HTTP
Internet-Draft
Intended status: Standards Track
Expires: June 2, 2019

M. West
Google

November 29, 2018

User Agent Client Hints
draft-west-ua-client-hints-00

Abstract

This document defines a set of Client Hints that aim to provide developers with the ability to perform agent-based content negotiation when necessary, while avoiding the historical baggage and passive fingerprinting surface exposed by the venerable "User-Agent" header.

Status of This Memo

This Internet-Draft will expire on June 2, 2019.

Copyright Notice

Table of Contents

1. Introduction

Today, user agents generally identify themselves to servers by sending a "User-Agent" HTTP request header field along with each request (defined in Section 5.5.3 of [RFC7231]). Ideally, this header would give servers the ability to perform content negotiation, sending down exactly those bits that best represent the requested resource in a given user agent, optimizing both bandwidth and user experience. In practice, however, this header's value exposes far more information about the user's device than seems appropriate as a default, on the one hand, and intentionally obscures the true user agent in order to bypass misguided server-side heuristics, on the other.

For example, a recent version of Chrome on iOS identifies itself as:

```
     User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 12_0 like Mac OS X)
                 AppleWebKit/605.1.15 (KHTML, like Gecko)
                 CriOS/69.0.3497.105 Mobile/15E148 Safari/605.1
```

While a recent version of Edge identifies itself as:

```
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
              AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.2704.79
              Safari/537.36 Edge/18.014
```

There's quite a bit of information packed into those strings (along with a fair number of lies). Version numbers, platform details, model information, etc. are all broadcast along with every request, and form the basis for fingerprinting schemes of all sorts. Individual vendors have taken stabs at altering their user agent strings, and have run into a few categories of feedback from developers that have stymied historical approaches:

1. Brand and version information (e.g. "Chrome 69") allows websites to work around known bugs in specific releases that aren't otherwise detectable. For example, implementations of Content Security Policy have varied wildly between vendors, and it's difficult to know what policy to send in an HTTP response without knowing what browser is responsible for its parsing and execution.

2. Developers will often negotiate what content to send based on the user agent and platform. Some application frameworks, for instance, will style an application on iOS differently from the same application on Android in order to match each platform's aesthetic and design patterns.

3. Similarly to #1, OS revisions and architecture can be responsible for specific bugs which can be worked around in website's code, and narrowly useful for things like selecting appropriate executables for download (32 vs 64 bit, ARM vs Intel, etc).

4. Sophisticated developers use model/make to tailor their sites to the capabilities of the device (e.g. [FacebookYearClass]) and to pinpoint performance bugs and regressions which sometimes are specific to model/make.

This document proposes a mechanism which might allow user agents to be a bit more aggressive about removing entropy from the "User-Agent" string generally by giving servers that really need some specific details about the client the ability to opt-into receiving them. It introduces four new Client Hints ([I-D.ietf-httpbis-client-hints]) that can provide the client's branding and version information, the

underlying operating system's branding and major version, as well as details about the underlying device. Rather than broadcasting this data to everyone, all the time, user agents can make reasonable decisions about how to respond to given sites' requests for more granular data, reducing the passive fingerprinting surface area exposed to the network.

1.1. Example

A user navigates to "https://example.com/" for the first time. Their user agent sends the following header along with the HTTP request:

```
Sec-CH-UA: "Examplary Browser 73"
```

The server is interested in rendering content consistent with the user's underlying platform, and asks for a little more information by sending an "Accept-CH" header (Section 2.2.1 of [I-D.ietf-httpbis-client-hints]) along with the initial response:

```
Accept-CH: UA, Platform
```

In response, the user agent includes more detailed version information, as well as information about the underlying platform in the next request:

```
Sec-CH-UA: "Examplary Browser 73.3R8.2H.1"
Sec-CH-Platform: "Windows 10"
```

