Network Working Group	A. Bryan
Internet-Draft	T. Kosse
Intended status: Experimental	D. Stenberg
Expires: October 10, 2010	April 08, 2010

TOC

FTP Extensions for Cryptographic Hashes
draft-bryan-ftp-hash-01

Abstract

The File Transfer Protocol does not offer any method to verify the integrity of a transferred file, nor can two files be compared against each other without actually transferring them first. Cryptographic hashes are a possible solution to this problem. In the past, several attempts have been made to add commands to obtain checksums and hashes, however none have been formally specified, leading to non-interoperability and confusion. To solve these issues, this document specifies a new FTP command to be used by clients to request cryptographic hashes of files.

Status of this Memo

Copyright Notice

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

TOC

The File Transfer Protocol [RFC0959] (Postel, J. and J. Reynolds, “File Transfer Protocol,” October 1985.) does not offer any method to verify the integrity of a transferred file, nor can two files be compared against each other without actually transferring them first. Cryptographic hashes are a possible solution to this problem. In the past, several attempts have been made to add commands to obtain checksums and hashes, however none have been formally specified, leading to non-interoperability and confusion. To solve these issues, this document specifies a new FTP command to be used by clients to request cryptographic hashes of files. HTTP has a similar feature named Instance Digests [RFC3230] (Mogul, J. and A. Van Hoff, “Instance Digests in HTTP,” January 2002.) which allows a client to request the cryptographic hash of a file.

[[Discussion of this draft should take place on apps-discuss@ietf.org.]]

1.1. Examples

TOC

Example of HASH client request:

```
HASH filename.ext
```

HASH server response with Positive Completion code and the requested hash using the currently selected algorithm:

```
213 80bc95fd391772fa61c91ed68567f0980bb45fd9
```

---

2. Notational Conventions

TOC

This specification describes conformance of FTP Extensions for cryptographic hashes.
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.), as scoped to those conformance targets.
This document also uses notation defined in STD 9, [RFC0959] (Postel, J. and J. Reynolds, “File Transfer Protocol,” October 1985.).
Syntax required is defined using the Augmented BNF defined in [RFC5234] (Crocker, D. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” January 2008.).

---

3. The HASH Command (HASH)

TOC

The HASH command allows for requesting the cryptographic hash of a file.
The syntax for the HASH command is:

```
hash = "HASH" SP <pathname>
```

As with all FTP commands, the "HASH" command label is interpreted in a case-insensitive manner.
The HASH command keyword MUST be followed by a single space (ASCII 32) followed by the pathname.
The pathname argument should reference the same file as other file based commands such as STOR or RETR which the same argument would reference.
The text returned in response to the HASH command MUST be:

```
hash-response = "213" SP 1*HEXDIGIT CRLF
```

All hash values MUST be encoded in lowercase hexadecimal format.
The standard negative error codes 500 and 501 are sufficient to handle all errors involving the HASH command (e.g., syntax errors). Response code 550 is used if the file can not be found. Response code 552 is used if the user isn't allowed to use the HASH command. Response code 450 is used to indicate the server is busy, e.g. already hashing other files yet inviting the client to retry in future.
The HASH command is useful for files transmitted in Image type mode (TYPE I) and Stream transfer mode (MODE S).

3.1. FEAT response for HASH

TOC

A server that supports HASH should advertise it in FEAT response [RFC2389] (Hethmon, P. and R. Elz, “Feature negotiation mechanism for the File Transfer Protocol,” August 1998.) with a list of all supported hash algorithms in a semicolon separated list. The hash algorithm that is currently selected is marked with an asterisk. In the example below, the "C>" lines are commands from user-PI to server-PI, the "S>" lines are server-PI replies.

```
C> feat
S> 211-Extensions supported:
S>  SIZE
S>  COMPRESSION
S>  HASH SHA-1*;MD5
S>  MDTM
S> 211 END
```

