Network Working Group
Request for Comments: 4467
Updates: 3501
Category: Standards Track

M. Crispin
University of Washington

May 2006

Internet Message Access Protocol (IMAP) - URLAUTH Extension

Status of This Memo

Copyright Notice

Abstract

This document describes the URLAUTH extension to the Internet Message Access Protocol (IMAP) (RFC 3501) and the IMAP URL Scheme (IMAPURL) (RFC 2192). This extension provides a means by which an IMAP client can use URLs carrying authorization to access limited message data on the IMAP server.

An IMAP server that supports this extension indicates this with a capability name of "URLAUTH".

1. Introduction

In [IMAPURL], a URL of the form imap://fred@example.com/INBOX/;uid=20 requires authorization as userid "fred". However, [IMAPURL] implies that it only supports authentication and confuses the concepts of authentication and authorization.

The URLAUTH extension defines an authorization mechanism for IMAP URLs to replace [IMAPURL]'s authentication-only mechanism. URLAUTH conveys authorization in the URL string itself and reuses a portion of the syntax of the [IMAPURL] authentication mechanism to convey the authorization identity (which also defines the default namespace in [IMAP]).

The URLAUTH extension provides a means by which an authorized user of an IMAP server can create URLAUTH-authorized IMAP URLs. A URLAUTH-authorized URL conveys authorization (not authentication) to the data

addressed by that URL. This URL can be used in another IMAP session to access specific content on the IMAP server, without otherwise providing authorization to any other data (such as other data in the mailbox specified in the URL) owned by the authorizing user.

Conceptually, a URLAUTH-authorized URL can be thought of as a "pawn ticket" that carries no authentication information and can be redeemed by whomever presents it. However, unlike a pawn ticket, URLAUTH has optional mechanisms to restrict the usage of a URLAUTH-authorized URL. Using these mechanisms, URLAUTH-authorized URLs can be usable by:

- anonymous (the "pawn ticket" model)
- authenticated users only
- a specific authenticated user only
- message submission acting on behalf of a specific user only

There is also a mechanism for expiration.

A URLAUTH-authorized URL can be used in the argument to the BURL command in message composition, as described in [BURL], for such purposes as allowing a client (with limited memory or other resources) to submit a message forward or to resend from an IMAP mailbox without requiring the client to fetch that message data.

The URLAUTH is generated using an authorization mechanism name and an authorization token, which is generated using a secret mailbox access key. An IMAP client can request that the server generate and assign a new mailbox access key (thus effectively revoking all current URLs using URLAUTH with the old mailbox access key) but cannot set the mailbox access key to a key of its own choosing.

1.1. Conventions Used in this Document

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as defined in [KEYWORDS].

The formal syntax uses the Augmented Backus-Naur Form (ABNF) notation including the core rules defined in Appendix A of [ABNF].

In examples, "C:" and "S:" indicate lines sent by the client and server, respectively. If a single "C:" or "S:" label applies to multiple lines, then the line breaks between those lines are for editorial clarity only and are not part of the actual protocol exchange.

2. Concepts

2.1. URLAUTH

The URLAUTH is a component, appended at the end of a URL, that conveys authorization to access the data addressed by that URL. It contains an authorized access identifier, an authorization mechanism name, and an authorization token. The authorization token is generated from the URL, the authorized access identifier, the authorization mechanism name, and a mailbox access key.

2.2. Mailbox Access Key

The mailbox access key is a random string with at least 128 bits of entropy. It is generated by software (not by the human user) and MUST be unpredictable.

Each user has a table of mailboxes and an associated mailbox access key for each mailbox. Consequently, the mailbox access key is per-user and per-mailbox. In other words, two users sharing the same mailbox each have a different mailbox access key for that mailbox, and each mailbox accessed by a single user also has a different mailbox access key.

2.3. Authorized Access Identifier

The authorized access identifier restricts use of the URLAUTH authorized URL to certain users authorized on the server, as described in section 3.

2.4. Authorization Mechanism

The authorization mechanism is the algorithm by which the URLAUTH is generated and subsequently verified, using the mailbox access key.

2.4.1. INTERNAL Authorization Mechanism

This specification defines the INTERNAL mechanism, which uses a token generation algorithm of the server's choosing and does not involve disclosure of the mailbox access key to the client.