1.2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. User Agent Hints

The following sections define a number of HTTP request header fields that expose detail about a given user agent, which servers can opt-into receiving via the Client Hints infrastructure defined in [I-D.ietf-httpbis-client-hints]. The definitions below assume that each user agent has defined a number of properties for itself (all of which are strings):

- "brand" (for example: "cURL", "Edge", "The World's Best Web Browser")

- "major version" (for example: "72", "3", or "28")
- "full version" (for example: "72.0.3245.12", "3.14159", or "297.70E04154A")
- "platform brand" (for example: "Windows NT", "iOS", or "AmazingOS")
- "platform version" (for example: "10", "12", or "17G")
- "platform architecture" (for example: "ARM64", or "ia32")
- "model" (for example: "", or "Pixel 2 XL")

User agents SHOULD keep these strings short and to the point, but servers MUST accept arbitrary values for each, as they are all values constructed at the user agent's whim.

2.1. The 'Sec-CH-Arch' Header Field

The "Sec-CH-Platform" request header field gives a server information about the architecture of the platform on which a given user agent is executing. It is a Structured Header ([I-D.ietf-httpbis-header-structure]) whose value MUST be a string ([I-D.ietf-httpbis-header-structure], Section 3.7).

The header's ABNF is:

```
Sec-CH-Arch = sh-string
```

To generate a "Sec-CH-Arch" header value for a given request, user agents MUST:

1. If the request's client-hints set includes "Arch", then:
   1. Let "value" be a Structured Header whose value is the user agent's "platform architecture".
   2. Set a header in request's header list whose name is "Sec-CH-Arch", and whose value is "value".

2.2. The 'Sec-CH-Model' Header Field

The "Sec-CH-Model" request header field gives a server information about the device on which a given user agent is executing. It is a Structured Header ([I-D.ietf-httpbis-header-structure]) whose value MUST be a string ([I-D.ietf-httpbis-header-structure], Section 3.7).

The header's ABNF is:

```
Sec-CH-Model = sh-string
```

To generate a "Sec-CH-Model" header value for a given request, user agents MUST:

1. If the request's client-hints set includes "Model", then:
    1. Let "value" be a Structured Header whose value is the user agent's "model".
    2. Set a header in request's header list whose name is "Sec-CH-Model", and whose value is "value".

2.3. The 'Sec-CH-Platform' Header Field

The "Sec-CH-Platform" request header field gives a server information about the platform on which a given user agent is executing. It is a Structured Header ([I-D.ietf-httpbis-header-structure]) whose value MUST be a string ([I-D.ietf-httpbis-header-structure], Section 3.7).

The header's ABNF is:

```
Sec-CH-Platform = sh-string
```

To generate a "Sec-CH-Platform" header value for a given request, user agents MUST:

1. If the request's client-hints set includes "Platform", then:
    1. Let "value" be a Structured Header whose value is the concatenation of the user agent's "platform brand", a U+0020 SPACE character, and the user agent's "platform version".
    2. Set a header in request's header list whose name is "Sec-CH-Platform", and whose value is "value".

2.4. The 'Sec-CH-UA' Header Field

The "Sec-CH-UA" request header field gives a server information about a user agent's branding and version. It is a Structured Header ([I-D.ietf-httpbis-header-structure]) whose value MUST be a list ([I-D.ietf-httpbis-header-structure], Section 3.2). Each item in the list MUST be a string ([I-D.ietf-httpbis-header-structure], Section 3.7).

The header's ABNF is:

```
Sec-CH-UA = sh-list
```

Unlike most Client Hints, the "Sec-CH-UA" header will be sent with all requests, whether or not the server opted-into receiving the header via an "Accept-CH" header. Prior to an opt-in, however, it will include only the user agent's branding information, and the major version number (both of which are fairly clearly sniffable by "examining the structure of other headers and by testing for the availability and semantics of the features introduced or modified between releases of a particular browser" [Janc2014]).