The IANA registry named "Hash Function Textual Names" defines values for hash types. Hash names should be presented in uppercase, but comparisons should be case-insensitive, e.g. MD5, md5, Md5 are all the same.

```
hash-feat = SP "HASH" SP hashlist CRLF
hashlist = 1*( hashname ["*"] ";" )
hashtype = 1*( hchar )
hchar = ALPHA / DIGIT / "-" / "_" / "/" / "." / ","
```

TOC

3.2. Changing the HASH algorithm

To query the current hash algorithm and to change it, the OPTS command as defined in [RFC2389] (Hethmon, P. and R. Elz, “Feature negotiation mechanism for the File Transfer Protocol,” August 1998.) is used with HASH as the first argument. If no second argument is passed, OPTS HASH simply returns the currently selected hash algorithm. To change the algorithm, a valid hashtype has to be given as second argument. If the command is successful, all future calls to HASH until the next successful OPTS HASH command or until the session is reinitialized (REIN) will use the selected hash algorithm.

```
C> OPTS HASH
S> 200 SHA-1
C> OPTS HASH SHA-512
S> 200 SHA-512
C> OPTS HASH CRC-37
S> 501 Unknown algorithm, current selection not changed


hashopts-cmd = "OPTS HASH" [ SP hashtype ] CRLF
hashopts-response = "200" SP hashtype CRLF
```

---

4. Command Usage

TOC

Client requests the cryptographic hash of a file with HASH command. Server replies with cryptographic hash of file. Client downloads file. Client hashes the downloaded file and compares its hash to the hash obtained from the server. This command could also be used to verify that an uploaded file is an exact copy.

---

5. IANA Considerations

TOC

This new command is added to the "FTP Commands and Extensions" registry created by [RFC5797] (Klensin, J. and A. Hoenes, “FTP Command and Extension Registry,” March 2010.).
Command Name: HASH
Description: Cryptographic Hash of a file
FEAT String: HASH
Command Type: Service execution
Conformance Requirements: Optional
Reference: This specification

6. Implementation Requirements

TOC

All conforming implementations MUST at least support the SHA-1 algorithm. Implementations SHOULD NOT make any algorithm the default that is known to be weaker than SHA-1. Support for any additional algorithms is optional.

7. Security Considerations

TOC

Calculating a file's hash is a CPU intensive operation and can easily consume the available disk I/O resources. If the HASH command isn't implemented carefully, a server could be vulnerable to a denial of service attack. On an affected server a malicious user could, for example, continuously send HASH commands over multiple connections and thus consume most of the FTP server's CPU and disk I/O resources, leaving little room for other operations. To mitigate this risk, a server SHOULD cache the calculated hashes so that the hash of a file is only calculated once even if multiple hash requests are sent for that file.

The performance of commonly used hard disk drives is adversely affected by the amount of time the device needs to reposition its read-and-write heads. A server SHOULD therefore avoid hashing multiple files at the same time which are located on the same physical media and SHOULD instead hash them sequentially. A possible solution is to use the 450 reply code of HASH to indicate that the server is already busy with another HASH operation.

In addition, the HASH command can be used to draw conclusions about the contents of a file. If the hash of a file on some server matches the hash of some known, local file, both files are likely identical. To prevent this scenario it suffices to limit use of the HASH command to users who would already be able to download the file.

8. References

TOC

8.1. Normative References

TOC

[RFC0959]	Postel, J. and J. Reynolds, “File Transfer Protocol,” STD 9, RFC 0959, October 1985.
[RFC2119]	

	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.
[RFC2389]	Hethmon, P. and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol," RFC 2389, August 1998.
[RFC5234]	Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," STD 68, RFC 5234, January 2008.

