



Update 2020-08-10 - It was fun while it lasted

Cloudflare is now refusing any ClientHello with an ESNI and an SNI, effectively breaking Noctilucent's firewall bypass ability. You can still "hide" a connection by using any Cloudflare DNS hosted domain to connect to while sending your domain in the ESNI, but you can no longer set the unencrypted SNI value to arbitrary domains.

Works:

```
1 $ ./noctilucent-client-macOS -TLSHost www.bitdefender.com -esni -  
   ESNIServerName defcon28.hackthis.computer -HostHeader defcon28.  
   hackthis.computer -serverName www.bitdefender.com  
2 [+] Using resolver: https://mozilla.cloudflare-dns.com/dns-query  
3 [+] Successfully queried _esni TXT record for host: www.bitdefender.com  
4 [=] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256  
5 [=] ESNI host set to: defcon28.hackthis.computer  
6 [=] SNI host has been unset  
7 [+] Connecting to https://www.bitdefender.com:443  
8 [+] TLS handshake complete  
9 [+] Sending GET request: GET / HTTP/1.1  
10 Host: defcon28.hackthis.computer  
11 User-Agent: ESNI_FRONT_TEST
```

```
12 Accept: */*
13 Connection: close
14
15
16 [+] GET request sent
17 [=] Reponse:
18 HTTP/1.1 200 OK
19 Date: Mon, 10 Aug 2020 17:51:07 GMT
20 Content-Type: text/plain; charset=utf-8
21 Content-Length: 14
22 Connection: close
23 Set-Cookie: __cfduid=dae00e2fadc81b286b86cce8f9db26c6d1597081866;
    expires=Wed, 09-Sep-20 17:51:06 GMT; path=/; domain=.hackthis.
    computer; HttpOnly; SameSite=Lax
24 CF-Cache-Status: DYNAMIC
25 cf-request-id: 047b186aa20000fd8613ae42000000001
26 Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.
    com/cdn-cgi/beacon/expect-ct"
27 Server: cloudflare
28 CF-RAY: 5c0b90243993fd86-ORD
29
30 Hello DEF CON!
31 [=] TLS 1.3 => Read 537 bytes
```

No longer works:

```
1 $ ./noctilucent-client-macOS -TLSHost www.bitdefender.com -esni -
    ESNIHostName defcon28.hackthis.computer -HostHeader defcon28.
    hackthis.computer -serverName www.bitdefender.com -preserveSNI
2 [+] Using resolver: https://doh-fi.blahdns.com/dns-query
3 [+] Successfully queried _esni TXT record for host: www.bitdefender.com
4 [=] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256
5 [=] ESNI host set to: defcon28.hackthis.computer
6 [=] SNI host set to: www.bitdefender.com
7 [+] Connecting to https://www.bitdefender.com:443
8 [E] handshake failed: remote error: tls: protocol version not supported
```

For a similar tool that only does ESNI and has a built in SOCK5 proxy (and has cleaner code), check out relaybaton.

Description

This is the code developed and presented as part of the DEF CON 28 (Safe Mode) talk “Domain Fronting is Dead, Long Live Domain Fronting: Using TLS 1.3 to evade censors, bypass network defenses, and blend in with the noise.”

Domain fronting, the technique of circumventing internet censorship and monitoring by obfuscating the domain of an HTTPS connection was killed by major cloud providers in April of 2018. However,

with the arrival of TLS 1.3, new technologies enable a new kind of domain fronting. This time, network monitoring and internet censorship tools are able to be fooled on multiple levels. This talk will give an overview of what domain fronting is, how it used to work, how TLS 1.3 enables a new form of domain fronting, and what it looks like to network monitoring. You can circumvent censorship and monitoring today without modifying your tools using an open source TCP and UDP transport tool (Cloak) that will be released alongside this talk.

Talk: Youtube

Slides are available in the [docs](#) folder.

Compiled test client, test server, and Cloak client binaries are available under “Releases.”

Demos

Noctilucent test client bypassing Palo Alto 10.0 TLS decryption: Youtube

Noctilucent Cloak client: Youtube

Noctilucent Cloak client with CobaltStrike: Youtube

Noctilucent built into DeimosC2 Agent: Youtube

Layout

1	Noctilucent	—
2	Cloak # The Cloak fork	
3	— build # Compiled Cloak binaries	
4	— cmd # Cloak client and server source code	
5	— example_config # Configs for Cloak and Shadowsocks	
6	— internal # Code for Cloak client and server	—
7	DeimosC2 # Modified HTTPS agent source code	—
8	_dev	
9	— GOROOT # Modified Go source tree (tls is placed in here)	—
10	client # Test client source code	
11	— build # Test client binaries	—
12	docs # Slides and other docs	—
13	example-traffic.pcapng # 2 requests made with the Noctilucent test client	
14	— screenshots	—
15	findfronts # Helper to find domains that can be used with Noctilucent	
16	server # Test server (HTTP and websockets)	—
17	tls # Noctilucent tls library (copied to _dev/GOROOT)	—

