Network Working Group
Request for Comments: 3230
Category: Standards Track

J. Mogul
Compaq WRL
A. Van Hoff
Marimba

January 2002

Instance Digests in HTTP

Status of this Memo

Copyright Notice

Abstract

HTTP/1.1 defines a Content-MD5 header that allows a server to include a digest of the response body. However, this is specifically defined to cover the body of the actual message, not the contents of the full file (which might be quite different, if the response is a Content-Range, or uses a delta encoding). Also, the Content-MD5 is limited to one specific digest algorithm; other algorithms, such as SHA-1 (Secure Hash Standard), may be more appropriate in some circumstances. Finally, HTTP/1.1 provides no explicit mechanism by which a client may request a digest. This document proposes HTTP extensions that solve these problems.

Table of Contents

1 Introduction..... 2
1.1 Other limitations of HTTP/1.1..... 3
2 Goals..... 4
3 Terminology..... 5
4 Specification..... 6
4.1 Protocol parameter specifications..... 6
4.1.1 Digest algorithms..... 6
4.2 Instance digests..... 7
4.3 Header specifications..... 8
4.3.1 Want-Digest..... 8
4.3.2 Digest..... 9
5 Negotiation of Content-MD5..... 9

6 IANA Considerations..... 10
7 Security Considerations..... 10
8 Acknowledgements..... 10
9 References..... 10
10 Authors' Addresses..... 12
11 Full Copyright Statement..... 13

1 Introduction

Although HTTP is typically layered over a reliable transport protocol, such as TCP, this does not guarantee reliable transport of information from sender to receiver. Various problems, including undetected transmission errors, programming errors, corruption of stored data, and malicious intervention can cause errors in the transmitted information.

A common approach to the problem of data integrity in a network protocol or distributed system, such as HTTP, is the use of digests, checksums, or hash values. The sender computes a digest and sends it with the data; the recipient computes a digest of the received data, and then verifies the integrity of this data by comparing the digests.

Checksums are used at virtually all layers of the IP stack. However, different digest algorithms might be used at each layer, for reasons of computational cost, because the size and nature of the data being protected varies, and because the possible threats to data integrity vary. For example, Ethernet uses a Cyclic Redundancy Check (CRC). The IPv4 protocol uses a ones-complement checksum over the IP header (but not the rest of the packet). TCP uses a ones-complement checksum over the TCP header and data, and includes a "pseudo-header" to detect certain kinds of programming errors.

HTTP/1.1 [4] includes a mechanism for ensuring message integrity, the Content-MD5 header. This header is actually defined for MIME-conformant messages in a standalone specification [10]. According to the HTTP/1.1 specification,

> The Content-MD5 entity-header field [...] is an MD5 digest of the entity-body for the purpose of providing an end-to-end message integrity check (MIC) of the entity-body.

HTTP/1.1 borrowed Content-MD5 from the MIME world based on an analogy between MIME messages (e.g., electronic mail messages) and HTTP messages (requests to or responses from an HTTP server).

As discussed in more detail in section 3, this analogy between MIME messages and HTTP messages has resulted in some confusion. In particular, while a MIME message is self-contained, an HTTP message might not contain the entire representation of the current state of a resource. (More precisely, an HTTP response might not contain an entire "instance"; see section 3 for a definition of this term.)

There are at least two situations where this distinction is an issue:

1. When an HTTP server sends a 206 (Partial Content) response, as defined in HTTP/1.1. The client may form its view of an instance (e.g., an HTML document) by combining a cache entry with the partial content in the message.

2. When an HTTP server uses a "delta encoding", as proposed in a separate document [9]. A delta encoding represents the changes between the current instance of a resource and a previous instance, and is an efficient way of reducing the bandwidth required for cache updates. The client forms its view of an instance by applying the delta in the message to one of its cache entries.

We include these two kinds of transformations in a potentially broader category we call "instance manipulations."

In each of these cases, the server might use a Content-MD5 header to protect the integrity of the response message. However, because the MIC in a Content-MD5 header field applies only to the entity in that message, and not to the entire instance being reassembled, it cannot protect against errors due to data corruption (e.g., of cache entries), programming errors (e.g., improper application of a partial content or delta), certain malicious attacks [9], or corruption of certain HTTP headers in transit.