8.2. Informative References

TOC

[RFC3230]	Mogul, J. and A. Van Hoff, "Instance Digests in HTTP," RFC 3230, January 2002.
[RFC5797]	Klensin, J. and A. Hoenes, "FTP Command and Extension Registry," RFC 5797, March 2010.
[draft-twine-ftpmd5]	Twine, J., "The MD5 and MMD5 FTP Command Extensions," draft-twine-ftpmd5-00 (work in progress), May 2002.

Appendix A. Acknowledgements and Contributors

TOC

Thanks to John C. Klensin, Alfred Hoenes, Daniel Stenberg, and James Twine.

Appendix B. List of Implementations with Non-standard Cryptographic Hash Command

TOC

[[to be removed by the RFC editor before publication as an RFC.]]
At least one previous Internet Draft [draft-twine-ftpmd5] (Twine, J., "The MD5 and MMD5 FTP Command Extensions," May 2002.) attempted to address this issue (it only supported one hash, MD5).
An incomplete list of FTP clients and servers that have implemented multiple commands (XMD5, XSHA1, SITE SHOHASH, etc) that are not formally specified, leading to non-interoperability and confusion.

*Akamai NetStorage p17-18 http://pigdogslow.dyndns.org/ NetStorage_UserGuide.pdf

*Apache Ftp Server (supports draft-twine-ftpmd5) http:// cwiki.apache.org/FTPSERVER/documentation.html

*Cerberus FTP server http://www.softpedia.com/progChangelog/ Cerberus-FTP-Server-Changelog-1904.html

*FileCOPA FTP Server http://www.filecopa-ftpserver.com/features.html

*FireFTP http://fireftp.mozdev.org/features.html

*Gene6 FTP Server http://www.g6ftpserver.com/en/information#features

*GoldenGate FTP (Ftp Full Java Server)

*IceWarp FTP Server http://www.icewarp.com/products/ftp_server/

*JAFS http://www.sbbi.net/site/jafs/features.html

*MOVEit DMZ

*Nofeel FTP server http://www.nftpserver.com/history.php

*Null FTP http://www.sharewareconnection.com/null-ftp-client-pro.htm

*ProFTPD module mod_digest http://www.smartftp.com/oss/proftpd/mod_digest.html

*SmartFTP client http://www.smartftp.com/features/

*Starksoft Ftp Component for .NET / Mono http://www.starksoft.com/prod_ftp.html

*RaidenFTPD32 FTP server

*WS_FTP client / server http://ipswitchft.custhelp.com/app/answers/detail/a_id/671/kw/xmd5/r_id/166/sno/1

*wuftpd ('SITE CHECKMETHOD' and 'SITE CHECKSUM')

*zFTPServer

Appendix C. Document History

TOC

[[to be removed by the RFC editor before publication as an RFC.]]
Known issues concerning this draft:

*Local HASH command: "Toggles number sign (#) printing for each data block that is transferred."

*Underspecification of the representation of the file that shall undergo the hash calculation.

*Correct response code to use for completion and errors.

*Need to include some more general advice on algorithms just in case algorithm X is found to be broken the day after this draft is released.

*Possibly we should suggest that servers calculate the hash numbers in advance, like when the file gets uploaded to avoid the risk of this becoming a DOS-vector.

*The FTP server's right to refuse to calculate the hash is of course important to help against DOS risks.

*Partial file hashes.

-01 : April 7, 2010.

*Changing HASH algorithm with OPTS.

*Reference RFC 5797 and add IANA Considerations section.

*Informative Reference to expired Internet Draft (draft-twine-ftpmd5) which attempted to address this issue (it only supported one hash, MD5).

-00 : October 19, 2009.

*Initial draft.

Authors' Addresses

TOC

	Anthony Bryan
	Pompano Beach, FL
	USA
Email:	anthonybryan@gmail.com
URI:	http://www.metalinker.org
	Tim Kosse
Email:	tim.kosse@filezilla-project.org
URI:	http://filezilla-project.org/
	Daniel Stenberg
Email:	daniel@haxx.se
URI:	http://www.haxx.se/