Mozilla's Position on Web Packaging

Web packaging proposes a significant change to the web platform in the way that content is delivered and authenticated.

From a technical standpoint, the changes are thorough and well-considered. There are some technical costs around security, operations, and complexity, but the specifications take steps to limit most of these costs.

The most disruptive feature of the proposal, origin substitution, describes a fundamental change to the security architecture of the web. In addition to a significant increase in complexity, origin substitution creates new angles of attack that site operators need to consider before they adopt the technology. Changes to the way sites operate could result in non-trivial security risks.

The main concern is web packaging might be employed to alter power dynamics between aggregators and publishers. At this moment, we don't understand enough to say definitively that this is damaging to the system. How this technology is deployed matters. Deployment without systems of accountability, oversight, and limitations on use could be harmful. There are no constraints on deployment in the proposals, so much depends on how the technology is used and the incentives around that use.

As a large suite of mechanisms, there are parts of web packaging that could be valuable on their own, such as the design of a common resource bundling format. There are also ways in which using web packaging could encourage better security and performance practices from sites.

As a whole, and for origin substitution in particular, until more information is available on the effect on the web ecosystem, Mozilla concludes that it would not be good for the web to deploy web packaging.

What is web packaging?

Web packaging is a combination of technologies:

- A means of signing an HTTP request/response exchange.
- A format for bundling multiple request/response exchanges into a single file.
- Changes to fetch that support the ability for signed content to be treated as being in a particular origin.

Combined this allows a site to create a package that can be distributed via any means to clients. Clients would be able to render the resources contained in that package as

though they were live resources on the origin. This idea that we call “origin substitution” - the ability to present content for an origin without contacting that origin - is the central problem that web packaging aims to tackle. Origin substitution is also the source of all the problems and trade-offs that are being made.

Why web packaging?

There has been a lot of discussion about two use cases for web packaging. Web packaging supports distribution of content in two major ways: offline and by third parties.

A use cases document describes many other potential uses, but examining these two cases is most important.

Offline distribution

The proponents originally claimed that this is primarily to enable offline sharing of online content. That is, people would be able to download web pages (or progressive web apps) and share them offline in a peer-to-peer fashion. Recipients of these packages could then use them without going online, with an expectation that if they did go online, the content would seamlessly transition to a fully connected experience.

Though the use of sites offline is a worthy goal, a more modest goal of making content available offline, but in a separate origin - and therefore without a story for transitioning to an online experience - is likely to be a more fruitful approach in the short term, even if it is only incremental.

This aspect of the feature is advertised as enabling distribution of censored or illegal[1] content. Given that this use case generally doesn’t depend on attribution to an origin, an incremental approach seems most likely to be more effective.

Third-party distribution

The second use case is “content distribution”. This is a far more difficult use case to understand, because it involves the complex relationship between entities that serve pages that link out to content (aggregators, like search engines and social networks), and those that publish that content (publishers, like journalism sites). Google’s Accelerated Mobile Pages (AMP), Facebook’s Instant Articles, Baidu’s MIP, and Apple’s News Format are all examples of aggregators that use similar techniques. All of these services aggregate content published by others.

---

[1] Just in case anyone thinks that the inclusion of signature might dissuade someone from providing illegal content in this form, remember that signatures don’t provide non-repudiation. The coupling of bundles to signatures therefore provides no meaningful attribution of illegal content to its source.

Aggregators have encouraged publishers to provide content that the aggregator can serve through the use of certain inducements like better search placement or presentation options. This provides the aggregator with some significant advantages.

Outbound links that might otherwise reference a publisher instead reference content hosted by the aggregator. This allows the aggregator to serve that content to users, without those users having to contact the publisher. The aggregator typically constrains the type and format of content that they will accept so that they can more effectively process and host the content. For instance, this allows the aggregator to verify that content does not contain malicious Javascript that could be a risk to their service.

For users of the aggregator, all of this means that content loads more quickly. It also means that information about their activities on the aggregator is only released to publishers when a link is followed. Technically, the aggregator has complete discretion about what information is released, but withholding capabilities like analytics could make any technology like this completely non-viable. The performance and privacy gains are significant, but the cost is that the origin of the final content is one controlled by the aggregator and not the publisher.