Thus, the Content-MD5 header, while useful and sufficient in many cases, is not sufficient for verifying instance integrity in all uses of HTTP.

The Digest Authentication mechanism [5] provides (in addition to its other goals) a message-digest function similar to Content-MD5, except that it includes certain header fields. Like Content-MD5, it covers a specific message, not an entire instance.

1.1 Other limitations of HTTP/1.1

Checksums are not free. Computing a digest takes CPU resources, and might add latency to the generation of a message. (Some of these costs can be avoided by careful caching at the sender’s end, but in

many cases such a cache would not have a useful hit ratio.) Transmitting a digest consumes HTTP header space (and therefore increases latency and network bandwidth requirements.) If the message recipient does not intend to use the digest, why should the message sender waste resources computing and sending it?

The Content-MD5 header, of course, implies the use of the MD5 algorithm [15]. Other algorithms, however, might be more appropriate for some purposes. These include the SHA-1 algorithm [12] and various "fingerprinting" algorithms [7]. HTTP currently provides no standardized support for the use of these algorithms.

HTTP/1.1 apparently assumes that the choice to generate a digest is up to the sender, and provides no mechanism for the recipient to indicate whether a checksum would be useful, or what checksum algorithms it would understand.

2 Goals

The goals of this proposal are:

1. Digest coverage for entire instances communicated via HTTP.

2. Support for multiple digest algorithms.

3. Negotiation of the use of digests.

The goals do not include:

- header integrity
  The digest mechanisms described here cover only the bodies of instances, and do not protect the integrity of associated "entity headers" or other message headers.

- authentication
  The digest mechanisms described here are not meant to support authentication of the source of a digest or of a message or instance. These mechanisms, therefore, are not sufficient defense against many kinds of malicious attacks.

- privacy
  Digest mechanisms do not provide message privacy.

- authorization
  The digest mechanisms described here are not meant to support authorization or other kinds of access controls.

The Digest Access Authentication mechanism [5] can provide some integrity for certain HTTP headers, and does provide authentication.

3 Terminology

HTTP/1.1 [4] defines the following terms:

resource — A network data object or service that can be identified by a URI, as defined in section 3.2. Resources may be available in multiple representations (e.g. multiple languages, data formats, size, resolutions) or vary in other ways.

entity — The information transferred as the payload of a request or response. An entity consists of metainformation in the form of entity-header fields and content in the form of an entity-body, as described in section 7.

variant — A resource may have one, or more than one, representation(s) associated with it at any given instant. Each of these representations is termed a ‘variant.’ Use of the term ‘variant’ does not necessarily imply that the resource is subject to content negotiation.

The dictionary definition for "entity" is "something that has separate and distinct existence and objective or conceptual reality" [8]. Unfortunately, the definition for "entity" in HTTP/1.1 is similar to that used in MIME [6], based on an entirely false analogy between MIME and HTTP.

In MIME, electronic mail messages do have distinct and separate existences. MIME defines "entity" as something that "refers specifically to the MIME-defined header fields and contents of either a message or one of the parts in the body of a multipart entity."

In HTTP, however, a response message to a GET does not have a distinct and separate existence. Rather, it is describing the current state of a resource (or a variant, subject to a set of constraints). The HTTP/1.1 specification provides no term to describe "the value that would be returned in response to a GET request at the current time for the selected variant of the specified resource." This leads to awkward wordings in the HTTP/1.1 specification in places where this concept is necessary.

It is too late to fix the terminological failure in the HTTP/1.1 specification, so we instead define a new term, for use in this document:

instance
: The entity that would be returned in a status-200 response to a GET request, at the current time, for the selected variant of the specified resource, with the application of zero or more content-codings, but without the application of any instance manipulations or transfer-codings.

It is convenient to think of an entity tag, in HTTP/1.1, as being associated with an instance, rather than an entity. That is, for a given resource, two different response messages might include the same entity tag, but two different instances of the resource should never be associated with the same (strong) entity tag.

