PASSIVE FINGERPRINTING OF HTTP/2 CLIENTS

Elad Shuster
Senior Security Researcher, Team Lead
Akamai
Before we begin....
Agenda

- Usage Statistics on Akamai’s Platform
- HTTP/2 Overview
- Passive Client Fingerprinting
- HTTP/2 Fingerprinting and it’s Use Cases
- HTTP/2 Threat landscape
Uptime ~ 37 years

Threat Research Team @ Akamai Technologies

Enjoying Big-Data

Love Single Malt Whiskeys!

CPA(il), MBA
Acknowledgments

This research was led by:

Ory Segal
Sr. Director Threat Research
Akamai

Aharon Friedman
Sr. Security Researcher
Akamai

http://akamai.me/2qW1qON
DATA COLLECTION
The Intelligent Platform

- 220,000+ Edge Servers
- 3,315+ Locations
- 1200+ Cities
- 129 Countries
- 1,227+ Networks
- 60 Tbps at last peak

The Data

- 3 trillion hits per day
- 1 Billion unique IPs seen quarterly
- 13+ trillion log lines per day
- 260+ TB of compressed daily logs

15 - 30% of all web traffic
HTTP/2 Usage Statistics

1 Billion Daily requests

10% Of Total Traffic

15.7K Hosts

413.4M Login requests

27.2M Unique IP Addresses

675.3K User Agents
HTTP/2 OVERVIEW
HTTP 1.x

GET /index.html HTTP/1.1
Host: www.fdsa.co
Connection: keep-alive
User-Agent: Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,he;q=0.8
HTTP/2 Overview

- Based on the SPDY Protocol (developed by Google)
- Published during 2015:
  - RFC 7540 Hypertext Transfer Protocol Version 2 (HTTP/2)
  - RFC 7541 HPACK: Header Compression for HTTP/2
- Binary Protocol
- Addresses (performance) challenges in HTTP/1.x
HTTP/1.x Challenges

Concurrency

Header Compression

Passive Server
Enter HTTP/2...

- **Concurrency**: Allows interleaving of request and response messages on the same TCP connection.
- **Compression**: Uses an efficient coding for HTTP header fields, as well as header compression (HPACK).
- **Server push**: Adds a new interaction mode whereby a server can push responses to a client, if it thinks the client will need them.
HTTP/2 Connection

Stream
an independent, bidirectional sequence of frames exchanged between the client and server
HTTP/2 Key Elements

Frame
smallest unit of communication in HTTP/2

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 Types

- SETTINGS
- HEADERS
- DATA
- PRIORITY
- WINDOW UPDATE
- PING
- GO AWAY
- CONTINUATION
- PUSH PROMISE
- RST STREAM
Frame Structure

```
+-----------------+             +-----------------+
| Length (24)     |             | Type (8)        |
+-----------------+             +-----------------+             +-----------------+
|                  |             | Flags (8)       |
+-----------------+             +-----------------+             +-----------------+  
| R               |             | Stream Identifier (31) |
+-----------------+             +-----------------+             +-----------------+  
|                  |             | Frame Payload (0...) |
+-----------------+             +-----------------+             +-----------------+  
```

Figure 1: Frame Layout
Frame Structure - Example

Length (24 bit) = XXXXX

Type = 0x1
    HEADERS_FRAME

Flags = 0x25
    END_HEADERS (0x4),
    END_STREAM (0x1),
    PRIORITY (0x20)

Stream Identifier (Stream ID = 73)

Pad Length = 0

E = 1

Stream Dependency = 0

Weight = 220

.method: GET
:authority: http2.akamai.com
:scheme: https
:path: /resources/h2.css
user-agent: Mozilla/5.0 (......) Chrome/62.0.3202.75
Single TCP Connection
HTTP/2 is the future of the Web, and it is here!

Your browser supports HTTP/2!

This is a demo of HTTP/2’s impact on your download of many small tiles making up the Akamai Spinning Globe.
Keep in mind...