The main drawback of these techniques is that the destination of a link is a page on the origin of the aggregator. This results in a mismatch between the source of the content and the identity of the host. Web packaging aims to address this by providing a standardized way in which this content can be hosted by an aggregator, but presented in the origin of the publisher.

While browsers could in theory preload directly from the publisher to realize the same performance gains, there are several reasons why preloading might be difficult or less efficient. Preloading content from a publisher is less efficient than loading from the aggregator as it requires new connections, which can add significant latency. Browsers are also less able to automatically identify candidates for preloading from the many links that might be presented on a page.

More seriously, a browser would be forced to trade performance gains for privacy. If browsers preload without releasing information about the source of a link, the preloading could be invalid and unusable. Many links include information about the entity that was given the link[2], making it impossible to prevent some information from leaking to a publisher. Attempting to ensure that the information is accurate could trigger unwanted side effects, such as what might happen when clicking a link from an advertisement. Preventing leakage to the same degree as provided by having content

[2] Many sites provide crawlers for search engines with a link that is specific to that search engine, so that they can tell where links came from even without relying on other mechanisms.

hosted on the aggregator is impossible, as even the most limited form of preloading exposes information about the browser including an IP address and characteristics of its networking stack.

Web packaging aims to provide the performance and privacy benefits without shifting the origin of content to that of the aggregator. It does that by having a publisher sign content that the aggregator can deliver. That content is attributed to the publisher, but the publisher does not need to be involved in the delivery of the content.

Through this, both the user and publisher see a faster transition from the aggregator to the publisher site. This improvement in navigation speed is seen as one of the primary benefits of packaging. The publisher sees few other benefits, other than a reduction in bandwidth costs, though the speed benefit might be significant enough to justify their costs. The costs to the publisher are somewhat harder to understand, and these are dealt with in more detail below.

Advantages of web packaging

The use cases document lists a range of interesting applications of bundling and origin substitution, both as individual primitives or composed as a whole. The biggest advantages are those that address the use cases already discussed, though there are some potential secondary properties that could be beneficial.

The ability to bundle a snapshot of the state of HTTP exchanges suggests interesting applications for archival, but existing tools provide this capability. Similarly, many of the use cases considered don't warrant the complexity of the proposed design. For these, it might be better to explore alternative designs or the repurposing of existing tools.

The individual primitives, origin substitution and bundling, are powerful tools. These tools can be composed in many ways that might provide advantages. For instance, the ability to have origins collectively attest to a single shared controller without first contacting every shared origin provides some interesting properties[3].

Concerns about web packaging

Mozilla's initial concerns with web packaging were focused on the security aspects of origin substitution. That is, the ability of an origin to provide a blob that could be obtained from anywhere, but be attributed to that origin.

[3] These properties seem limited to an attempt to convince browsers not to restrict access to certain capabilities, so a lot depends on to what extent cross-origin content is necessary for legitimate sites, as opposed to things like tracking.

Origin substitution is the key piece of this technology. It is what enables offline content to seamlessly transition into online content and it is what allows an aggregator to host content for another origin.

At its core, origin substitution enables a fundamental change to the way the web works. Content is no longer constrained to follow connections to origins, where that content is produced and where it is obtained can become completely decoupled.

Origin substitution can be an appealing proposition as it enables new and interesting deployment topologies. Though CDNs have generally shown more disdain than interest in the idea, it might be possible for a CDN to host content for customers without also being able to impersonate those customers. Limiting the privileges a CDN gains in this way might be appealing to customers who might wish to use technical measures to retain close control over their origins.

Concentrating on security issues is relatively easy. Coming to terms with a fundamental change to the security and content delivery model of the web is a more difficult task. This document tries to go further and explore other potentially problematic parts in the technology.

Attacker compromise of signed exchanges

The potential for an attacker to compromise a server key or fraudulently obtain a certificate and use that to produce fake content for the victim origin is a new risk this design adds.

The current design stipulates that only certain certificates can be used to sign exchanges. This means that a site that does not choose to use web packaging does not need to worry about attacks that use web packaging, outside of the possibility of misissuance of a certificate that has the certificate extension.