> Note: The token generation algorithm chosen by the server implementation should be modern and reasonably secure. At the time of the writing of this document, an [HMAC] such as HMAC-SHA1 is recommended.

If it becomes necessary to change the token generation algorithm of the INTERNAL mechanism (e.g., because an attack against the current algorithm has been discovered), all currently existing URLAUTH-authorized URLs are invalidated by the change in algorithm. Since this would be an unpleasant surprise to applications that depend upon the validity of a URLAUTH-authorized URL, and there is no good way to do a bulk update of existing deployed URLs, it is best to avoid this situation by using a secure algorithm as opposed to one that is "good enough".

Server implementations SHOULD consider the possibility of changing the algorithm. In some cases, it may be desirable to implement the change of algorithm in a way that newly-generated tokens use the new algorithm, but that for a limited period of time tokens using either the new or old algorithm can be validated. Consequently, the server SHOULD incorporate some means of identifying the token generation algorithm within the token.

Although this specification is extensible for other mechanisms, none are defined in this document. In addition to the mechanism name itself, other mechanisms may have mechanism-specific data, which is to be interpreted according to the definition of that mechanism.

2.5. Authorization Token

The authorization token is a deterministic string of at least 128 bits that an entity with knowledge of the secret mailbox access key and URL authorization mechanism can use to verify the URL.

3. IMAP URL Extensions

[IMAPURL] is extended by allowing the addition of ";EXPIRE=<datetime>" and ";URLAUTH=<access>:<mech>:<token>" to IMAP URLs that refer to a specific message or message parts.

The URLAUTH is comprised of ";URLAUTH=<access>:<mech>:<token>" and MUST be at the end of the URL.

URLAUTH does not apply to, and MUST NOT be used with, any IMAP URL that refers to an entire IMAP server, a list of mailboxes, an entire IMAP mailbox, or IMAP search results.

When ";EXPIRE=<datetime>" is used, this indicates the latest date and time that the URL is valid. After that date and time, the URL has expired, and server implementations MUST reject the URL. If ";EXPIRE=<datetime>" is not used, the URL has no expiration, but still can be revoked as discussed below.

The URLAUTH takes the form ";URLAUTH=<access>:<mech>:<token>". It is composed of three parts. The <access> portion provides the authorized access identifiers, which may constrain the operations and users that are permitted to use this URL. The <mech> portion provides the authorization mechanism used by the IMAP server to generate the authorization token that follows. The <token> portion provides the authorization token.

The "submit+" access identifier prefix, followed by a userid, indicates that only a userid authorized as a message submission entity on behalf of the specified userid is permitted to use this URL. The IMAP server does not validate the specified userid but does validate that the IMAP session has an authorization identity that is authorized as a message submission entity. The authorized message submission entity MUST validate the userid prior to contacting the IMAP server.

The "user+" access identifier prefix, followed by a userid, indicates that use of this URL is limited to IMAP sessions that are logged in as the specified userid (that is, have authorization identity as that userid).

> Note: If a SASL mechanism that provides both authorization and authentication identifiers is used to authenticate to the IMAP server, the "user+" access identifier MUST match the authorization identifier.

The "authuser" access identifier indicates that use of this URL is limited to IMAP sessions that are logged in as an authorized user (that is, have authorization identity as an authorized user) of that IMAP server. Use of this URL is prohibited to anonymous IMAP sessions.

The "anonymous" access identifier indicates that use of this URL is not restricted by session authorization identity; that is, any IMAP session in authenticated or selected state (as defined in [IMAP]), including anonymous sessions, may issue a URLFETCH using this URL.

The authorization token is represented as an ASCII-encoded hexadecimal string, which is used to authorize the URL. The length and the calculation of the authorization token depends upon the mechanism used; but, in all cases, the authorization token is at least 128 bits (and therefore at least 32 hexadecimal digits).

4. Discussion of URLAUTH Authorization Issues

In [IMAPURL], the userid before the "@" in the URL has two purposes:

1) It provides context for user-specific mailbox paths such as "INBOX".

2) It specifies that resolution of the URL requires logging in as that user and limits use of that URL to only that user.

An obvious limitation of using the same field for both purposes is that the URL can only be resolved by the mailbox owner.

