Data Integrity for Mildly Active Content

Hilarie K. Orman
Volera, Inc.
ho@alum.mit.edu

August 14, 2001

Abstract

Web pages are becoming more complicated, both in appearance and in the underlying business models. Just as page composition for print media mix news, advertising, and regional content, web pages are increasingly the result of many hands. The Internet facilitates a "delayed evaluation" model for customization, and the composition process for pages is becoming a pipelined process, with each stage controlled by a different machine, in some cases under the auspices of different authorities, such as content delivery network operators. There is currently no formal model for defining data integrity in such an amorphous environment. An authorization and validation model can be constructed based on well-defined message parts and a simple composition language. The W3C XML signature standard serves as a base model for the definitions proposed in this paper.

1 Introduction

The HTTP protocol which delivers web pages to the world, has a simple "client-server" interaction model, and an even simpler "request-get semantic (request a named object, get the object), yet, in the real world protocol proxies do much of the processing. Proxies are of two sorts: those explicitly denoted as proxies by the client and those placed at the ingress point to a website in order to offload processing overhead from the origin servers. By extension, proxies can form a content distribution network in which the content is replicated to and served from diverse sites.

Although the original intent of proxies was for caching purposes that could reduce bandwidth costs and latency, today's proxies do much more than simply cache. They enforce access policy, they authenticate users, they present a uniform view of federated websites, they route requests through content delivery networks, they customize web pages, and they offload backend processing. The result of all this activity is to leave the website designer with the problem of designing content rather than worrying about how to scale his web server farm.

In accomplishing these goals, proxies carry out changes to the requests and content responses that represent the intention of the content author and/or the content owner (publisher). However, the proxies may be owned and administered by other parties. Thus, the right to modify the content has been implicitly delegated to these parties.

The model presented here allows the content owners intentions to be expressed as part of the content message, it allows fine-grained delegation of modification rights, and it allows any party to validate the message with respect to owner's intentions, even if several proxies are involved in the modification process. In particular, the requester can validate the final message. The method is friendly to caches in that partially modified message forms can be cached and because the normal content is separate from the authorization and validation information.

The model also supports delegation of modification rights from users to trusted proxies. The trusted proxies perform policy resolution functions.

2 Overview of Method

In our model, the content owner specifies content as a set of parts, some of which are immutable and some of which are replaceable. Each part has permissions, and the set of parts and their permissions is the message "manifest", an index to the message. The content owner's signature on the manifest specifies his intentions.

In order for the owner to delegate modification rights verifiably to parts of a message, the message must have well-defined part boundaries. This can be accomplished by specifying byte ranges or "chunk" parts or formal parts with MIME or other standards. The manifest names each part and its hash value: non-invertible, collision-resistant function of each byte of the part.

The modification rights for a part include both the permitted action and the identification of the parties authorized to perform the action. The rights may be further modified to specify content type, size, resolution, method, etc.

Each party that modifies the message (an "editor") in accordance with the owner's instructions must attach an action notification to the message. This refers to the permission in the manifest, the message part, the action, the hash of manifest, the identity of the editor, and a signature over these items.

Because the manifest and the signed actions are separate from the "content", the content remains cacheable, even in a partially modified form (i.e., only some of the permitted editing actions have been carried out).

3 The Manifest

The manifest is the most important element of the validation scheme. The publisher of complex content must provide a complete description of the message contents, part by part, and what can be done to these parts and by whom.