The intent of these defenses is to not significantly change the risk profile for sites, except for those sites that opt in to this mechanism. The certificate used to generate signed exchanges can be separate from online keys used to terminate connections[4], meaning that stronger protections might be feasible for those certificates. Whether this will be useful is hard to know, as operational practices around the use of these keys and certificates are not settled.

The only offered defense against compromise or misissuance is certificate revocation. For an offline consumer of this content, this is unlikely to reach them, but then there is fundamentally nothing that can be done without a means of communication. The same

[4] Nothing prevents that certificate being used to terminate a connection as the extension is non-critical.

problem exists for service workers, or any other technology that might be used to take content offline.

In general terms, should an attacker gain the ability to compromise packages, the validity period of the package bounds any exposure. Web caching and the aforementioned service workers can already be used to effect similar persistent attacks. There however, requirements for constant revalidation limit the duration of attacks. In this case, attacks might be persisted for as long as the recipient remains unaware of the certificate revocation. The 7 day limit on the validity period, which corresponds with current advice on OCSP lifetimes, limits this. From the time that a certificate is revoked, an attacker has at most 7 days to continue to provide content that is valid for the victim origin, limited both by OCSP and signature validity periods.

Note that the same considerations apply to attempts to misrepresent old content as being current. This might be used to have compromised content, such as content that contains a vulnerability for which a fix has been deployed, accepted by recipients.

In all these cases, online recipients might be able to check with the origin, but that either negates the performance and security advantages, or presents an attacker with an opportunity to race that validation attempt with their attack. An important consideration here is that a transient exposure to attack can be turned into a persistent threat by an attacker. An origin can terminate persistent attacks if they are aware of the need, but it requires that they be contacted by the browser.

For these security issues, while the measures offered are effective within the constraints imposed by the requirements, this represents a small regression in effective security. Recipients of packages will be vulnerable for longer than they would when using direct connections to origins, but the effect is largely consistent with the effect of caching or service workers. Sites that don't opt to use the mechanism are not affected by key compromise, but remain at risk from errors in misissuance. The CAA extension defined in the specification mitigates this partially by forcing an attacker to overwrite a CAA record.

These weaknesses are all relatively minor. The mitigations that are proposed go a long way to limiting the potential scope and severity of attacks, ensuring that problems are roughly equivalent to other technologies that are either pre-existing or in development. As a whole, these might reasonably be presented as a trade-off. Sites opt-in to using this mechanism, and in doing so need to be aware that this comes with some risks, but in doing so they enable a new feature.

For security then, a lot depends on the value proposition. If the value that might be realized by using web packaging were significant, that might outweigh these disadvantages.

Content customization

Web packaging depends on a view of content that is relatively static. Ideally, a single page can be represented by a single static bundle of content. For a completely offline case, this is a hard requirement, the package has to be ready for use by all potential audiences.

Reduced personalization

Sites routinely customize content to individuals. Many sites produce different content based on the browser used, the form factor of the device, the estimated location of the requester, and things like pixel density of the screen. It's possible to build packages with all possible variations of content and to use browser-based mechanisms like CSS media queries, the <picture> element, or script to manage selecting the right variants of content. However, this could increase the size of the package, which might reduce or reverse any potential performance benefit.

The extent to which a single package can support personalization will depend on the quantity and variety that can be bundled into the package without causing it to expand too much in size. If the goal is to minimize package size for performance reasons, then the degree of variation that a bundle can support will be limited. More expansive personalization might require the use of multiple bundles. However, using multiple bundles limits the scope of client-side content adaptation techniques in negative ways. This presents publishers with trade-offs in how they construct bundles.

Keeping the number of package variants small will be a goal for several reasons. Every variant has to be generated and signed separately. This constrains the number of variants that can be generated, stored, and served. The provider of a web package therefore is limited in what sorts of customizations are possible. While certain customizations are possible through accessing browser storage and providing the logic and resources necessary in every package, the model for customization changes.

For instance, a publisher might ordinarily provide differing content for different screen sizes. If a single bundle cannot support client-side adaptation, the packages that are provided to an aggregator need to be constructed based on how the aggregator serves content based on screen size. The aggregator needs to be able to both acquire information about screen size from clients and apply that information in selecting the right package. If the publisher interacts with multiple aggregators, then it needs to

ensure that it provides a different set of packages for each aggregator, based on aggregator capabilities.