URLAUTH overrides the second purpose of the userid in the IMAP URL and by default permits the URL to be resolved by any user permitted by the access identifier.

The "user+<userid>" access identifier limits resolution of that URL to a particular userid, whereas the "submit+<userid>" access identifier is more general and simply requires that the session be authorized by a user that has been granted a "submit" role within the authentication system. Use of either of these access identifiers makes it impossible for an attacker, spying on the session, to use the same URL, either directly or by submission to a message submission entity.

The "authuser" and "anonymous" access identifiers do not have this level of protection and should be used with caution. These access identifiers are primarily useful for public export of data from an IMAP server, without requiring that it be copied to a web or anonymous FTP server. Refer to the Security Considerations for more details.

5. Generation of URLAUTH-Authorized URLs

A URLAUTH-authorized URL is generated from an initial URL as follows:

An initial URL is built, ending with ";URLAUTH=<access>" but without the ":<mech>:<token>" components. An authorization mechanism is selected and used to calculate the authorization token, with the initial URL as the data and a secret known to the IMAP server as the key. The URLAUTH-authorized URL is generated by taking the initial URL and appending ":", the URL authorization mechanism name, ":", and the ASCII-encoded hexadecimal representation of the authorization token.

Note: ASCII-encoded hexadecimal is used instead of BASE64 because a BASE64 representation may have "=" padding characters, which would be problematic in a URL.

In the INTERNAL mechanism, the mailbox access key for that mailbox is the secret known to the IMAP server, and a server-selected algorithm is used as described in section 2.4.1.

6. Validation of URLAUTH-authorized URLs

A URLAUTH-authorized URL is validated as follows:

The URL is split at the ":" that separates "<access>" from "<mech>:<token>" in the ";URLAUTH=<access>:<mech>:<token>" portion of the URL. The "<mech>:<token>" portion is first parsed and saved as the authorization mechanism and the authorization token. The URL is truncated, discarding the ":" described above, to create a "rump URL" (the URL minus the ":" and the "<mech>:<token>" portion). The rump URL is then analyzed to identify the mailbox.

If the mailbox cannot be identified, an authorization token is calculated on the rump URL, using random "plausible" keys (selected by the server) as needed, before returning a validation failure. This prevents timing attacks aimed at identifying mailbox names.

If the mailbox can be identified, the authorization token is calculated on the rump URL and a secret known to the IMAP server using the given URL authorization mechanism. Validation is successful if, and only if, the calculated authorization token for that mechanism matches the authorization token supplied in ";URLAUTH=<access>:<mech>:<token>".

Removal of the ":<mech>:<token>" portion of the URL MUST be the only operation applied to the URLAUTH-authorized URL to get the rump URL. In particular, URL percent escape decoding and case-folding (including to the domain part of the URL) MUST NOT occur.

In the INTERNAL mechanism, the mailbox access key for that mailbox is used as the secret known to the IMAP server, and the same server-selected algorithm used for generating URLs is used to calculate the authorization token for verification.

7. Additional Commands

These commands are extensions to the [IMAP] base protocol.

The section headings of these commands are intended to correspond with where they would be located in the base protocol document if they were part of that document.

BASE.6.3.RESETKEY. RESETKEY Command

Arguments: optional mailbox name
optional mechanism name(s)

Responses: none other than in result

Result: OK - RESETKEY completed, URLMECH containing new data
NO - RESETKEY error: can't change key of that mailbox
BAD - command unknown or arguments invalid

The RESETKEY command has two forms.

The first form accepts a mailbox name as an argument and generates a new mailbox access key for the given mailbox in the user's mailbox access key table, replacing any previous mailbox access key (and revoking any URLs that were authorized with a URLAUTH using that key) in that table. By default, the mailbox access key is generated for the INTERNAL mechanism; other mechanisms can be specified with the optional mechanism argument.

The second form, with no arguments, removes all mailbox access keys in the user's mailbox access key table, revoking all URLs currently authorized using URLAUTH by the user.

Any current IMAP session logged in as the user that has the mailbox selected will receive an untagged OK response with the URLMECH status response code (see section BASE.7.1.URLMECH for more details about the URLMECH status response code).