- HTTP/2 is binary (you can’t use netcat to draft traffic)
- HTTP/2 implementations use TLS
- Most intercepting proxies (e.g. Burp) don’t support H2
Inspection Tools
Server Side

Web server debug logs
Client Side
Let’s get familiarized with the logs....
Stream 0

<table>
<thead>
<tr>
<th>Source</th>
<th>Frame Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>SETTINGS</td>
<td>[SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]</td>
</tr>
</tbody>
</table>
## Stream 0

<table>
<thead>
<tr>
<th>Source</th>
<th>Frame Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>SETTINGS</td>
<td>[SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]</td>
</tr>
<tr>
<td>Client</td>
<td>SETTINGS</td>
<td>[SETTINGS_HEADER_TABLE_SIZE(0x01):65536]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[SETTINGS_INITIAL_WINDOW_SIZE(0x04):131072]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[SETTINGS_MAX_FRAME_SIZE(0x05):16384]</td>
</tr>
</tbody>
</table>
Stream 15

<table>
<thead>
<tr>
<th>Source</th>
<th>Frame Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>HEADERS</td>
<td>&lt;Flags, Headers&gt;</td>
</tr>
</tbody>
</table>
Stream 15

<table>
<thead>
<tr>
<th>Source</th>
<th>Frame Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>HEADERS</td>
<td>&lt;Flags, Headers&gt;</td>
</tr>
<tr>
<td>Client</td>
<td>WINDOW_UPDATE</td>
<td>(window_size_increment=12451840)</td>
</tr>
</tbody>
</table>
Stream 15

<table>
<thead>
<tr>
<th>Source</th>
<th>Frame Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>HEADERS</td>
<td>&lt;Flags, Headers&gt;</td>
</tr>
<tr>
<td>Client</td>
<td>WINDOW_UPDATE</td>
<td>(window_size_increment=12451840)</td>
</tr>
<tr>
<td>Server</td>
<td>HEADERS</td>
<td>&lt;Flags, Headers&gt;</td>
</tr>
</tbody>
</table>
HTTP/2 Conversation

- HTTP/2 is negotiated via the TLS ALPN extension (Application Level Protocol Negotiation)

```plaintext
[ALPN] client offers:
  * h2
  * http/1.1

[ALPN] client offers:
  * h2
  * http/1.1

SSL/TLS handshake completed
The negotiated protocol: h2
```
HTTP/2 Conversation

- Connection Established
- Settings Exchanged
- Request
- Response

- SETTINGS – Always Stream ID = 0
HTTP/2 Conversation

- Client Send a HEADERS frame
- Stream ID = 1
HTTP/2 Conversation

Connection Established

Settings Exchanged

Request

Response

- Server Responds with a Message
- Message = HEADERS and DATA frames
- Stream ID = 1
Passive Client Fingerprinting
Passive collection of attributes that might expose consistent unique behavior

Fingerprinting software **clients** NOT end users

Deduce about up-time, OS (type and version), Running Software, etc…

**Passive Client Fingerprinting**
- Observe client’s behaviors while establishing a connection

- Attributes sent by the client that might expose consistent unique behavior:
  - Initial connection settings
  - Initial flow control settings
  - Prioritization
  - (Pseudo) Header Order

HTTP/2 Passive Client Fingerprinting
Proposed Fingerprint Elements

- SETTINGS
- WINDOW UPDATE
- DATA
- PRIORITY
- HEADERS
- PING
- GO AWAY
- CONTINUATION
- SETTINGS[:]
- WINDOW UPDATE
- PRIORITY
- PSH-Order

Proposed Fingerprint Elements
Proposed Fingerprint

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

- SETTINGS frame Conveys configuration parameters
- SETTINGS MUST be sent by BOTH endpoints at the start of a connection
- Parameter default values vary between implementations
- Stream identifier for a SETTINGS frame MUST be zero
## SETTINGS PARAMETERS

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETTINGS_HEADER_TABLE_SIZE (0x1) (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) (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 2^16-1 (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>
Firefox/55.0 - Mac OS X 10.11.6

[44.130] recv SETTINGS frame <length=18, flags=0x00, stream_id=0>
(niv=3)