For scalable resources, like images, bundling the highest quality and relying on clients being able to scale the resource for more limited devices is probably superior to providing multiple copies of the same content. There are performance costs for doing so if the largest size is not needed.

Flash of unpersonalized content

It is possible that the personalization problem can be partly addressed by avoiding packaging content that might be personalized. Online resources are used to provide personalized content when possible and fallback objects are used when offline. For instance, a small version of an image might be provided, with any larger variant being requested later if possible. The same might be done for the sorts of customization we see in advertising: stubbed content can be provided, with active connections being used to emplace custom advertising.

The risk with creating views of pages without customization is that adding personalization after page load can be jarring. Restoring details about a user that is logged in to the site might result in a need to add and remove page elements or provide new styling. Upgrading images to those suitable is probably less disruptive.

There are many tools that might be used to manage a transition from generic content to a personalized page, but that represents a complexity burden for those that provide a web package.

Minority interests

The pressure to reduce the number of variants of content could produce deleterious secondary effects. Of most concern is the accommodations sites might make for minority users. For instance, until recently the WEBP image format wasn't widely supported by browsers, but it was supported by browsers with a significant portion of the market. Sites producing web packages in that sort of environment might decide that providing JPEG images is inconvenient on the basis that most users can read the more efficient WEBP.

To some extent, this is no different to the existing pressure on minorities to conform, but the constraints of the format make the pressure more acute. For online content, it is possible for a publisher to provide ways to fill gaps or omissions dynamically, but no recourse is available if the use is purely offline.

This is a situation where the calculus for browsers with market dominance is very different than for others.

Producing and consuming content variants

There is a challenge in deciding what to select when presented with alternative representations of resources, or even if the provided representation is acceptable at all. HTTP/2 server push suffers from a lesser version of the same problem. In HTTP/2, a server decides what a request might look like when it generates a server push. There, the server has other requests on the same connection to model this information on, so the guess can be tuned.

The creator of a web package has no such information, and so needs to rely on predictions of its expected recipients. Thus sites cannot be reactive in their decision-making, greater awareness of the technical capabilities of their target audience is necessary.

In accepting content, a browser that finds that a packaged exchange contains a request that is sufficiently different than what it might have generated is forced to make a difficult choice: make a new request, or find ways to be more lenient in its acceptance criteria. The variants work should help this, by carrying information that might allow a browser to determine if a different request would produce a different response.

The operational complexity of this aspect is hard to assess without experience. In that it is similar to the challenges in HTTP/2 server push deployment, it's likely that the industry will struggle with this aspect.

Privacy

An interaction over HTTPS provides confidentiality for both the identity of a resource and the content that is ultimately provided. This property does not hold for web packaging. The entity that provides a web package gains both pieces of information.

The obvious limitation is that the provider of content in many cases is also the one providing the link. On that basis, the provider of the content is no more privileged than they would otherwise be. It is already possible, through the use of onclick handlers, link target rewriting, redirection, and the sendBeacon API to collect information about which links on a page are being followed. The content that they learn would then be the same content that they would obtain if they followed the link themselves.

There is a small risk of leakage from personalized and confidential content, but the specifications place constraints on the use of signed exchanges that limit this. Importantly, they prohibit the use of cookies, which is the most obvious way in which

confidential content is requested. They also advise servers to only use responses that have a Cache-Control header field value of public. Here, the biggest risk is likely from inline generation of packages, that is, the spontaneous generation of packages in response to requests that contain personal content. Problems like this only arise if sites fail to implement fairly obvious and well-documented safeguards.

On this basis, the use of web packaging doesn't present a significantly enhanced risk to privacy if implemented moderately carefully by publishers.

Complexity

Significant effort has been invested into making web packaging simple, and that shows in the designs. The designs are in many ways improved over earlier iterations.

However, this represents a completely parallel system for securing web content. While this uses the Web PKI infrastructure for authentication, the means by which content is delivered is entirely new. That naturally means additional platform complexity.

This general design space is not new. SHTTP was proposed at around the same time as HTTPS. Though there are aspects of the SHTTP design that are likely now redundant (like leaving certain metadata unprotected), it bears a degree of similarity to web packaging. Part of the appeal of HTTPS over SHTTP was the simplicity of HTTPS: you just added a TLS (or SSL) layer to the stack that handled all the difficult security bits, like authentication, encryption, anti-replay, liveness checks, and all of that.

