Domain Fronting is Dead, Long Live Domain Fronting

Using TLS 1.3 to evade censors, bypass network defenses, and blend in with the noise
Outline

0| Domain Fronting 101
  - HTTP Basics
  - HTTPS Basics
  - Classic Domain Fronting

1| TLS 1.3 + ESNI for Domain Hiding
  - DNS over TLS/HTTPS
  - TLS 1.3 with ESNI
  - Domain Hiding

2| Demos
  - ESNI for Domain Hiding
  - Bypassing SNI and full decrypt firewalls
  - Alternate protocols

3| What is Blue to do?
Domain Fronting 101
Domain Fronting 101

- If you wish to make an apple pie from scratch, you must first invent the universe
- To understand Domain Fronting, you must first understand HTTP over TLS (aka HTTPS)
- Server Name Indication (SNI) allows multiple sites to be hosted on the same IP
- TLS 1.3 enables encrypted certificates and encrypted Server Name Identification (ESNI)
- DNS over TLS or HTTPS + TLS 1.3 = domain fronting 2.0 or “domain hiding”
HTTP Basics

- First, a user requests the IP of a server via DNS
- This is an unencrypted packet sent to UDP port 53
HTTP Basics

- The DNS server responds with an IP address
- The response is also unencrypted
HTTP Basics

- The user sends a GET request, using the domain in the “Host” header
- TCP port 80 - unencrypted
HTTP Basics

- The server responds with HTML content
HTTP Basics

- Obviously bad because nothing is encrypted
- Both the DNS and HTTP request and response are in plaintext
HTTP Basics

- Obviously bad because nothing is encrypted
- Both the DNS and HTTP request and response are in plaintext
HTTPS Basics

- Starts off the same way, a user requests the IP of a server via DNS
- This is an unencrypted connection on UDP port 53
HTTPS Basics

- The DNS server responds with an IP address
- The response is also unencrypted
HTTPS Basics

- The user sends a ClientHello to start the TLS handshake
- Server uses the “server_name” field (plaintext) to lookup how to respond
HTTPS Basics

- The server responds with a certificate (in plaintext unless TLS 1.3) and completes the handshake.
- All data after the handshake is encrypted.
HTTPS Basics

- Much better than HTTP
- Entire DNS process and the certificate exchange process are still unencrypted
HTTPS Basics

- Much better than HTTP
- Entire DNS process and the certificate exchange process are still unencrypted

```sh
server {
    listen 443;
    server_name defcon28.hackthis.computer;
    ssl_certificate /path/to/cert.pem;
    ...
}
server {
    listen 443;
    server_name test.hackthis.computer;
    ssl_certificate /path/to/test-cert.pem;
    ...
}
```
Domain Fronting

- Circumvent censorship by obfuscating the domain of an HTTPS connection
- Connect to an approved server, but send an HTTP request for the actual destination
- Uses a hosting service to host the true destination - false destination must be on the same service
  - Google App Engine
  - Amazon S3/CloudFront
  - Microsoft Azure CDN
  - Others
Domain Fronting

- DNS lookup as before for any site hosted by the hosting service
- Client and Server handshake as usual
Domain Fronting

- Client and Server handshake as usual
Domain Fronting

- Client sends an HTTP request with the Host header set to the actual destination
- The CDN forwards the request as long as the destination is hosted by the service
  - Any site on GAE could be used to front for an otherwise censored GAE server
Domain Fronting

- Like a letter delivered to a house with multiple residents
- The mailman can see the address on the outside
- Letter on the inside goes to the correct person
Domain Fronting

- April 2018 - The music stops
- Google
  - “Domain fronting has never been a supported feature at Google”¹
- Amazon
  - Implemented “Enhanced Domain Protections”
  - “no customer ever wants to find that someone else is masquerading as their innocent, ordinary domain”²
  - Think of the innocent children ordinary domains!
- Cloudflare
  - Only HTTP works
- Azure
  - Still works... For now

[1] A Google update just created a big problem for anti-censorship tools
[2] Enhanced Domain Protections for Amazon CloudFront Requests
Domain Fronting - Issues

- Major providers shut it down
- Limited fronting options
  - Only sites hosted on the same provider can be used to front
- Must host an “app” or have an account with the provider
  - Not free
    - Bandwidth
    - CPU time
  - Complex sign up requirements
    - Identity checks
    - Phone required
    - Credit card required
TLS 1.3 + ESNI for “Domain Hiding”
The Growth of TLS 1.3