[SETTINGS_HEADER_TABLE_SIZE(0x01):65536]
[SETTINGS_INITIAL_WINDOW_SIZE(0x04):131072]
[SETTINGS_MAX_FRAME_SIZE(0x05):16384]
recv `SETTINGS` frame <length=12, flags=0x00, stream_id=0>

(niv=2)

[SETTINGS_ENABLE_PUSH(0x02):0]

[SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
EDGE 15.15063 – Windows 10

```
[28297.704] recv SETTINGS frame <length=12, flags=0x00, stream_id=0>
  (niv=2)
  [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):1024]
  [SETTINGS_INITIAL_WINDOW_SIZE(0x04):10485760]
```

Chrome 60 – Android 8.0.0 Pixel XL

```
[30336.100] 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]
```
<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>
Proposed Fingerprint

EDGE 15.15063 – Windows 10

[28297.704] recv SETTINGS frame <length=12, flags=0x00, stream_id=0>
  (niv=2)
  [SETTINGS_MAX_CONCURRENT_STREAMS(0x03) 1024]
  [SETTINGS_INITIAL_WINDOW_SIZE(0x04): 10485760]

[ 3:1024 ; ]
Proposed Fingerprint

EDGE 15.15063 – Windows 10

```
[28297.704] recv SETTINGS frame <length=12, flags=0x00, stream_id=0>
  (niv=2)
  [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):1024]
  [SETTINGS_INITIAL_WINDOW_SIZE(0x04):10485760]
```

[ 3:1024 ; 4:10485760 ]
Proposed Fingerprint

Firefox/55.0 - Mac OS X 10.11.6

```
[44.130] recv SETTINGS frame <length=18, flags=0x00, stream_id=0>
(niv=3)

[SETTINGS_HEADER_TABLE_SIZE(0x01):65536]
[SETTINGS_INITIAL_WINDOW_SIZE(0x04):131072]
[SETTINGS_MAX_FRAME_SIZE(0x05):16384]
```

[1:65536 ; 4:131072 ; 5:16384 ]
Proposed Fingerprint

- Flow control element
- Window size can be set for entire connection or per stream
  - Connection – Initial size can be set in SETTINGS
  - RFC set default window sizes if not specified
Proposed Fingerprint

Chrome 60 – Android 8.0.0 Pixel XL

[35556.053] recv SETTINGS frame <length=18, flags=0x00, stream_id=0>
(niv-3)

[35556.053] [SETTINGS_HEADER_TABLE_SIZE(0x01):65536]
[SETTINGS_MAX_CONCURRENT_STREAMS(0x03):1000]
[SETTINGS_INITIAL_WINDOW_SIZE(0x04):6291456]

[35556.054] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=0>
(window_size_increment=15663105)

[1:65536; 4:131072; 5:16384|15663105]
Proposed Fingerprint

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

Chrome 60 – Android 8.0.0 Pixel XL

```
[35556.053] 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]
[35556.054] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=0>
  (window_size_increment=15663105)
```

```
[1:65536 ; 4:131072 ; 5:16384] [15663105]
```

* If frame is not sent – use 0 instead
Proposed Fingerprint

- Set stream dependencies and priorities
- Priority is set by assigning weights to streams
- Weights express preference of resources allocation
- No guarantees

“only a suggestion”
Proposed Fingerprint

- Used by some at the beginning of each connection

- Each frame has **three fields**:
  - Weight
  - Stream Dependency
  - Exclusivity Bit
Proposed Fingerprint

Firefox/54.0

Collect dependency, weight, exclusivity

```plaintext
[39000.283] recv PRIORITY frame <length=5, flags=0x00, stream_id=3>
  (dep_stream_id=0, weight=201, exclusive=0)
[39000.283] recv PRIORITY frame <length=5, flags=0x00, stream_id=5>
  (dep_stream_id=0, weight=101, exclusive=0)
[39000.283] recv PRIORITY frame <length=5, flags=0x00, stream_id=7>
  (dep_stream_id=0, weight=1, exclusive=0)
[39000.283] recv PRIORITY frame <length=5, flags=0x00, stream_id=9>
  (dep_stream_id=7, weight=1, exclusive=0)
[39000.284] recv PRIORITY frame <length=5, flags=0x00, stream_id=11>
  (dep_stream_id=3, weight=1, exclusive=0)
[39000.284] recv PRIORITY frame <length=5, flags=0x00, stream_id=13>
  (dep_stream_id=0, weight=241, exclusive=0)
```
Proposed Fingerprint