We also define this term:

instance manipulation
: An operation on one or more instances which may result in an instance being conveyed from server to client in parts, or in more than one response message. For example, a range selection or a delta encoding. Instance manipulations are end-to-end, and often involve the use of a cache at the client.

4 Specification

In this specification, the key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" are to be interpreted as described in RFC 2119 [2].

4.1 Protocol parameter specifications

4.1.1 Digest algorithms

Digest algorithm values are used to indicate a specific digest computation. For some algorithms, one or more parameters may be supplied.

```
digest-algorithm = token
```

The BNF for "parameter" is as is used in RFC 2616 [4]. All digest-algorithm values are case-insensitive.

The Internet Assigned Numbers Authority (IANA) acts as a registry for digest-algorithm values. Initially, the registry contains the following tokens:

MD5	The MD5 algorithm, as specified in RFC 1321 [15]. The output of this algorithm is encoded using the base64 encoding [1].
SHA	The SHA-1 algorithm [12]. The output of this algorithm is encoded using the base64 encoding [1].
UNIXsum	The algorithm computed by the UNIX "sum" command, as defined by the Single UNIX Specification, Version 2 [13]. The output of this algorithm is an ASCII decimal-digit string representing the 16-bit checksum, which is the first word of the output of the UNIX "sum" command.
UNIXcksum	The algorithm computed by the UNIX "cksum" command, as defined by the Single UNIX Specification, Version 2 [13]. The output of this algorithm is an ASCII digit string representing the 32-bit CRC, which is the first word of the output of the UNIX "cksum" command.

If other digest-algorithm values are defined, the associated encoding MUST either be represented as a quoted string, or MUST NOT include ";" or "," in the character sets used for the encoding.

4.2 Instance digests

An instance digest is the representation of the output of a digest algorithm, together with an indication of the algorithm used (and any parameters).

```
instance-digest = digest-algorithm "="
                          <encoded digest output>
```

The digest is computed on the entire instance associated with the message. The instance is a snapshot of the resource prior to the application of of any instance manipulation or transfer-coding (see section 3). The byte order used to compute the digest is the transmission byte order defined for the content-type of the instance.

Note: the digest is computed before the application of any instance manipulation. If a range or a delta-coding [9] is used, the computation of the digest after the computation of the range or delta would not provide a digest useful for checking the integrity of the reassembled instance.

The encoded digest output uses the encoding format defined for the specific digest-algorithm. For example, if the digest-algorithm is "MD5", the encoding is base64; if the digest-algorithm is "UNIXsum", the encoding is an ASCII string of decimal digits.

Examples:

```
MD5=HUXZLQLMuI/KZ5KDcJPcOA==
sha=thvDyvhfIqlvFe+A9MYgxAfm1q5=
UNIXsum=30637
```

4.3 Header specifications

The following headers are defined.

4.3.1 Want-Digest

The Want-Digest message header field indicates the sender's desire to receive an instance digest on messages associated with the Request-URI.

```
Want-Digest = "Want-Digest" ":"
                  #(digest-algorithm [ ";" "q" "=" qvalue])
```

If a digest-algorithm is not accompanied by a qvalue, it is treated as if its associated qvalue were 1.0.

The sender is willing to accept a digest-algorithm if and only if it is listed in a Want-Digest header field of a message, and its qvalue is non-zero.

If multiple acceptable digest-algorithm values are given, the sender's preferred digest-algorithm is the one (or ones) with the highest qvalue.

Examples:

```
Want-Digest: md5
Want-Digest: MD5;q=0.3, sha;q=1
```

4.3.2 Digest

The Digest message header field provides a message digest of the instance described by the message.

```
Digest = "Digest" ":" #(instance-digest)
```

The instance described by a message might be fully contained in the message-body, partially-contained in the message-body, or not at all contained in the message-body. The instance is specified by the Request-URI and any cache-validator contained in the message.

A Digest header field MAY contain multiple instance-digest values. This could be useful for responses expected to reside in caches shared by users with different browsers, for example.

A recipient MAY ignore any or all of the instance-digests in a Digest header field.