To generate a "Sec-CH-UA" header value for a given request, user agents MUST:

1. Let "value" be a Structured Header whose value is a list ([I-D.ietf-httpbis-header-structure]).

2. If the request's client-hints set includes "UA", then add an item to "value" whose value is the concatenation of the user agent's "brand", a U+0020 SPACE character, and the user agent's "full version".

   Otherwise, add an item to "value" whose value is the concatenation of the user agent's "brand", a U+0020 SPACE character, and the user agent's "major version".

3. The user agent MAY execute the following steps:

   1. Append additional items to "value" containing arbitrary brand and version combinations.

   2. Randomize the order of the items in "value".

   Note: See Section 4.2 for more details on why these steps might be appropriate.

4. Set a header in request's header list whose name is "Sec-CH-UA", and whose value is "value".

2.5. Integration with Fetch

The Fetch specification should call into the following algorithm in place of the current Step 5.11 in its HTTP-network-or-cache fetch algorithm.

To set the user agent metadata for a request ("r"), the user agent MUST execute the following steps:

1. If request's header list does not contain "User-Agent", then the user agent MAY append "User-Agent"/default "User-Agent" value to "request"'s header list.

2. Set request's "Sec-CH-Arch" header, as described in Section 2.1.

3. Set request's "Sec-CH-Model" header, as described in Section 2.2.

4. Set request's "Sec-CH-Platform" header, as described in Section 2.3.

5. Set request's "Sec-CH-UA" header, as described in Section 2.4.

3. Security and Privacy Considerations

3.1. Secure Transport

Client Hints will not be delivered to non-secure endpoints (see the secure transport requirements in Section 2.2.1 of [I-D.ietf-httpbis-client-hints]). This means that user agent information will not be leaked over plaintext channels, reducing the opportunity for network attackers to build a profile of a given agent's behavior over time.

3.2. Delegation

Client Hints will be delegated from top-level pages via Feature Policy (once a few patches against Fetch and Client Hints and Feature Policy land. This reduces the likelihood that user agent information will be delivered along with subresource requests, which reduces the potential for passive fingerprinting.

- Fetch integration of Accept-CH opt-in: https://github.com/whatwg/fetch/issues/773

- HTML integration of Accept-CH-Lifetime and the ACHL cache: https://github.com/whatwg/html/issues/3774

- Adding new CH features to the CH list in Fetch: https://github.com/whatwg/fetch/issues/725

- Other PRs for adding the Feature Policy 3rd party opt-in: https://github.com/whatwg/fetch/issues/811 and https://github.com/wicg/feature-policy/issues/220

3.3. Access Restrictions

The information in the Client Hints defined above reveals quite a bit of information about the user agent and the platform/device upon which it runs. User agents ought to exercise judgement before granting access to this information, and MAY impose restrictions above and beyond the secure transport and delegation requirements noted above. For instance, user agents could choose to reveal "platform architecture" only on requests it intends to download, giving the server the opportunity to serve the right binary. Likewise, they could offer users control over the values revealed to servers, or gate access on explicit user interaction via a permission prompt or via a settings interface.

4. Implementation Considerations

4.1. The 'User-Agent' Header

User agents SHOULD deprecate the "User-Agent" header in favor of the Client Hints model described in this document. The header, however, is likely to be impossible to remove entirely in the near-term, as existing sites' content negotiation code will continue to require its presence (see [Rossi2015] for a recent example of a new browser's struggles in this area).

One approach which might be advisable could be for each user agent to lock the value of its "User-Agent" header, ensuring backwards compatibility by maintaining the crufty declarations of "like Gecko" and "AppleWebKit/537.36" on into eternity. This can ratchet over time, first freezing the version number, then shifting platform and model information to something reasonably generic in order to reduce the fingerprint the header provides.

