Akamai

FASTER FORWARD

Passive Fingerprinting of HTTP/2 Clients

Elad Shuster

Security Data Analyst

Threat Research @ Akamai

#> uname -a

Elad Shuster

- Uptime ~ 37 years
- Security Data Analyst @ Akamai Technologies
- Deeply in love with my work!
- CPA(il), MBA
- Enjoying Big-Data, Research, Music, Beers and Single Malt Whiskeys!

Acknowledgments

This research was led by:

Ory Segal – Sr. Director, Threat Research @ Akamai

Aharon Friedman – Sr. Security Researcher @ Akamai

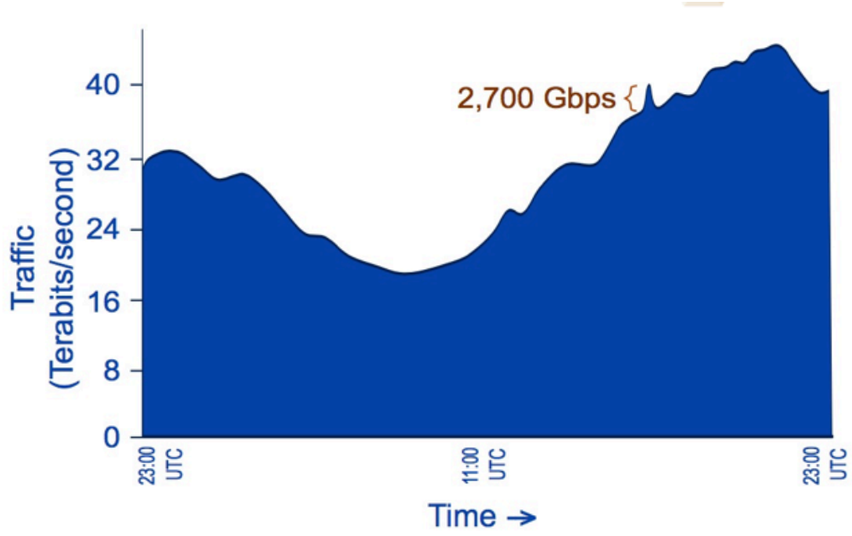
Passive Client Fingerprinting

- Fingerprinting clients NOT end users!
- Passive collection of attributes
- May be collected from:
  - **Transport layer** (e.g. TCP properties)
  - **Session layer** (e.g. TLS capabilities)
  - **Application layer** (e.g. HTTP implementation characteristics)
- Deduce about OS (type and version), Running Software, up-time, etc…

Starting out with the research…

Akamai By The Numbers

- 233,000 servers in over 130 countries within 1,600 networks
- Regularly serving 35+ Million HTTP reqs/sec - 3 Trillion per day
- Daily Web traffic reaching more than 30 Terabits per second
- Peak traffic over 46 Terabits/sec
- ~ 2 Hexabytes of storage
- 20 TB daily attack data



Data Corpus

- 10 million HTTP/2 connections
- Over 40,000 unique user agents
- Hundreds of implementations
- The data set for the research was anonymized

HTTP/2 – 101 - Overview

- Addresses the following performance issues in HTTP/1.1:
  - **Request Concurrency** - requires multiple TCP connections
  - **Header Compression** - repetitive and verbose
  - No Concept of **Server Push**
- **2012 :** Work on SPDY began
- **May 2015**: RFC 7540 (HTTP/2) and RFC 7541 (HPACK)

HTTP/2 – 101 - Overview

- Enter HTTP/2…
  - Single TCP connection
  - Interleaving of requests
  - Header Compression via HPACK
  - Introducing Server Push

