

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 21, 2020

M. Thomson
M. Nottingham
September 18, 2019

Report from the IAB Workshop on Exploring Synergy between Content
Aggregation and the Publisher Ecosystem (ESCAPE)
draft-iab-escape-report-00

Abstract

The Exploring Synergy between Content Aggregation and the Publisher Ecosystem (ESCAPE) Workshop was convened by the Internet Architecture Board (IAB) in July 2019. This report summarizes its significant points of discussion and identifies topics that may warrant further consideration.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 21, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Mention of Specific Entities	3
2. Use Cases	3
2.1. Instant Navigation	4
2.2. Offline Content Sharing	5
2.3. Other Use Cases	5
2.3.1. Book Publishing	6
2.3.2. Web Archiving	7
3. Interactions Between Web Publishers and Aggregators	8
3.1. Incentives for Web Packages	8
3.2. Operational Costs	9
3.3. Content Regulation	9
3.4. Web Performance	10
4. Systemic Effects	11
4.1. Consolidation	11
4.1.1. Consolidation of Power in Linking Sites	11
4.1.2. Consolidation of Power in Publishers	12
4.1.3. Consolidation of User Preferences	12
4.2. Effect on Web Security	13
4.3. Privacy of Content	14
5. AMP Issues Unrelated to Web Packaging	15
5.1. AMP Governance	15
5.2. Constraints on the AMP Format	16
5.3. Performance	16
5.4. Implementation of Paywalls	16
6. Venues for Future Discussion	17
7. Security Considerations	17
8. References	17
8.1. Informative References	17
8.2. URIs	20
Appendix A. About the Workshop	20
A.1. Agenda	20
A.1.1. Thursday 2019-07-18	20
A.1.2. Friday 2019-07-19	21
A.2. Workshop Attendees	21
Appendix B. Web Packaging Overview	22
B.1. Authority in HTTPS	23
B.2. Authority in Web Packaging	23
B.3. Applicability	24
B.4. The AMP Format, Google Search Results, and Web Packaging	24
Authors' Addresses	25

1. Introduction

The IAB convened this workshop to examine some proposed changes to the Internet and the Web, and their potential effects on the Internet publishing landscape. Of particular interest was the Web Packaging proposal from Google, under consideration in the IETF, the W3C's Web Incubator Community Group (WICG), and the Web Hypertext Application Technology Working Group (WHATWG).

In considering these proposals, we heard about both positive effects of Web Packaging, and concerns that it could have significant effects on the relationship between publishers (e.g., news Web sites) and content aggregators (e.g., search engines and social networks). As such, our focus was primarily on this relationship, rather than being a technical discussion.

Online publishers do not regularly participate in standards activities directly. A Workshop format was used to solicit input from them. The workshop had 27 participants from a diverse set of backgrounds, including a small number of attendees from publishers, one aggregator (Google), plus representatives from browsers, the AMP community, CDNs, network operators, academia, and standards bodies. See the Workshop Call for Participation [CFP] for more information and a complete listing of submissions.

As intended, the Workshop was primarily a forum for discussion, so it did not reach definite conclusions. Instead, this report is the primary output of the Workshop, as a record of that discussion.

This report documents the use cases discussed in Section 2 and explains the interactions between publishers and aggregators that might be affected by it in Section 3. Appendix A includes more details about the Workshop itself. For those unfamiliar with Web Packaging, Appendix B provides a summary as background material.

1.1. Mention of Specific Entities

Participants agreed to conduct the Workshop under the Chatham House Rule [CHATHAM-HOUSE], so this report does not attribute statements to individuals or organizations without express permission. Submissions to the Workshop were public, and thus attributable; they are used here to provide substance and context.

2. Use Cases

Much of the Workshop concentrated on discussion of the validity and relative merits of the use cases that might be enabled by Web Packaging. See Appendix B for an overview of what Web Packaging is.

2.1. Instant Navigation

The largest use of Web Packaging so far is in Google Search, where packages are intended to improve the perceived performance of navigation to pages that are linked from search results when "clicked".

To enable this, when a linking (or referring) web page includes links to pages on another site, it also provides the browser with a packaged copy of the target content, signed by the origin of the target content. In effect, the referring page provides a cache for the target page's content. If navigation to one of those links occurs, having the Web Package gives a browser the assurance that the cache didn't change the content, so it can treat that content as if it were acquired directly from the server for the target page - even though it came from a different server. In many cases, this results in significantly lower perceived delay in displaying the target page.

A vital characteristic of this technique is that the browser does not contact the target site before navigation. The browser does not make any requests to sites until after navigation occurs, and only then if the site requires additional content or makes a request directly.

Similar improvements could also be realized by downloading content (packaged or otherwise) directly from the target site through a technique called prefetching. However, doing so would reveal information about the user's activity on the linking page to those sites - even when the user never actually navigates to it.

Note: This technique that uses Web Packaging is also referred to as "privacy-preserving prefetch". This document avoids that term as there was some contention at the workshop about what aspects of privacy might be preserved by the technique.

Sites bundled with Web Packaging can additionally be constructed in a way that ensures that they render without needing any additional network access. This makes it possible to provide near-instantaneous navigation. The proposed changes to web navigation in support of loading Web Packages is designed to support this use case.

Workshop participants recognized the value of web performance for usability, as well as for business metrics like retention and bounce rates. Such improvements were seen as a valuable goal, but publishers raised questions about whether they justified the cost of supporting an additional format, while others raised concerns about different aspects of the Web Packaging proposal.

2.2. Offline Content Sharing

Another primary use case discussed was the ability to share Web content between devices where neither has an active connection to the Internet. One of the stated goals of Web Packaging is to enable sharing of content offline.

Several participants reported that in areas where Internet access is expensive, slow, or intermittent, the use of direct peer-to-peer file exchange (e.g., "saving a Web site and sharing it on a USB stick") is commonplace. Most Web browsers already have some affordances for this, but these are recognized as in need of improvements.

In the discussion, several rejected an assumed requirement of this use case – that there be no difference between the treatment of a "normal" Web page and that of one loaded from an offline Web Package.

The ability for a Web Package to provide clear attribution for content was seen as valuable by some participants for a range of reasons. However, reservations were expressed about the subtleties of the properties that signatures provide and the effect of this on Web security; see also Section 4.2 and Section 2.3.2.

Many participants pointed out that using "unsigned bundles" – that is, Web Packages without Signed Exchanges – could be adequate for this use case, since most users don't need cryptographic proof of the site's identity. However, some expressed concerns that this might worsen the propagation of falsehood.

Some suggested that the value of Signed Exchanges was not realized in small-scale interpersonal exchange of information, but in the building of systems for content delivery that might include capabilities like discovery and automated distribution. The contention here was that effective use of digital signatures in offline distribution of content implied considerably more infrastructure than was described in current proposals.

No definite conclusions about offline sharing were reached during the workshop.

2.3. Other Use Cases

A session on the second morning concentrated on two other significant potential use cases for Web Packages: book publishing and Web archiving. These were not seen as "primary" by the proponents of Web Packaging; the original intent was not to spend significant time on these subjects, but there was considerable interest from attendees.

2.3.1. Book Publishing

The potential application of a packaging format to book publishing was discussed, with particular reference to ways that books differ from web content. Specialists from that industry pointed out that book delivery can vary greatly from typical web content delivery.

Workshop participants briefly explored existing solutions. PDF was seen as particularly challenging for this use case, due to its limitations, and EPUB has constraints that also make it challenging for publishers.

Although Web Packaging might help to address this use case, the question of how to identify book content was not resolved. The use of Signed Exchanges in this context might offer means of tying content in books to a Web site, but several limitations inherent in doing that were identified.

In particular, book publication specialists represented that books don't have the same requirements for timeliness or currency as web pages. For instance, Dave Cramer's submission [CRAMER] observed that Moby Dick was published over 61,000 days ago, which is considerably longer than the proposed limit of 7 days for Signed Exchanges. The limited length of time that a Web Package can be considered valid was discussed at some length.

Additionally, the risk of a publisher going out of business during the lifetime of a book is significant, because books - at least successful ones - often span generations in their applicability. To that end, having a means of attributing content to a publisher was considered less practical, and potentially undesirable (much like the discussion above regarding "unsigned bundles").

There were other aspects of book publication that participants saw as challenging for packaging. For example, it is currently not understood what it is to refer to distinct parts of a book. Participants saw this as an area where providing stable references for bundles of content might offer possibilities, but nothing concrete came from that discussion.

The potential for active content in a bundle to use Web APIs to enrich content or enable new features was considered valuable. Models for enabling paywalls were discussed at some length (see Section 5.4).

2.3.2. Web Archiving

Web archiving is a complicated discipline that is made more difficult by the complex nature of the web itself.

From an archival standpoint, the potential for Web content to be provided in a self-contained form was viewed positively. Several improvements to the structure of Web Packaging were considered, such as providing complete sets of content and the use of Memento [MEMENTO].

Though there were potential applications of a packaging scheme, many challenges were recognized as requiring additional work on the part of content producers to be fully effective. For example, JavaScript is needed to render some archived content faithfully, but attributing that content to an origin in all scenarios is challenging.

If packaging were to be widely deployed it might improve the situation for archival replay. In particular, the speculation is that there would be less "live leakage" as packaged content might be less likely to refer to live resources that currently tend to "leak" into views of archives. It was also noted that subresources might also be more likely to be packaged, especially those that are needed for deferred representations (i.e., after JavaScript execution on the page or some user interactions). Other potential applications and enhancements are discussed in [ALAM].

Participants discussed the use of a signature for non-repudiation at some length. In one case related to the Internet Archive, a public figure disputed the accuracy of archived content, asserting that either the original content was modified at the source, or in the archive.

Some participants initially saw digital signatures as a way to address such issues of provenance. As similar problems exist in other areas, such as in book publication, medical research, and news, a solution to this problem was considered to have broad applicability.

However, the discussion ultimately concluded that providing non-repudiation in retrospect is challenging. Signing keys are not expected to remain secure for long periods. If keys are leaked afterwards, an attacker could retroactively generate fraudulent signatures. Alternative solutions were discussed, such as providing independent archives for the same data, using consensus protocols, or using an append-only construct like a Haber-Stornetta log [AOLOG], all of which can be used to increase the difficulty of altering or misrepresenting established archives.

3. Interactions Between Web Publishers and Aggregators

A significant motivation for holding the Workshop was to provide a forum where publishers could discuss the impact of Web Packaging on the online publishing ecosystem. Of primary interest was whether Web Packages might effectively enable a transfer of power from publishers to aggregators.

Both publishers and aggregators at the workshop expressed the importance of maintaining a positive relationship. Publishers in particular expressed the need to be able to trust that aggregators won't misrepresent their work, or de-emphasize it for reasons unrelated to quality and perceived value to the user.

One key question from [BERJON] was discussed:

Web Packaging has other uses, but it is primarily seen by a large proportion of its stakeholders as a solution to problems that AMP created. Before we agree to solve those issues, should we not ask if AMP was a useful approach in the first place - and useful to whom?

In examining this issue, discussion focused on the current incentive model offered by aggregators. The costs that publishers incur for participation in that system were considered. Considerable time was spent on AMP, a summary of that discussion can be found in Section 5.

We also considered the question of whether standardizing Web Packaging confers credibility to aggregators exercising unwelcome control over publisher content, or whether the technical safeguards Web Packaging provides could allow aggregators to relax their restrictions on the kinds of content they're willing to cache and serve. No conclusions were drawn.

3.1. Incentives for Web Packages

Submissions to the Workshop indicated that the use of inducements involving better placement and formatting of links to publisher content had a significant effect on the uptake of related technology. For example, in [DEPUYDT-NELSON]:

[...] The Washington Post has always placed a great deal of trust in Google to represent its content--and their reward for doing so is more traffic, which positively impacts the business.

During the Workshop, several online publishers indicated that if it weren't for the privileged position in the Google Search carousel given to AMP content, they would not publish in that format.

Publishers that do produce AMP said they see a non-trivial increase in traffic as a result of deploying AMP content. For example, Yahoo Japan reported a 60% increase in traffic as a result of deploying AMP on Yahoo Travel [OTSU]. There was no data presented as to whether this increase was due to better placement in Google Search results, from the inherent benefits of the AMP cache, or the use of the AMP format.

Anecdotal evidence was offered by another large publisher that saw a 10% drop in traffic as a result of accidentally disabling AMP content. However, increases in traffic might not result in similarly proportioned increases in revenue, as observed in [BREWSTER].

3.2. Operational Costs

Several participants pointed out that introducing a new, parallel format for Web content incurs operational costs. In particular, supporting any new format – such as Web Packaging, Apple News, or Facebook Instant Articles – requires not only initial development of tooling (some generic, some specific to a site's requirements) but also an ongoing investment in maintaining its operability. Some participants expressed concern about the impact upon small publishers with limited technical and financial resources, especially in the current publishing climate.

Increased exposure from new formats might not always justify the added expense of providing articles in that format [BREWSTER]. However, a standardized format might help publishers reduce the cost of maintaining multiple formats.

3.3. Content Regulation

The use of Web Packaging as a tool for avoiding censorship was not a significant topic of discussion, except to note that publishers often have regulatory requirements regarding removal or correction of content.

Reference was made to the desire to remove videos of a recent shooting [CHRISTCHURCH] and the potential difficulty in doing so if content were available as Web Packages. Legal requirements to remove content come from multiple angles: copyright violations, illegal content, editorial corrections or errors, and right to erasure provisions in the European Union General Data Protection Regulation [GDPR] were mentioned. One participant speculated that making it more difficult to remove material in this way might discourage regulators from censoring content.

In this context, participants observed that it would be difficult to create mechanisms to track and control content served as a Web Package without compromising the stated goal of censorship resistance.

3.4. Web Performance

Understanding the effect that Web Packaging might have on web performance was a matter of some contention.

Some informal analysis from the Google Search deployment was presented (later published in [AMP-PERF]) that showed significant performance improvements in metrics related to navigation time resulting from the combination of prefetch, prerendering, and the AMP format. These results are suggestive of a possibility that Web Packaging could provide some of that improvement on its own, but no data was presented that apportioned the improvement among the three components.

Though data was presented to demonstrate potential rather than be a definitive result, discussions raised a number of questions that suggest the need for further study. Attendees suggested that future measurements consider the effect of signed bundles distinct from the enhancements derived from the AMP format. Future research in this area might also consider the effectiveness of different strategies on devices with varying capabilities, bandwidth, power consumption requirements, or network conditions.

Of particular interest is the additional work required to fetch and render multiple web pages in preparation for navigation. This might ultimately use fewer connections, but comes with an increased network and CPU cost for clients. Some participants pointed out that different clients or applications might require different tuning; for example, when users have limited (or expensive) bandwidth, or for sites with less clear knowledge about the use of outbound links.

Workshop participants also expressed interest in learning about the effect of Web Packages on subsequent navigations within the target site.

In discussion, some participants suggested that their experience supported a theory that operating a cache at the linking site was most effective and the additional work done prior to navigation in terms of fetching and preparing content was what provided the most gains; others suggested that the benefits inherent in the AMP format was a dominant factor.

Understanding the complete effect of Web Packaging on web performance will require further work.

4. Systemic Effects

It is not straightforward to estimate how a proposed technology change might affect all of the parts of a system – including not only other components but also things like end-user rights and the balance of power between parties – ahead of time. To date, when evaluating proposals, the IETF has generally focused on more immediate concerns, such as interoperability and security.

Moreover, people often find new uses for successful standards [SUCCESS] after they are deployed. It is rarely possible to accurately predict all applications of a protocol or format, whether they are harmful or beneficial. Refusing standardization only impedes both outcomes.

With the understanding that predictions are difficult to make, there was considerable speculation at the Workshop about the possible effect of Web Packaging on the Web. Some of that speculation is informed by experience, but that experience is necessarily limited in scope. This section attempts to capture that discussion.

4.1. Consolidation

Concerns about the consolidation of power on the Internet have significantly increased lately, as a result of several factors. While the IAB, the Internet Society, and others are examining this phenomenon to understand it better, it is nevertheless prudent to consider whether proposals for changes to how the Internet works favors or counters consolidation. Favoring entities with existing advantages – like resources, size, or market share – is not necessarily a factor that disqualifies a new proposal, but it needs to be considered as a cost of enabling that technology.

While it isn't clear what all of the outcomes of adopting Web Packaging would be, the Workshop revealed several concerns for consolidation risks for all involved parties: users, publisher sites, linking sites, and services they each rely on.

4.1.1. Consolidation of Power in Linking Sites

Several participants noted that Web Packaging's enablement of instant navigation (Section 2.1) might advantage larger linking sites – such as social networks or search engines – over smaller ones in the same industry because doing so requires careful selections of which links to optimize, so as not to create unneeded traffic.

For example, a news article often has many links, but not all of them are equally likely to be followed. Deciding which ones to pre-fetch requires considerable data collection and engineering, so this technique might not be feasible for smaller entities. Additionally, some participants noted that this technique favors sites that have a linear set of ranked links, like search results; it is more difficult to apply to a page of news (for example) because predicting what link a user will follow is less obvious.

This technique also requires access to a cache with terms of use compatible with the requirements of the site. It was pointed out that the Google AMP Cache has policies that might be acceptable to many, and there are other caches. Sites operated by entities other than Google already use this cache, though it was observed that a site that does not host its own cache suffers a minor performance degradation.

4.1.2. Consolidation of Power in Publishers

Participants seemed to agree that if performance is strong enough differentiator, the effective use of Web Packaging might turn out to be a condition for success for online publishers. Google Search's choice to privilege content that is served using HTTPS was pointed out as showing that this sort of influence can be effective. Equally, it is not necessarily the case that standardization of new capabilities will affect such policies materially, as noted in [YASSKIN]:

It seems unlikely that any decisions we make in a packaging or distribution system will affect the considerations aggregators use when deciding how to rank recommendations or the power this gives them over publishers.

The most common concern raised in the discussion was the effect of this technology on smaller publishers who might be less able to optimize the packages they produce, where their primary differentiation in the market has previously been the quality of their content.

4.1.3. Consolidation of User Preferences

In typical operation of the Web, servers have an opportunity to tailor content to the needs of their users. In contrast, a static Web Package has few options for individualization, as the content is generated once and used by many.

As a result, publishers noted that AMP provides less opportunity to customize content for their customers. Their concerns included not

only personalizing content based on what they know about the user but also optimizing the package for specific browsers. Other participants observed in relation to this that Web Packaging might also have a consolidating effect in the browser market.

Some participants brought up the possibility of customization by providing multiple packages, including multiple variants of resources in a single package, or performing customization after the package was loaded. However, other participants pointed out that all of these options have negative side effects, either in complexity or reduced performance arising from larger bundles or delayed customization.