Web packaging aims for a more limited set of security properties than SHTTP, but it nonetheless inherits much of its complexity.

As with other considerations, complexity is not a strong basis for rejecting a feature. But it does speak to a need for careful consideration. Complex features produce more bugs and expand the risk profile for security problems.

Performance

Signed exchanges require significantly more computational effort to consume than a typical resource. In TLS, particularly with newer HTTP versions, the cost of the more expensive cryptographic operations, like signing and key exchange are amortized across multiple exchanges. Here, every resource requires an independent signature verification[5].

Signatures also add a non-trivial number of bytes to each signed exchange. The smallest known signature scheme adds 32 bytes to every exchange, though most are at least twice that, plus overheads. For content that is suitable for signing, it is possible to assume that

[5] This is a feature of the design as of May 2019 which is very likely to change.

it is all public content, and so the usual concerns about mixing public data with secret data under compression do not apply. This allows compression to be used across the exchanges in a bundle, so fixed overheads can be greatly reduced.

It is anticipated that signing will be infrequent and offline. Performance costs for a publisher are therefore amortized over many uses of the same package and so the added cost is tolerable. The only potential source of additional computational cost is in signing and maintaining different variants of bundles.

More significant is the storage overhead for publishers and aggregators. Where there are multiple pages, each page will require one or more packages, depending on the number of variants the site provides, potentially increased for aggregators that have different capabilities. Each bundle will contain copies of common resources, like CSS, script and images. Without deduplication, this could amplify storage costs for content. The advantage of the design here is that it is relatively straightforward to do this sort of deduplication, and the cost in additional complexity is largely borne by aggregators.

Overall, the increase in computational cost for clients is likely tolerable, signature validation is generally very fast, but this would need to be tested. The increase in the size of resources might be counteracted by more effective use of compression. The cost to publishers is negligible, and we will assume for these purposes that participation is discretionary.

Origin uniformity

The design of web packaging assumes that origins are uniform. It is not possible to deploy signed exchanges with a key that only attests to a subset of the resources on an origin. This might be incompatible with site deployment strategies, that rely on a centralized controller (the entity that runs the HTTP server) to manage who gets to sign for different resources.

If there are mutually distrustful entities that share the same origin - a bad idea, but nevertheless still practiced - then those same controls would need to be replicated for signed exchanges. Otherwise entities might make claims about resources they don't control. This is probably a matter for server operators to manage; if there are mutual distrustful entities sharing an origin, then it could be unwise to enable signed exchanges. An offline signing key might still be used for a limited subset of resources.

Centralization and control

It is possible that this technology gives aggregators better means of controlling content. The existing technologies demonstrate that this power already exists, so the question is whether this would in any way alter the power dynamics.

It might be argued that the existing use of aggregator hosting is strictly worse than the proposed system. The proposed system might allow aggregators to impose fewer limitations on what they might host because content no longer runs on their origin. This might give publishers more control over what content gains access to better treatment from aggregators.

It is unlikely that aggregators will host arbitrary content for this purpose for reasons of security (for instance, content sniffing presents some risks), reputation, or more practical concerns like the amount of storage required. With greater access to the content that is being distributed, there are more opportunities to strengthen this control.

This is perhaps the hardest question to assess in this entire space. Does this technology represent a benefit to publishers? Or does the benefit they might receive come from the inducements that aggregators might offer (like improved placement or visibility)?

To what extent then is this the application of a power that aggregators possess, a power that is derived from their ability to control whether and how their audience sees a link, over publishers?

Clearly, aggregators have the power of refusal. They can refuse to link to a site, or they can refuse to host content that does not meet their standards. Without web packaging, this refusal might be based on a need to prevent malicious content from interfering with the origin of the aggregator, and the origin of other publishers. But can an aggregator refuse to serve content that does not meet arbitrary criteria?

What controls ensure that this power is used responsibly? We already see ways in which search engines exercise political power, reducing visibility of certain results or eliminating them entirely. For instance, sites that have poor performance or security are deprioritized in results. While we might see these particular choices as virtuous, they are not value-neutral. They are a manifestation of power over other web users.