https://http2.akamai.com/demo

HTTP/2 – 101 - Overview

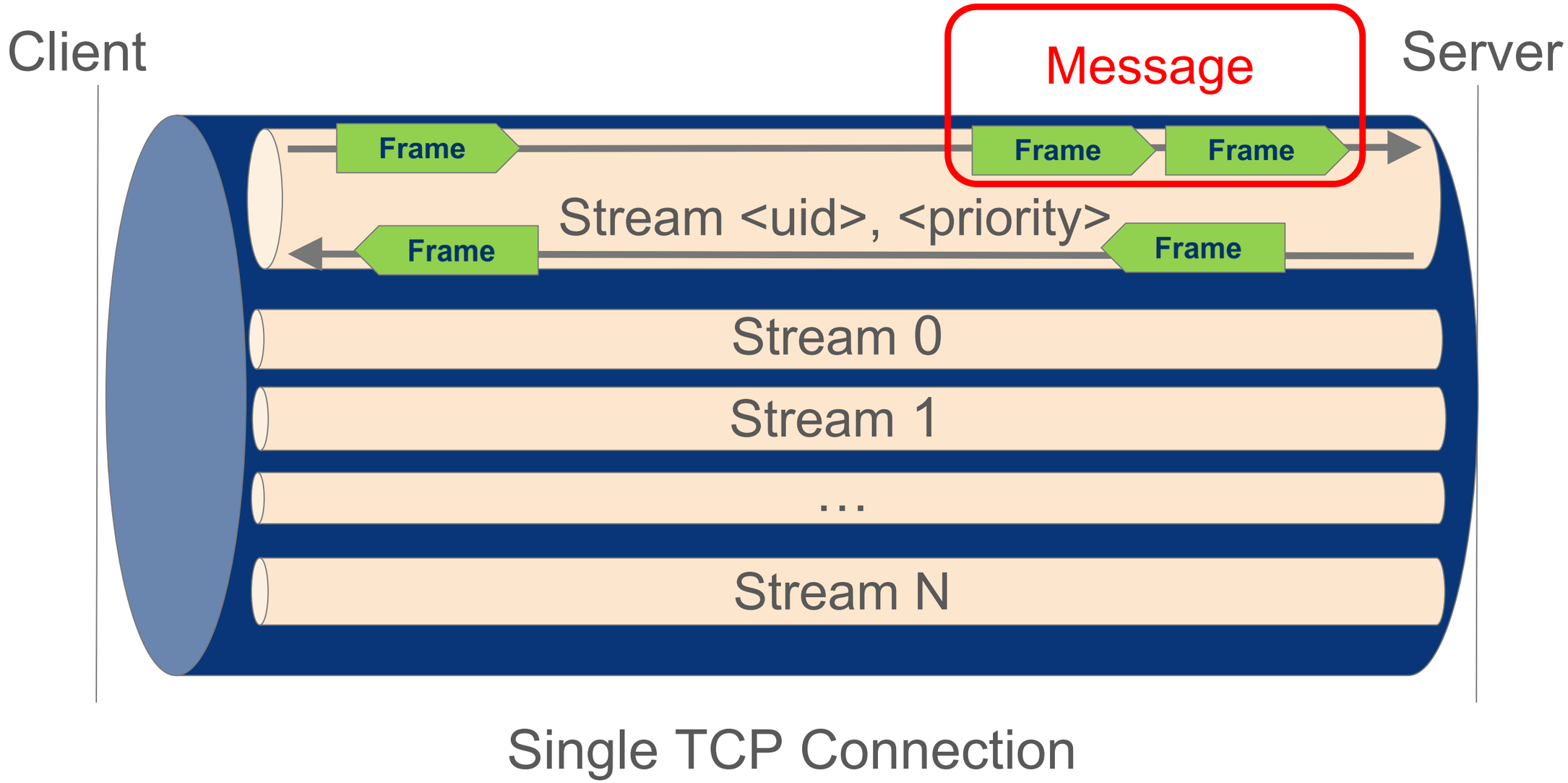
- Protocol negotiation
  - Over TLS* in the ALPN (Application Level Protocol Negotiation) extension
  - Over HTTP – using the “Upgrade: “ header

```
GET / HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS payload>
```