4.2. Effect on Web Security

One session explored the impact of introducing a new security model for the Web. Currently, sites rely on connection-oriented security (provided by TLS [TLS]), but Web Packaging adds a limited form of object security. That is, the package protects the integrity of a message, rather than providing integrity and confidentiality for its delivery. Object security is not a new concept in the context of the Web; designs like SHTTP [SHTTP] are as old as HTTPS. Though the intent is for Web Packaging to have a far more narrow applicability, it provides fewer security guarantees than HTTPS, since it provides only authentication, no confidentiality with respect to the cache, and no assurance of liveness.

Object-based security – such as proposed in Web Packaging – allows the use of content regardless of how it is obtained; some participants noted that third parties gain greater control over the distribution of content, reducing the ability of publishers to retract or alter content over the validity period of signed content.

Another topic of discussion was composition attacks. In its proposed form, Web Packaging only provides authentication of independent resources, not a web page as a single unit, allowing an attacker to control the composition of resources. This weakness was acknowledged as a known shortcoming of the current proposal that would be addressed.

The issue of managing the trade-off between control and performance in caches arose. While participants recognized that problems with resource composition already occur by accident – for example, when a cache stores different versions of resources – Web Packaging allows an attacker more direct control over what resources are available to clients.

For example, an attacker might be able to cause content with a security flaw to be used up to a week past the time that the defect was fixed.

As an example of how Web Packaging might change the risk profile for sites, participants discussed recovery from cross-site scripting attacks. It is already the case that a brief exposure to this class of attack can result in an attacker gaining persistent access, but mechanisms exist that can be used to avoid or correct issues, like cache validation and Clear Site Data [CLEAR-DATA]. These measures are not available to clients unless they connect to the site.

The discussion pointed out that these concerns are not new or uniquely enabled by Web Packaging. However, it was pointed out that new features are routinely subject to higher security and privacy expectations. In an example unrelated to Web Packaging but with similar tradeoffs, shared compression of multiple resources has significant performance benefits. The risk with shared compression exposes is the potential for exposing encrypted information through side-channels. Though sites can use shared compression without this exposure, shared compression will likely only be enabled once it is clear that measures to prevent accidental information exposure are understood to be effective in a broad set of deployments.

The discussion also addressed the question of whether concerns might equally apply to the typical use of a Content Distribution Network (CDN) as a third-party provider of the content. Some participants concluded that CDNs are typically in a contractual relationship with the sites they serve and so are more likely to have their interests aligned.

4.3. Privacy of Content

Discussion and submissions raised concerns regarding how serving content using Web Packages might adversely affect privacy of individuals. There are challenges here, but the very narrow applicability of Web Packaging to what is effectively static content limits the privacy risk. The conclusion was that provided sufficient care is taken in implementation, use of Web Packages does not substantially increase the information that an aggregator gains about what content is consumed.

Concretely, an aggregator knows what content it serves in anticipation of navigation. This is - at least in theory - substantially the same as the content that the aggregator might receive if it performed the navigation itself. Assuming that content is stripped of personalization, the aggregator gains no new information.

5. AMP Issues Unrelated to Web Packaging

On multiple occasions, discussion at the Workshop concentrated on problems that arise as a result of constraints on the AMP format or details of its inclusion in Google Search. For instance, the requirement to make metadata about pages to be exposed by pages is unlikely to be affected by any standardization of a packaging format as that requirement is independent of the process of delivering content.

This section provides some detail on aspects of the discussion that touched on AMP more generally in this way. Some treatment of these points is considered relevant as some of the discussion at the workshop, even under the remit of discussing Web Packaging, concentrated on the effect of AMP on the ecosystem.

Note: Of the four formats mentioned in the workshop call for papers [CFP], only AMP sent representatives to the workshop. The discussion was therefore concentrated around AMP; this section should not be read to imply anything about other formats.

Discussion and submissions referred to a commitment [AMP-LESSONS] to allow publishers to use content that met specific criteria to access privileged positions in search results, regardless of their adoption of AMP. Participants felt that this approach might address some of these concerns if it were adopted and durable. For instance, the use of Web Packaging might be sufficient to remove some constraints on active content on the basis that the active content would be attributed to the publisher and not the AMP cache.

5.1. AMP Governance

There was interest from workshop participants in the governance model used for AMP. In particular, the question of how independent the AMP project would be of Google and Google Search.

Three of the seven members of the AMP Technical Steering Committee, the body that governs AMP, are Google employees, which gives Google considerable influence over the project. It was asserted that the governance structure was intended to be more independent of Google over time. The understanding was that any consumer of the format, such as Google Search, would make an independent assessment about whether to use or require different aspects of the AMP project products.

5.2. Constraints on the AMP Format

Sites often implement AMP by creating a separate set of content in parallel to their regular HTML content. Publishers noted this as a high cost, particularly for smaller sites. It was pointed out that websites can serve AMP-compliant content exclusively. However, several publishers referred to limitations in the format that made it unsuitable for their needs.

Many cited reasons for this duplication were related to the necessity of running arbitrary active content (typically, JavaScript). For example:

- o AMP provides a framework for supporting user authentication, but publishers asserted that using this framework was not considered practical.
- o AMP content does not support rendering of certain content, which can affect the ability of publishers to innovate in how they produce content.
- o The AMP model for the implementation of paywalls (Section 5.4) was claimed to be inimical to some publisher business models.

More broadly, they considered AMP's constraints on the use of active content as problematic, since they prevent the use of capabilities that are provided on equivalent non-AMP pages. Reference was made to a proposed `<amp-script>` element – which has since been made fully available – that seeks to provide limited access to some dynamic content.

5.3. Performance

Publishers observed that using the AMP format does not provide any guarantee of performance gains and in some cases could contribute to performance degradation. It was suggested that this was most problematic for sites that are already well-tuned for performance.

5.4. Implementation of Paywalls

The use of "paywalls" by Web publishers to control access to content in return for payment is increasingly common. One popular approach is to offer a limited number of articles without payment while insisting on a paid subscription to access further articles.

On several occasions, participants expressed dissatisfaction with the difficulty of integrating paywall authorization when using AMP. In particular, they said AMP encourages publishers to include an

article's full content, hidden by default but easily accessible to motivated users. The discussion extended to workarounds like cookie syncing [COOKIE-SYNC] that is used as part of authorization, a consequence of having cached content hosted on the linking site rather than the target site.

The same topic came up concerning book publication, where publishers indicated that having a means of enabling different methods of distribution without also facilitating unconstrained copying of book content was necessary.

This conflation of AMP issues with those addressed by Web Packaging was recurrent in the discussion. As observed in [DAS], these concerns might be addressed by linking to a signed bundle.

6. Venues for Future Discussion

Web Packaging work continues in multiple forums. Questions about the core format and signatures is being discussed on the wpack@ietf.org mailing list [1]. Changes to web browsers as proposed in [LOADING] will be discussed on the Fetch specification repository [2].

7. Security Considerations

Proposals discussed at the Workshop might have a significant security impact, and these topics were discussed in some depth; see Section 4.2.

8. References

8.1. Informative References

[ALAM] Alam, S., Weigle, M., Nelson, M., Klein, M., and H. Van de Sompel, "Supporting Web Archiving via Web Packaging", June 2019, <<https://www.iab.org/wp-content/IAB-uploads/2019/06/sawood-alam-2.pdf>>.

[AMP-LESSONS] Ubl, M., "Standardizing lessons learned from AMP", March 2018, <<https://blog.amp.dev/2018/03/08/standardizing-lessons-learned-from-amp/>>.

[AMP-PERF] Steinlauf, E., "The Speed Benefit of AMP Prerendering", August 2019, <<https://developers.googleblog.com/2019/08/the-speed-benefit-of-amp-prerendering.html>>.

- [AOLOG] Haber, S. and W. Stornetta, "How to time-stamp a digital document", Journal of Cryptology Vol. 3, DOI 10.1007/bf00196791, 1991.
- [BERJON] Berjon, R., "ESCAPE: The New York Times Position", July 2019, <<https://www.iab.org/wp-content/IAB-uploads/2019/07/ NYT-ESCAPE.pdf>>.
- [BREWSTER] Brewster, A., "ESCAPE Position / Patch.com", June 2019, <<https://www.iab.org/wp-content/IAB-uploads/2019/07/ NYT-ESCAPE.pdf>>.
- [BUNDLE] Yasskin, J., "Web Packaging", draft-yasskin-dispatch-web-packaging-00 (work in progress), June 2017.
- [CFP] IAB, ., "Exploring Synergy between Content Aggregation and the Publisher Ecosystem Workshop 2019", May 2019, <<https://www.iab.org/activities/workshops/ escape-workshop/>>.
- [CHATHAM-HOUSE] Chatham House, "Chatham House Rule", n.d., <<https://www.chathamhouse.org/chatham-house-rule>>.
- [CHRISTCHURCH] Stevenson, R. and J. Anthony, "'Thousands' of Christchurch shootings videos removed from YouTube, Google says", March 2019, <<https://www.stuff.co.nz/business/111330323/ facebook-working-around-the-clock-to-block-christchurch-shootings-video>>.
- [CLEAR-DATA] West, M., "Clear Site Data", W3C Working Draft, November 2017, <<https://www.w3.org/TR/clear-site-data/>>.
- [COOKIE-SYNC] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., and C. Diaz, "The Web Never Forgets", Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14, DOI 10.1145/2660267.2660347, 2014.
- [CRAMER] Cramer, D., "Packaging Books", June 2019, <<https://www.iab.org/wp-content/IAB-uploads/2019/06/ cramer-position-paper.pdf>>.

- [DAS] Das, S., "The Implication of Signed Exchanges on E-Commerce", June 2019, <https://www.iab.org/wp-content/IAB-uploads/2019/06/IAB-Position-Paper_-_Signed-Exchanges.pdf>.
- [DEPUYDT-NELSON] DePuydt, M. and M. Nelson, "Signed Exchanges and The Importance of Trust in Aggregator/Publisher relationships", June 2019, <<https://www.iab.org/wp-content/IAB-uploads/2019/06/washpost.pdf>>.
- [GDPR] European Union, "General Data Protection Regulation", EU Regulation 2016/679, April 2016, <<https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN#d1e2606-1-1>>.
- [HTTP] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [LOADING] Yasskin, J., "Loading Signed Exchanges", September 2019, <<https://wicg.github.io/webpackage/loading.html>>.
- [MEMENTO] Van de Sompel, H., Nelson, M., and R. Sanderson, "HTTP Framework for Time-Based Access to Resource States -- Memento", RFC 7089, DOI 10.17487/RFC7089, December 2013, <<https://www.rfc-editor.org/info/rfc7089>>.
- [ORIGIN] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <<https://www.rfc-editor.org/info/rfc6454>>.
- [OTSU] Ohtsu, S., "Deployment Experience of Signed HTTP Exchanges with AMP as a Publisher", June 2019, <<https://www.iab.org/wp-content/IAB-uploads/2019/06/shigeki-ohtsu.pdf>>.
- [SHTTP] Rescorla, E. and A. Schiffman, "The Secure HyperText Transfer Protocol", RFC 2660, DOI 10.17487/RFC2660, August 1999, <<https://www.rfc-editor.org/info/rfc2660>>.
- [SUCCESS] Thaler, D. and B. Aboba, "What Makes for a Successful Protocol?", RFC 5218, DOI 10.17487/RFC5218, July 2008, <<https://www.rfc-editor.org/info/rfc5218>>.
- [SXG] Yasskin, J., "Signed HTTP Exchanges", draft-yasskin-http-origin-signed-responses-06 (work in progress), July 2019.

- [TAG-DC] Betts, A., "Distributed and syndicated content", July 2017, <https://www.iab.org/wp-content/IAB-uploads/2019/06/IAB-Position-Paper_Signed-Exchanges.pdf>.
- [TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [YASSKIN] Yasskin, J., "Chrome's position on the ESCAPE workshop", June 2019, <<https://www.iab.org/wp-content/IAB-uploads/2019/06/chrome.html>>.

8.2. URIs

- [1] <https://www.ietf.org/mailman/listinfo/wpack>
- [2] <https://github.com/whatwg/fetch/issues/784>
- [3] <https://amp.dev/>
- [4] <https://schema.org/>
- [5] <https://developers.google.com/amp/cache/>

Appendix A. About the Workshop

The ESCAPE Workshop was held on 2019-07-18 and the morning of 2019-07-19 at Cisco's facility in Herndon, Virginia USA.

Attendees to the Workshop were asked to submit position papers. These papers are published on the IAB website [CFP].

The Workshop was conducted under Chatham House rule [CHATHAM-HOUSE], meaning that statements cannot be attributed to individuals or organizations without explicit authorization.

A.1. Agenda

This section outlines the broad areas of discussion on each day.

A.1.1. Thursday 2019-07-18

Web Packaging Overview: A technical summary of Web Packaging was provided, plus a longer discussion of a range of use cases.

Web Packaging and Aggregators: The use of web packaging from the perspective of a content aggregator was given.

Web Packaging and Publishers: After a break, presentations from web publishers talked about the benefits and costs of Web Packaging. This included some discussion of the effect of developing AMP-conformant versions of content from a publisher perspective.

Web Packaging and Security: This session concentrated on how the Web Packaging proposal might affect the Web security model.

Alternatives to Web Packaging: This session looked at alternative technologies, including those that were attempted in the past and some more recent ideas for addressing the use case of making web navigations more performant.

A.1.2. Friday 2019-07-19

Web Archival: This session talked about the potential application of a technology like Web Packaging in addressing some of the myriad problems faced by web archival systems.

Book Publishing: A discussion of the effect of technologies for bundling and distribution of books.

Conclusions: A wrap up session attempted to capture key learnings from the Workshop.

A.2. Workshop Attendees

Attendees to the Workshop are listed with their primary affiliation as it appeared in submissions. Attendees from the program committee (PC), the Internet Architecture Board (IAB), and Internet Engineering Steering Group (IESG) are also marked.

- o Sawood Alam, Old Dominion University
- o Jari Arkko, Ericsson (IAB)
- o Richard Barnes, Cisco
- o Robin Berjon, New York Times (PC)
- o Zack Bloom, Cloudflare
- o Abraham Brewster, Patch.com
- o Alissa Cooper, Cisco (IESG, IAB)
- o Dave Cramer, Hachette Book Group

- o Melissa DePuydt, Washington Post
- o Levi Durfee, AMP Advisory Committee
- o Rudy Galfi, Google
- o Joseph Lorenzo Hall, Center for Democracy & Technology (PC)
- o Matthew Nelson, Washington Post
- o Michael Nelson, Old Dominion University
- o Mark Nottingham, Fastly (IAB, PC)
- o Shigeki Ohtsu, Yahoo
- o Eric Rescorla, Mozilla
- o Adam Roach, Mozilla (IESG)
- o Rich Salz, Akamai Technologies
- o Wendy Seltzer, W3C
- o David Strauss, Pantheon (PC)
- o Chi-Jiun Su, Hughes
- o Ralph Swick, W3C
- o Martin Thomson, Mozilla (IAB, PC)
- o Jeffrey Yasskin, Google
- o Dan York, Internet Society
- o Benjamin Young, John Wiley & Sons

Appendix B. Web Packaging Overview

Web Packaging is comprised of two separate technologies: resource bundling [BUNDLE] and signed exchanges [SXG].

In both the submissions and Workshop discussion, the most controversial aspect of the technology is the use of signed exchanges as an alternative means of providing authority over a particular resource, for a few different reasons.

This appendix explains how authority works on the Web and how Web Packaging proposes to change that.

B.1. Authority in HTTPS

The web currently uses HTTPS [HTTP] to establish a server's authority - that is, to give an assurance that the content came from where the URL implies. The combination of URI scheme (https), domain name (or host), and port number are formed into a single identifier, the origin [ORIGIN] to which content is attributed.

Web browsers use the certificate offered as part of a TLS connection [TLS] to servers in determining whether a server is authoritative for that origin; see [ORIGIN] and Section 9.1 of [HTTP]. Content is attributed to a given URL only if it is received from a connection to a server that is authoritative for the associated origin.

As an example, a web browser seeking to load "https://example.com/index.html" makes a TLS connection to a server. As part of the TLS connection establishment, the server offers a certificate for the name "example.com". If the browser accepts the certificate, it will then make requests for URLs on the "https://example.com" origin on that connection and consider any answers the server to be authoritative.

This notion of authority is a crucial property of web security: only content that is attributed to the same web origin can access all information in that origin, including the content of most resources as well as state associated with the origin, such as cookies. This separation ensures that sites can keep secrets from each other, even when they are both loaded in the same browser.

B.2. Authority in Web Packaging

Web Packaging, through the use of signed exchanges, aims to provide an alternative means of establishing authority. A signed exchange is an expression of an HTTP request and response (an exchange) with certain information stripped and a digital signature applied.

The signature is made with a similar certificate to the one a server might offer in HTTPS - that certificate can also be used for HTTPS - but it includes a special attribute that denotes its suitability for signed exchanges.

A web browser that has been provided with a signed exchange can verify the signature, and - if the signature is valid and the certificate is acceptable - use the content from the signed exchange.

Critically, the web browser does not make an HTTPS connection to a server to get the content or to verify the signature.

In effect, Web Packaging moves from a model where authority is derived from the delivery method (i.e., TLS) to an object security model, where authority is derived from a signature on objects. In doing so, it aims to render the means of delivery irrelevant to determinations of security.

B.3. Applicability

Web Packaging does not claim to supplant the authority model of the Web completely, but to provide an alternative that might be used under certain narrow conditions. In particular, Web Packaging is intended for use with content that is not secret from an entity that is aware of the existence of that content.

In aid of this goal, web packaging does not include information from exchanges that is related either the process of acquiring content as well as any information that relates to individual requests. For instance, use of the Set-Cookie header field is expressly forbidden, as it often contains information that is related to a particular user.

B.4. The AMP Format, Google Search Results, and Web Packaging

The relationship between the AMP Project <https://amp.dev/> [3] and Web Packaging is complicated. The AMP Project, sponsored by Google, establishes a profile of HTML with a stated goal of providing support for the best practices for the format, with a strong emphasis on performance. The format tightly constrains the use of HTML features but also offers a library of components that provide sanitized implementations of many commonly used capabilities.

The connection to Web Packaging is bound up in the way that Google Search treats AMP content specially. AMP content provides two properties that Google Search exploits: metadata exposure and static analysis of active content.

AMP content provides metadata in a form that can be reliably extracted, using the microformats defined by the Schema.org project <https://schema.org/> [4]. This aspect of AMP has no effect on the discussion, except to the extent that this relates to Google Search and their use of this metadata in populating the carousel.