Qualys SSL Labs - Protocol Support of Alexa Top Sites

- SSL v2.0
- SSL v3.0
- TLS v1.0
- TLS v1.1
- TLS v1.2
- TLS v1.3

31.7%
The Growth of TLS 1.3

TLS Versions as seen by CloudFlare

- Other
- TLS 1.0
- TLS 1.1
- TLS 1.2
- TLS 1.3

59%
DNS - Fixing the Problem

- What if you wrap a DNS request in TLS?
- How about HTTPS? (RFC 8484)
- "The unmitigated usage of encrypted DNS, particularly DNS over HTTPS, could allow attackers and insiders to bypass organizational controls." - SANS³
  - Great!

[³] A New Needle and Haystack: Detecting DNS over HTTPS Usage
TLS 1.3 and ESNI

- Now that DNS is encrypted, it can be used to fetch a public key before an HTTPS connection is started
- Classic Diffie-Hellman key exchange to symmetrically encrypt the server_name
  - All data required is sent in a single Client Hello (the client’s public key + extras)
TLS 1.3 and ESNI - Step by Step

- Client requests the IP address and ESNI public key via DNS over TLS or HTTPS
TLS 1.3 and ESNI - Step by Step

- DNS server returns the IP address and ESNI public key via DNS over TLS or HTTPS
- Cloudflare rotates their key every ~1 hour (with a few hours of buffer allowed by the servers)
TLS 1.3 and ESNI - Step by Step

- Client sends a TLS 1.3 ClientHello with an encrypted_server_name extension

```sh
server {
  listen 443;
  server_name defcon28.hackthis.computer;
  ssl_certificate /path/to/cert.pem;
  ...  
}
server {
  listen 443;
  server_name test.hackthis.computer;
  ssl_certificate /path/to/test-cert.pem;
  ...  
}
```
TLS 1.3 and ESNI - Step by Step

- Web server responds with a ServerHello that includes the encrypted certificate

```plaintext
server {
  listen 443;
  server_name defcon28.hackthis.computer;
  ssl_certificate /path/to/cert.pem;
  ...
}
server {
  listen 443;
  server_name test.hackthis.computer;
  ssl_certificate /path/to/test-cert.pem;
  ...
}
```
TLS 1.3 and ESNI - Weak Spots

- DNS response for IP or _esni could be tampered with (poisoned resolver cache)
- DNSSEC!\(^4\)

TLS 1.3 and ESNI - Weak Spots

- DNS over TLS or HTTPS is completely blocked
- Preload ESNI keys to bootstrap the connection
TLS 1.3 and ESNI - Weak Spots

- The TLS connection goes to an IP address
- An IP may be shared by many domains, but may not
- Is there a generally applicable way to route to any domain via any other?
  - No, but how close can we get?
<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS is unencrypted</td>
<td>DNS over TLS or HTTPS</td>
</tr>
<tr>
<td>DNS is untrustworthy</td>
<td>DNSSEC</td>
</tr>
<tr>
<td>Encrypted DNS is blocked</td>
<td>Bootstrap ESNI keys</td>
</tr>
<tr>
<td>SNI is unencrypted</td>
<td>ESNI</td>
</tr>
<tr>
<td>IPs (or IPs that domains resolve to) are blocked</td>
<td>Domain hiding with the largest CDN!</td>
</tr>
</tbody>
</table>
Demos
Cloudflare

- Founded in 2009
- Content Delivery Network (CDN)
  - Highest number of internet exchange points of any network
  - Largest CDN in the world
  - Supports TLS 1.3, ESNI, Websockets, and QUIC
- Authoritative DNS
  - Over 26,000,000 domains
  - Supports DNSSEC
Domain Hiding with Cloudflare

- A TLS 1.3 connection with ESNI is sent to ANY Cloudflare server
  - SNI can be included as well - does not have to match ESNI value
- A HTTP request is sent using that connection can have any Host header
  - True domain must have DNS provided by Cloudflare
- Cloudflare will forward the request to the true destination - just like domain fronting!
- Robin Wood (@digininja) was the first to show this was possible