A sender MAY send an instance-digest using a digest-algorithm without knowing whether the recipient supports the digest-algorithm, or even knowing that the recipient will ignore it.

Examples:

```
Digest: md5=HUXZLQLMuI/KZ5KDcJPcOA==
Digest: SHA=thvDyvhfIqlvFe+A9MYgxAfm1q5=,unixsum=30637
```

5 Negotiation of Content-MD5

HTTP/1.1 provides a Content-MD5 header field, but does not provide any mechanism for requesting its use (or non-use). The Want-Digest header field defined in this document provides the basis for such a mechanism.

First, we add to the set of digest-algorithm values (in section 4.1.1) the token "contentMD5", with the provision that this digest-algorithm MUST NOT be used in a Digest header field.

The presence of the "contentMD5" digest-algorithm with a non-zero qvalue in a Want-Digest header field indicates that the sender wishes to receive a Content-MD5 header on messages associated with the Request-URI.

The presence of the "contentMD5" digest-algorithm with a zero qvalue in a Want-Digest header field indicates that the sender will ignore Content-MD5 headers on messages associated with the Request-URI.

6 IANA Considerations

The Internet Assigned Numbers Authority (IANA) administers the name space for digest-algorithm values. Values and their meaning must be documented in an RFC or other peer-reviewed, permanent, and readily available reference, in sufficient detail so that interoperability between independent implementations is possible. Subject to these constraints, name assignments are First Come, First Served (see RFC 2434 [11]).

7 Security Considerations

This document specifies a data integrity mechanism that protects HTTP instance data, but not HTTP entity headers, from certain kinds of accidental corruption. It is also useful in detecting at least one spoofing attack [9]. However, it is not intended as general protection against malicious tampering with HTTP messages.

The HTTP Digest Access Authentication mechanism [5] provides some protection against malicious tampering.

8 Acknowledgements

It is not clear who first realized that the Content-MD5 header field is not sufficient to provide data integrity when ranges or deltas are used.

Laurent Demailly may have been the first to suggest an algorithm-independent checksum header for HTTP [3]. Dave Raggett suggested the use of the term "digest" instead of "checksum" [14].

9 References

[1] Freed, N. and N. Borenstein, N., "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 2049, November 1996.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Laurent Demailly. Re: Revised Charter. http://www.ics.uci.edu/pub/ietf/http/hypermail/1995q4/0165.html.

[4] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1.", RFC 2616, June 1999.

[5] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[6] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[7] Nevin Heintze. Scalable Document Fingerprinting. Proc. Second USENIX Workshop on Electronic Commerce, USENIX, Oakland, CA, November, 1996, pp. 191-200. http://www.cs.cmu.edu/afs/cs/user/nch/www/koala/main.html.

[8] Merriam-Webster. Webster's Seventh New Collegiate Dictionary. G. & C. Merriam Co., Springfield, MA, 1963.

[9] Mogul, J., Krishnamurthy, B., Douglis, F., Feldmann, A., Goland, Y. and A. van Hoff, "Delta encoding in HTTP", RFC 3229, December 2001.

[10] Myers, J. and M. Rose, "The Content-MD5 Header Field", RFC 1864, October 1995.

[11] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[12] National Institute of Standards and Technology. Secure Hash Standard. FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION 180-1, U.S. Department of Commerce, April, 1995. http://csrc.nist.gov/fips/fip180-1.txt.

[13] The Open Group. The Single UNIX Specification, Version 2 - 6 Vol Set for UNIX 98. Document number T912, The Open Group, February, 1997.

[14] Dave Raggett. Re: Revised Charter. http://www.ics.uci.edu/pub/ietf/http/hypermail/1995q4/0182.html.

[15] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

10 Authors' Addresses

Jeffrey C. Mogul
Western Research Laboratory
Compaq Computer Corporation
250 University Avenue
Palo Alto, California, 94305, U.S.A.

EMail: JeffMogul@acm.org
Phone: 1 650 617 3304 (email preferred)

Arthur van Hoff
Marimba, Inc.
440 Clyde Avenue
Mountain View, CA 94043

EMail: avh@marimba.com
Phone: 1 (650) 930 5283

11 Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.