Constrained use of active content – such as JavaScript – in AMP makes it possible to analyze content to verify that actions taken are narrowly limited. This static analysis assures that AMP content can

be served without affecting other content on the same site. For Google Search, this is what enables the loading of AMP content alongside search content and other AMP resources.

To provide preloading, Google operates an AMP Cache <https://developers.google.com/amp/cache/> [5], from which AMP content is served. As a consequence, browsers attribute the content to the origin [ORIGIN] of the AMP Cache and not the publisher, creating some confusion about how content is attributed, as discussed in the W3C finding on distributed content [TAG-DC].

An important goal of Web Packaging is to attribute content loaded from a cache, such as the AMP cache, to the publisher that created that content. For more on this see Section 2.1.

Authors' Addresses

Martin Thomson

Email: mt@lowentropy.net

Mark Nottingham

Email: mnot@mnot.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 January 2021

J. Yasskin
Google
27 July 2020

Signed HTTP Exchanges
draft-yasskin-http-origin-signed-responses-09

Abstract

This document specifies how a server can send an HTTP exchange--a request URL, content negotiation information, and a response--with signatures that vouch for that exchange's authenticity. These signatures can be verified against an origin's certificate to establish that the exchange is authoritative for an origin even if it was transferred over a connection that isn't. The signatures can also be used in other ways described in the appendices.

These signatures contain countermeasures against downgrade and protocol-confusion attacks.

Note to Readers

Discussion of this draft takes place on the HTTP working group mailing list (ietf-http-wg@w3.org), which is archived at <https://lists.w3.org/Archives/Public/ietf-http-wg/> (<https://lists.w3.org/Archives/Public/ietf-http-wg/>).

The source code and issues list for this draft can be found in <https://github.com/WICG/webpackage> (<https://github.com/WICG/webpackage>).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 January 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Terminology	4
3. Signing an exchange	5
3.1. The Signature Header	6
3.1.1. Examples	7
3.1.2. Open Questions	9
3.2. CBOR representation of exchange response headers	9
3.2.1. Example	9
3.3. Loading a certificate chain	10
3.4. Canonical CBOR serialization	11
3.5. Signature validity	12
3.5.1. Open Questions	16
3.6. Updating signature validity	16
3.6.1. Examples	17
3.7. The Accept-Signature header	19
3.7.1. Integrity identifiers	20
3.7.2. Key type identifiers	20
3.7.3. Key value identifiers	20
3.7.4. Examples	21
3.7.5. Open Questions	21
4. Cross-origin trust	22
4.1. Uncached header fields	23
4.1.1. Stateful header fields	24
4.2. Certificate Requirements	25
4.2.1. Extensions to the CAA Record: cansignhttpexchanges Parameter	26
5. Transferring a signed exchange	26
5.1. Same-origin response	27
5.1.1. Serialized headers for a same-origin response	27
5.1.2. The Signed-Headers Header	28

5.2.	HTTP/2 extension for cross-origin Server Push	28
5.2.1.	Indicating support for cross-origin Server Push	28
5.2.2.	NO_TRUSTED_EXCHANGE_SIGNATURE error code	29
5.2.3.	Validating a cross-origin Push	29
5.3.	application/signed-exchange format	30
5.3.1.	Cross-origin trust in application/signed-exchange	31
5.3.2.	Example	32
5.3.3.	Open Questions	32
6.	Security considerations	32
6.1.	Over-signing	32
6.1.1.	Session fixation	33
6.1.2.	Misleading content	33
6.2.	Off-path attackers	33
6.2.1.	Mis-issued certificates	33
6.2.2.	Stolen private keys	34
6.3.	Downgrades	35
6.4.	Signing oracles are permanent	35
6.5.	Unsigned headers	35
6.6.	application/signed-exchange	36
6.7.	Key re-use with TLS	36
6.8.	Content sniffing	36
7.	Privacy considerations	37
7.1.	Visibility of resource requests	38
7.2.	User ID transfer	39
8.	IANA considerations	39
8.1.	Signature Header Field Registration	39
8.2.	Accept-Signature Header Field Registration	39
8.3.	Signed-Headers Header Field Registration	40
8.4.	HTTP/2 Settings	40
8.5.	HTTP/2 Error code	40
8.6.	Internet Media Type application/signed-exchange	41
8.7.	Internet Media Type application/cert-chain+cbor	42
8.8.	The cansignhttpexchanges CAA Parameter	43
9.	References	43
9.1.	Normative References	43
9.2.	Informative References	46
Appendix A.	Use cases	49
A.1.	PUSHed subresources	49
A.2.	Explicit use of a content distributor for subresources	49
A.3.	Subresource Integrity	50
A.4.	Binary Transparency	50
A.5.	Static Analysis	51
A.6.	Offline websites	51
Appendix B.	Requirements	51
B.1.	Proof of origin	51
B.1.1.	Certificate constraints	51
B.1.2.	Signature constraints	52
B.1.3.	Retrieving the certificate	52

B.2. How much to sign	53
B.2.1. Conveying the signed headers	53
B.3. Response lifespan	54
B.3.1. Certificate revocation	54
B.3.2. Response downgrade attacks	54
B.4. Low implementation complexity	55
B.4.1. Limited choices	55
B.4.2. Bounded-buffering integrity checking	55
Appendix C. Determining validity using cache control	56
C.1. Example of updating cache control	56
C.2. Downsides of updating cache control	57
Appendix D. Change Log	57
Appendix E. Acknowledgements	60
Author's Address	60

1. Introduction

Signed HTTP exchanges provide a way to prove the authenticity of a resource in cases where the transport layer isn't sufficient. This can be used in several ways:

- * When signed by a certificate ([RFC5280]) that's trusted for an origin, an exchange can be treated as authoritative for that origin, even if it was transferred over a connection that isn't authoritative (Section 9.1 of [RFC7230]) for that origin. See Appendix A.1 and Appendix A.2.
- * A top-level resource can use a public key to identify an expected publisher for particular subresources, a system known as Subresource Integrity ([SRI]). An exchange's signature provides the matching proof of authorship. See Appendix A.3.
- * A signature can vouch for the exchange in some way, for example that it appears in a transparency log or that static analysis indicates that it omits certain attacks. See Appendix A.4 and Appendix A.5.

Subsequent work toward the use cases in [I-D.yasskin-wpack-use-cases] will provide a way to group signed exchanges into bundles that can be transmitted and stored together, but single signed exchanges are useful enough to standardize on their own.

2. Terminology

Absolute URL A string for which the URL parser (<https://url.spec.whatwg.org/#concept-url-parser>) ([URL]), when run without a base URL, returns a URL rather than a failure, and for which that URL has a null fragment. This is similar to the

absolute-URL string (<https://url.spec.whatwg.org/#absolute-url-string>) concept defined by ([URL]) but might not include exactly the same strings.

Author The entity that wrote the content in a particular resource. This specification deals with publishers rather than authors.

Publisher The entity that controls the server for a particular origin [RFC6454]. The publisher can get a CA to issue certificates for their private keys and can run a TLS server for their origin.

Exchange (noun) An HTTP request URL, content negotiation information, and an HTTP response. This can be encoded into a request message from a client with its matching response from a server, into the request in a PUSH_PROMISE with its matching response stream, or into the dedicated format in Section 5.3, which uses [I-D.ietf-httpbis-variants] to encode the content negotiation information. This is not quite the same meaning as defined by Section 8 of [RFC7540], which assumes the content negotiation information is embedded into HTTP request headers.

Intermediate An entity that fetches signed HTTP exchanges from a publisher or another intermediate and forwards them to another intermediate or a client.

Client An entity that uses a signed HTTP exchange and needs to be able to prove that the publisher vouched for it as coming from its claimed origin.

Unix time Defined by [POSIX] section 4.16 (http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_16).

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.

3. Signing an exchange

In the response of an HTTP exchange the server MAY include a "Signature" header field (Section 3.1) holding a list of one or more parameterised signatures that vouch for the content of the exchange. Exactly which content the signature vouches for can depend on how the exchange is transferred (Section 5).

The client categorizes each signature as "valid" or "invalid" by validating that signature with its certificate or public key and other metadata against the exchange's URL, response headers, and content (Section 3.5). This validity then informs higher-level protocols.

Each signature is parameterised with information to let a client fetch assurance that a signed exchange is still valid, in the face of revoked certificates and newly-discovered vulnerabilities. This assurance can be bundled back into the signed exchange and forwarded to another client, which won't have to re-fetch this validity information for some period of time.

3.1. The Signature Header

The "Signature" header field conveys a list of signatures for an exchange, each one accompanied by information about how to determine the authority of and refresh that signature. Each signature directly signs the exchange's URL and response headers and identifies one of those headers that enforces the integrity of the exchange's payload.

The "Signature" header is a Structured Header as defined by [I-D.ietf-httpbis-header-structure]. Its value MUST be a parameterised list (Section 3.4 of [I-D.ietf-httpbis-header-structure]). Its ABNF is:

Signature = sh-param-list

Each parameterised identifier in the list MUST have parameters named "sig", "integrity", "validity-url", "date", and "expires". Each parameterised identifier MUST also have either "cert-url" and "cert-sha256" parameters or an "ed25519key" parameter. This specification gives no meaning to the identifier itself, which can be used as a human-readable identifier for the signature (however, this is likely to change soon; see Section 3.1.2, Paragraph 1). The present parameters MUST have the following values:

"sig" Byte sequence (Section 3.10 of [I-D.ietf-httpbis-header-structure]) holding the signature of most of these parameters and the exchange's URL and response headers.

"integrity" A string (Section 3.8 of

[I-D.ietf-httpbis-header-structure]) containing a "/"-separated sequence of names starting with the lowercase name of the response header field that guards the response payload's integrity. The meaning of subsequent names depends on the response header field, but for the "digest" header field, the single following name is the name of the digest algorithm that guards the payload's integrity.

"cert-url" A string (Section 3.8 of [I-D.ietf-httpbis-header-structure]) containing an absolute URL (Section 2) with a scheme of "https" or "data".

"cert-sha256" Byte sequence (Section 3.10 of [I-D.ietf-httpbis-header-structure]) holding the SHA-256 hash of the first certificate found at "cert-url".

"ed25519key" Byte sequence (Section 3.10 of [I-D.ietf-httpbis-header-structure]) holding an Ed25519 public key ([RFC8032]).

"validity-url" A string (Section 3.8 of [I-D.ietf-httpbis-header-structure]) containing an absolute URL (Section 2) with a scheme of "https".

"date" and "expires" An integer (Section 3.6 of [I-D.ietf-httpbis-header-structure]) representing a Unix time.

The "cert-url" parameter is not signed, so intermediates can update it with a pointer to a cached version.

3.1.1. Examples

The following header is included in the response for an exchange with effective request URI "https://example.com/resource.html". Newlines are added for readability.

Signature:

```

sig1;
  sig=*MEUCIQDXLI2gN3RNB1gFiuRNFpZXcDiaUpX6HIEwcZEc0cZYLAIGA9DsVOMM+g5YpwEBdGW3sS
+bvnmAJJiSMwhuBdqp5UY=*;
  integrity="digest/mi-sha256";
  validity-url="https://example.com/resource.validity.1511128380";
  cert-url="https://example.com/oldcerts";
  cert-sha256=*W7uB969dFW3Mb5ZefPS9Tq5ZbH5iSmOILpjv2qEArmI=*;
  date=1511128380; expires=1511733180,
sig2;
  sig=*MEQCIGjZRqTRf9iKNkGFyzRMTFgwf/BrY2ZNIP/dykhUV0aYAiBTXg+8wujoT4n/W+cNgb7pGq
QvIUgYZ8u8HZJ5YH26Qg==*;
  integrity="digest/mi-sha256";
  validity-url="https://example.com/resource.validity.1511128380";
  cert-url="https://example.com/newcerts";
  cert-sha256=*J/1Em9kNR0DdCmINbvitpvdYKNQ+YgBj99DlYp4fEXw=*;
  date=1511128380; expires=1511733180,
srisig;
  sig=*1GZVaJJM5f2oGczF1LmBdKTDL+QADza4BgeO494ggACYJOvrof6uh5OJCcwKrk7DK+LBch0jss
DYPp5CLc1SDA==*;
  integrity="digest/mi-sha256";
  validity-url="https://example.com/resource.validity.1511128380";
  ed25519key=*zsSevyFsxyZHiUluVBDD4eypdRLTqyWRVOJuuKUz+A8=*
  date=1511128380; expires=1511733180,
thirdpartysig;
  sig=*MEYCIQCnXJzn6Rh2fNxsobktir8TkiaJYQFhWTuWi1i4PewQaQIhAMS2TVjc4rTshDtXbgQEOW
gj2mRXALhfxPztXgPupii+*;
  integrity="digest/mi-sha256";
  validity-url="https://thirdparty.example.com/resource.validity.1511161860";
  cert-url="https://thirdparty.example.com/certs";
  cert-sha256=*UeOWUPkvxlGRTyvHcsMUN0A2oNsZbU8EUvg8A9ZAnNc=*;
  date=1511133060; expires=1511478660,

```

There are 4 signatures: 2 from different secp256r1 certificates within "https://example.com/", one using a raw ed25519 public key that's also controlled by "example.com", and a fourth using a secp256r1 certificate owned by "thirdparty.example.com".

All 4 signatures rely on the "Digest" response header with the mi-sha256 digest algorithm to guard the integrity of the response payload.

The signatures include a "validity-url" that includes the first time the resource was seen. This allows multiple versions of a resource at the same URL to be updated with new signatures, which allows clients to avoid transferring extra data while the old versions don't have known security bugs.

The certificates at "https://example.com/oldcerts" and "https://example.com/newcerts" have "subjectAltName"s of "example.com", meaning that if they and their signatures validate, the exchange can be trusted as having an origin of

"https://example.com/". The publisher might be using two certificates because their readers have disjoint sets of roots in their trust stores.

The publisher signed with all three certificates at the same time, so they share a validity range: 7 days starting at 2017-11-19 21:53 UTC.

The publisher then requested an additional signature from "thirdparty.example.com", which did some validation or processing and then signed the resource at 2017-11-19 23:11 UTC. "thirdparty.example.com" only grants 4-day signatures, so clients will need to re-validate more often.

3.1.2. Open Questions

The next revision of [I-D.ietf-httpbis-header-structure] will provide a way to parameterise byte sequences, at which point the signature itself is likely to become the main list item.

Should the cert-url and validity-url be lists so that intermediates can offer a cache without losing the original URLs? Putting lists in dictionary fields is more complex than [I-D.ietf-httpbis-header-structure] allows, so they're single items for now.

3.2. CBOR representation of exchange response headers

To sign an exchange's response headers, they need to be serialized into a byte string. Since intermediaries and distributors (Appendix A.2) might rearrange, add, or just reserialize headers, we can't use the literal bytes of the headers as this serialization. Instead, this section defines a CBOR representation that can be embedded into other CBOR, canonically serialized (Section 3.4), and then signed.

The CBOR representation of a set of response metadata and headers is the CBOR ([RFC7049]) map with the following mappings:

- * The byte string ':status' to the byte string containing the response's 3-digit status code, and
- * For each response header field, the header field's lowercase name as a byte string to the header field's value as a byte string.

3.2.1. Example

Given the HTTP exchange:

```
GET / HTTP/1.1
Host: example.com
Accept: */*
```

```
HTTP/1.1 200
Content-Type: text/html
Digest: mi-sha256=dcRDgR2GM35DluAV13PzgnG6+pvQwPywFvAulUeFrs=
Signed-Headers: "content-type", "digest"
```

```
<!doctype html>
<html>
...
```

The cbor representation consists of the following item, represented using the extended diagnostic notation from [CDDL] appendix G:

```
{
  'digest': 'mi-sha256=dcRDgR2GM35DluAV13PzgnG6+pvQwPywFvAulUeFrs=',
  'status': '200',
  'content-type': 'text/html'
}
```

3.3. Loading a certificate chain

The resource at a signature's "cert-url" MUST have the "application/cert-chain+cbor" content type, MUST be canonically-encoded CBOR (Section 3.4), and MUST match the following CDDL:

```
cert-chain = [
  "", ; U+1F4DC U+26D3
  + augmented-certificate
]
augmented-certificate = {
  cert: bytes,
  ? ocsf: bytes,
  ? sct: bytes,
  * tstr => any,
}
```

The first map (second item) in the CBOR array is treated as the end-entity certificate, and the client will attempt to build a path ([RFC5280]) to it from a trusted root using the other certificates in the chain.

1. Each "cert" value MUST be a DER-encoded X.509v3 certificate ([RFC5280]). Other key/value pairs in the same array item define properties of this certificate.

2. The first certificate's "ocsp" value MUST be a complete, DER-encoded OCSF response for that certificate (using the ASN.1 type "OCSPResponse" defined in [RFC6960]). Subsequent certificates MUST NOT have an "ocsp" value.
3. Each certificate's "sct" value if any MUST be a "SignedCertificateTimestampList" for that certificate as defined by Section 3.3 of [RFC6962].

Loading a "cert-url" takes a "forceFetch" flag. The client MUST:

1. Let "raw-chain" be the result of fetching ([FETCH]) "cert-url". If "forceFetch" is `_not_set`, the fetch can be fulfilled from a cache using normal HTTP semantics [RFC7234]. If this fetch fails, return "invalid".
 2. Let "certificate-chain" be the array of certificates and properties produced by parsing "raw-chain" using the CDDL above. If any of the requirements above aren't satisfied, return "invalid". Note that this validation requirement might be impractical to completely achieve due to certificate validation implementations that don't enforce DER encoding or other standard constraints.
 3. Return "certificate-chain".
- 3.4. Canonical CBOR serialization

Within this specification, the canonical serialization of a CBOR item uses the following rules derived from Section 3.9 of [RFC7049] with erratum 4964 applied:

- * Integers and the lengths of arrays, maps, and strings MUST use the smallest possible encoding.
- * Items MUST NOT be encoded with indefinite length.
- * The keys in every map MUST be sorted in the bitwise lexicographic order of their canonical encodings. For example, the following keys are correctly sorted:
 1. 10, encoded as 0A.
 2. 100, encoded as 18 64.
 3. -1, encoded as 20.
 4. "z", encoded as 61 7A.

5. "aa", encoded as 62 61 61.
6. [100], encoded as 81 18 64.
7. [-1], encoded as 81 20.
8. false, encoded as F4.

Note: this specification does not use floating point, tags, or other more complex data types, so it doesn't need rules to canonicalize those.

3.5. Signature validity

The client MUST parse the "Signature" header field as the parameterised list (Section 4.2.5 of [I-D.ietf-httpbis-header-structure]) described in Section 3.1. If an error is thrown during this parsing or any of the requirements described there aren't satisfied, the exchange has no valid signatures. Otherwise, each member of this list represents a signature with parameters.