Use in Go projects

HTTPS

1. Replace the `crpto/tls` import with `github.com/SixGenInc/Noctilucent/tls`.
2. Copy `esni_DoH.go` from `client` into your project (or use your own method for getting the keys).
3. Aquire and parse the ESNI Keys for Cloudflare:

```
1 esniKeysBytes, err := QueryESNIKeysForHostDoH("cloudflare.com", true)
2 if err != nil {
3     fmt.Println("[E] Failed to retrieve ESNI keys for host via DoH: %s"
4         , err)
5 }
6 esnikeys, err := tls.ParseESNIKeys(esniKeysBytes)
7 if err != nil {
8     fmt.Println("[E] Failed to parse ESNI keys: %s", err)
9 }
```

4. Set your `tls.Config` with the new options:

```
1 tlsConfig := &tls.Config{
2     InsecureSkipVerify: true,
3     ClientESNIKeys:     esnikeys,
4     MinVersion:          tls.VersionTLS13, // Force TLS 1.3
5     MaxVersion:          tls.VersionTLS13,
6     ESNIHostName:        actualDomain,
7     PreserveSNI:         true,
8     ServerName:          frontDomain
9 }
```

5. Dial using this `tls.Config`

```
1 httpClient = &http.Client{
2     Transport: &http.Transport{
3         DialTLS: func(network, addr string) (net.Conn, error) {
4             conn, err = tls.Dial("tcp", host+":port", tlsConfig)
5             return conn, err
6         },
7     },
8 }
```

6. Make hidden requests!

```
1 r, err := httpClient.Post(("https://" + actualDomain + ":" + port + URL
    ), "application/json", bytes.NewBuffer(someJSON))
```

Websockets

Websockets are a bit different depending on which websocket library you are using.

If you are currently using golang.org/x/net/websocket:

1. Replace it with github.com/SixGenInc/Noctilucent/websocket.
2. Acquire ESNI keys, parse, and setup a `tls.Config` like HTTPS.
3. Setup and Dial with the websocket library as so:

```
1 config, _ := websocket.NewConfigWithHost("wss://" + frontDomain + URL, "
    https://" + frontDomain, ESNIKeyName)
2 config.TlsConfig = tlsConfig
3 ws, err := websocket.DialConfig(config)
```

4. Send and receive using `ws` as normal, now hidden!

If you are currently using github.com/gorilla/websocket:

1. Acquire ESNI keys, parse, and setup a `tls.Config` like HTTPS.
2. Dial using the TLS config:

```
1 con, err := tls.Dial("tcp", fontDomainAndPort, tlsConfig)
```

3. Setup and force the `Host` header and use the TLS connection with the new websocket client:

```
1 header := http.Header{}
2 header.Add("Host", ESNIKeyName)
3 u, _ := url.Parse("ws://" + fontDomainAndPort)
4 c, _, err := websocket.NewClient(con, u, header, 16480, 16480)
```

4. Use the websocket client as normal, now hidden!

Build from source

Setup - Ubuntu 18.04

Install latest go from golang.org

Note: Developed with Go 1.14

```
1 sudo apt install make git gcc
2 make client # this will take a minute, we are replacing tls in the
   standard library and recompiling it
3 make cloak
```

Setup - macOS 10.15

Install latest go from golang.org

Note: Developed with Go 1.14

```
1 xcode-select --install
2 make client # this will take a minute, we are replacing tls in the
   standard library and recompiling it
3 make cloak
```

Client binaries are available in [client/build](#) and Cloak client binaries are available in [Cloak/build](#).

The modifications made to Cloak can be seen in [Cloak/git_diff.patch](#)

Testing

Test Client

The [server/server.go](#) contains a sample HTTP and websocket server to test against. You can build the server and run it on your own VPS with `go build` inside the server directory. Setup a Cloudflare account and point a domain at your VPS. Ensure that SSL/TLS is set to “Flexible” as the server is not using TLS (or setup a reverse-proxy to handle TLS). Run the server binary (as root, it binds to port 80) on the VPS and replace [defcon28.hackthis.computer](#) in the examples with your domain. You can replace `-serverName` with anything (really, even non-domain strings) and `-TLSHost` with any site that is hosted behind Cloudflare (lots, try [medium.com](#)). Cloudflare has a helpful site for finding frontable domains [here](#), or you can choose any from [findfronts/frontable100k.txt](#).