We don't know what people think about this particular aspect of this technology. A lot may depend on how it is used. That usage isn't static either. Responsible usage today does not imply any guarantee of continued respect for publishers.

Resource composition attacks

A fundamental change to the security architecture of the web changes the way in which systems need to be built. The addition of this feature requires that sites consider new permutations of attack.

Control over the composition of resources creates some opportunities for attackers that don't exist in the current system.[6]

Consider the case where a critical security check is moved from resource A to resource B as part of a change to the system. As a result, there exists a composition of resources where the security check doesn't exist at all (version 2 of A with version 1 of B). If these resources are independently signed, an attacker can cause that system configuration to be used by a browser.

Similarly, content with security vulnerabilities that is signed can be provided to clients past the time that the problem is fixed. This allows attackers to extend the availability of zero-day bugs, allowing exploitation of that bug for the entire validity period of the content.

For these, the response of proponents of web packaging has generally been to identify ways in which these problems exist in the current system. For instance, incompatible versions of resources could be served by existing systems during upgrades. Caching might make this more than a transitory situation. It is also possible for caching to cause content that includes security vulnerabilities to persist for longer than is ideal.

The critical distinction between the existing system and the proposed system of signed exchanges is in the degree to which an attacker has influence over a situation. In the current Web, the origin can control the rollout timing and cache settings to avoid having inconsistent/mismatched resources. Signed exchanges gives an attacker complete control over the composition of resources, whereas the degree to which an attacker can influence resource composition is limited.

Importantly, an offline recipient has no recourse when consuming signed content. Revocation systems only work for online clients. The argument presented is that caching - and service workers in particular - can cause this situation to persist in ways that are very similar to web packaging. This argument suggests that limits on validity for caching and service workers are looser than in the proposed system, so it is potentially worse. The theory is that the opt-in characteristics are substantially similar if an attacker is able to deny active access to servers.

---

[6] This feature of the design as of May 2019 is likely to change, which might invalidate this style of attack.

Web packaging enables a shift from largely accidental exposure to attacker control. For a client with an active connection, security vulnerabilities persist only until the affected resources are revalidated (or clear site data is invoked). Thus, we conclude that this changes the risk profile for sites that opt in to signing of resources.

This style of attack might provide motivation for sites to move to a stronger model for managing the composition of resources. For instance, moving new versions of content to new URLs prevents an attacker from controlling composition as described. Though that could require a significant restructuring of sites and it contravenes long-standing advice about identifier stability, it has some advantages for performance and security even when signed exchanges are not used.

Is web packaging good for the web?

There is a lot to consider with web packaging. Many of the technical concerns are relatively minor. There are security problems, but most are well managed. There are operational concerns, but those can be overcome. It's a complex addition to the platform, but we can justify complication in exchange for significant benefits.

The question remains about whether this fundamental change to the way that content is delivered on the web represents a problematic shift in the power balance between actors. We have to consider whether aggregators could use this technology to impose their will on publishers.

Web packaging certainly has the effect of applying pressure toward consolidation of market share in a few worrying ways. It provides an incentive to support majority client populations at the expense of minorities. It increases the cost for an aggregator to provide optimal outbound links since the state-of-the-art that now requires package-based links and click prediction, and smaller sites may not be able to afford to do that.

Sites are being given significant incentive to deploy the technology. However, this incentive downplays the accompanying costs and increased exposure to new security problems that comes with deployment. There are some tools that help in making web packaging function, but these are incapable of also providing any structural changes to site architecture that would be needed to support a transition to securely providing origin substitution.

Big changes need strong justification and support. This particular change is bigger than most and presents a number of challenges. The increased exposure to security problems

and the unknown effects of this on power dynamics is significant enough that we have to regard this as harmful until more information is available.

We're actively working to understand this technology better. The Internet Architecture Board are organizing a workshop that aims to gather information about the bigger questions. That workshop is specifically structured to collect input from the publishing community. The technical details of the proposal will also be discussed at upcoming IETF meetings. Based on what we learn through these processes and our own investigation, we might be able to revise this position.

In addition to this, any assessment needs to look at not just harm, but weigh the benefits against costs. This is a technically complex set of changes to the web platform. In making an assessment about value, we have to see what benefits are realized, by whom, and to better understand who bears the costs.