The client MUST use the following algorithm to determine whether each signature with parameters is invalid or potentially-valid for an exchange's

- * "requestUrl", a byte sequence that can be parsed into the exchange's effective request URI (Section 5.5 of [RFC7230]),
- * "responseHeaders", a byte sequence holding the canonical serialization (Section 3.4) of the CBOR representation (Section 3.2) of the exchange's response metadata and headers, and
- * "payload", a stream of bytes constituting the exchange's payload body (Section 3.3 of [RFC7230]). Note that the payload body is the message body with any transfer encodings removed.

Potentially-valid results include:

- * The signed headers of the exchange so that higher-level protocols can avoid relying on unsigned headers, and
- * Either a certificate chain or a public key so that a higher-level protocol can determine whether it's actually valid.

This algorithm accepts a "forceFetch" flag that avoids the cache when fetching URLs. A client that determines that a potentially-valid certificate chain is actually invalid due to an expired OCSP response MAY retry with "forceFetch" set to retrieve an updated OCSP from the original server.

1. Let:
 - * "signature" be the signature (byte sequence in the parameterised identifier's "sig" parameter).
 - * "integrity" be the signature's "integrity" parameter.
 - * "validity-url" be the signature's "validity-url" parameter.
 - * "cert-url" be the signature's "cert-url" parameter, if any.
 - * "cert-sha256" be the signature's "cert-sha256" parameter, if any.
 - * "ed25519key" be the signature's "ed25519key" parameter, if any.
 - * "date" be the signature's "date" parameter, interpreted as a Unix time.
 - * "expires" be the signature's "expires" parameter, interpreted as a Unix time.
2. Set "publicKey" and "signing-alg" depending on which key fields are present:
 1. If "cert-url" is present:
 1. Let "certificate-chain" be the result of loading the certificate chain at "cert-url" passing the "forceFetch" flag (Section 3.3). If this returns "invalid", return "invalid".
 2. Let "main-certificate" be the first certificate in "certificate-chain".
 3. Set "publicKey" to "main-certificate"'s public key.
 4. If "publicKey" is an RSA key, return "invalid".

5. If "publicKey" is a key using the secp256r1 elliptic curve, set "signing-alg" to ecdsa_secp256r1_sha256 as defined in Section 4.2.3 of [RFC8446].
 6. Otherwise, either return "invalid" or set "signing-alg" to a non-legacy signing algorithm defined by TLS 1.3 or later ([RFC8446]). This choice MUST depend only on "publicKey"'s type and not on any other context.
2. If "ed25519key" is present, set "publicKey" to "ed25519key" and "signing-alg" to ed25519, as defined by [RFC8032]
 3. If "expires" is more than 7 days (604800 seconds) after "date", return "invalid".
 4. If the current time is before "date" or after "expires", return "invalid".
 5. Let "message" be the concatenation of the following byte strings. This matches the [RFC8446] format to avoid cross-protocol attacks if anyone uses the same key in a TLS certificate and an exchange-signing certificate.
 1. A string that consists of octet 32 (0x20) repeated 64 times.
 2. A context string: the ASCII encoding of "HTTP Exchange 1".

Note: RFC EDITOR PLEASE DELETE THIS NOTE; The implementation of the final RFC MUST use this context string, but implementations of drafts MUST NOT use it and MUST use another draft-specific string beginning with "HTTP Exchange 1 " instead. This ensures that signers can predict how their signatures will be used.
 3. A single 0 byte which serves as a separator.
 4. If "cert-sha256" is set, a byte holding the value 32 followed by the 32 bytes of the value of "cert-sha256". Otherwise a 0 byte.
 5. The 8-byte big-endian encoding of the length in bytes of "validity-url", followed by the bytes of "validity-url".
 6. The 8-byte big-endian encoding of "date".
 7. The 8-byte big-endian encoding of "expires".

8. The 8-byte big-endian encoding of the length in bytes of "requestUrl", followed by the bytes of "requestUrl".
9. The 8-byte big-endian encoding of the length in bytes of "responseHeaders", followed by the bytes of "responseHeaders".
6. If "cert-url" is present and the SHA-256 hash of "main-certificate"'s "cert_data" is not equal to "cert-sha256" (whose presence was checked when the "Signature" header field was parsed), return "invalid".

Note that this intentionally differs from TLS 1.3, which signs the entire certificate chain in its Certificate Verify (Section 4.4.3 of [RFC8446]), in order to allow updating the stapled OCSP response without updating signatures at the same time.

7. If "signature" is not a valid signature of "message" by "publicKey" using "signing-alg", return "invalid".
8. If "headers", interpreted according to Section 3.2, does not contain a "Content-Type" response header field (Section 3.1.1.5 of [RFC7231]), return "invalid".

Clients MUST interpret the signed payload as this specified media type instead of trying to sniff a media type from the bytes of the payload, for example by attaching an "X-Content-Type-Options: nosniff" header field ([FETCH]) to the extracted response.

9. If "integrity" names a header field and parameter that is not present in "responseHeaders" or which the client cannot use to check the integrity of "payload" (for example, the header field is new and hasn't been implemented yet), then return "invalid". If the selected header field provides integrity guarantees weaker than SHA-256, return "invalid". If validating integrity using the selected header field requires the client to process records larger than 16384 bytes, return "invalid". Clients MUST implement at least the "Digest" header field with its "mi-sha256" digest algorithm (Section 3 of [I-D.thomson-http-mice]).

Note: RFC EDITOR PLEASE DELETE THIS NOTE; Implementations of drafts of this RFC MUST recognize the draft spelling of the content encoding and digest algorithm specified by [I-D.thomson-http-mice] until that draft is published as an RFC. For example, implementations of draft-thomson-http-mice-03 would use "mi-sha256-03" and MUST NOT use "mi-sha256" itself. This

ensures that final implementations don't need to handle compatibility with implementations of early drafts of that content encoding.

If "payload" doesn't match the integrity information in the header described by "integrity", return "invalid".

10. Return "potentially-valid" with whichever is present of "certificate-chain" or "ed25519key".

Note that the above algorithm can determine that an exchange's headers are potentially-valid before the exchange's payload is received. Similarly, if "integrity" identifies a header field and parameter like "Digest:mi-sha256" ([I-D.thomson-http-mice]) that can incrementally validate the payload, early parts of the payload can be determined to be potentially-valid before later parts of the payload. Higher-level protocols MAY process parts of the exchange that have been determined to be potentially-valid as soon as that determination is made but MUST NOT process parts of the exchange that are not yet potentially-valid. Similarly, as the higher-level protocol determines that parts of the exchange are actually valid, the client MAY process those parts of the exchange and MUST wait to process other parts of the exchange until they too are determined to be valid.

3.5.1. Open Questions

Should the signed message use the TLS format (with an initial 64 spaces) even though these certificates can't be used in TLS servers?

3.6. Updating signature validity

Both OCSF responses and signatures are designed to expire a short time after they're signed, so that revoked certificates and signed exchanges with known vulnerabilities are distrusted promptly.

This specification provides no way to update OCSF responses by themselves. Instead, clients need to re-fetch the "cert-url" (Section 3.5, Paragraph 6) to get a chain including a newer OCSF response.

The "validity-url" parameter (Section 3.1) of the signatures provides a way to fetch new signatures or learn where to fetch a complete updated exchange.

Each version of a signed exchange SHOULD have its own validity URLs, since each version needs different signatures and becomes obsolete at different times.

The resource at a "validity-url" is "validity data", a CBOR map matching the following CDDL ([CDDL]):

```
validity = {  
  ? signatures: [ + bytes ]  
  ? update: {  
    ? size: uint,  
  }  
}
```

The elements of the "signatures" array are parameterised identifiers (Section 4.2.6 of [I-D.ietf-httpbis-header-structure]) meant to replace the signatures within the "Signature" header field pointing to this validity data. If the signed exchange contains a bug severe enough that clients need to stop using the content, the "signatures" array MUST NOT be present.

If the the "update" map is present, that indicates that a new version of the signed exchange is available at its effective request URI (Section 5.5 of [RFC7230]) and can give an estimate of the size of the updated exchange ("update.size"). If the signed exchange is currently the most recent version, the "update" SHOULD NOT be present.

If both the "signatures" and "update" fields are present, clients can use the estimated size to decide whether to update the whole resource or just its signatures.

3.6.1. Examples

For example, say a signed exchange whose URL is "https://example.com/resource" has the following "Signature" header field (with line breaks included and irrelevant fields omitted for ease of reading).

Signature:

```

sig1;
  sig=*MEUCIQ...*;
  ...
  validity-url="https://example.com/resource.validity.1511157180";
  cert-url="https://example.com/oldcerts";
  date=1511128380; expires=1511733180,
sig2;
  sig=*MEQCIG...*;
  ...
  validity-url="https://example.com/resource.validity.1511157180";
  cert-url="https://example.com/newcerts";
  date=1511128380; expires=1511733180,
thirdpartysig;
  sig=*MEYCIQ...*;
  ...
  validity-url="https://thirdparty.example.com/resource.validity.1511161860";
  cert-url="https://thirdparty.example.com/certs";
  date=1511478660; expires=1511824260

```

At 2017-11-27 11:02 UTC, "sig1" and "sig2" have expired, but "thirdpartysig" doesn't expire until 23:11 that night, so the client needs to fetch "https://example.com/resource.validity.1511157180" (the "validity-url" of "sig1" and "sig2") if it wishes to update those signatures. This URL might contain:

```

{
  "signatures": [
    'sig1; '
    'sig=*MEQCIC/I9Q+7BZFP6cSDsWx43pBAL0ujTbON/+7RwKVk+ba5AiB3FSFLZqpzmDJ0NumNwN0
4pqqJZE99fcK86UjkPbj4jw==*; '
    'validity-url="https://example.com/resource.validity.1511157180"; '
    'integrity="digest/mi-sha256"; '
    'cert-url="https://example.com/newcerts"; '
    'cert-sha256=*J/lEm9kNRODdCmINbvitpvdYKNQ+YgBj99DlYp4fEXw=*; '
    'date=1511733180; expires=1512337980'
  ],
  "update": {
    "size": 5557452
  }
}

```

This indicates that the client could fetch a newer version at "https://example.com/resource" (the original URL of the exchange), or that the validity period of the old version can be extended by replacing the first two of the original signatures (the ones with a validity-url of "https://example.com/resource.validity.1511157180") with the single new signature provided. (This might happen at the end of a migration to a new root certificate.) The signatures of the updated signed exchange would be:

Signature:

```
sig1;
sig=*MEQCIC...*;
...
validity-url="https://example.com/resource.validity.1511157180";
cert-url="https://example.com/newcerts";
date=1511733180; expires=1512337980,
thirdpartysig;
sig=*MEYCIQ...*;
...
validity-url="https://thirdparty.example.com/resource.validity.1511161860";
cert-url="https://thirdparty.example.com/certs";
date=1511478660; expires=1511824260
```

"https://example.com/resource.validity.1511157180" could also expand the set of signatures if its "signatures" array contained more than 2 elements.

3.7. The Accept-Signature header

"Signature" header fields cost on the order of 300 bytes for ECDSA signatures, so servers might prefer to avoid sending them to clients that don't intend to use them. A client can send the "Accept-Signature" header field to indicate that it does intend to take advantage of any available signatures and to indicate what kinds of signatures it supports.

When a server receives an "Accept-Signature" header field in a client request, it SHOULD reply with any available "Signature" header fields for its response that the "Accept-Signature" header field indicates the client supports. However, if the "Accept-Signature" value violates a requirement in this section, the server MUST behave as if it hadn't received any "Accept-Signature" header at all.

The "Accept-Signature" header field is a Structured Header as defined by [I-D.ietf-httpbis-header-structure]. Its value MUST be a parameterised list (Section 3.4 of [I-D.ietf-httpbis-header-structure]). Its ABNF is:

Accept-Signature = sh-param-list

The order of identifiers in the "Accept-Signature" list is not significant. Identifiers, ignoring any initial "-" character, MUST NOT be duplicated.

Each identifier in the "Accept-Signature" header field's value indicates that a feature of the "Signature" header field (Section 3.1) is supported. If the identifier begins with a "-" character, it instead indicates that the feature named by the rest of the identifier is not supported. Unknown identifiers and parameters MUST be ignored because new identifiers and new parameters on existing identifiers may be defined by future specifications.

3.7.1. Integrity identifiers

Identifiers starting with "digest/" indicate that the client supports the "Digest" header field ([RFC3230]) with the parameter from the HTTP Digest Algorithm Values Registry (<https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml>) registry named in lower-case by the rest of the identifier. For example, "digest/mi-blake2" indicates support for Merkle integrity with the as-yet-unspecified mi-blake2 parameter, and "-digest/mi-sha256" indicates non-support for Merkle integrity with the mi-sha256 content encoding.

If the "Accept-Signature" header field is present, servers SHOULD assume support for "digest/mi-sha256" unless the header field states otherwise.

3.7.2. Key type identifiers

Identifiers starting with "ecdsa/" indicate that the client supports certificates holding ECDSA public keys on the curve named in lower-case by the rest of the identifier.

If the "Accept-Signature" header field is present, servers SHOULD assume support for "ecdsa/secp256r1" unless the header field states otherwise.

3.7.3. Key value identifiers

The "ed25519key" identifier has parameters indicating the public keys that will be used to validate the returned signature. Each parameter's name is re-interpreted as a byte sequence (Section 3.10 of [I-D.ietf-httpbis-header-structure]) encoding a prefix of the public key. For example, if the client will validate signatures using the public key whose base64 encoding is

"11qYAYKxCrfVS/7TyWQHOG7hcvPapiMlrwIaaPcHUro=", valid "Accept-Signature" header fields include:

```
Accept-Signature: ..., ed25519key; *11qYAYKxCrfVS/7TyWQHOG7hcvPapiMlrwIaaPcHUro=*
Accept-Signature: ..., ed25519key; *11qYAYKxCrfVS/7TyWQHOG==*
Accept-Signature: ..., ed25519key; *11qYAQ==*
Accept-Signature: ..., ed25519key; **
```

but not

```
Accept-Signature: ..., ed25519key; *11qYA===*
```

because 5 bytes isn't a valid length for encoded base64, and not

```
Accept-Signature: ..., ed25519key; 11qYAQ
```

because it doesn't start or end with the "*"s that indicate a byte sequence.

Note that "ed25519key; **" is an empty prefix, which matches all public keys, so it's useful in subresource integrity (Appendix A.3) cases like "<link rel=preload as=script href=...">" where the public key isn't known until the matching "<script src=..." integrity=...">" tag.

3.7.4. Examples

```
Accept-Signature: digest/mi-sha256
```

states that the client will accept signatures with payload integrity assured by the "Digest" header and "mi-sha256" digest algorithm and implies that the client will accept signatures from ECDSA keys on the secp256r1 curve.

```
Accept-Signature: -ecdsa/secp256r1, ecdsa/secp384r1
```

states that the client will accept ECDSA keys on the secp384r1 curve but not the secp256r1 curve and payload integrity assured with the "Digest: mi-sha256" header field.

3.7.5. Open Questions

Is an "Accept-Signature" header useful enough to pay for itself? If clients wind up sending it on most requests, that may cost more than the cost of sending "Signature"s unconditionally. On the other hand, it gives servers an indication of which kinds of signatures are supported, which can help us upgrade the ecosystem in the future.

Is "Accept-Signature" the right spelling, or do we want to imitate "Want-Digest" (Section 4.3.1 of [RFC3230]) instead?

Do I have the right structure for the identifiers indicating feature support?

4. Cross-origin trust

To determine whether to trust a cross-origin exchange, the client takes a "Signature" header field (Section 3.1) and the exchange's

- * "requestUrl", a byte sequence that can be parsed into the exchange's effective request URI (Section 5.5 of [RFC7230]),
- * "responseHeaders", a byte sequence holding the canonical serialization (Section 3.4) of the CBOR representation (Section 3.2) of the exchange's response metadata and headers, and
- * "payload", a stream of bytes constituting the exchange's payload body (Section 3.3 of [RFC7230]).

The client MUST parse the "Signature" header into a list of signatures according to the instructions in Section 3.5, and run the following algorithm for each signature, stopping at the first one that returns "valid". If any signature returns "valid", return "valid". Otherwise, return "invalid".

1. If the signature's "validity-url" parameter (Section 3.1) is not same-origin (<https://html.spec.whatwg.org/multipage/origin.html#same-origin>) with "requestUrl", return "invalid".
2. Use Section 3.5 to determine the signature's validity for "requestUrl", "responseHeaders", and "payload", getting "certificate-chain" back. If this returned "invalid" or didn't return a certificate chain, return "invalid".
3. Let "response" be the response metadata and headers parsed out of "responseHeaders".
4. If Section 3 of [RFC7234] forbids a shared cache from storing "response", return "invalid".
5. If "response"'s headers contain an uncached header field, as defined in Section 4.1, return "invalid".
6. Let "authority" be the host component of "requestUrl".

7. Validate the "certificate-chain" using the following substeps. If any of them fail, re-run Section 3.5 once over the signature with the "forceFetch" flag set, and restart from step 2. If a substep fails again, return "invalid".
 1. Use "certificate-chain" to validate that its first entry, "main-certificate" is trusted as "authority"'s server certificate ([RFC5280] and other undocumented conventions). Let "path" be the path that was used from the "main-certificate" to a trusted root, including the "main-certificate" but excluding the root.
 2. Validate that "main-certificate" has the CanSignHttpExchanges extension (Section 4.2).
 3. Validate that "main-certificate" has an "ocsp" property (Section 3.3) with a valid OCSP response whose lifetime ("nextUpdate - thisUpdate") is less than 7 days ([RFC6960]). Note that this does not check for revocation of intermediate certificates, and clients SHOULD implement another mechanism for that.
 4. Validate that valid SCTs from trusted logs are available from any of:
 - * The "SignedCertificateTimestampList" in "main-certificate"'s "sct" property (Section 3.3),
 - * An OCSP extension in the OCSP response in "main-certificate"'s "ocsp" property, or
 - * An X.509 extension in the certificate in "main-certificate"'s "cert" property,as described by Section 3.3 of [RFC6962].
 8. Return "valid".
- 4.1. Uncached header fields
- Hop-by-hop and other uncached headers MUST NOT appear in a signed exchange. These will eventually be listed in [I-D.ietf-httpbis-cache], but for now they're listed here:
- * Hop-by-hop header fields listed in the Connection header field (Section 6.1 of [RFC7230]).

- * Header fields listed in the no-cache response directive in the Cache-Control header field (Section 5.2.2.2 of [RFC7234]).
- * Header fields defined as hop-by-hop:
 - Connection
 - Keep-Alive
 - Proxy-Connection
 - Trailer
 - Transfer-Encoding
 - Upgrade
- * Stateful headers as defined below.