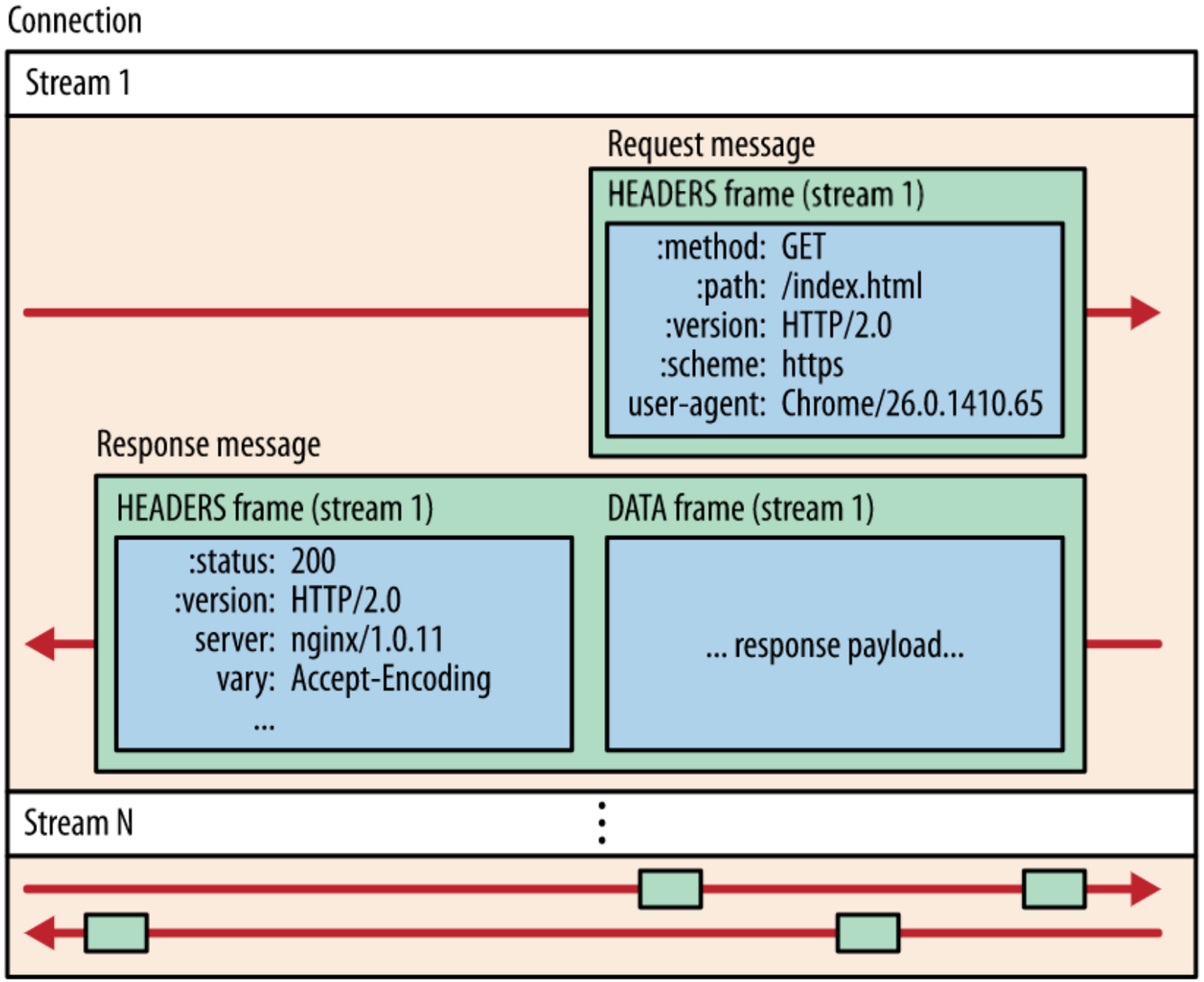
- Technically, the RFC does not mandate the use of TLS

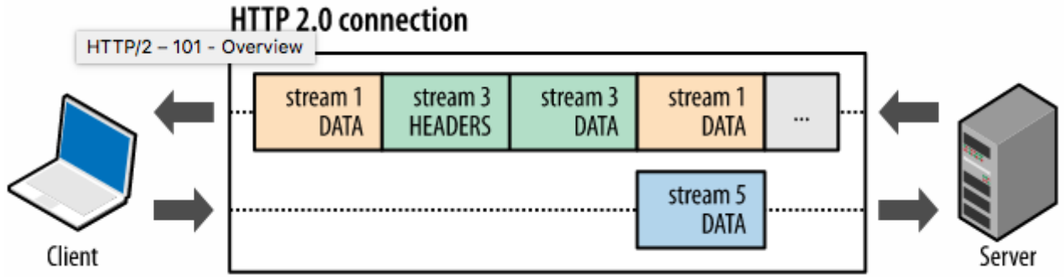
  “… The string "h2c" identifies the protocol where HTTP/2 is run over cleartext TCP. …”

