blacklisted or blocked keywords. This affects any system that has to pass through the Great Firewall before it reaches the end-user/client. Most of the keywords relate to VPN software, proxies, adult sites, file and text sharing and torrents. This is consistent with the kinds of material that China is known to censor within its borders.

This theory is also supported by the behaviour of the keyword filtering which seems to imply greedy filtering. Rather than blocking a specific single domain such as say tracker.thepiratebay.org this system appears to be triggering if the keyword is embedded anywhere in the domain. This is why we poisoned DNS responses for non-existent domains.

This does beg the question that if censorship is the goal why not simply sinkhole to a non-functional IP address? This one is a bit harder to answer. There is a lot of diversity in the what networks the IPs in these poisoned records point to. Infrastructure for large social media networks, CDNs, cloud platforms and even random blogs. Our best guess is that this is simply random and not intentional and the prevalence of these sorts of organizations is simply that they are more likely to own a large IP space.

What Can Be Done to Mitigate the Risk?

Unfortunately there is no simple way for organizations to address this risk. Operating infrastructure in China is not as simple as signing up to some hosting provider in China with a credit card. Any business that wants to host in China or even have any kind of Internet presence inside China needs to apply for and obtain a valid Chinese Internet Content Provider (ICP) license before anything can be hosted.

There are a number of very specific requirements to have this license including:

- You must have a domain name registered with a Chinese domain name registrar.
- You must use servers located in China.
- You must use a hosting provider that is licensed by the Chinese government.

These requirements mean that any business that wants to have an Internet presence in China must host their infrastructure in China and thus are susceptible to the DNS poisoning that we have highlighted in this post. While simply not operating in China would solve this it is obviously not a realistic solution.

One solution would be to host your nameservers outside of China. While this would mitigate against the DNS poisoning it does present concerns around speed, reliability and general user experience for the Chinese users of these sites.

One thing that organization can more easily control that will limit the impact of some of the issues demonstrated in this post is to ensure that basic web security hygiene is implemented. This means ensuring that all cookies have the “Secure” flag, so that they are not accessible by web servers that are not being communicated to via HTTPs / TLS. Enforcing the “HTTPOnly” flag on all cookies will mean that a simple XSS vulnerability cannot be escalated to steal sensitive cookies.

Unfortunately, while basic web security hygiene goes a long way towards mitigating the attacks presented here defacement and phishing still pose a real and exploitable risk.

Special Thanks

- Eric Head (todayisnew) for sharing additional keywords and research related to this DNS poisoning attack.
- Sean Yeoh with help in the data analysis of returned poisoned IP addresses.
- Our customers of Assetnote who contributed valuable insights into this attack and the difficulties in mitigating the issue.

Appendix

Known Keywords Triggering Poisoning

Show Known Keywords

Whois and Known IPs

Show Known IPs

Testing Tool

Do you have any domains that are being routed through China that you want to test?

Enter in the domain you would like to test, i.e. `domain.cn`, and we will let you know if it's vulnerable to the attack vectors in this blog post.

Enter domain: Lookup