As of 2020-08-10 ESNI and SNI cannot be used together
Noctilucent

- Go (Golang) rewrite of crypto/tls based on Cloudflare’s tls-tris project
- TLS 1.3 support
  - Config options specific for domain hiding
    - ESNIServerName - does not need to match the SNI extension or Host header
    - PreserveSNI - Allows the sending of a ClientHello with both SNI and ESNI extensions
- Drop in replacement for standard crypto/tls - backwards compatible
- Test client application
  - Attempts DNS over HTTPS (DoH) - Falls back to DNS over TLS, then system default DNS
  - Allows command line tuning of nearly every part of the TLS and HTTP connection
  - HTTPS and Websocket support
  - Cross platform!
Normal HTTPS Connection

```bash
.../esni/client $ ./client-macOS -TLSTransport cloudflare.com -serverName cloudflare.com -HostHeader cloudflare.com
[==] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256
[==] ESNI host has not been set
[==] SNI host set to: cloudflare.com
[+] Connecting to https://cloudflare.com:443
[+] TLS handshake complete
[+] Sending GET request: GET / HTTP/1.1
Host: cloudflare.com
User-Agent: ESNI_FRONT_TEST
Accept: */*
Connection: close

[+] GET request sent
[==] Response:
HTTP/1.1 301 Moved Permanently
Date: Tue, 24 Mar 2020 00:23:42 GMT
Transfer-Encoding: chunked
Connection: close
Cache-Control: max-age=3600
Expires: Tue, 24 Mar 2020 01:23:42 GMT
Location: https://www.cloudflare.com/
Set-Cookie: __cf_bm=166e3a90f25654e23ea8574af6e0cac3fc530e4-1585009422-1800-AQQvE0Fo8e2IeEE7TuneGnj4E3aeOv382tFA2iBp3BDl8Vv+5qaljvAEDCySkqCMj0VSS2fpgaVUrSDCQjZv+nU=; path=/; expires=Tue, 24-Mar-20 00:53:42 GMT; domain=.cloudflare.com; HttpOnly; Secure; SameSite=None
Strict-Transport-Security: max-age=15780000; includeSubDomains
Server: cloudflare
CF-Ray: 578c3eb81d10740d-IAD
alt-svc: h3-27:"443"; ma=86400, h3-25:"443"; ma=86400, h3-24:"443"; ma=86400, h3-23:"443"; ma=86400
0
[==] TLS 1.3 => Read 819 bytes
```
TLS 1.3 + ESNI HTTPS Connection

.../esni/client $ ./client-macOS -TLSHost cloudflare.com -esni -ESNIServerName cloudflare.com -HostHeader cloudflare.com
[+] Using resolver: https://doh-2.seby.io/dns-query
[+] Successfully queried _ensi TXT record for host: cloudflare.com
[=] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256
[=] ESNI host set to: cloudflare.com
[=] SNI host has been unset
[+] Connecting to https://cloudflare.com:443
[+] TLS handshake complete
[+] Sending GET request: GET / HTTP/1.1
Host: cloudflare.com
User-Agent: ESNI_FRONT_TEST
Accept: */*
Connection: close

[+] GET request sent
[=] Response:
HTTP/1.1 301 Moved Permanently
Date: Tue, 24 Mar 2020 23:41:52 GMT
Transfer-Encoding: chunked
Connection: close
Cache-Control: max-age=3600
Expires: Wed, 25 Mar 2020 00:41:52 GMT
Location: https://www.cloudflare.com/
Set-Cookie: _cf_bm=47e9c445d7f4fdeee6433be8bb7e9e65ed24bdf-1585093312-1800-AeZ9reipfUK2D9ZS6D2s4M17fc/L0/sVCpQGE3nDmlQbyRKS9kuJ23NAXIw3lFaASrle2MNv50UNuRL80VLO84=; path=/; expires=Wed, 25-Mar-20 00:11:52 GMT; domain=.cloudflare.com; HttpOnly; Secure; SameSite=None
Strict-Transport-Security: max-age=15780000; includeSubDomains
Server: cloudflare
CF-RAY: 57943ed30b4f747e-IAH
alt-svc: h3-27:"443"; ma=86400, h3-25:"443"; ma=86400, h3-24:"443"; ma=86400, h3-23:"443"; ma=86400