HTTP/2 – 101 - Overview



HTTP/2 – 101 - Overview





HTTP/2 – 101 - Overview

- **Stream** - bidirectional flow of frames within an established connection

  - Assigned with a **Unique ID** and a **Priority**

- **Message** - sequence of frames that map to a logical request or response

- **Frame** - smallest unit of communication in HTTP/2 – 10 Types:

  - SETTINGS
  - HEADERS
  - DATA
  - WINDOW UPDATE
  - PRIORITY
  - PUSH_PROMISE
  - PING
  - GOAWAY
  - RST_STREAM
  - CONTINUATION

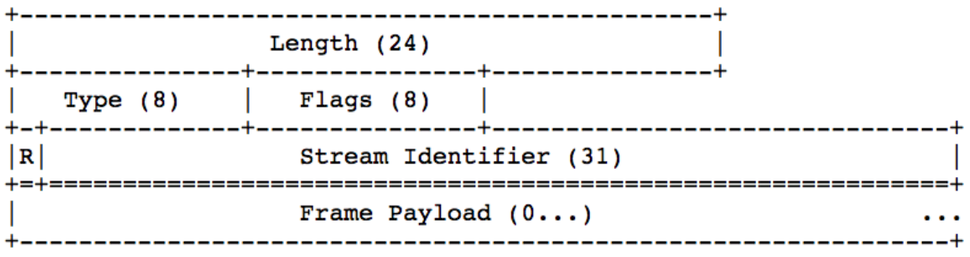


Figure 1: Frame Layout

HTTP/2 – 101 - Overview

Frame Type

Stream ID

```
[1320512.528] send SETTINGS frame <length=6, flags=0x00, stream_id=0>
 (niv=1)
 [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
[1320512.530] recv SETTINGS frame <length=18, flags=0x00, stream_id=0>
 (niv=3)
 [SETTINGS_HEADER_TABLE_SIZE(0x01):65536]
 [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):1000]
 [SETTINGS_INITIAL_WINDOW_SIZE(0x04):6291456]
[1320512.531] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=0>
 (window_size_increment=15663105)
[1320512.532] recv (stream_id=1) :method: GET
[1320512.532] recv (stream_id=1) :authority: www.f
[1320512.532] recv (stream_id=1) :scheme: https
[1320512.533] recv (stream_id=1) :path: /
[1320512.533] recv (stream_id=1) user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.36
[1320512.533] recv (stream_id=1) upgrade-insecure-requests: 1
[1320512.534] recv (stream_id=1) accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
[1320512.534] recv (stream_id=1) accept-encoding: gzip, deflate, br
[1320512.534] recv (stream_id=1) accept-language: en-US,en;q=0.8,he;q=0.6
[1320512.534] recv HEADERS frame <length=239, flags=0x25, stream_id=1>
 ; END_STREAM | END_HEADERS | PRIORITY
 (padlen=0, dep_stream_id=0, weight=256, exclusive=1)
 ; Open new stream
```

HTTP/2 – Client Fingerprinting

- Searching for:
  - flows or messages in the protocol
  - different clients expose a **consistent unique behavior**
- Proposed Fingerprint Based On:
  - Setting Parameters in SETTINGS frame
  - WINDOW_UPDATE increment size
  - PRIORITY attributes
  - Pseudo-Header Fields Order

HTTP/2 – SETTINGS FRAME

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

- Conveys configuration parameters
- MUST be sent by both endpoints at the start of a connection
- The stream identifier for a SETTINGS frame MUST be zero

```
[1320512.528] send SETTINGS frame <length=6, flags=0x00, stream_id=0>
 (niv=1)
 [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
[1320512.530] recv SETTINGS frame <length=18, flags=0x00, stream_id=0>
 (niv=3)
 [SETTINGS_HEADER_TABLE_SIZE(0x01):65536]
 [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):1000]
 [SETTINGS_INITIAL_WINDOW_SIZE(0x04):6291456]
```