Example:

```
C: a31 RESETKEY
S: a31 OK All keys removed
C: a32 RESETKEY INBOX
S: a32 OK [URLMECH INTERNAL] mechs
C: a33 RESETKEY INBOX XSAMPLE
S: a33 OK [URLMECH INTERNAL XSAMPLE=P34OKhO7VEkCbsiYY8rGEg==] done
```

BASE.6.3.GENURLAUTH. GENURLAUTH Command

Argument: one or more URL/mechanism pairs

Response: untagged response: GENURLAUTH

Result: OK - GENURLAUTH completed
NO - GENURLAUTH error: can't generate a URLAUTH
BAD - command unknown or arguments invalid

The GENURLAUTH command requests that the server generate a URLAUTH-authorized URL for each of the given URLs using the given URL authorization mechanism.

The server MUST validate each supplied URL as follows:

(1) The mailbox component of the URL MUST refer to an existing mailbox.

(2) The server component of the URL MUST contain a valid userid that identifies the owner of the mailbox access key table that will be used to generate the URLAUTH-authorized URL. As a consequence, the iserver rule of [IMAPURL] is modified so that iuserauth is mandatory.

Note: the server component of the URL is generally the logged in userid and server. If not, then the logged in userid and server MUST have owner-type access to the mailbox access key table owned by the userid and server indicated by the server component of the URL.

(3) There is a valid access identifier that, in the case of "submit+" and "user+", will contain a valid userid. This userid is not necessarily the same as the owner userid described in (2).

(4) The server MAY also verify that the iuid and/or isection components (if present) are valid.

If any of the above checks fail, the server MUST return a tagged BAD response with the following exception. If an invalid userid is supplied as the mailbox access key owner and/or as part of the access identifier, the server MAY issue a tagged OK response with a generated mailbox key that always fails validation when used with a URLFETCH command. This exception prevents an attacker from validating userids.

If there is currently no mailbox access key for the given mailbox in the owner's mailbox access key table, one is automatically generated. That is, it is not necessary to use RESETKEY prior to first-time use of GENURLAUTH.

If the command is successful, a GENURLAUTH response code is returned listing the requested URLs as URLAUTH-authorized URLs.

Examples:

```
C: a775 GENURLAUTH "imap://joe@example.com/INBOX/;uid=20/
   ;section=1.2" INTERNAL
S: a775 BAD missing access identifier in supplied URL
C: a776 GENURLAUTH "imap://example.com/Shared/;uid=20/
   ;section=1.2;urlauth=submit+fred" INTERNAL
S: a776 BAD missing owner username in supplied URL
C: a777 GENURLAUTH "imap://joe@example.com/INBOX/;uid=20/
   ;section=1.2;urlauth=submit+fred" INTERNAL
S: * GENURLAUTH "imap://joe@example.com/INBOX/;uid=20/;section=1.2
   ;urlauth=submit+fred:internal:91354a473744909de610943775f92038"
S: a777 OK GENURLAUTH completed
```

BASE.6.3.URLFETCH. URLFETCH Command

```
Argument:   one or more URLs

Response:   untagged response: URLFETCH

Result:     OK - urlfetch completed
            NO - urlfetch failed due to server internal error
            BAD - command unknown or arguments invalid
```

The URLFETCH command requests that the server return the text data associated with the specified IMAP URLs, as described in [IMAPURL] and extended by this document. The data is returned for all validated URLs, regardless of whether or not the session would otherwise be able to access the mailbox containing that data via SELECT or EXAMINE.

Note: This command does not require that the URL refer to the selected mailbox; nor does it require that any mailbox be selected. It also does not in any way interfere with any selected mailbox.

The URLFETCH command effectively executes with the access of the userid in the server component of the URL (which is generally the userid that issued the GENURLAUTH). By itself, the URLAUTH does NOT grant access to the data; once validated, it grants whatever access to the data is held by the userid in the server component of the URL. That access may have changed since the GENURLAUTH was done.

The URLFETCH command MUST return an untagged URLFETCH response and a tagged OK response to any URLFETCH command that is syntactically valid. A NO response indicates a server internal failure that may be resolved on later retry.

> Note: The possibility of a NO response is to accommodate implementations that would otherwise have to issue an untagged BYE with a fatal error due to an inability to respond to a valid request. In an ideal world, a server SHOULD NOT issue a NO response.

The server MUST return NIL for any IMAP URL that references an entire IMAP server, a list of mailboxes, an entire IMAP mailbox, or IMAP search results.