4.2. GREASE-like UA Strings

History has shown us that there are real incentives for user agents to lie about their branding in order to thread the needle of sites' sniffing scripts. While I'm optimistic that we can reset expectations around sniffing by freezing the thing that's sniffed-upon today, and creating a sane set of options for developers, it's likely that this is hopelessly naive. It's reasonable to ponder what we should do to encourage sniffing in the right way, if we believe it's going to happen one way or another.

User agents may choose to model "UA" as a set, rather than a single entry. This could encourage standardized processing of the "UA" string by Randomly including additional, intentionally incorrect, comma-separated entries with arbitrary ordering (similar conceptually

to [I-D.ietf-tls-grease]) could encourage standardized processing if the "UA" string by servers, and reduce the chance that we ossify on a few required strings. For example, Chrome 73's "Sec-CH-UA" header might be ""Chrome 73", "NotBrowser 12"", or ""BrowsingIsFun Version 12b", "Chrome 73"", or something completely different.

4.3. The 'Sec-CH-' prefix

Based on some discussion in https://github.com/w3ctag/design-reviews/issues/320, it seems reasonable to forbid access to these headers from JavaScript, and demarcate them as browser-controlled client hints so they can be documented and included in requests without triggering CORS preflights. A "Sec-CH-" prefix seems like a viable approach, but this bit might shift as the broader Client Hints discussions above coalesce into something more solid that lands in specs.

5. IANA Considerations

This document intends to define the "Sec-CH-Arch", "Sec-CH-Model", "Sec-CH-Platform", and "Sec-CH-UA" HTTP request header fields, and register them in the permanent message header field registry ([RFC3864]).

It also intends to deprecate the "User-Agent" header field.

5.1. 'Sec-CH-Arch' Header Field

Header field name: Sec-CH-Arch

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document: this specification (Section 2.1)

5.2. 'Sec-CH-Model' Header Field

Header field name: Sec-CH-Model

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document: this specification (Section 2.4)

5.3. 'Sec-CH-Platform' Header Field

Header field name: Sec-CH-Platform

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document: this specification (Section 2.3)

5.4. 'Sec-CH-UA' Header Field

Header field name: Sec-CH-UA

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document: this specification (Section 2.4)

5.5. 'User-Agent' Header Field

Header field name: User-Agent

Applicable protocol: http

Status: deprecated

Author/Change controller: IETF

Specification document: this specification (Section 4.1), and Section 5.5.3 of [RFC7231]

6. References

6.1. Normative References

[I-D.ietf-httpbis-client-hints]
Grigorik, I., "HTTP Client Hints", draft-ietf-httpbis-client-hints-06 (work in progress), July 2018.

[I-D.ietf-httpbis-header-structure]
Nottingham, M. and P. Kamp, "Structured Headers for HTTP", draft-ietf-httpbis-header-structure-08 (work in progress), October 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

6.2. Informative References

[FacebookYearClass]
Marra, C. and D. Weaver, "Year class: A classification system for Android", November 2014, <https://code.fb.com/android/year-class-a-classification-system-for-android/>.

[I-D.ietf-tls-grease]
Benjamin, D., "Applying GREASE to TLS Extensibility", draft-ietf-tls-grease-01 (work in progress), June 2018.

[Janc2014]
Zalewski, M. and A. Janc, "Technical analysis of client identification mechanisms", September 2014, <https://dev.chromium.org/Home/chromium-security/client-identification-mechanisms#TOC-Browser-level-fingerprints>.

[Rossi2015]
Rossi, J., "The Microsoft Edge Rendering Engine that makes the Web just work", May 2015, <https://channel9.msdn.com/Events/WebPlatformSummit/2015/The-Microsoft-Edge-Rendering-Engine-that-makes-the-Web-just-work#time=9m45s>.

Appendix A. Changes

A.1. draft-west-ua-client-hints-00

- This specification sprang, fully-formed, from the head of Zeus.

Author's Address

Mike West
Google

Email: mkwst@google.com