HTTP/2 – SETTINGS FRAME

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- er Fields Order

Parameter Name	Scope
SETTINGS_HEADER_TABLE_SIZE (0x1)	Allows the sender to inform the remote endpoint of the maximum size of the header compression table used to decode header blocks, in octets.
SETTINGS_ENABLE_PUSH (0x2)	This setting can be used to disable server push (Section 8.2).
SETTINGS_MAX_CONCURRENT_STREAMS (0x3)	Indicates the maximum number of concurrent streams that the sender will allow.
SETTINGS_INITIAL_WINDOW_SIZE (0x4)	Indicates the sender's initial window size (in octets) for stream-level flow control. The initial value is 216-1 (65,535) octets.
SETTINGS_MAX_FRAME_SIZE (0x5)	Indicates the size of the largest frame payload that the sender is willing to receive, in octets.
SETTINGS_MAX_HEADER_LIST_SIZE (0x6)	This advisory setting informs a peer of the maximum size of header list that the sender is prepared to accept, in octets.

HTTP/2 – SETTINGS FRAME

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- eader Fields Order

User-Agent	MAX CONCURRENT STREAMS	HEADER TABLE SIZE	MAX HEADER LIST SIZE	MAX FRAME SIZE	INITIAL WINDOW SIZE	ENABLE PUSH
Mozilla/5.0 (Android 6.0; Mobile; rv:52.0) Gecko/52.0 Firefox/52.0	[]	['4096']	[]	['16384']	['32768']	[]
Mozilla/5.0 (Android 6.0.1; Tablet; rv:47.0) Gecko/47.0 Firefox/47.0	[]	[]	[]	['16384']	['32768']	[]
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; McAfee)	['1024']	[]	[]	[]	['10485760']	[]
Mozilla/5.0 (Linux; Android 7.1; Pixel XL...	['100']	['4096']	['131072']	['16384']	['163840']	['0']

HTTP/2 – WINDOW_UPDATE

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

- Implements flow control
- New streams are created with an initial flow-control window size of 65,535 octets
- WINDOW_UPDATE frame is used to adjust the initial window size

```
[1323385.669] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=0>
 (window_size_increment=12517377)
```

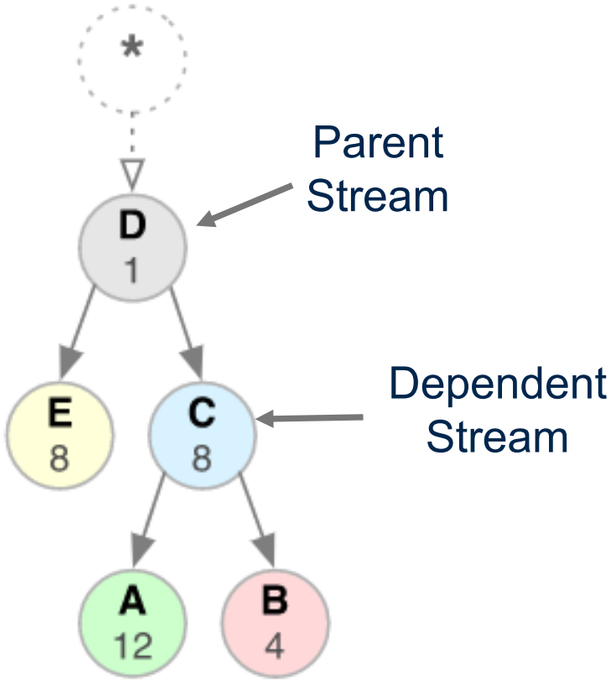
```
[1323385.669] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=13>
 (window_size_increment=12451840)
```

```
[1323386.100] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=17>
 (window_size_increment=12451840)
```

HTTP/2 – PRIORITY Frame

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

- Sets a priority of any given stream
- Express preference of resources allocation
- Defines Dependencies
- Some clients (e.g. Firefox) set the PRIORITY for reserved stream at the beginning of each connection
- No guarantees!