Firefox/54.0

Http2Session.cpp

```
// The Hello is comprised of
// 1] 24 octets of magic, which are designed to
// flush out silent but broken intermediaries
// 2] a settings frame which sets a small flow control window for pushes
// 3] a window update frame which creates a large session flow control window
// 4] 5 priority frames for streams which will never be opened with headers
//    these streams (3, 5, 7, 9, b) build a dependency tree that all other
//    streams will be direct leaves of.
```
# Proposed Fingerprint

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

<table>
<thead>
<tr>
<th>User-Agent</th>
<th>SETTINGS</th>
<th>WINDOW UPDATE</th>
<th>PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome 58.0 Mac OS X</td>
<td>1:65536; 3:1000; 4:6291456</td>
<td>15663105</td>
<td>0</td>
</tr>
<tr>
<td>okhttp/3.6.0</td>
<td>4:16777216</td>
<td>16711681</td>
<td>0</td>
</tr>
<tr>
<td>curl/7.54.0</td>
<td>3:100; 4:1073741824</td>
<td>1073676289</td>
<td>0</td>
</tr>
<tr>
<td>nghttp2/1.22.0</td>
<td>3:100; 4:65535</td>
<td>00</td>
<td>3:0:0:20,5:0:0:101, 7:0:0:1,9:0:7:1,11:0:3:1</td>
</tr>
</tbody>
</table>
ALMOST THERE...

STAY ON TARGET.
Nice. But still…

not enough entropy
Proposed Fingerprint

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

**Pseudo Headers**

**Request Pseudo Headers**

- :method
- :scheme
- :authority
- :path

**Response Pseudo Headers**

- :status
Proposed Fingerprint

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

🟦 **HTTP/1.1 Request**
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html

🟦 **HTTP/2 Request**
:method: GET
:path: /
:authority: www.example.com
:scheme: https
User-Agent: Mozilla/5.0
Accept: text/html
## Proposed Fingerprint

<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>
Proposed Fingerprint

Example from Chrome’s source code:

```c
void CreateSpdyHeadersFromHttpRequest(const HttpRequestInfo& info,
                                       const HttpRequestHeaders& request_headers,
                                       bool direct,
                                       SpdyHeaderBlock* headers) {

    (*headers)["method"] = info.method;

    if (info.method == "CONNECT") {
        (*headers)["authority"] = GetHostAndPort(info.url);
    } else {
        (*headers)["authority"] = GetHostAndOptionalPort(info.url);
        (*headers)["scheme"] = info.url.scheme();
        (*headers)["path"] = info.url.PathForRequest();
    }
```
Proposed Fingerprint

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

SETTINGS | WINDOW_UPDATE | PRIORITY | PSH-Order

| SETTINGS | Window Update | PRIORITY | Pseudo Header Order |
USE CASES
Use Cases

- Positive Security
- Detect Browser Impersonators
- Tool Detection
- Anonymous Proxy / VPN Detection

* Fingerprinting should also combine other layers
HTTP/2 THREAT LANDSCAPE
Most **security tools** lack H2 support:

- Burp Suite
- Zed Attack Proxy
- Fiddler
- SQLmap
- Acunetix
- AppScan
- NetSparker
- SentryMBA
- THC-Hydra
Why?

- Not enough incentive for Attackers
  - Web servers support both HTTP/1.X and HTTP/2
  - HTTP/2 libraries are not common
  - Cost exceeds the Gain
- Server Implementation Weaknesses found in 2016
  - Handling of Compression, Stream management
Key Takeaways

- Basic understanding of how HTTP/2 works
- Key differences between HTTP 1.x and 2.0
- Passive Fingerprinting
- Proposed fingerprint mechanism and Use Cases
- (Lack of) Threat Landscape
Questions
THANK YOU

Elad Shuster
eshuster@akamai.com