Example:

> Note: For clarity, this example uses the LOGIN command, which SHOULD NOT be used over a non-encrypted communication path.
>
> This example is of a submit server, obtaining a message segment for a message that it has already validated was submitted by "fred".

```
S: * OK [CAPABILITY IMAP4REV1 URLAUTH] example.com IMAP server
C: a001 LOGIN submitserver secret
S: a001 OK submitserver logged in
C: a002 URLFETCH "imap://joe@example.com/INBOX/;uid=20/
   ;section=1.2;urlauth=submit+fred:internal
   :91354a473744909de610943775f92038"
S: * URLFETCH "imap://joe@example.com/INBOX/;uid=20/;section=1.2
   ;urlauth=submit+fred:internal
   :91354a473744909de610943775f92038" {28}
S: Si vis pacem, para bellum.
S:
S: a002 OK URLFETCH completed
```

8. Additional Responses

These responses are extensions to the [IMAP] base protocol.

The section headings of these responses are intended to correspond with where they would be located in the base protocol document if they were part of that document.

BASE.7.1.URLMECH. URLMECH Status Response Code

The URLMECH status response code is followed by a list of URL authorization mechanism names. Mechanism names other than INTERNAL may be appended with an "=" and BASE64-encoded form of mechanism-specific data.

This status response code is returned in an untagged OK response in response to a RESETKEY, SELECT, or EXAMINE command. In the case of the RESETKEY command, this status response code can be sent in the tagged OK response instead of requiring a separate untagged OK response.

Example:

```
C: a33 RESETKEY INBOX XSAMPLE
S: a33 OK [URLMECH INTERNAL XSAMPLE=P34OKhO7VEkCbsiYY8rGEg==] done
```

In this example, the server supports the INTERNAL mechanism and an experimental mechanism called XSAMPLE, which also holds some mechanism-specific data (the name "XSAMPLE" is for illustrative purposes only).

BASE.7.4.GENURLAUTH. GENURLAUTH Response

Contents: One or more URLs

The GENURLAUTH response returns the URLAUTH-authorized URL(s) requested by a GENURLAUTH command.

Example:

```
C: a777 GENURLAUTH "imap://joe@example.com/INBOX/;uid=20/
   ;section=1.2;urlauth=submit+fred" INTERNAL
S: * GENURLAUTH "imap://joe@example.com/INBOX/;uid=20/;section=1.2
   ;urlauth=submit+fred:internal:91354a473744909de610943775f92038"
S: a777 OK GENURLAUTH completed
```

BASE.7.4.URLFETCH. URLFETCH Response

Contents: One or more URL/nstring pairs

The URLFETCH response returns the message text data associated with one or more IMAP URLs, as described in [IMAPURL] and extended by this document. This response occurs as the result of a URLFETCH command.

The returned data string is NIL if the URL is invalid for any reason (including validation failure). If the URL is valid, but the IMAP fetch of the body part returned NIL (this should not happen), the returned data string should be the empty string ("") and not NIL.

Note: This command does not require that the URL refer to the selected mailbox; nor does it require that any mailbox be selected. It also does not in any way interfere with any selected mailbox.

Example:

```
C: a002 URLFETCH "imap://joe@example.com/INBOX/;uid=20/
   ;section=1.2;urlauth=submit+fred:internal
   :91354a473744909de610943775f92038"
S: * URLFETCH "imap://joe@example.com/INBOX/;uid=20/;section=1.2
   ;urlauth=submit+fred:internal
   :91354a473744909de610943775f92038" {28}
S: Si vis pacem, para bellum.
S:
S: a002 OK URLFETCH completed
```

9. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [ABNF].