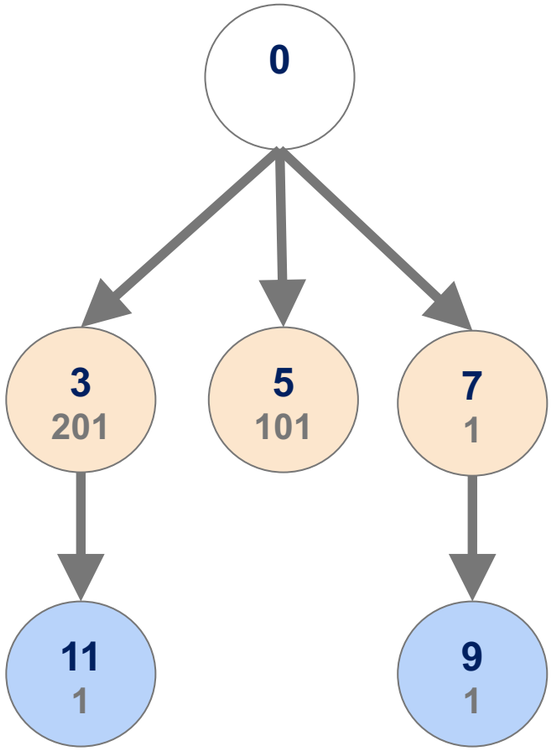
HTTP/2 – PRIORITY Frame

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

```
[1323385.669] recv PRIORITY frame <length=5, flags=0x00, stream_id=3>
 (dep_stream_id=0, weight=201, exclusive=0)
[1323385.669] recv PRIORITY frame <length=5, flags=0x00, stream_id=5>
 (dep_stream_id=0, weight=101, exclusive=0)
[1323385.669] recv PRIORITY frame <length=5, flags=0x00, stream_id=7>
 (dep_stream_id=0, weight=1, exclusive=0)
[1323385.669] recv PRIORITY frame <length=5, flags=0x00, stream_id=9>
 (dep_stream_id=7, weight=1, exclusive=0)
[1323385.669] recv PRIORITY frame <length=5, flags=0x00, stream_id=11>
 (dep_stream_id=3, weight=1, exclusive=0)
```

Stream ID	Exclusivity Bit	Dependent Stream ID	Weight
3	0	0	201
5	0	0	101
7	0	0	1
9	0	7	1
11	0	3	1

Firefox/54.0



HTTP/2 Fingerprinting – Let's Recap…

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

- Three Fingerprint Elements:
  - SETTINGS
  - WINDOW_UPDATE
  - PRIORITY

- Proposed Structure:

SETTINGS[;] | WINDOW_UPDATE | PRIORITY[,]

HTTP/2 Fingerprinting – Let's Recap…

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

User-Agent: okhttp/3.6.0

HTTP/2 fingerprint:
4:16777216|16711681|0

User-Agent: Go-http-client/2.0

HTTP/2 fingerprint:
2:0;4:4194304;6:10485760|1073741824|0

User-Agent: Curl/7.54.0

HTTP/2 fingerprint:
3:100;4:1073741824;2:0|1073676289|0

User-Agent: nghttp2/1.22.0

HTTP/2 fingerprint:
3:100;4:65535|00|3:0:0:201,5:0:0:101,7:0:0:1,9:0:7:1,11:0:3:1

Parameter Name
SETTINGS_HEADER_TABLE_SIZE (0x1)
SETTINGS_ENABLE_PUSH (0x2)
SETTINGS_MAX_CONCURRENT_STREAMS (0x3)
SETTINGS_INITIAL_WINDOW_SIZE (0x4)
SETTINGS_MAX_FRAME_SIZE (0x5)
SETTINGS_MAX_HEADER_LIST_SIZE (0x6)

HTTP/2 Fingerprinting – Let's Recap…

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

Example 1: Chrome Browser on Mac OS X

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.96 Safari/537.36

HTTP/2 fingerprint:

1:65536;3:1000;4:6291456|15663105|0

Example 2: Chrome Browser on Windows 10 (Identical to Chrome in Example #1)

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.96 Safari/537.36

HTTP/2 fingerprint:

1:65536;3:1000;4:6291456|15663105|0

Example 3: Microsoft Edge Browser on Windows 10

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393

HTTP/2 fingerprint:

3:1024;4:10485760|10420225|0


YOU TRIED YOUR BEST?
NOT GOOD ENOUGH

HTTP/2 – Pseudo-Header Fields Order

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

- Pseudo-header fields are not HTTP header fields
- MUST be defined in the RFC
- Request or response with undefined header – considered malformed!
- Request Pseudo-Header Fields:
  - :method
  - :scheme
  - :authority
  - :path