[=] TLS 1.3 => Read 819 bytes
Hidden Request (no SNI)

```
.../esni/client $ ./client-macos -TLSHost cloudflare.com -esni -ESNIServerName defcon28.hackthis.com
-HostHeader defcon28.hackthis.com
[+] Using resolver: https://dns.rubyfish.cn/dns-query
[+] Successfully queried _ensi TXT record for host: cloudflare.com
[=] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256
[=] ENSI host set to: defcon28.hackthis.com
[=] SNI host has been unset
[+] Connecting to https://cloudflare.com:443
[+] TLS handshake complete
[+] Sending GET request: GET / HTTP/1.1
Host: defcon28.hackthis.com
User-Agent: ENSI_FRONT_TEST
Accept: */*
Connection: close

[+] GET request sent
[=] Reponses:
HTTP/1.1 200 OK
Date: Tue, 24 Mar 2020 23:47:47 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 14
Connection: close
Set-Cookie: __cfduid=d4f30bc8fdaaaa9f40faea6ad776822a71585093667; expires=Thu, 23-Apr-20 23:47:47 GMT;
path=/; domain=.hackthis.com; HttpOnly; SameSite=Lax
CF-Cache-Status: DYNAMIC
Server: cloudflare
CF-RAY: 5794477ebfd7745d-IAD

Hello DEF CON!
[=] TLS 1.3 => Read 488 bytes
```
Fronted Request (with SNI)

As of 2020-08-10 this no longer works due to a change at Cloudflare
So what?

- Any domain protected by Cloudflare is able to arbitrarily front to any IP
  - Target IP can be hosted anywhere
  - DNS must be run via Cloudflare
- Cloudflare Managed DNS is free
- Signup requirements?
  - Email (disposable is ok)
  - Password
  - That's it!
What Can You Hide With?

- Source: Alexa top 100,000 domains
- HTTPS GET request looking for cloudflare cookies, Expect-CT header, or Server
- Results:
  - At least **21% of the top 100,000 domains** are available to front (21,298)
  - A few examples:
    - myshopify.com
    - medium.com
    - discordapp.com (on the PA whitelist)
    - udemy.com
    - zendesk.com
    - coinbase.com (on the PA whitelist)
    - hdfcbank.com
    - mozilla.org (on the PA whitelist)
    - teamviewer.com
    - blackboard.com
    - okta.com
    - bitdefender.com
    - ny.gov
    - mlb.com
    - stanford.edu
    - plex.tv
    - coronavirus.gov.hk
    - So much porn...

[5] List of Domains and Applications Excluded from SSL Decryption
Just doing some Single Sign On...

As of 2020-08-10 this no longer works due to a change at Cloudflare

Hello DEF CON!
SNI Based Web Filters

- Many products only look at SNI
  - ESNI completely ignored
- By preserving the SNI along with ESNI, filters and analytics can be tricked
- Example: Untangle
  - Installed with strict web filter settings
Fooling SNI based Firewalls

As of 2020-08-10 this no longer works due to a change at Cloudflare
### Fooling SNI based Firewalls

<table>
<thead>
<tr>
<th>Domain</th>
<th>[domain]</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu.com</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>firefox.com</td>
<td></td>
<td>91</td>
</tr>
<tr>
<td>nextdns.io</td>
<td></td>
<td>68</td>
</tr>
<tr>
<td>cloudflare-dns.com</td>
<td></td>
<td>63</td>
</tr>
<tr>
<td>digicert.com</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>blahdns.com</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>mozilla.net</td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>pki.goog</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>bypass-untangle.com</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>google.com</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>mozilla.com</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>secureddns.eu</td>
<td></td>
<td>26</td>
</tr>
</tbody>
</table>
HTTPS Decrypting Firewalls

- Seen in enterprise environments
  - Install a root certificate on endpoints
  - Break and re-encrypt traffic that passes through
  - Corporate Man-in-the-Middle
- Allows analysis of full packet data!
- Kazakhstan attempted this nation-wide in July 2019\(^6\)
- Does TLS 1.3/ESNI offer a way around these firewalls?

\(^6\) Kazakhstan government is now intercepting all HTTPS traffic
HTTPS Decrypting Firewalls

- Palo Alto PA-VM 10.0.0
  - Released 2020-06-17
  - Major new feature: TLS 1.3 decryption
  - Default decryption profile does not include TLS 1.3!
    - Allows TLS 1.3 to pass with an error

<table>
<thead>
<tr>
<th>TLS</th>
<th>Client and decrypt profile version mismatch. Supported client version bitmask: 0x40. Supported decrypt profile version bitmask: 0x38.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS1.3</td>
<td>Client and decrypt profile version mismatch. Supported client version bitmask: 0x40. Supported decrypt profile version bitmask: 0x38.</td>
</tr>
<tr>
<td>TLS1.3</td>
<td>Client and decrypt profile version mismatch. Supported client version bitmask: 0x40. Supported decrypt profile version bitmask: 0x38.</td>
</tr>
</tbody>
</table>
[Palo Alto bypass via Mozilla fronting]
What Else?

