Passive Fingerprinting of HTTP/2 Clients

Elad Shuster
Security Data Analyst
Threat Research @ Akamai
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:
  o **Transport layer** (e.g. TCP properties)
  o **Session layer** (e.g. TLS capabilities)
  o **Application layer** (e.g. HTTP implementation characteristics)

• Deduce about OS (type and version), Running Software, up-time, etc…
Starting out with the research…

IF ONLY I COULD GET THE DAMN DATA!
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

    ```plaintext
    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

Client

Single TCP Connection

Server

Frame

Stream <uid>, <priority>

Frame

Frame

Frame

Frame

Message

Stream 0

Stream 1

…

Stream N
HTTP/2 – 101 - Overview

Connection

Stream 1
Request message
HEADERS frame (stream 1)
  :method: GET
  :path: /index.html
  :version: HTTP/2.0
  :scheme: https
  user-agent: Chrome/26.0.1410.65

Response message
HEADERS frame (stream 1)
  :status: 200
  :version: HTTP/2.0
  server: nginx/1.0.11
  vary: Accept-Encoding
  ...
DATA frame (stream 1)
  ... response payload...

Stream N

HTTP 2.0 connection

©2015 AKAMAI | FASTER FORWARD™
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
HTTP/2 – 101 - Overview

Frame Type

Stream ID
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

• 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

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETTINGS_HEADER_TABLE_SIZE (0x1)</td>
<td>Allows the sender to inform the remote endpoint of the maximum size of the header compression table used to decode header blocks, in octets.</td>
</tr>
<tr>
<td>SETTINGS_ENABLE_PUSH (0x2)</td>
<td>This setting can be used to disable server push (Section 8.2).</td>
</tr>
<tr>
<td>SETTINGS_MAX_CONCURRENT_STREAMS (0x3)</td>
<td>Indicates the maximum number of concurrent streams that the sender will allow.</td>
</tr>
<tr>
<td>SETTINGS_INITIAL_WINDOW_SIZE (0x4)</td>
<td>Indicates the sender’s initial window size (in octets) for stream-level flow control. The initial value is 216-1 (65,535) octets.</td>
</tr>
<tr>
<td>SETTINGS_MAX_FRAME_SIZE (0x5)</td>
<td>Indicates the size of the largest frame payload that the sender is willing to receive, in octets.</td>
</tr>
<tr>
<td>SETTINGS_MAX_HEADER_LIST_SIZE (0x6)</td>
<td>This advisory setting informs a peer of the maximum size of header list that the sender is prepared to accept, in octets.</td>
</tr>
</tbody>
</table>
## HTTP/2 – SETTINGS FRAME

<table>
<thead>
<tr>
<th>User-Agent</th>
<th>MAX CONCURRENT STREAMS</th>
<th>HEADER TABLE SIZE</th>
<th>MAX HEADER LIST SIZE</th>
<th>MAX FRAME SIZE</th>
<th>INITIAL WINDOW SIZE</th>
<th>ENABLE PUSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla/5.0 (Android 6.0; Mobile; rv:52.0) Gecko/52.0 Firefox/52.0</td>
<td>()</td>
<td>'4096'</td>
<td>()</td>
<td>'16384'</td>
<td>'32768'</td>
<td>()</td>
</tr>
<tr>
<td>Mozilla/5.0 (Android 6.0.1; Tablet; rv:47.0) Gecko/47.0 Firefox/47.0</td>
<td>()</td>
<td>()</td>
<td>()</td>
<td>'16384'</td>
<td>'32768'</td>
<td>()</td>
</tr>
<tr>
<td>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)</td>
<td>['1024']</td>
<td>()</td>
<td>()</td>
<td>()</td>
<td>'10485760'</td>
<td>()</td>
</tr>
<tr>
<td>Mozilla/5.0 (Linux; Android 7.1; Pixel XL...)</td>
<td>['100']</td>
<td>'4096'</td>
<td>['131072']</td>
<td>'16384'</td>
<td>'163840'</td>
<td>['0']</td>
</tr>
</tbody>
</table>
HTTP/2 – WINDOW_UPDATE

• 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

- 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

```
[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)
```

### Table: PRIORITY Frame Details

<table>
<thead>
<tr>
<th>Stream ID</th>
<th>Exclusivity Bit</th>
<th>Dependent Stream ID</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>201</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
HTTP/2 Fingerprinting – Let’s Recap…

• Three Fingerprint Elements:
  • SETTINGS
  • WINDOW_UPDATE
  • PRIORITY

• Proposed Structure:
  SETTINGS[;] | WINDOW_UPDATE | PRIORITY[,]
HTTP/2 Fingerprinting – Let’s Recap…

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

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

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…

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

- 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

<table>
<thead>
<tr>
<th>HTTP request</th>
<th>Header Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /index.html HTTP/1.1</td>
<td>:method: GET</td>
</tr>
<tr>
<td>Host: example.com</td>
<td>:scheme: http</td>
</tr>
<tr>
<td>Accept: text/html</td>
<td>:path: /index.html</td>
</tr>
<tr>
<td></td>
<td>:authority: example.com</td>
</tr>
<tr>
<td></td>
<td>accept: text/html</td>
</tr>
</tbody>
</table>
# HTTP/2 – Pseudo-Header Fields Order

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