HTTP request	Header Frame
GET /index.html HTTP/1.1 Host: example.com Accept: text/html	:method: GET :scheme: http :path: /index.html :authority: example.com accept: text/html

HTTP/2 – Pseudo-Header Fields Order

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

Client / Implementation	Pseudo Headers Name Order
Google Chrome (58.0.3029.110 on Mac OS X)	`:method, :authority, :scheme, :path`
Firefox v53.0 (Mac OS X)	`:method, :path, :authority, :scheme`
Safari v10.1 (Mac OS X)	`:method, :scheme, :path, :authority`
Curl v7.54.0 (Mac OS X)	`:method, :path, :scheme, :authority`
Go-http-client v2.0	`:authority, :method, :path, :scheme`
Jetty HTTP2 Client v9.3.4.v20151007	`:scheme, :method, :authority, :path`

HTTP/2 Fingerprinting – Proposed Fingerprint

- Setting Parameters in SETTINGS frame
- WINDOW_UPDATE increment size
- PRIORITY attributes
- Pseudo-Header Fields Order

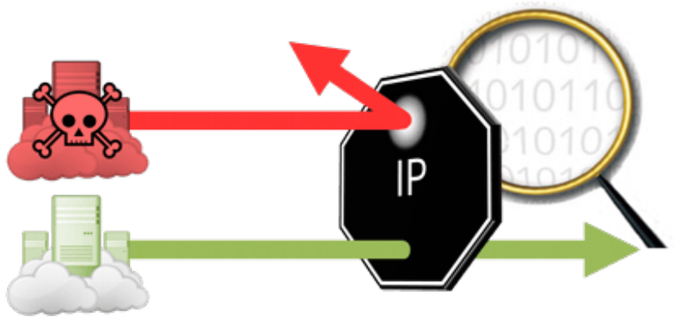
SETTINGS[;] | WINDOW_UPDATE | PRIORITY[,] | PSH-Order

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:53.0) Gecko/20100101 Firefox/53.0

HTTP/2 fingerprint:

```
1:65536;4:131072;5:16384|12517377|3:0:0:201,5:0:0:101,7:0:0:1,9:0:7:1,11:0:3:1|m,p,a,s
```

HTTP/2 Fingerprinting – Use Cases

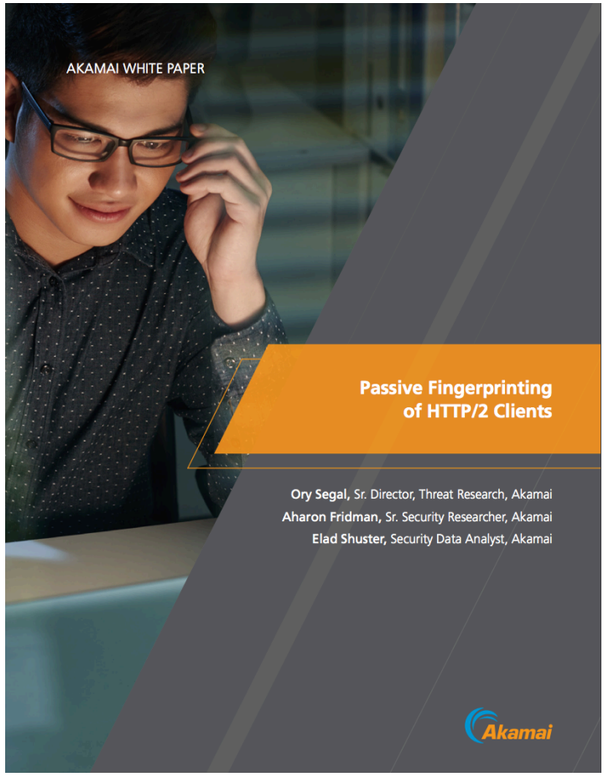


- Positive Security
- Spoofed User-Agent Detection
  - Headless Browsers
  - Crawlers / Other Bots
- Anonymous Proxy / VPN Detection

HTTP/2 Attack Landscape

HTTP/2 Fingerprinting

This presentation has been based on the following white paper:



http://akamai.me/2sv42WP

Questions? Suggestions?

Thank You!

Elad Shuster
eshuster@akamai.com