As a simple example, not based on any particular encoding standard, consider the following:

```
<Manifest>
<Part><PartID one><Part Digest A34FC22B84>
          <Permission Replace><Replace-Type Transcode>
  <Replace-Restriction size lessthan 40K>
                  <EditorID "www.languages.com">
</Part>
<Part>
<PartID two><Part Digest 43DDC019A2>
<PartCombine StrictOR>
<PartID four><PartDigest 324089AB10>
          <Permission Delete>
                  <EditorID any>

</Part>
<Part><PartID three><Part Digest BF96C4E31>
          <Permission Append><Append-Type Content-type Text>
                  <Delegate "www.contentnetwork.com">
                  <Delegate "regionaltext.com">
</Part>
<ManifestDigest 85E3A3D077>
<ContentPublisher "OnlineNewsMagazine.com">
<Signature 437892F3982ED190A8C928B40DE3FAC14E89A0843902>
</Manifest>
```

The permissions constitute a small editing language. In the above, the first part can be replaced by a transcoder at www.languages.com if the replacement content is less than 40K bytes. The second or fourth part can be deleted by any entity. The third part can be immediately followed by new part inserted by either www.contentnetwork.com or regionaltext.com, or a party delegated by either one of them.

The manifest concept is similar to the W3C's XML Digital Signature standard. That defines a complete set of identifiers to support the cryptography behind using digital signatures for validating content. The encoding method for the data, the hash digest algorithm, the digest itself, the digital signature method, the identifiers, and the public key representation are all detailed in that specification.

The W3C specification further allows the manifest to contain information about content that might not be part of the current content, such as the contents associated with a URL. This kind of implicitly included content is also useful for complex content, if the referenced data is relatively long-lived (if it isn't, the hash value would change and the content would not be valid for the cited message).

3.1 Permissions

The editing language is a simple one, encompassing the actions necessary for supporting a template with a combination of immutable content, unspecified content, and content options. The permissions are

1. delete: removed the named message part.
2. replace: replace the named part with a new (possibly restricted) part.
3. insert: add a new message part after the named part.
4. transcode: similar to “replace”, the transcode option allows a new version of a message part to replace the original one.
5. delegate: allow the subject to delegate his authority to another
6. alternatives: the publisher specifies a boolean combination of message parts from which editors can select any valid combination

A further, less simple, action is “execute”, discussed later in this document.

4 Signed Actions and Validation

Each action taken by an editor must be unambiguously noted and bound to the editor’s identity. By using special message parts, the editor can do this without causing security information to intrude into the normal message content.

The required parts of an action are:

1. The hash value of the manifest for the message
2. The action (e.g., “delete”)
3. The identifier of message part on which the action was performed (e.g., “Part 2”
4. The editor’s identity
5. The editor’s signature over the hash of the above items

The editor must carry out the signed action on the normal message content. If the editor does not do this, the validation will fail. For example, if the editor signs a “delete” action for the part identified as “Part 2” but leaves the part intact, the message is invalid. A receiver should reject the message because it is possible that an unauthorized party inserted the content after an authorized editor deleted it.

Although it should be possible for the recipient to determine that this happened and to elect to delete the message part itself, this would so greatly complicate the editing semantics that it should be disallowed.

Interestingly, an editor may re-introduce content deleted by another editor without violating security properties. This is because the original message is valid with or without content with "delete" permission. Because both forms are legal, if an editor has access to the original version and the version with a valid deletion, the editor is free to use either version.

Moreover, many versions of messages with "replace" permission in the manifest may exist, and they are all valid. Is there any message integrity with respect to the intended audience of a modified message. That is, if an editor created two versions of a message, one intended for ırecipient1 and the other for ırecipient2, is there an enforceable way to restrict delivery of the two messages, so that they do not reach the wrong parties?

This security model does not prevent misdelivery, but it can address the problem by making it detectable. An editor can add an element to a signed action that limits the delivery intentions. This is the "destination" item, and it can name the intended recipient directly (e.g. "johndoe@dotcom.com") or some aspect of the acceptable delivery set (e.g., "p3p sports").

The recommended security policy is intended to be "monotonic" in the sense that privileges and the set of possible actions can only decrease as the message moves between editors.

4.1 Caching Parts

These methods are friendly to caches because each message part can be considered an independent, cacheable item. The hash value of the part uniquely identifies it, and caches could use fine-grained caching strategies to keep the parts and manifests as separate items.