TLS 1.3

```
Normal HTTPS Connection

.../esni/client $ ./client-macOS -TLSHost 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
[=] Reponse:
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/
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Set-Cookie: __cf_bm=166e3a90f25654e23ea8574aef6e0cac3fc530e4-1585009422-1800-
AQQvEoF8e21eEE7TrueGnj4E3ae0Fv38ZtFA2iBp3Dxl8Vv+5qalJvAEDCy5kqcTMj0VSs2fgaVUrSDCQjZv+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 with ESNI (ECH)

```
TLS 1.3 + ESNI HTTPS Connection

.../esni/client $ ./client-macOS -TLSHost cloudflare.com -esni -ESNIServerName cloudflare.com -H
cloudflare.com
[+] Using resolver: https://doh-2.seby.io/dns-query
[+] Successfully queried _esni 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
[=] Reponse:
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/
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Set-Cookie: __cf_bm=47e9c445d7f4ffdeec643bbe8bb7e0e65ed24bfd-1585093312-1800-
AezT9reipfUk2D9ZS6D2s4ML7fC/Lo/sVCpGQE3nDmLqByRKS9kuJ23NAXIw3lFaASr1e2MNV50UNuRL80VLQ84=; 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-IAD
alt-svc: h3-27=":443"; ma=86400, h3-25=":443"; ma=86400, h3-24=":443"; ma=86400, h3-23=":443"; m

0
[=] TLS 1.3 => Read 819 bytes
```

TLS 1.3 with ESNI (ECH) and Hiding

```
TLS1.3 + ESNI HTTPS Connection

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

[+] GET request sent
[+] Reponse:
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=d4f30bc8fdad9a9f40faea6ad776822a71585093667; expires=Thu, 23-Apr-2020 23:59:59 GMT; path=/; domain=.hackthis.computer; HttpOnly; SameSite=Lax
CF-Cache-Status: DYNAMIC
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 5794477ebfd7745d-IAD

Hello DEF CON!
[+] TLS 1.3 => Read 488 bytes
```

TLS 1.3 with ESNI (ECH), Hiding, and decoy SNI

```
TLS1.3 + ESNI HTTPS Connection

.../esni/client $ ./client-macOS -TLSHost cloudflare.com -esni -ESNIServerName defcon28.hackthis.computer -serverName cloudflare.com -pre
-HostHeader defcon28.hackthis.computer
[+] Using resolver: https://dns.rubyfish.cn/dns-query
[+] Successfully queried _ensl TXT record for host: cloudflare.com
[+] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256
[+] ENSI host set to: defcon28.hackthis.computer
[+] SNI host set to: cloudflare.com
[+] Connecting to https://cloudflare.com:443
[+] TLS handshake complete
[+] Sending GET request: GET / HTTP/1.1
Host: defcon28.hackthis.computer
User-Agent: ESNI_FRONT_TEST
Accept: */*
Connection: close

[+] GET request sent
[+] Reponse:
HTTP/1.1 200 OK
Date: Tue, 24 Mar 2020 23:53:26 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 14
Connection: close
Set-Cookie: __cfduid=db2a920903e1a64ddd9a144770c84265d1585094006; expires=Thu, 23-Apr-2020 23:59:59 GMT; path=/; domain=.hackthis.computer; HttpOnly; SameSite=Lax
CF-Cache-Status: DYNAMIC
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 57944fc418b4ceec-IAD

Hello DEF CON!
[+] TLS 1.3 => Read 488 bytes
```

TLS 1.3 with ESNI (ECH), Hiding, and decoy SNI - Websocket

```
Fronting Streaming data
.../esni/client $ ./client-macOS -TLSHost www.okta.com -
-serverName www.okta.com -preserveSNI -useWebsocket -URI
[+] Using resolver: https://dns.rubyfish.cn/dns-query
[+] Successfully queried _ensl TXT record for host: www.
[+] 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 Client

1. Setup a standard Cloak + Shadowsocks server using this script.
2. Download a shadowsocks-rust binary for your platform.
3. Use the `noctilucent-cloak-client` and `sslocal` to create a local SOCKS proxy that is hidden behind a Cloudflare hosted domain. Example configs are available in `Cloak/example_config` and should be edited to match the values given by the Cloak + Shadowsocks setup script.

Thanks

This project is based on cloudflare's `tls-tris` and inspired by DigiNinja's rough openssl PoC work. It also includes a modified version of ahhh's DNS over HTTPS code, `godns` and of course Cloak.