4.1.1. Stateful header fields

As described in Section 6.1, a publisher can cause problems if they sign an exchange that includes private information. There's no way for a client to be sure an exchange does or does not include private information, but header fields that store or convey stored state in the client are a good sign.

A stateful response header field modifies state, including authentication status, in the client. The HTTP cache is not considered part of this state. These include but are not limited to:

- * "Authentication-Control", [RFC8053]
- * "Authentication-Info", [RFC7615]
- * "Clear-Site-Data", [W3C.WD-clear-site-data-20171130]
- * "Optional-WWW-Authenticate", [RFC8053]
- * "Proxy-Authenticate", [RFC7235]
- * "Proxy-Authentication-Info", [RFC7615]
- * "Public-Key-Pins", [RFC7469]
- * "Sec-WebSocket-Accept", [RFC6455]
- * "Set-Cookie", [RFC6265]

- * "Set-Cookie2", [RFC2965]
- * "SetProfile", [W3C.NOTE-OPS-OverHTTP]
- * "Strict-Transport-Security", [RFC6797]
- * "WWW-Authenticate", [RFC7235]

4.2. Certificate Requirements

We define a new X.509 extension, `CanSignHttpExchanges` to be used in the certificate when the certificate permits the usage of signed exchanges. When this extension is not present the client MUST NOT accept a signature from the certificate as proof that a signed exchange is authoritative for a domain covered by the certificate. When it is present, the client MUST follow the validation procedure in Section 4.

```
id-ce-canSignHttpExchanges OBJECT IDENTIFIER ::= { TBD }
```

```
CanSignHttpExchanges ::= NULL
```

Note that this extension contains an ASN.1 NULL (bytes "05 00") because some implementations have bugs with empty extensions.

Leaf certificates without this extension need to be revoked if the private key is exposed to an unauthorized entity, but they generally don't need to be revoked if a signing oracle is exposed and then removed.

CA certificates, by contrast, need to be revoked if an unauthorized entity is able to make even one unauthorized signature.

Certificates with this extension MUST be revoked if an unauthorized entity is able to make even one unauthorized signature.

Certificates with this extension MUST have a Validity Period no greater than 90 days.

Conforming CAs MUST NOT mark this extension as critical.

A conforming CA MUST NOT issue certificates with this extension unless, for each `dnsName` in the `subjectAltName` extension of the certificate to be issued:

1. An "issue" or "issuwild" CAA property ([RFC6844]) exists that authorizes the CA to issue the certificate; and

2. The "cansignhttpexchanges" parameter (Section 4.2.1) is present on the property and is equal to "yes"

Clients MUST NOT accept certificates with this extension in TLS connections (Section 4.4.2.2 of [RFC8446]).

RFC EDITOR PLEASE DELETE THE REST OF THE PARAGRAPHS IN THIS SECTION

```
id-ce-google OBJECT IDENTIFIER ::= { 1 3 6 1 4 1 11129 }
id-ce-canSignHttpExchangesDraft OBJECT IDENTIFIER ::= { id-ce-google 2 1 22 }
```

Implementations of drafts of this specification MAY recognize the "id-ce-canSignHttpExchangesDraft" OID as identifying the CanSignHttpExchanges extension. This OID might or might not be used as the final OID for the extension, so certificates including it might need to be reissued once the final RFC is published.

Some certificates have already been issued with this extension and with validity periods longer than 90 days. These certificates will not immediately be treated as invalid. Instead:

- * Clients MUST reject certificates with this extension that were issued after 2019-05-01 and have a Validity Period longer than 90 days.
- * After 2019-08-01, clients MUST reject all certificates with this extension that have a Validity Period longer than 90 days.

The above requirements on CAs to limit the Validity Period and check for a CAA parameter are effective starting 2019-05-01.

4.2.1. Extensions to the CAA Record: cansignhttpexchanges Parameter

A CAA parameter "cansignhttpexchanges" is defined for the "issue" and "issuewild" properties defined by [RFC6844]. The value of this parameter, if specified, MUST be "yes".

5. Transferring a signed exchange

A signed exchange can be transferred in several ways, of which three are described here.

5.1. Same-origin response

The signature for a signed exchange can be included in a normal HTTP response. Because different clients send different request header fields, clients don't know how the server's content negotiation algorithm works, and intermediate servers add response header fields, it can be impossible to have a signature for the exchange's exact request, content negotiation, and response. Therefore, when a client calls the validation procedure in Section 3.5) to validate the "Signature" header field for an exchange represented as a normal HTTP request/response pair, it MUST pass:

- * The "Signature" header field,
- * The effective request URI (Section 5.5 of [RFC7230]) of the request,
- * The serialized headers defined by Section 5.1.1, and
- * The response's payload.

If the client relies on signature validity for any aspect of its behavior, it MUST ignore any header fields that it didn't pass to the validation procedure.

If the signed response includes a "Variants" header field, the client MUST use the cache behavior algorithm in Section 4 of [I-D.ietf-httpbis-variants] to check that the signed response is an appropriate representation for the request the client is trying to fulfil. If the response is not an appropriate representation, the client MUST treat the signature as invalid.

5.1.1. Serialized headers for a same-origin response

The serialized headers of an exchange represented as a normal HTTP request/response pair (Section 2.1 of [RFC7230] or Section 8.1 of [RFC7540]) are the canonical serialization (Section 3.4) of the CBOR representation (Section 3.2) of the response status code (Section 6 of [RFC7231]) and the response header fields whose names are listed in that response's "Signed-Headers" header field (Section 5.1.2). If a response header field name from "Signed-Headers" does not appear in the response's header fields, the exchange has no serialized headers.

If the exchange's "Signed-Headers" header field is not present, doesn't parse as a Structured Header ([I-D.ietf-httpbis-header-structure]) or doesn't follow the constraints on its value described in Section 5.1.2, the exchange has no serialized headers.

5.1.1.1. Open Questions

Do the serialized headers of an exchange need to include the "Signed-Headers" header field itself?

5.1.2. The Signed-Headers Header

The "Signed-Headers" header field identifies an ordered list of response header fields to include in a signature. The request URL and response status are included unconditionally. This allows a TLS-terminating intermediate to reorder headers without breaking the signature. This *can* also allow the intermediate to add headers that will be ignored by some higher-level protocols, but Section 3.5 provides a hook to let other higher-level protocols reject such insecure headers.

This header field appears once instead of being incorporated into the signatures' parameters because the signed header fields need to be consistent across all signatures of an exchange, to avoid forcing higher-level protocols to merge the header field lists of valid signatures.

"Signed-Headers" is a Structured Header as defined by [I-D.ietf-httpbis-header-structure]. Its value MUST be a list (Section 3.2 of [I-D.ietf-httpbis-header-structure]). Its ABNF is:

```
Signed-Headers = sh-list
```

Each element of the "Signed-Headers" list must be a lowercase string (Section 3.8 of [I-D.ietf-httpbis-header-structure]) naming an HTTP response header field. Pseudo-header field names (Section 8.1.2.1 of [RFC7540]) MUST NOT appear in this list.

Higher-level protocols SHOULD place requirements on the minimum set of headers to include in the "Signed-Headers" header field.

5.2. HTTP/2 extension for cross-origin Server Push

To allow servers to Server-Push (Section 8.2 of [RFC7540]) signed exchanges (Section 3) signed by an authority for which the server is not authoritative (Section 9.1 of [RFC7230]), this section defines an HTTP/2 extension.

5.2.1. Indicating support for cross-origin Server Push

Clients that might accept signed Server Pushes with an authority for which the server is not authoritative indicate this using the HTTP/2 SETTINGS parameter `ENABLE_CROSS_ORIGIN_PUSH` (0xSETTING-TBD).

An `ENABLE_CROSS_ORIGIN_PUSH` value of 0 indicates that the client does not support cross-origin Push. A value of 1 indicates that the client does support cross-origin Push.

A client **MUST NOT** send a `ENABLE_CROSS_ORIGIN_PUSH` setting with a value other than 0 or 1 or a value of 0 after previously sending a value of 1. If a server receives a value that violates these rules, it **MUST** treat it as a connection error (Section 5.4.1 of [RFC7540]) of type `PROTOCOL_ERROR`.

The use of a `SETTINGS` parameter to opt-in to an otherwise incompatible protocol change is a use of "Extending HTTP/2" defined by Section 5.5 of [RFC7540]. If a server were to send a cross-origin Push without first receiving a `ENABLE_CROSS_ORIGIN_PUSH` setting with the value of 1 it would be a protocol violation.

5.2.2. `NO_TRUSTED_EXCHANGE_SIGNATURE` error code

The signatures on a Pushed cross-origin exchange may be untrusted for several reasons, for example that the certificate could not be fetched, that the certificate does not chain to a trusted root, that the signature itself doesn't validate, that the signature is expired, etc. This draft conflates all of these possible failures into one error code, `NO_TRUSTED_EXCHANGE_SIGNATURE` (0xERROR-TBD).

5.2.2.1. Open Questions

How fine-grained should this specification's error codes be?

5.2.3. Validating a cross-origin Push

If the client has set the `ENABLE_CROSS_ORIGIN_PUSH` setting to 1, the server **MAY** Push a signed exchange for which it is not authoritative, and the client **MUST NOT** treat a `PUSH_PROMISE` for which the server is not authoritative as a stream error (Section 5.4.2 of [RFC7540]) of type `PROTOCOL_ERROR`, as described in Section 8.2 of [RFC7540], unless there is another error as described below.

Instead, the client **MUST** validate such a `PUSH_PROMISE` and its response against the following list:

1. If the `PUSH_PROMISE` includes any non-pseudo request header fields, the client **MUST** treat it as a stream error (Section 5.4.2 of [RFC7540]) of type `PROTOCOL_ERROR`.
2. If the `PUSH_PROMISE`'s method is not "GET", the client **MUST** treat it as a stream error (Section 5.4.2 of [RFC7540]) of type `PROTOCOL_ERROR`.

3. Run the algorithm in Section 4 over:

- * The "Signature" header field from the response.
- * The effective request URI from the PUSH_PROMISE.
- * The canonical serialization (Section 3.4) of the CBOR representation (Section 3.2) of the pushed response's status and its headers except for the "Signature" header field.
- * The response's payload.

If this returns "invalid", the client MUST treat the response as a stream error (Section 5.4.2 of [RFC7540]) of type NO_TRUSTED_EXCHANGE_SIGNATURE. Otherwise, the client MUST treat the pushed response as if the server were authoritative for the PUSH_PROMISE's authority.

5.2.3.1. Open Questions

Is it right that "validity-url" is required to be same-origin with the exchange? This allows the mitigation against downgrades in Section 6.3, but prohibits intermediates from providing a cache of the validity information. We could do both with a list of URLs.

5.3. application/signed-exchange format

To allow signed exchanges to be the targets of "<link rel=prefetch>" tags, we define the "application/signed-exchange" content type that represents a signed HTTP exchange, including a request URL, response metadata and header fields, and a response payload.

When served over HTTP, a response containing an "application/signed-exchange" payload MUST include at least the following response header fields, to reduce content sniffing vulnerabilities (Section 6.8):

- * Content-Type: application/signed-exchange;v=_version_
- * X-Content-Type-Options: nosniff

This content type consists of the concatenation of the following items:

1. 8 bytes consisting of the ASCII characters "sxg1" followed by 4 0x00 bytes, to serve as a file signature. This is redundant with the MIME type, and recipients that receive both MUST check that they match and stop parsing if they don't.

Note: RFC EDITOR PLEASE DELETE THIS NOTE; The implementation of the final RFC MUST use this file signature, but implementations of drafts MUST NOT use it and MUST use another implementation-specific 8-byte string beginning with "sxml-".

2. 2 bytes storing a big-endian integer "fallbackUrlLength".
3. "fallbackUrlLength" bytes holding a "fallbackUrl", which MUST UTF-8 decode to an absolute URL with a scheme of "https".

Note: The byte location of the fallback URL is intended to remain invariant across versions of the "application/signed-exchange" format so that parsers encountering unknown versions can always find a URL to redirect to.

Issue: Should this fallback information also include the method?

4. 3 bytes storing a big-endian integer "sigLength". If this is larger than 16384 (16×1024), parsing MUST fail.
5. 3 bytes storing a big-endian integer "headerLength". If this is larger than 524288 (512×1024), parsing MUST fail.
6. "sigLength" bytes holding the "Signature" header field's value (Section 3.1).
7. "headerLength" bytes holding "signedHeaders", the canonical serialization (Section 3.4) of the CBOR representation of the response headers of the exchange represented by the "application/signed-exchange" resource (Section 3.2), excluding the "Signature" header field.
8. The payload body (Section 3.3 of [RFC7230]) of the exchange represented by the "application/signed-exchange" resource.

Note that the use of the payload body here means that a "Transfer-Encoding" header field inside the "application/signed-exchange" header block has no effect. A "Transfer-Encoding" header field on the outer HTTP response that transfers this resource still has its normal effect.

5.3.1. Cross-origin trust in application/signed-exchange

To determine whether to trust a cross-origin exchange stored in an "application/signed-exchange" resource, pass the "Signature" header field's value, "fallbackUrl" as the effective request URI, "signedHeaders", and the payload body to the algorithm in Section 4.

5.3.2. Example

An example "application/signed-exchange" file representing a possible signed exchange with `https://example.com/` (`https://example.com/`) follows, with lengths represented by descriptions in "<>"s, CBOR represented in the extended diagnostic format defined in Appendix G of [CDDL], and most of the "Signature" header field and payload elided with a:

```
sxg1\0\0\0\0<2-byte length of the following url string>
https://example.com/<3-byte length of the following header
value><3-byte length of the encoding of the
following map>sig1; sig=*...; integrity="digest/mi-sha256"; ...{
  'status': '200',
  'content-type': 'text/html'
}<!doctype html>\r\n<html>...
```

5.3.3. Open Questions

Should this be a CBOR format, or is the current mix of binary and CBOR better?

Are the mime type, extension, and magic number right?

6. Security considerations

6.1. Over-signing

If a publisher blindly signs all responses as their origin, they can cause at least two kinds of problems, described below. To avoid this, publishers SHOULD design their systems to opt particular public content that doesn't depend on authentication status into signatures instead of signing by default.

Signing systems SHOULD also incorporate the following mitigations to reduce the risk that private responses are signed:

1. Strip the "Cookie" request header field and other identifying information like client authentication and TLS session IDs from requests whose exchange is destined to be signed, before forwarding the request to a backend.
2. Only sign exchanges where the response includes a "Cache-Control: public" header. Clients are not required to fail signature-checking for exchanges that omit this "Cache-Control" response header field to reduce the risk that naive signing systems blindly add it.

6.1.1. Session fixation

Blind signing can sign responses that create session cookies or otherwise change state on the client to identify a particular session. This breaks certain kinds of CSRF defense and can allow an attacker to force a user into the attacker's account, where the user might unintentionally save private information, like credit card numbers or addresses.

This specification defends against cookie-based attacks by blocking the "Set-Cookie" response header, but it cannot prevent Javascript or other response content from changing state.

6.1.2. Misleading content

If a site signs private information, an attacker might set up their own account to show particular private information, forward that signed information to a victim, and use that victim's confusion in a more sophisticated attack.

Stripping authentication information from requests before sending them to backends is likely to prevent the backend from showing attacker-specific information in the signed response. It does not prevent the attacker from showing their victim a signed-out page when the victim is actually signed in, but while this is still misleading, it seems less likely to be useful to the attacker.

6.2. Off-path attackers

Relaxing the requirement to consult DNS when determining authority for an origin means that an attacker who possesses a valid certificate no longer needs to be on-path to redirect traffic to them; instead of modifying DNS or IP routing, they need only convince the user to visit another Web site in order to serve responses signed as the target. This consideration and mitigations for it are shared by the combination of [RFC8336] and [I-D.ietf-httpbis-http2-secondary-certs], and are discussed further in [I-D.bishop-httpbis-origin-fed-up].

6.2.1. Mis-issued certificates

If a CA mis-issues a certificate for a domain, this specification provides a way to detect the mis-issuance and mitigate harm within approximately two weeks. Specifically, because all signed exchanges must include a "SignedCertificateTimestampList" ([RFC6962], a CT log has promised to publish the mis-issued certificate within that log's Maximum Merge Delay, 1 day for many logs. The domain owner can then detect the mis-issued certificate and notify the CA to revoke it,

which the [BRs], section 4.9.1.1, say they must do within another 5 days.

Once the mis-issued certificate is revoked, existing OCSP responses begin to expire. The [BRs], section 4.9.10, require that OCSP responses have a maximum expiration time of 10 days, after which they can't be used to validate a certificate chain (Section 3.3). This leads to a total compromised time of 16 days after a mis-issuance.

However, CAs might future-date their OCSP responses, in which case the mitigation doesn't work.

CAs are forbidden from future-dating their OCSP responses by the [BRs] section 4.9.9, "OCSP responses MUST conform to RFC6960 and/or RFC5019." [RFC6960] includes, "The time at which the status was known to be correct SHALL be reflected in the thisUpdate field of the response.", and [RFC5019] includes, "When pre-producing OCSPResponse messages, the responder MUST set the thisUpdate, nextUpdate, and producedAt times as follows: thisUpdate: The time at which the status being indicated is known to be correct."

However, if a CA violates the [BRs] to sign future-dated OCSP responses, attempts to keep the nonconformant OCSP responses private, but then leaks them, it could cause clients to trust a hostile signed exchange long after its certificate has been revoked.

Clients could use systems like [CRLSets] and [OneCrl] to revoke the intermediate certificate that signed the future-dated OCSP responses.

6.2.2. Stolen private keys

If the private key for a CanSignHttpExchanges certificate is stolen, it can be used at scale until the certificate expires or is revoked, and unlike for a stolen key for a normal TLS-terminating certificate, the rightful owner can't detect the problem by watching for attacks on the DNS or routing infrastructure.

This specification does not currently propose a way for the rightful owner to detect that their keys are being used by an attacker, after they've opted into the risk by requesting a CanSignHttpExchanges certificate in the first place. Clients can fetch a signature's "validity-url" (Section 3.1) to help owners detect key compromise, but that compromises some of the privacy properties of this specification.

6.3. Downgrades

Signing a bad response can affect more users than simply serving a bad response, since a served response will only affect users who make a request while the bad version is live, while an attacker can forward a signed response until its signature expires. Publishers should consider shorter signature expiration times than they use for cache expiration times.

Clients MAY also check the "validity-url" (Section 3.1) of an exchange more often than the signature's expiration would require. Doing so for an exchange with an HTTPS request URI provides a TLS guarantee that the exchange isn't out of date (as long as Section 5.2.3.1 is resolved to keep the same-origin requirement).