4.2 Handling message variants

The publisher of the complex messages is actually publishing several possible messages under one name (the URL). Because any authorized modification results in a new message, caches must introduce the notion of page variants. While there are provisions for variants today, they are assumed to be controlled by the publisher, only. In order to be effective at caching complex messages, the caches will need to retain more information about when each variant is to be used.

5 Delegation

The permission to "delegate" is an important one for the security model. With it, an editor can formally give permission to another party to carry out actions in his permissible action set. The delegation is a traceable, verifiable action. Delegation is essential for systems with several independent entities bound is business partnerships because it allows organizations to extend their business relationships in a scalable manner.

In basic delegation, the content publisher includes a delegation permission in the manifest. The permission names an editor and the actions which he can delegate to others for modifying the message. Additional options for delegation include:

1. Depth. The number of redelegations allowed. By default, the depth is zero.
2. Subjects. The allowable subjects to whom the delegating editor can authorize for assuming modification permissions.

In "part delegation" the delegation information is associated with a particular manifest item, such as "modify part2". By including a "delegate" permission with the "modify" permission, the content publisher can permit delegation of that single right for the current message.

An editor named as the main subject in a delegation right can either carry out the editing action himself or delegate it to another. The delegation is an action, requiring a signed action payload. This payload must include the identity of the new editor and a reference to the manifest item giving delegation permission to the current editor. This action is signed and attached to the message.

A delegate carries out the delegated actions just as the original editor would, by attaching signed action payloads.

6 Requests, Responses, Protocol and Content

Most of the editing privileges make sense in the context of web pages that are like newspaper pages with space for advertising, personal ads, etc. around the news content and distinctive publisher graphics. But web-based protocols include requests for content, not just responses with the content. What are the security issues surrounding requests?

The most important issue concerns the binding of the request to the response. If there are proxy servers between the requester and the site serving the requests, it is possible that the proxy might alter the request, invisibly to the requester, and return unrelated content. For example, the browser line might read "//www.securitysite.org/importantstuff.html", the proxy would change the request to "//www.intrudersite.com" and return content from there. What verifiable measures are there to relate request and response?

This is not a simple problem, because today's content distribution networks need to have many, dispersed servers prepared to serve up content when requests come in. They accomplish the distribution through DNS manipulations, redirections via the HTTP protocol, and name mapping at the proxy servers. For example, a request for `a.b.com/today.html` might turn into a request for `xxx0012.contentnetwork.com/a.b.com.today.html`.

What we would like is a verified binding between the client request and the server's reply:

```
Alice: I'd like to see the page at a.b.com/today.html.
Response:
  1.  You requested a.b.com/today.html in a message (with hash xxxxx)
  2.  The page following this header is
     from xxx0012.contentnetwork.com/a.b.com.today.html
  3.  a.b.com authorized distribution of its web content from the
       contentnetwork.com organization
  4.  contentnetwork.com authorized URL rewriting within its
     xxx*.contentnetwork.com domain to yourisp.com
  5.  yourisp.com rewrote the request URL from
            a.b.com/today.html
            to  xxx0012.contentnetwork.com/a.b.com.today.html
  6.  yourisp.com has permission from you to rewrite URL's for all
     domains with which it has a business relationship
  7.  contentnetwork.com attests that this page is a valid
    interpretation of the original page, a.b.com/today.html,
  (as received in a request from you with hash xxxxx).
```

This is a rather lengthy authorization chain, but it does show how it is possible to establish a strong binding between the exact request and the response. The key elements are the authorizations to rewrite URL's and to act as a content server. These are policy statements, relating to delegated authority for classes of actions.

6.1 Policy

Policy statements apply to a class of messages, not just one message. The manifest format can be used to encode and transmit policy statements to receivers that will need them to verify message transformations. These policies can be cached, and the caching can reduce the bulk of request/response binding messages such as the one above.