The following modifications are made to the Formal Syntax in [IMAP]:

```
resetkey        = "RESETKEY" [SP mailbox *(SP mechanism)]

capability      =/ "URLAUTH"

command-auth    =/ resetkey / genurlauth / urlfetch

resp-text-code  =/ "URLMECH" SP "INTERNAL" *(SP mechanism ["=" base64])

genurlauth      = "GENURLAUTH" 1*(SP url-rump SP mechanism)

genurlauth-data = "*" SP "GENURLAUTH" 1*(SP url-full)
```

```
url-full        = astring
                    ; contains authimapurlfull as defined below

url-rump        = astring
                    ; contains authimapurlrump as defined below

urlfetch        = "URLFETCH" 1*(SP url-full)

urlfetch-data   = "*" SP "URLFETCH" 1*(SP url-full SP nstring)
```

The following extensions are made to the Formal Syntax in [IMAPURL]:

```
authimapurl     = "imap://" enc-user [iauth] "@" hostport "/"
                    imessagepart
                    ; replaces "imapurl" and "iserver" rules for
                    ; URLAUTH authorized URLs

authimapurlfull = authimapurl iurlauth

authimapurlrump = authimapurl iurlauth-rump

enc-urlauth     = 32*HEXDIG

enc-user        = 1*achar
                    ; same as "enc_user" in RFC 2192

iurlauth        = iurlauth-rump ":" mechanism ":" enc-urlauth

iurlauth-rump   = [expire] ";URLAUTH=" access

access          = ("submit+" enc-user) / ("user+" enc-user) /
                  "authuser" / "anonymous"

expire          = ";EXPIRE=" date-time
                     ; date-time defined in [DATETIME]

mechanism       = "INTERNAL" / 1*(ALPHA / DIGIT / "-" / ".")
                    ; case-insensitive
                    ; new mechanisms MUST be registered with IANA
```

10. Security Considerations

Security considerations are discussed throughout this memo.

The mailbox access key SHOULD have at least 128 bits of entropy (refer to [RANDOM] for more details) and MUST be unpredictable.

The server implementation of the INTERNAL mechanism SHOULD consider the possibility of needing to change the token generation algorithm, and SHOULD incorporate some means of identifying the token generation algorithm within the token.

The URLMECH status response code may expose sensitive data in the mechanism-specific data for mechanisms other than INTERNAL. A server implementation MUST implement a configuration that will not return a URLMECH status response code unless some mechanism is provided that protects the session from snooping, such as a TLS or SASL security layer that provides confidentiality protection.

The calculation of an authorization token with a "plausible" key if the mailbox can not be identified is necessary to avoid attacks in which the server is probed to see if a particular mailbox exists on the server by measuring the amount of time taken to reject a known bad name versus some other name.

To protect against a computational denial-of-service attack, a server MAY impose progressively longer delays on multiple URL requests that fail validation.

The decision to use the "authuser" access identifier should be made with caution. An "authuser" access identifier can be used by any authorized user of the IMAP server; therefore, use of this access identifier should be limited to content that may be disclosed to any authorized user of the IMAP server.

The decision to use the "anonymous" access identifier should be made with extreme caution. An "anonymous" access identifier can be used by anyone; therefore, use of this access identifier should be limited to content that may be disclosed to anyone. Many IMAP servers do not permit anonymous access; in this case, the "anonymous" access identifier is equivalent to "authuser", but this MUST NOT be relied upon.

Although this specification does not prohibit the theoretical capability to generate a URL with a server component other than the logged in userid and server, this capability should only be provided

when the logged in userid/server has been authorized as equivalent to the server component userid/server, or otherwise has access to that userid/server mailbox access key table.

11. IANA Considerations

This document constitutes registration of the URLAUTH capability in the imap4-capabilities registry.

URLAUTH authorization mechanisms are registered by publishing a standards track or IESG-approved experimental RFC. The registry is currently located at:

http://www.iana.org/assignments/urlauth-authorization-mechanism-registry

This registry is case-insensitive.

This document constitutes registration of the INTERNAL URLAUTH authorization mechanism.

IMAP URLAUTH Authorization Mechanism Registry

Mechanism Name	Reference
INTERNAL	[RFC4467]

12. Normative References

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[BURL] Newman, C., "Message Submission BURL Extension", RFC 4468, May 2006.

[DATETIME] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[IMAP] Crispin, M., "Internet Message Access Protocol - Version 4rev1", RFC 3501, March 2003.

[IMAPURL] Newman, C., "IMAP URL Scheme", RFC 2192, September 1997.

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

13. Informative References

[HMAC] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RANDOM] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

Author’s Address

Mark R. Crispin
Networks and Distributed Computing
University of Washington
4545 15th Avenue NE
Seattle, WA 98105-4527

Phone: (206) 543-5762
EMail: MRC@CAC.Washington.EDU

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).