6.4. Signing oracles are permanent

An attacker with temporary access to a signing oracle can sign "still valid" assertions with arbitrary timestamps and expiration times. As a result, when a signing oracle is removed, the keys it provided access to MUST be revoked so that, even if the attacker used them to sign future-dated exchange validity assertions, the key's OCSF assertion will expire, causing the exchange as a whole to become untrusted.

6.5. Unsigned headers

The use of a single "Signed-Headers" header field prevents us from signing aspects of the request other than its effective request URI (Section 5.5 of [RFC7230]). For example, if a publisher signs both "Content-Encoding: br" and "Content-Encoding: gzip" variants of a response, what's the impact if an attacker serves the brotli one for a request with "Accept-Encoding: gzip"? This is mitigated by using [I-D.ietf-httpbis-variants] instead of request headers to describe how the client should run content negotiation.

The simple form of "Signed-Headers" also prevents us from signing less than the full request URL. The SRI use case (Appendix A.3) may benefit from being able to leave the authority less constrained.

Section 3.5 can succeed when some delivered headers aren't included in the signed set. This accommodates current TLS-terminating intermediates and may be useful for SRI (Appendix A.3), but is risky for trusting cross-origin responses (Appendix A.1, Appendix A.2, and Appendix A.6). Section 5.2 requires all headers to be included in the signature before trusting cross-origin pushed resources, at Ryan Sleevi's recommendation.

6.6. application/signed-exchange

Clients MUST NOT trust an effective request URI claimed by an "application/signed-exchange" resource (Section 5.3) without either ensuring the resource was transferred from a server that was authoritative (Section 9.1 of [RFC7230]) for that URI's origin, or calling the algorithm in Section 5.3.1 and getting "valid" back.

6.7. Key re-use with TLS

In general, key re-use across multiple protocols is a bad idea.

Using an exchange-signing key in a TLS (or other directly-internet-facing) server increases the risk that an attacker can steal the private key, which will allow them to mint packages (similar to Section 6.4) until their theft is discovered.

Using a TLS key in a CanSignHttpExchanges certificate makes it less likely that the server operator will discover key theft, due to the considerations in Section 6.2.

This specification uses the CanSignHttpExchanges X.509 extension (Section 4.2) to discourage re-use of TLS keys to sign exchanges or vice-versa.

We require that clients reject certificates with the CanSignHttpExchanges extension when making TLS connections to minimize the chance that servers will re-use keys like this. Ideally, we would make the extension critical so that even clients that don't understand it would reject such TLS connections, but this proved impossible because certificate-validating libraries ship on significantly different schedules from the clients that use them.

Even once all clients reject these certificates in TLS connections, this will still just discourage and not prevent key re-use, since a server operator can unwisely request two different certificates with the same private key.

6.8. Content sniffing

While modern browsers tend to trust the "Content-Type" header sent with a resource, especially when accompanied by "X-Content-Type-Options: nosniff", plugins will sometimes search for executable content buried inside a resource and execute it in the context of the origin that served the resource, leading to XSS vulnerabilities. For example, some PDF reader plugins look for "%PDF" anywhere in the first 1kB and execute the code that follows it.

The "application/signed-exchange" format (Section 5.3) includes a URL and response headers early in the format, which an attacker could use to cause these plugins to sniff a bad content type.

To avoid vulnerabilities, in addition to the response header requirements in Section 5.3, servers are advised to only serve an "application/signed-exchange" resource (SXG) from a domain if it would also be safe for that domain to serve the SXG's content directly, and to follow at least one of the following strategies:

1. Only serve signed exchanges from dedicated domains that don't have access to sensitive cookies or user storage.
2. Generate signed exchanges "offline", that is, in response to a trusted author submitting content or existing signatures reaching a certain age, rather than in response to untrusted-reader queries.
3. Do all of:
 1. If the SXG's fallback URL (Section 5.3) is derived from the request URL, percent-encode (<https://url.spec.whatwg.org/#percent-encode>) ([URL]) any bytes that are greater than 0x7E or are not URL code points (<https://url.spec.whatwg.org/#url-code-points>) ([URL]) in the fallback URL . It is particularly important to make sure no unescaped nulls (0x00) or angle brackets (0x3C and 0x3E) appear.
 2. Do not reflect request header fields into the set of response headers.

There are still a few binary length fields that an attacker may influence to contain sensitive bytes, but they're always followed by lowercase alphabetic strings from a small set of possibilities, which reduces the chance that a client will sniff them as indicating a particular content type.

To encourage servers to include the "X-Content-Type-Options: nosniff" header field, clients SHOULD reject signed exchanges served without it.

7. Privacy considerations

7.1. Visibility of resource requests

Normally, when a client follows a link from `https://source.example/page.html` to `"https://publisher.example/page.html"`, `"publisher.example"` learns that the client is interested in the resource. `"source.example"` also has several ways of discovering that the client has clicked the link, including the use of Javascript to record the click or having the link point to a URL that serves a 302 redirect to the real target.

If `"publisher.example"` signs `"page.html"` into `"page.sxg"`, `"distributor.example"` serves it as `"https://distributor.example/publisher/page.sxg"`, and the client fetches it from there, then `"distributor.example"` learns that the client is interested, and if the client executes some Javascript on the page or makes subresource requests, that could also report the client's interest back to `"publisher.example"`.

To prevent network operators other than `"distributor.example"` or `"publisher.example"` from learning which exchanges were read, clients SHOULD only load exchanges fetched over a transport that's protected from eavesdroppers. This can be difficult to determine when the exchange is being loaded from local disk, but when the client itself requested the exchange over a network it SHOULD require TLS ([RFC8446]) or a successor transport layer, and MUST NOT accept exchanges transferred over plain HTTP without TLS.

If `"source.example"` and `"distributor.example"` are controlled by the same entity, no extra information escapes here. If they are run by different entities, a similar amount of information escapes as if `"source.example"` had implemented its click tracking by outsourcing to a service like `https://bit.ly/` (`https://bit.ly/`).

There has been discussion of allowing a publisher to restrict the set of distributors that can host its signed content. If that's added, then the privacy situation becomes more similar to the situation with CDNs, where a publisher chooses a CDN to serve their content, and the CDN learns about all requests for that content. Here the publisher would choose one or more distributors, and the distributor(s) would learn about requests for the content.

For non-executable resource types, a signed response can improve the privacy situation by hiding the client's interest from the original publisher.

7.2. User ID transfer

If a request for "https://distributor.example/publisher/page.sxg" comes with the source's or distributor's user ID for the user, either because it's sent with the distributor's cookies or because the source stashes an encoded user ID into either the request's path or a subdomain, the distributor has a few ways to pass that user ID on to the publisher that signed the page:

1. If the distributor has the publisher's signing keys, it can sign a new page with its user ID directly embedded.
2. Otherwise, the publisher can sign lots of copies of their package, and the distributor can choose a particular copy to send a subset of the bits in its user ID to the publisher on each click, which will eventually transfer the whole thing.

To prevent this, the request for a signed exchange needs to omit credentials and block them from appearing in the URL in the same way it would block them from appearing in a cross-origin URL. We're exploring ways the link can mark the request so user agents can take the right counter-measures.

8. IANA considerations

TODO: possibly register the validity-url format.

8.1. Signature Header Field Registration

This section registers the "Signature" header field in the "Permanent Message Header Field Names" registry ([RFC3864]).

Header field name: "Signature"

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): Section 3.1 of this document

8.2. Accept-Signature Header Field Registration

This section registers the "Accept-Signature" header field in the "Permanent Message Header Field Names" registry ([RFC3864]).

Header field name: "Accept-Signature"

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): Section 3.7 of this document

8.3. Signed-Headers Header Field Registration

This section registers the "Signed-Headers" header field in the "Permanent Message Header Field Names" registry ([RFC3864]).

Header field name: "Signed-Headers"

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): Section 5.1.2 of this document

8.4. HTTP/2 Settings

This section establishes an entry for the HTTP/2 Settings Registry that was established by Section 11.3 of [RFC7540]

Name: ENABLE_CROSS_ORIGIN_PUSH

Code: 0xSETTING-TBD

Initial Value: 0

Specification: This document

8.5. HTTP/2 Error code

This section establishes an entry for the HTTP/2 Error Code Registry that was established by Section 11.4 of [RFC7540]

Name: NO_TRUSTED_EXCHANGE_SIGNATURE

Code: 0xERROR-TBD

Description: The client does not trust the signature for a cross-origin Pushed signed exchange.

Specification: This document

8.6. Internet Media Type application/signed-exchange

IANA is requested to register the MIME media type ([IANA.media-types]) for signed exchanges, application/signed-exchange, as follows:

Type name: application

Subtype name: signed-exchange

Required parameters:

- * v: A string denoting the version of the file format. ([RFC5234] ABNF: "version = DIGIT/%x61-7A") The version defined in this specification is "1". When used with the "Accept" header field (Section 5.3.2 of [RFC7231]), this parameter can be a comma (,)-separated list of version strings. ([RFC5234] ABNF: "version-list = version *("," version)") The server is then expected to reply with a resource using a particular version from that list.

Note: RFC EDITOR PLEASE DELETE THIS NOTE; Implementations of drafts of this specification MUST NOT use simple integers to describe their versions, and MUST instead define implementation-specific strings to identify which draft is implemented. The newest version of [I-D.yasskin-httpbis-origin-signed-exchanges-impl] describes the meaning of one such string.

Optional parameters: N/A

Encoding considerations: binary

Security considerations: see Section 6.6

Interoperability considerations: N/A

Published specification: This specification (see Section 5.3).

Applications that use this media type: N/A

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): 73 78 67 31 00

File extension(s): .sxxg

Macintosh file type code(s): N/A

Person and email address to contact for further information: See Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Authors' Addresses section.

Change controller: IESG

Provisional registration? Yes

8.7. Internet Media Type application/cert-chain+cbor

IANA is requested to register the MIME media type ([IANA.media-types]) for CBOR-format certificate chains, application/cert-chain+cbor, as follows:

Type name: application

Subtype name: cert-chain+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: N/A

Interoperability considerations: N/A

Published specification: This specification (see Section 3.3).

Applications that use this media type: N/A

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): 1*9(??) 67 F0 9F 93 9C E2 9B 93

File extension(s): N/A

Macintosh file type code(s): N/A

Person and email address to contact for further information: See Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Authors' Addresses section.

Change controller: IESG

Provisional registration? Yes

8.8. The cansignhttpexchanges CAA Parameter

There are no IANA considerations for this parameter.

9. References

9.1. Normative References

[CDDL] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

[FETCH] WHATWG, "Fetch", July 2020, <<https://fetch.spec.whatwg.org/>>.

[I-D.ietf-httpbis-header-structure] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-header-structure-19, 3 June 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-httpbis-header-structure-19.txt>>.

[I-D.ietf-httpbis-variants] Nottingham, M., "HTTP Representation Variants", Work in Progress, Internet-Draft, draft-ietf-httpbis-variants-06, 3 November 2019, <<http://www.ietf.org/internet-drafts/draft-ietf-httpbis-variants-06.txt>>.

- [I-D.thomson-http-mice]
Thomson, M. and J. Yasskin, "Merkle Integrity Content Encoding", Work in Progress, Internet-Draft, draft-thomson-http-mice-03, 13 August 2018, <<http://www.ietf.org/internet-drafts/draft-thomson-http-mice-03.txt>>.
- [IANA.media-types]
IANA, "Media Types", <<http://www.iana.org/assignments/media-types>>.
- [POSIX] IEEE and The Open Group, "The Open Group Base Specifications Issue 7", value 1003.1-2008, 2016 Edition, name IEEE, 2016, <<http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/>>.
- [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>>.
- [RFC3230] Mogul, J. and A. Van Hoff, "Instance Digests in HTTP", RFC 3230, DOI 10.17487/RFC3230, January 2002, <<https://www.rfc-editor.org/info/rfc3230>>.
- [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>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6844] Hallam-Baker, P. and R. Stradling, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 6844, DOI 10.17487/RFC6844, January 2013, <<https://www.rfc-editor.org/info/rfc6844>>.

- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.
- [RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <<https://www.rfc-editor.org/info/rfc6962>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [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>>.
- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/info/rfc8032>>.
- [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>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [URL] WHATWG, "URL", July 2020, <<https://url.spec.whatwg.org/>>.

9.2. Informative References

- [BRs] CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates", 10 December 2018, <<https://cabforum.org/baseline-requirements-documents/>>.
- [CRLSets] Langley, A., "Revocation checking and Chrome's CRL", 5 February 2012, <<https://www.imperialviolet.org/2012/02/05/crlsets.html>>.
- [I-D.bishop-httpbis-origin-fed-up] Bishop, M. and E. Nygren, "DNS Security with HTTP/2 ORIGIN", Work in Progress, Internet-Draft, draft-bishop-httpbis-origin-fed-up-00, 8 January 2019, <<http://www.ietf.org/internet-drafts/draft-bishop-httpbis-origin-fed-up-00.txt>>.
- [I-D.burke-content-signature] Burke, B., "HTTP Header for digital signatures", Work in Progress, Internet-Draft, draft-burke-content-signature-00, 7 March 2011, <<http://www.ietf.org/internet-drafts/draft-burke-content-signature-00.txt>>.
- [I-D.cavage-http-signatures] Cavage, M. and M. Sporny, "Signing HTTP Messages", Work in Progress, Internet-Draft, draft-cavage-http-signatures-12, 21 October 2019, <<http://www.ietf.org/internet-drafts/draft-cavage-http-signatures-12.txt>>.
- [I-D.ietf-httpbis-cache] Fielding, R., Nottingham, M., and J. Reschke, "HTTP Caching", Work in Progress, Internet-Draft, draft-ietf-httpbis-cache-10, 12 July 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-httpbis-cache-10.txt>>.
- [I-D.ietf-httpbis-http2-secondary-certs] Bishop, M., Sullivan, N., and M. Thomson, "Secondary Certificate Authentication in HTTP/2", Work in Progress, Internet-Draft, draft-ietf-httpbis-http2-secondary-certs-06, 14 May 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-httpbis-http2-secondary-certs-06.txt>>.

- [I-D.thomson-http-content-signature]
Thomson, M., "Content-Signature Header Field for HTTP",
Work in Progress, Internet-Draft, draft-thomson-http-
content-signature-00, 2 July 2015, <[http://www.ietf.org/
internet-drafts/draft-thomson-http-content-signature-
00.txt](http://www.ietf.org/internet-drafts/draft-thomson-http-content-signature-00.txt)>.
- [I-D.yasskin-httpbis-origin-signed-exchanges-impl]
Yasskin, J. and K. Ueno, "Signed HTTP Exchanges
Implementation Checkpoints", Work in Progress, Internet-
Draft, draft-yasskin-httpbis-origin-signed-exchanges-impl-
03, 25 July 2019, <[http://www.ietf.org/internet-drafts/
draft-yasskin-httpbis-origin-signed-exchanges-impl-
03.txt](http://www.ietf.org/internet-drafts/draft-yasskin-httpbis-origin-signed-exchanges-impl-03.txt)>.
- [I-D.yasskin-wpack-use-cases]
Yasskin, J., "Use Cases and Requirements for Web
Packages", Work in Progress, Internet-Draft, draft-
yasskin-wpack-use-cases-00, 30 October 2019,
<[http://www.ietf.org/internet-drafts/draft-yasskin-wpack-
use-cases-00.txt](http://www.ietf.org/internet-drafts/draft-yasskin-wpack-use-cases-00.txt)>.
- [OneCrl] Goodwin, M., "Revoking Intermediate Certificates:
Introducing OneCRL", 3 March 2015,
<[https://blog.mozilla.org/security/2015/03/03/revoking-
intermediate-certificates-introducing-onecrl/](https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/)>.
- [RFC2965] Kristol, D. and L. Montulli, "HTTP State Management
Mechanism", RFC 2965, DOI 10.17487/RFC2965, October 2000,
<<https://www.rfc-editor.org/info/rfc2965>>.
- [RFC5019] Deacon, A. and R. Hurst, "The Lightweight Online
Certificate Status Protocol (OCSP) Profile for High-Volume
Environments", RFC 5019, DOI 10.17487/RFC5019, September
2007, <<https://www.rfc-editor.org/info/rfc5019>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS)
Extensions: Extension Definitions", RFC 6066,
DOI 10.17487/RFC6066, January 2011,
<<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265,
DOI 10.17487/RFC6265, April 2011,
<<https://www.rfc-editor.org/info/rfc6265>>.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454,
DOI 10.17487/RFC6454, December 2011,
<<https://www.rfc-editor.org/info/rfc6454>>.

- [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <<https://www.rfc-editor.org/info/rfc6455>>.
- [RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <<https://www.rfc-editor.org/info/rfc6797>>.
- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<https://www.rfc-editor.org/info/rfc7235>>.
- [RFC7469] Evans, C., Palmer, C., and R. Slevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <<https://www.rfc-editor.org/info/rfc7469>>.
- [RFC7615] Reschke, J., "HTTP Authentication-Info and Proxy-Authentication-Info Response Header Fields", RFC 7615, DOI 10.17487/RFC7615, September 2015, <<https://www.rfc-editor.org/info/rfc7615>>.
- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [RFC8053] Oiwa, Y., Watanabe, H., Takagi, H., Maeda, K., Hayashi, T., and Y. Ioku, "HTTP Authentication Extensions for Interactive Clients", RFC 8053, DOI 10.17487/RFC8053, January 2017, <<https://www.rfc-editor.org/info/rfc8053>>.
- [RFC8336] Nottingham, M. and E. Nygren, "The ORIGIN HTTP/2 Frame", RFC 8336, DOI 10.17487/RFC8336, March 2018, <<https://www.rfc-editor.org/info/rfc8336>>.
- [SRI] Akhawe, D., Braun, F., Marier, F., and J. Weinberger, "Subresource Integrity", World Wide Web Consortium Recommendation REC-SRI-20160623, 23 June 2016, <<http://www.w3.org/TR/2016/REC-SRI-20160623>>.
- [W3C.NOTE-OPS-OverHTTP] Hensley, P., Metral, M., Shardanand, U., Converse, D., and M. Myers, "Implementation of OPS Over HTTP", W3C NOTE NOTE-OPS-OverHTTP, 2 June 1997, <<http://www.w3.org/TR/NOTE-OPS-OverHTTP>>.

[W3C.WD-clear-site-data-20171130]

West, M., "Clear Site Data", World Wide Web Consortium WD
WD-clear-site-data-20171130, 30 November 2017,
<<https://www.w3.org/TR/2017/WD-clear-site-data-20171130>>.

Appendix A. Use cases

A.1. PUSHed subresources

To reduce round trips, a server might use HTTP/2 Push (Section 8.2 of [RFC7540]) to inject a subresource from another server into the client's cache. If anything about the subresource is expired or can't be verified, the client would fetch it from the original server.