- Many connections to a fronted site may be suspicious
  - What protocols can we use?
    - HTTP ✅
    - Websockets ✅
    - Arbitrary TCP/UDP via a helper ✅
Fronting Streaming data with Websockets

As of 2020-08-10 this no longer works due to a change at Cloudflare

```
.../esni/client $ ./client-macOS -TLSHost www.okta.com -esni -ESNIServerName defcon28.hackthis.computer
-serverName www.okta.com -preserveSNI -useWebsocket -URL /websocket
[+] Using resolver: https://dns.rubyfish.cn/dns-query
[+]
[+]
[+] Successfully queried _esni TXT record for host: www.okta.com
[=]
[=] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256
[=] ENSI host set to: defcon28.hackthis.computer
[=] SNI host set to: www.okta.com
[+] Connecting to wss://www.okta.com/websocket
[+]
[+] TLS handshake complete
[+]
[+] Websocket Send: Hello DEF CON 28!
[+] Websocket Receive: Hello DEF CON 28!
```
[Cloak Demo]
[Cloak+CobaltStrike Demo]
[DeimosC2 Demo]
What is Blue to Do?
What is Blue to Do?

- Disable TLS 1.3 (25-50% of traffic)
- Block Cloudflare (26 million domains)?
- Block ClientHellos with an encrypted_server_name extension?
  - Technically possible
  - Netsweeper >=6.4.1 (2020-02-25) can block all ESNI traffic
    - Cannot categorize sites
    - All or none
  - No other vendors currently support ESNI blocking as of today
What is Blue to Do?

- Flag on ClientHello packets with both “server_name” and/or “encrypted_server_name”
  - Snort
    - Possible with custom rules?
    - ssl_state:client_hello
    - tls.version can narrow down to 1.3
    - SNI extension type is 0x0000
    - ESNI extension type is 0xffce
  - Securicata
    - Snort with extra features
    - has tls.sni but no tls.esni
  - Zeek
    - The third field of JA3 is TLS extensions
    - Alert on any traffic with 65468 (0xffce)
What is Blue to Do?

- “Good old fashioned police work”
  - Beaconing detection and anomaly network analytics
    - RITA or BeaKer (or fancy AI/ML)
    - How much should a user interact with a site? How often? How much data?
  - JA3 and JA3S mismatches
    - Should svchost.exe have a JA3 fingerprint of Go?
    - Cloak and utls are working to defeat this
  - Well instrumented EDR, application whitelisting, etc
    - Don’t get pwned in the first place!
    - “Server administrators should never allow untrusted code to run on the server”
      - Microsoft
Domain Hiding - Noctilucent - The Future

- Latest advancement in censorship resistant communication
- Useable today
  - Go - drop in replacement
  - Cloak fork - use with anything that can be proxied (CobaltStrike, etc)
- Will be harder to block as TLS 1.3 and ESNI adoption grows
- ESNI RFC is in flux (last updated 2020-06-01) - might break tomorrow
  - Actually called ECH as of May 2020
- Currently relies on a single (massive) CDN
  - Cloudflare please don’t break this
    - 2020-08-10: They broke it! ClientHellos with both ESNI and SNI are rejected
    - ESNI to any Cloudflare IP still works
    - China now blocks any traffic with an ESNI at the great firewall
- Blue Team: Your move!
Special Thanks

- Robin Wood (@digininja) - freelance pen-tester, researcher and developer
- Andy Wang (cbeuw) - developer of Cloak
- Nick Sullivan (@grittygrease) - Head of Research and Cryptography at Cloudflare

References/Resources

- A New Needle and Haystack: Detecting DNS over HTTPS Usage
- Wikipedia: Domain Fronting
- Blocking-resistant communication through domain fronting
- Encrypt it or lose it: how encrypted SNI works
- Encrypted Server Name Indication for TLS 1.3
- Godns - a simple client lib for doing dns over https
- How DNSSEC Works
- SSL Labs - SSL Pulse
- Domain fronting through Cloudflare
- TLS-tris
- RITA (Real Intelligence Threat Analytics)
Questions

Erik Hunstad

https://github.com/SixGenInc/Noctilucent

185 Admiral Cochrane Dr. | Suite 210
Annapolis, MD 21401
erik@sixgen.io
www.sixgen.io
@SixGenInc

Personal Twitter: @badsectorlabs
Personal Blog: blog.badsectorlabs.com