In that example, the sixth statement refers to a user-supplied policy statement that delegates rewriting authority from the user to his Internet Service Provider (ISP). The "request_rewrite" permission is a new one for the manifest. It requires the identity of the rewriting authority and an optional set of target domains for which the rewriting is permitted. This rewriting permission is implicitly given by users of proxies today, but it is useful to have it expressed in formal protocol elements as well. This applies to proxies acting on behalf of users and proxies acting on behalf of web servers (reverse proxies).

Another new privilege (or permission) is that of acting as a content distributor: surrogate. This allows the surrogate organization to distribute content as if it were the publisher's organization. Content distribution networks do this today, again, with implicit permission from the publisher.

Endusers can use the permission system to set policy with regard to message modification. As an example, a user might specify `delegate myenterprise_proxy delete any part inserted by 'badguys.com'`

or
`delegate myisp delete all URLs referring to badguys.com/*.gif notify`
or
`delegate myisp replace any content-type=gif authorized by owner with content-type=gif from goodguys.com`.

User policy is specified in a signed manifest. The delegated authority must add a signed action to messages modified in accordance with user policies that require "notify". Users may also indicate that they do not want to see the data integrity items (manifest, action signatures, etc.).

Other user-based policies can apply to fine-grained operations such as authentication delegation (sometimes a part of "single sign-on"), long-term state information management such as cookies, user preference information, etc.

Policy is intended to be "soft state" information, because the recipient of the policy need not permanently hold the information, but instead should be able to query for it at any time. The protocol for sending and querying for policy must be defined, as must the caching rules.

7 Executable Content

The manifest supplies a small language that defines many possible legal messages from one, and some of the message parts are left to editors. Suppose the publisher would like to control what the editors can do more precisely than these permissions allow. One way of doing this is to use programs embedded in the content.

There are several possible forms of programs in content. One form merely affects the cacheability of a particular message part. The program specifies the conditions under which the message part is valid, and it supplies a URL to use when refreshing the part.

Another form of executable content is demonstrated in Edge Side Includes, a tagged language embedded in HTML documents. These indicate content alternatives.

A third form goes under various names, including "proxylets", "cachelets", "active message", etc. These are programs written in a general purpose programming language of some sort, and the editors are given permission to execute the programs locally. Fully executable content may damage editors or have unintended consequences when run in the editor's environment, so its use is controversial. However, the concept is attractive because of its potential for moving processing from overloaded web servers to distributed application servers provisioned for high intensity use.

Some languages are more suited to proxylet use than others, and specialized scripting languages, perhaps coded in XML, are a possibility. But there is a second part to the security question, which is whether or not the message actually derives from the program. If it were easy to verify that an editor ran a program supplied by the publisher correctly, and that such a program produced the output shown in a content message, executable content might be much less controversial. Limited scripting languages, specialized to content manipulations such as image compression, for example, might be verifiable enough for practical purposes.

As a practical first step towards safe use of executable content, the "execute" permission should designate not just the subjects allowed to replace a message part with the execution of a program, but also the subjects allowed to execute the program. This second subject is meant to be an organization with security controls in place so that the organization will agree to be held accountable for the faithful execution of the program. An editor, for example, might contract with such an organization for secure execution of proxylets, and the second organization would sign both the result of the computation and the hash value of the program used to generation the result.

8 Conclusions

It is possible to introduce at least one definition of data integrity into a protocol that transports complex content. While many details remain to be determined before deployment, there is existing work that shows that it can be done. The methods may have extra overhead in carrying out the required digital signatures, but they are "cache friendly" and allow fine-grained caching.

There are several possible directions to focus on to progress the concepts in this paper. User policy and encoding standards are the ones of most immediate importance. Further refinement of the permissions and their parameters will follow from the intended use environments.

References

[1] Eastlake, Reagle, Solo, eds., "XML-Signature Syntax and Processing W3C Candidate Recommendation", 19-April-2001, http://www.w3.org/TR/xmldsig-core/

[2] Fielding, et al., Hypertext Transfer Protocol – HTTP/1.1, http://www.w3.org/Protocols/rfc2068/rfc2068