For example, if "<https://example.com/index.html>" includes

```
<script src="https://jquery.com/jquery-1.2.3.min.js">
```

Then to avoid the need to look up and connect to "jquery.com" in the critical path, "example.com" might push that resource signed by "jquery.com".

A.2. Explicit use of a content distributor for subresources

In order to speed up loading but still maintain control over its content, an HTML page in a particular origin "O.com" could tell clients to load its subresources from an intermediate content distributor that's not authoritative, but require that those resources be signed by "O.com" so that the distributor couldn't modify the resources. This is more constrained than the common CDN case where "O.com" has a CNAME granting the CDN the right to serve arbitrary content as "O.com".

```

```

To make it easier to configure the right distributor for a given request, computation of the "physicalsrc" could be encapsulated in a custom element:

```
<dist-img src="https://O.com/img.png"></dist-img>
```

where the "<dist-img>" implementation generates an appropriate "" based on, for example, a "<meta name='dist-base'>" tag elsewhere in the page. However, this has the downside that the preloader (<https://calendar.perfplanet.com/2013/big-bad-preloader/>) can no longer see the physical source to download it. The resulting delay might cancel out the benefit of using a distributor.

This could be used for some of the same purposes as SRI (Appendix A.3).

To implement this with the current proposal, the distributor would respond to the physical request to "<https://distributor.com/O.com/img.png>" with first a signed PUSH_PROMISE for "<https://O.com/img.png>" and then a redirect to "<https://O.com/img.png>".

A.3. Subresource Integrity

The W3C WebAppSec group is investigating using signatures (<https://github.com/mikewest/signature-based-sri>) in [SRI]. They need a way to transmit the signature with the response, which this proposal provides.

Their needs are simpler than most other use cases in that the "integrity="ed25519-[public-key]" attribute and CSP-based ways of expressing a public key don't need that key to be wrapped into a certificate.

The "ed25519key" signature parameter supports this simpler way of attaching a key.

The current proposal for signature-based SRI describes signing only the content of a resource, while this specification requires them to sign the request URI as well. This issue is tracked in <https://github.com/mikewest/signature-based-sri/issues/5> (<https://github.com/mikewest/signature-based-sri/issues/5>). The details of what they need to sign will affect whether and how they can use this proposal.

A.4. Binary Transparency

So-called "Binary Transparency" may eventually allow users to verify that a program they've been delivered is one that's available to the public, and not a specially-built version intended to attack just them. Binary transparency systems don't exist yet, but they're likely to work similarly to the successful Certificate Transparency logs described by [RFC6962].

Certificate Transparency depends on Signed Certificate Timestamps that prove a log contained a particular certificate at a particular time. To build the same thing for Binary Transparency logs containing HTTP resources or full websites, we'll need a way to provide signatures of those resources, which signed exchanges provides.

A.5. Static Analysis

Native app stores like the Apple App Store (<https://www.apple.com/ios/app-store/>) and the Android Play Store (<https://play.google.com/store>) grant their contents powerful abilities, which they attempt to make safe by analyzing the applications before offering them to people. The web has no equivalent way for people to wait to run an update of a web application until a trusted authority has vouched for it.

While full application analysis probably needs to wait until the authority can sign bundles of exchanges, authorities may be able to guarantee certain properties by just checking a top-level resource and its [SRI]-constrained sub-resources.

A.6. Offline websites

Fully-offline websites can be represented as bundles of signed exchanges, although an optimization to reduce the number of signature verifications may be needed. Work on this is in progress in the <https://github.com/WICG/webpackage> (<https://github.com/WICG/webpackage>) repository.

Appendix B. Requirements

B.1. Proof of origin

To verify that a thing came from a particular origin, for use in the same context as a TLS connection, we need someone to vouch for the signing key with as much verification as the signing keys used in TLS. The obvious way to do this is to re-use the web PKI and CA ecosystem.

B.1.1. Certificate constraints

If we re-use existing TLS server certificates, we incur the risks that:

1. TLS server certificates must be accessible from online servers, so they're easier to steal or use as signing oracles than an offline key. An exchange's signing key doesn't need to be online.
2. A server using an origin-trusted key for one purpose (e.g. TLS) might accidentally sign something that looks like an exchange, or vice versa.

These risks are considered too high, so we define a new X.509 certificate extension in Section 4.2 that requires CAs to issue new certificates for this purpose. We expect at least one low-cost CA to be willing to sign certificates with this extension.

B.1.2. Signature constraints

In order to prevent an attacker who can convince the server to sign some resource from causing those signed bytes to be interpreted as something else the new X.509 extension here is forbidden from being used in TLS servers. If Section 4.2 changes to allow re-use in TLS servers, we would need to:

1. Avoid key types that are used for non-TLS protocols whose output could be confused with a signature. That may be just the "rsaEncryption" OID from [RFC8017].
2. Use the same format as TLS's signatures, specified in Section 4.4.3 of [RFC8446], with a context string that's specific to this use.

The specification also needs to define which signing algorithm to use. It currently specifies that as a function from the key type, instead of allowing attacker-controlled data to specify it.

B.1.3. Retrieving the certificate

The client needs to be able to find the certificate vouching for the signing key, a chain from that certificate to a trusted root, and possibly other trust information like SCTs ([RFC6962]). One approach would be to include the certificate and its chain in the signature metadata itself, but this wastes bytes when the same certificate is used for multiple HTTP responses. If we decide to put the signature in an HTTP header, certificates are also unusually large for that context.

Another option is to pass a URL that the client can fetch to retrieve the certificate and chain. To avoid extra round trips in fetching that URL, it could be bundled (Appendix A.6) with the signed content

or PUSHed (Appendix A.1) with it. The risks from the "client_certificate_url" extension (Section 11.3 of [RFC6066]) don't seem to apply here, since an attacker who can get a client to load an exchange and fetch the certificates it references, can also get the client to perform those fetches by loading other HTML.

To avoid using an unintended certificate with the same public key as the intended one, the content of the leaf certificate or the chain should be included in the signed data, like TLS does (Section 4.4.3 of [RFC8446]).

B.2. How much to sign

The previous [I-D.thomson-http-content-signature] and [I-D.burke-content-signature] schemes signed just the content, while ([I-D.cavage-http-signatures] could also sign the response headers and the request method and path. However, the same path, response headers, and content may mean something very different when retrieved from a different server. Section 5.1.1 currently includes the whole request URL in the signature, but it's possible we need a more flexible scheme to allow some higher-level protocols to accept a less-signed URL.

Servers might want to sign other request headers in order to capture their effects on content negotiation. However, there's no standard algorithm to check that a client's actual request headers match request headers sent by a server. The most promising attempt at this is [I-D.ietf-httpbis-variants], which encodes the content negotiation algorithm into the "Variants" and "Variant-Key" response headers. The proposal here (Section 3) assumes that is in use and doesn't sign request headers.

B.2.1. Conveying the signed headers

HTTP headers are traditionally munged by proxies, making it impossible to guarantee that the client will see the same sequence of bytes as the publisher published. In the HTTPS world, we have more end-to-end header integrity, but it's still likely that there are enough TLS-terminating proxies that the publisher's signatures would tend to break before getting to the client.

There's no way in current HTTP for the response to a client-initiated request (Section 8.1 of [RFC7540]) to convey the request headers it expected to respond to, but we sidestep that by conveying content negotiation information in response headers, per [I-D.ietf-httpbis-variants].

Since proxies are unlikely to modify unknown content types, we can wrap the original exchange into an "application/signed-exchange" format (Section 5.3) and include the "Cache-Control: no-transform" header when sending it.

To reduce the likelihood of accidental modification by proxies, the "application/signed-exchange" format includes a file signature that doesn't collide with other known signatures.

To help the PUSHed subresources use case (Appendix A.1), we might also want to extend the "PUSH_PROMISE" frame type to include a signature, and that could tell intermediates not to change the ensuing headers.

B.3. Response lifespan

A normal HTTPS response is authoritative only for one client, for as long as its cache headers say it should live. A signed exchange can be re-used for many clients, and if it was generated while a server was compromised, it can continue compromising clients even if their requests happen after the server recovers. This signing scheme needs to mitigate that risk.

B.3.1. Certificate revocation

Certificates are mis-issued and private keys are stolen, and in response clients need to be able to stop trusting these certificates as promptly as possible. Online revocation checks don't work (<https://www.imperialviolet.org/2012/02/05/crlsets.html>), so the industry has moved to pushed revocation lists and stapled OCSP responses [RFC6066].

Pushed revocation lists work as-is to block trust in the certificate signing an exchange, but the signatures need an explicit strategy to staple OCSP responses. One option is to extend the certificate download (Appendix B.1.3) to include the OCSP response too, perhaps in the TLS 1.3 CertificateEntry (<https://tlswg.github.io/tls13-spec/draft-ietf-tls-tls13.html#ocsp-and-sct>) format.

B.3.2. Response downgrade attacks

The signed content in a response might be vulnerable to attacks, such as XSS, or might simply be discovered to be incorrect after publication. Once the author fixes those vulnerabilities or mistakes, clients should stop trusting the old signed content in a reasonable amount of time. Similar to certificate revocation, I expect the best option to be stapled "this version is still valid" assertions with short expiration times.

These assertions could be structured as:

1. A signed minimum version number or timestamp for a set of request headers: This requires that signed responses need to include a version number or timestamp, but allows a server to provide a single signature covering all valid versions.
2. A replacement for the whole exchange's signature. This requires the publisher to separately re-sign each valid version and requires each version to include a different update URL, but allows intermediates to serve less data. This is the approach taken in Section 3.
3. A replacement for the exchange's signature and an update for the embedded "expires" and related cache-control HTTP headers [RFC7234]. This naturally extends publishers' intuitions about cache expiration and the existing cache revalidation behavior to signed exchanges. This is sketched and its downsides explored in Appendix C.

The signature also needs to include instructions to intermediates for how to fetch updated validity assertions.

B.4. Low implementation complexity

Simpler implementations are, all things equal, less likely to include bugs. This section describes decisions that were made in the rest of the specification to reduce complexity.

B.4.1. Limited choices

In general, we're trying to eliminate unnecessary choices in the specification. For example, instead of requiring clients to support two methods for verifying payload integrity, we only require one.

B.4.2. Bounded-buffering integrity checking

Clients can be designed with a more-trusted network layer that decides how to trust resources and then provides those resources to less-trusted rendering processes along with handles to the storage and other resources they're allowed to access. If the network layer can enforce that it only operates on chunks of data up to a certain size, it can avoid the complexity of spooling large files to disk.

To allow the network layer to verify signed exchanges using a bounded amount of memory, Section 5.3 requires the signature to be less than 16kB and the headers to be less than 512kB, and Section 3.5 requires that the MI record size be less than 16kB. This allows the network

layer to validate a bounded chunk at a time, and pass that chunk on to a renderer, and then forget about that chunk before processing the next one.

The "Digest" header field from [RFC3230] requires the network layer to buffer the entire response body, so it's disallowed.

Appendix C. Determining validity using cache control

This draft could expire signature validity using the normal HTTP cache control headers ([RFC7234]) instead of embedding an expiration date in the signature itself. This section specifies how that would work, and describes why I haven't chosen that option.

The signatures in the "Signature" header field (Section 3.1) would no longer contain "date" or "expires" fields.

The validity-checking algorithm (Section 3.5) would initialize "date" from the resource's "Date" header field (Section 7.1.1.2 of [RFC7231]) and initialize "expires" from either the "Expires" header field (Section 5.3 of [RFC7234]) or the "Cache-Control" header field's "max-age" directive (Section 5.2.2.8 of [RFC7234]) (added to "date"), whichever is present, preferring "max-age" (or failing) if both are present.

Validity updates (Section 3.6) would include a list of replacement response header fields. For each header field name in this list, the client would remove matching header fields from the stored exchange's response header fields. Then the client would append the replacement header fields to the stored exchange's response header fields.

C.1. Example of updating cache control

For example, given a stored exchange of:

```
GET / HTTP/1.1
Host: example.com
Accept: */*

HTTP/1.1 200
Date: Mon, 20 Nov 2017 10:00:00 UTC
Content-Type: text/html
Date: Tue, 21 Nov 2017 10:00:00 UTC
Expires: Sun, 26 Nov 2017 10:00:00 UTC

<!doctype html>
<html>
...
```

And an update listing the following headers:

```
Expires: Fri, 1 Dec 2017 10:00:00 UTC
Date: Sat, 25 Nov 2017 10:00:00 UTC
```

The resulting stored exchange would be:

```
GET / HTTP/1.1
Host: example.com
Accept: */*
```

```
HTTP/1.1 200
Content-Type: text/html
Expires: Fri, 1 Dec 2017 10:00:00 UTC
Date: Sat, 25 Nov 2017 10:00:00 UTC
```

```
<!doctype html>
<html>
...
```

C.2. Downsides of updating cache control

In an exchange with multiple signatures, using cache control to expire signatures forces all signatures to initially live for the same period. Worse, the update from one signature's "validity-url" might not match the update for another signature. Clients would need to maintain a current set of headers for each signature, and then decide which set to use when actually parsing the resource itself.

This need to store and reconcile multiple sets of headers for a single signed exchange argues for embedding a signature's lifetime into the signature.

Appendix D. Change Log

RFC EDITOR PLEASE DELETE THIS SECTION.

draft-09

- * No change

draft-08

- * Improve the privacy considerations.

draft-07

- * Provisionally register application/signed-exchange and application/cert-chain+cbor.

draft-06

- * Add a security consideration for future-dated OCSP responses and for stolen private keys.
- * Define a CAA parameter to opt into certificate issuance.
- * Limit certificate lifetimes to 90 days.
- * UTF-8 decode the fallback URL.

draft-05

- * Define absolute URLs, and limit the schemes each instance can use.
- * Fill in TBD size limits.
- * Update to mice-03 including the Digest header.
- * Refer to draft-yasskin-httpbis-origin-signed-exchanges-impl for draft version numbers.
- * Require "exchange"'s response to be cachable by a shared cache.
- * Define the "integrity" field of the Signature header to include subfields of the main integrity-protecting header, including the digest algorithm.
- * Put a fallback URL at the beginning of the "application/signed-exchange" format, which replaces the ':url' key from the CBOR representation of the exchange's request and response metadata and headers.
- * Remove the rest of the request headers from the signed data, in favor of representing content negotiation with the "Variants" response header.
- * Make the signed message format a concatenation of byte sequences, which helps implementations avoid re-serializing the exchange's request and response metadata and headers.
- * Explicitly check the response payload's integrity instead of assuming the client did it elsewhere in processing the response.
- * Reject uncached header fields.

- * Update to draft-ietf-httpbis-header-structure-09.

- * Update to the final TLS 1.3 RFC.

draft-04

- * Update to draft-ietf-httpbis-header-structure-06.

- * Replace the application/http-exchange+cbor format with a simpler application/signed-exchange format that:

- Doesn't require a streaming CBOR parser parse it from a network stream.
- Doesn't allow request payloads or response trailers, which don't fit into the signature model.
- Allows checking the signature before parsing the exchange headers.

- * Require absolute URLs.

- * Make all identifiers in headers lower-case, as required by Structured Headers.

- * Switch back to the TLS 1.3 signature format.

- * Include the version and draft number in the signature context string.

- * Remove support for integrity protection using the Digest header field.

- * Limit the record size in the mi-sha256 encoding.

- * Forbid RSA keys, and only require clients to support secp256r1 keys.

- * Add a test OID for the CanSignHttpExchanges X.509 extension.

draft-03

- * Allow each method of transferring an exchange to define which headers are signed, have the cross-origin methods use all headers, and remove the "allResponseHeaders" flag.

- * Describe footguns around signing private content, and block certain headers to make it less likely.

- * Define a CBOR structure to hold the certificate chain instead of re-using the TLS1.3 message. The TLS 1.3 parser fails on unexpected extensions while this format should ignore them, and apparently TLS implementations don't expose their message parsers enough to allow passing a message to a certificate verifier.
- * Require an X.509 extension for the signing certificate.

draft-02

- * Signatures identify a header (e.g. Digest or MI) to guard the payload's integrity instead of directly signing over the payload.
- * The validityUrl is signed.
- * Use CBOR maps where appropriate, and define how they're canonicalized.
- * Remove the update.url field from signature validity updates, in favor of just re-fetching the original request URL.
- * Define an HTTP/2 extension to use a setting to enable cross-origin Server Push.
- * Define an "Accept-Signature" header to negotiate whether to send Signatures and which ones.
- * Define an "application/http-exchange+cbor" format to fetch signed exchanges without HTTP/2 Push.
- * 2 new use cases.

Appendix E. Acknowledgements

Thanks to Andrew Ayer, Devin Mullins, Ilari Liusvaara, John Wilander, Justin Schuh, Mark Nottingham, Mike Bishop, Ryan Sleevi, and Yoav Weiss for comments that improved this draft.

Author's Address

Jeffrey Yasskin
Google

Email: jyasskin@chromium.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2021

J. Yasskin
Google
7 March 2021

Web Bundles
draft-yasskin-wpack-bundled-exchanges-04

Abstract

Web bundles provide a way to bundle up groups of HTTP responses, with the request URLs and content negotiation that produced them, to transmit or store together. They can include multiple top-level resources with one identified as the default by a `primaryUrl` metadata, provide random access to their component exchanges, and efficiently store 8-bit resources.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the WPACK Working Group mailing list (wpack@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/wpack/>.

Source for this draft and an issue tracker can be found at <https://github.com/WICG/webpackage>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology and Conventions	3
2. Semantics	3
2.1. Operations	3
2.2. Naming a representation	3
3. Expected performance	4
3.1. Random access	4
3.2. Streaming	4
4. Format	4
4.1. Top-level structure	4
4.1.1. Trailing length	6
4.1.2. Draft version numbers	6
4.2. Bundle sections	6
4.2.1. The index section	7
4.2.2. The manifest section	9
4.2.3. The critical section	9
4.3. Responses	9
4.4. Serving constraints	10
5. Security Considerations	10
5.1. Version skew	10
5.2. Content sniffing	11
6. IANA considerations	12
6.1. Internet Media Type Registration	12
6.2. Web Bundle Section Name Registry	13
7. References	14
7.1. Normative References	14
7.2. Informative References	15
Appendix A. Change Log	16
Appendix B. Acknowledgements	17
Author's Address	17

1. Introduction

To satisfy the use cases in [I-D.yasskin-wpack-use-cases], this document proposes a new bundling format to group HTTP resources. The format is structured as an initial table of "sections" within the bundle followed by the content of those sections.

1.1. Terminology and 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.

This specification uses the conventions and terminology defined in the Infra Standard ([INFRA]).

2. Semantics

A bundle is logically a set of HTTP representations (Section 7 of [I-D.ietf-httpbis-semantics]), themselves represented by HTTP response messages (Section 2.1 of [I-D.ietf-httpbis-semantics]). The bundle can include an optional URL identifying the primary resource within the bundle and can include other optional metadata. Particular applications can require that the primary URL and/or other metadata is present.

While the order of the representations is not semantically meaningful, it can significantly affect performance when the bundle is loaded from a network stream.

2.1. Operations

Bundle parsers support two primary operations:

1. They can load the bundle's metadata given a prefix of the bundle.
2. They can find a representation within the bundle given that representation's URL (Section 2.2) and the content-negotiation information that would appear in an HTTP request's headers.

2.2. Naming a representation

Representations within a bundle are named by their "Content-Location" (Section 7.8 of [I-D.ietf-httpbis-semantics]), which holds a URL. This is also known as the representation's URL.

Multiple representations within a bundle can have the same URL, in which case they are distinguished by the content negotiation information contained in their "Variants" and "Variant-Key" headers ([I-D.ietf-httpbis-variants]).

This identifying information for each representation is stored in an index (Section 4.2.1) rather than in that representation's HTTP response message.

3. Expected performance

Bundles can be used in two different situations: they can be loaded from storage that provides $O(1)$ access to any byte within the bundle, or they can be sent across a stream that provides bytes incrementally. An implementation MAY prefer either or both situations and SHOULD provide the following performance characteristics in its preferred situations:

3.1. Random access

To load a resource when seeing a bundle for the first time, the implementation reads $O(\text{size of the metadata and resource index})$ before starting to return bytes of the resource.

TODO: Is big- O notation the right way to express expectations here?

3.2. Streaming

When sending a bundle over a stream, the implementation will need to wait until it has the sizes of all contained resources before starting to send the resource index.

When reading a bundle from a stream, the implementation starts returning bytes of a resource after receiving $O(1)$ bytes of that resource, which comes after the $O(\# \text{ of resources})$ bytes of the index.

4. Format

4.1. Top-level structure

A bundle is a CBOR array ([CBORbis]) with the following CDDL ([CDDL]) schema:

```
webbundle = [  
  magic: h'F0 9F 8C 90 F0 9F 93 A6',  
  version: bytes .size 4,  
  primary-url: whatwg-url,  
  section-lengths: bytes .cbor section-lengths,  
  sections: [* any ],  
  length: bytes .size 8, ; Big-endian number of bytes in the bundle.  
]
```

```
whatwg-url = tstr
```

When serialized, the bundle MUST satisfy the core deterministic encoding requirements from Section 4.2.1 of [CBORbis]. This format does not use floating point values or tags, so this specification does not add any deterministic encoding rules for them. If an item doesn't follow these requirements, or a byte-sequence being decoded as a CBOR item contains extra bytes, the parser MUST signal an error instead of any data it can extract from that item.

High-level fields in the bundle format are designed to provide their length-in-bytes before the field starts so that a recipient trying to stream a bundle from the network can always wait for a known number of bytes instead of needing to implement a streaming CBOR parser.

The "magic" number is "" (U+1F310 U+1F4E6) encoded in UTF-8. With the CBOR initial bytes for the array and bytestring, this makes the format identifiable by looking for "8? 48" (in base 16) followed by that UTF-8 encoding. Parsers MUST only check the initial nibble of the initial "8?" byte in order to accommodate any future version's change in the number of array elements (up to 15).

The "version" bytestring MUST be "31 00 00 00" in base 16 (an ASCII "1" followed by 3 0s) for this version of bundles. If the recipient doesn't support the version in this field, it MUST either ignore the bundle or fetch and use the content of the "primary-url" field instead.

The "primary-url" field identifies both a fallback when the recipient doesn't understand the bundle and a default resource inside the bundle to use when the recipient doesn't have more specific instructions. This field MAY be an empty string, although protocols using bundles MAY themselves forbid that empty value.

The "section-lengths" and "sections" arrays contain the actual content of the bundle and are defined in Section 4.2. The "section-lengths" array is embedded in a byte string to facilitate reading it from a network. This byte string **MUST** be less than 8192 (8*1024) bytes long, and parsers **MUST NOT** load any data from a "section-lengths" item longer than this.

The bundle ends with an 8-byte integer holding the length of the whole bundle.

4.1.1. Trailing length

A bundle ends with an 8-byte CBOR byte string holding a big-endian integer that represents the byte-length of the whole bundle.

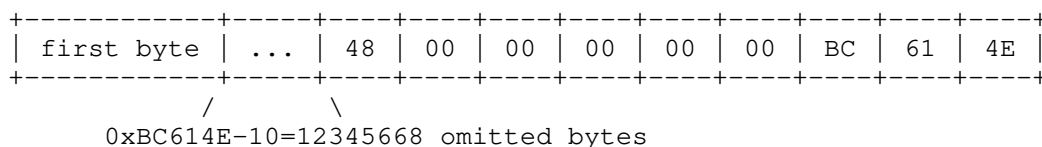


Figure 1: Example trailing bytes

Recipients loading the bundle in a random-access context **SHOULD** start by reading the last 8 bytes and seeking backwards by that many bytes to find the start of the bundle, instead of assuming that the start of the file is also the start of the bundle. This allows the bundle to be appended to another format such as a generic self-extracting executable.

4.1.2. Draft version numbers

This section is to be removed before publishing as an RFC.

Implementations of drafts of this specification **MUST NOT** use a "version" string of "31 00 00 00" (base 16). They **MUST** instead define an implementation-specific 4-byte string starting with "62" ("b") to identify which draft is implemented.

4.2. Bundle sections

A bundle's content is in a series of sections, which can be accessed randomly using the information in the "section-lengths" CBOR item:

```
section-lengths = [* (section-name: tstr, length: uint) ],
```

This field lists the named sections in the bundle in the order they appear, with each section name followed by the length in bytes of the corresponding CBOR item in the "sections" array. This allows a random-access parser (Section 3) to jump directly to the section it needs. This specification defines the following sections:

- * "index" (Section 4.2.1)
- * "manifest" (Section 4.2.2)
- * "critical" (Section 4.2.3)
- * "responses" (Section 4.3)

Future specifications can register new section names as described in Section 6.2, in order to extend the format without incrementing its version number.

The "responses" section MUST appear after the other three sections defined here, and parsers MUST NOT load any data if that is not the case.

The "sections" array contains the sections' content. The length of this array MUST be exactly half the length of the "section-lengths" array, and parsers MUST NOT load any data if that is not the case.

The bundle MUST contain the "index" and "responses" sections. All other sections are optional.

4.2.1. The index section

```
index = {* whatwg-url => [ variants-value, +location-in-responses ] }
variants-value = bstr
location-in-responses = (offset: uint, length: uint)
```

The "index" section defines the set of HTTP representations in the bundle and identifies their locations in the "responses" section. It consists of a CBOR map whose keys are the URLs of the representations in the bundle (Section 2.2). The value of an index entry is an array whose first item is a "Variants" header field value ([I-D.ietf-httpbis-variants]) or the empty string. This is followed by a sequence of offset/length pairs, one for each representation of this resource. The offset is relative to the start of the "responses" section, with an offset of 0 referring to the head of the CBOR "responses" array itself. The length is the length in bytes of the "response" CBOR item holding this representation (Section 4.3).

If the first item in the value of an index entry is empty, it MUST be followed by exactly one offset/length pair. This means there is a single representation for this resource, with no content negotiation.

Otherwise, the first item MUST be followed by one offset/length pair for each of the possible combinations of available-values within the "Variants" value (the first item of the array) in lexicographic (row-major) order.

For example, given a "Variants" value of "accept-encoding=(gzip br), accept-language=(en fr ja)", the list of offset/length pairs will correspond to the "Variant-Key"s:

- * (gzip en)
- * (gzip fr)
- * (gzip ja)
- * (br en)
- * (br fr)
- * (br ja)

The order of variant-axes is important. If the "Variants" value were "accept-language=(en fr ja), accept-encoding=(gzip br)" instead, the "location-in-responses" pairs would instead correspond to:

- * (en gzip)
- * (en br)
- * (fr gzip)
- * (fr br)
- * (ja gzip)
- * (ja br)

If the wrong number of offset/length pairs is present in a resource's array, the entire index MUST fail to parse.

A combination of available-values that is omitted from the bundle MUST be signaled by setting its offset and length to 0.

4.2.2. The manifest section

```
manifest = whatwg-url
```

The "manifest" section records a single URL identifying the manifest of the bundle. The URL MUST refer to a resource with representations contained in the bundle itself.

The bundle can contain multiple representations at this URL, and the client is expected to content-negotiate for the best one. For example, a client might select the one matching an "accept" header of "application/manifest+json" ([appmanifest]) and an "accept-language" header of "es-419".

Many bundles have a choice between identifying their manifest in this section or in their primary resource, especially if that resource is an HTML file. Identifying the manifest in this section can help recipients apply fields in the manifest sooner, for example to show a splash screen before parsing the primary resource.

4.2.3. The critical section

```
critical = [*tstr]
```

The "critical" section consists of the names of sections of the bundle that the client needs to understand in order to load the bundle correctly. Other sections are assumed to be optional.

If the client has not implemented a section named by one of the items in this list, the client MUST fail to parse the bundle as a whole.

4.3. Responses

```
responses = [*response]  
response = [headers: bstr .cbor headers, payload: bstr]  
headers = {* bstr => bstr}
```

The "responses" section holds the HTTP responses that represent the HTTP representations in the bundle. It consists of a CBOR array of responses, each of which is pointed to by one or more entries in the "index" section (Section 4.2.1).

The length of the "headers" byte string in a response MUST be less than 524288 (512*1024) bytes, and recipients MUST fail to load a response with longer headers.

When receiving a bundle in a stream, the recipient MAY process the headers before the payload has been received and MAY start processing the beginning of the payload before the end of the payload has been received.

The keys of the headers map MUST consist of lowercase ASCII as described in Section 8.1.2 of [RFC7540]. Response pseudo-headers (Section 8.1.2.4 of [RFC7540]) are included in this headers map.

Each response's headers MUST include a ":status" pseudo-header with exactly 3 ASCII decimal digits and MUST NOT include any other pseudo-headers.

If a response's payload is not empty, its headers MUST include a "Content-Type" header (Section 7.4 of [I-D.ietf-httpbis-semantics]). The client MUST interpret the following payload as this specified media type instead of trying to sniff a media type from the bytes of the payload, for example by appending an artificial "X-Content-Type-Options: nosniff" header field ([FETCH]) to downstream protocols.

4.4. Serving constraints

When served over HTTP, a response containing an "application/webbundle" payload MUST include at least the following response header fields, to reduce content sniffing vulnerabilities (Section 5.2):

- * Content-Type: application/webbundle
- * X-Content-Type-Options: nosniff

5. Security Considerations

5.1. Version skew

Bundles currently have no mechanism for ensuring that any signed exchanges they contain constitute a consistent version of those resources. Even if a website never has a security vulnerability when resources are fetched at a single time, an attacker might be able to combine a set of resources pulled from different versions of the website to build a vulnerable site. While the vulnerable site could have occurred by chance on a client's machine due to normal HTTP caching, bundling allows an attacker to guarantee that it happens. Future work in this specification might allow a bundle to constrain its resources to come from a consistent version.

5.2. Content sniffing

While modern browsers tend to trust the "Content-Type" header sent with a resource, especially when accompanied by "X-Content-Type-Options: nosniff", plugins will sometimes search for executable content buried inside a resource and execute it in the context of the origin that served the resource, leading to XSS vulnerabilities. For example, some PDF reader plugins look for "%PDF" anywhere in the first 1kB and execute the code that follows it.

The "application/webbundle" format defined above includes URLs and request headers early in the format, which an attacker could use to cause these plugins to sniff a bad content type.

To avoid vulnerabilities, in addition to the response header requirements in Section 4.4, servers are advised to only serve an "application/webbundle" resource from a domain if it would also be safe for that domain to serve the bundle's content directly, and to follow at least one of the following strategies:

1. Only serve bundles from dedicated domains that don't have access to sensitive cookies or user storage.
2. Generate bundles "offline", that is, in response to a trusted author submitting content or existing signatures reaching a certain age, rather than in response to untrusted-reader queries.
3. Do all of:
 1. If the bundle's contained URLs (e.g. in the manifest and index) are derived from the request for the bundle, percent-encode (<https://url.spec.whatwg.org/#percent-encode>) ([URL]) any bytes that are greater than 0x7E or are not URL code points (<https://url.spec.whatwg.org/#url-code-points>) ([URL]) in these URLs. It is particularly important to make sure no unescaped nulls (0x00) or angle brackets (0x3C and 0x3E) appear.
 2. Similarly, if the request headers for any contained resource are based on the headers sent while requesting the bundle, only include request header field names *and values* that appear in a static allowlist. Keep the set of allowed request header fields smaller than 24 elements to prevent attackers from controlling a whole CBOR length byte.
 3. Restrict the number of items a request can direct the server to include in a bundle to less than 12, again to prevent attackers from controlling a whole CBOR length byte.

4. Do not reflect request header fields into the set of response headers.

If the server serves responses that are written by a potential attacker but then escaped, the "application/webbundle" format allows the attacker to use the length of the response to control a few bytes before the start of the response. Any existing mechanisms that prevent polyglot documents probably keep working in the face of this new attack, but we don't have a guarantee of that.

To encourage servers to include the "X-Content-Type-Options: nosniff" header field, clients SHOULD reject bundles served without it.

6. IANA considerations

6.1. Internet Media Type Registration

IANA is requested to register the MIME media type ([IANA.media-types]) for web bundles, application/webbundle, as follows:

- * Type name: application
- * Subtype name: webbundle
- * Required parameters:
 - v: A string denoting the version of the file format.
([RFC5234] ABNF: "version = 1*(DIGIT/%x61-7A)") The version defined in this specification is "1".

Note: RFC EDITOR PLEASE DELETE THIS NOTE; Implementations of drafts of this specification MUST NOT use simple integers to describe their versions, and MUST instead define implementation-specific strings to identify which draft is implemented.

- * Optional parameters: N/A
- * Encoding considerations: binary
- * Security considerations: See Section 5 of this document.
- * Interoperability considerations: N/A
- * Published specification: This document

- * Applications that use this media type: None yet, but it is expected that web browsers will use this format.
- * Fragment identifier considerations: N/A
- * Additional information:
 - Deprecated alias names for this type: N/A
 - Magic number(s): 86 48 F0 9F 8C 90 F0 9F 93 A6
 - File extension(s): .wbn
 - Macintosh file type code(s): N/A
- * Person & email address to contact for further information: See the Author's Address section of this specification.
- * Intended usage: COMMON
- * Restrictions on usage: N/A
- * Author: See the Author's Address section of this specification.
- * Change controller: The IESG iesg@ietf.org (<mailto:iesg@ietf.org>)
- * Provisional registration? Yes.

6.2. Web Bundle Section Name Registry

IANA is directed to create a new registry with the following attributes:

Name: Web Bundle Section Names

Review Process: Specification Required

Initial Assignments:

Section Name	Specification
"index"	Section 4.2.1
"manifest"	Section 4.2.2
"critical"	Section 4.2.3
"responses"	Section 4.3

Table 1

Requirements on new assignments:

Section Names MUST be encoded in UTF-8.

A section's specification MAY say that, if it is present, another section is not processed.

7. References

7.1. Normative References

- [appmanifest] Caceres, M., Christiansen, K., Lamouri, M., Kostiainen, A., Dolin, R., and M. Giuca, "Web App Manifest", World Wide Web Consortium WD WD-appmanifest-20180523, 23 May 2018, <<https://www.w3.org/TR/2018/WD-appmanifest-20180523>>.
- [CBORbis] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/rfc/rfc8949>>.
- [CDDL] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/rfc/rfc8610>>.
- [FETCH] WHATWG, "Fetch", March 2021, <<https://fetch.spec.whatwg.org/>>.

- [I-D.ietf-httpbis- semantics]
Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis- semantics-14, 12 January 2021, <<https://tools.ietf.org/html/draft-ietf-httpbis- semantics-14>>.
- [I-D.ietf-httpbis- variants]
Nottingham, M., "HTTP Representation Variants", Work in Progress, Internet-Draft, draft-ietf-httpbis- variants-06, 3 November 2019, <<https://tools.ietf.org/html/draft-ietf- httpbis- variants-06>>.
- [IANA.media- types]
IANA, "Media Types", <<http://www.iana.org/assignments/media- types>>.
- [INFRA] WHATWG, "Infra", March 2021, <<https://infra.spec.whatwg.org/>>.
- [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/rfc/rfc2119>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/rfc/rfc5234>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/rfc/rfc7540>>.
- [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/rfc/rfc8174>>.
- [URL] WHATWG, "URL", March 2021, <<https://url.spec.whatwg.org/>>.

7.2. Informative References

[I-D.yasskin-wpack-use-cases]

Yasskin, J., "Use Cases and Requirements for Web Packages", Work in Progress, Internet-Draft, draft-yasskin-wpack-use-cases-01, 27 July 2020, <<https://tools.ietf.org/html/draft-yasskin-wpack-use-cases-01>>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

draft-04

- * Rewrite to be more declarative and less algorithmic.
- * Make a bundle represent a set of HTTP Representations, with the Content-Location replacing what was the request URL, and the Variants information, as before, driving content negotiation.
- * Make the primary URL optional.
- * Remove the signatures section.
- * Update Variants examples for the latest Variants draft.
- * Removed the distinction between "metadata" and non-metadata sections.

draft-03

- * Make the manifest optional.
- * Update the reference to draft-yasskin-wpack-use-cases.
- * Retitle to "web bundles".

draft-02

- * Fix the initial bytes of the format.
- * Allow empty responses to omit their content type.
- * Provisionally register application/webbundle.

draft-01

- * Include only section lengths in the section index, requiring sections to be listed in order.

- * Have the "index" section map URLs to sets of responses negotiated using the Variants system ([I-D.ietf-httpbis-variants]).
- * Require the "manifest" to be embedded into the bundle.
- * Add a content sniffing security consideration.
- * Add a version string to the format and its mime type.
- * Add a fallback URL in a fixed location in the format, and use that fallback URL as the primary URL of the bundle.
- * Add a "signatures" section to let authorities (like domain-trusted X.509 certificates) vouch for subsets of a bundle.
- * Use the CBORbis "deterministic encoding" requirements instead of "canonicalization" requirements.

Appendix B. Acknowledgements

Thanks to the Chrome loading team, especially Kinuko Yasuda and Kouhei Ueno for making the format work well when streamed.

Author's Address

Jeffrey Yasskin
Google

Email: jyasskin@chromium.org