3GPP TSG SA WG3 Security — S3#22 **S3-020067**

25 - 28 February 2002

Bristol, UK

Source: **Nortel Networks**

Title: **SIP Message Integrity Protection Work in the IETF**

Document for: **Discussion**

Agenda Item: ***TBD*, IM Subsystem Access Security**

It was agreed in SA3 meeting S3#19bis to progress a SIP-level solution mechanism for (SIP) message integrity protection in the IMS for the UE-to-Proxy CSCF hop. In meeting S3#20, SA3 agreed to include text in draft TS 33.203 that outlines such a mechanism. The mechanism selected to be progressed is the HTTP Digest security framework with extensions. Extensions that will permit complete bi-directional message integrity protection between a User Agent (e.g., an IMS UE) and an Outbound Proxy (e.g., P-CSCF) must be agreed within the IETF. To this end, an Internet Draft entitled "SIP Digest Authentication: Extensions to HTTP Digest Authentication" (draft-undery-sip-auth-00.txt) has been written for submission to IETF.

The key objectives of this draft are to: (a) make possible more complete integrity protection in 'one-hop' environments (e.g., User Agent to Outbound Proxy); and (b) enable last-hop (proxy to User Agent Server) authentication to work properly, as originally scoped in Digest.

It is believed that the proposed extensions will permit Digest to satisfy the Release 5 requirements for IMS message integrity protection. SA3 feedback and analysis on this issue is welcome.

The attached draft will be presented in the next IETF meeting (IETF #53 - Minneapolis, USA March 17-22, 2002). It is submitted to this SA3 meeting for discussion.

Internet Engineering Task Force
Internet Draft
Document: draft-undery-sip-auth-00.txt
Category: Standards Track

J. Undery
Ubiquity

Sanjoy Sen
Nortel Networks

Vesa Torvinen
Ericsson

January 2002

SIP Digest Authentication: Extensions to HTTP Digest Authentication

Status of this Memo

1. Abstract

The Session Initiation Protocol [2] currently uses the HTTP digest authentication algorithm [3] for simple authentication of SIP requests. This scheme does not allow for the inclusion of headers important to the integrity of a SIP request including To, From, Call-ID, CSeq, Contact and Expires headers. This document attempts to address this issue.

Examples of possible attacks involve capturing authenticated messages on the wire, modifying headers and resending the message. This makes it possible to modify registration details and initiate sessions with entities requiring authentication.

HTTP Digest [3] has also other shortcomings if applied for SIP. Firstly, SIP UA has serious difficulties to distinguish the source of Authentication-Info headers in SIP forking situations. Secondly,

HTTP Digest [3] cannot be utilized for proxy to UAS authentication because the existing headers do not provide the target of the challange. This document introduces extensions to HTTP Digest to solve these problems.

2. Introduction

RFC 2617 [3] defines two methods for a HTTP client to authenticate itself to a server, Basic and Digest. Digest authentication as used by SIP uses a cryptographic hash of a number of elements including the request method, URI and optionally the body of the message.

Unfortunately Digest authentication fails to provide authentication of a number of headers critical to the integrity of a SIP request. This enables a number of attacks against servers, by attackers pretending to be the client and modifying authenticated messages.

There are also other known problems related to the use of HTTP authentication headers in SIP environment. These problems are related to the lack of source and target parameters in different authentication headers.

To fix these problems it is necessary to extend the digest authentication scheme. This document attempts to 'patch' digest authentication in RFC 2617 to provide a better solution for SIP. Existing headers are supplemented with new parameters and parameter values. Defining new authentication headers will enable proxy-to-UAS authentication and enable the UAS to target specific proxies. Existing headers could not be re-used because of backward compatibility and efficiency reasons.

3. Perceived Threats

In this section we will describe the threats to security that we attempt to address and those that will be ignored. This section will not discuss the complexity of exploiting these threats because according to [4], "A threat is, by definition, a vulnerability available to a motivated and capable adversary". The fact it exists is enough.

3.1. Addressed Threats

Replay attacks: this is really an issue of how the server creates and expires its nonces, section 4.2.1 describes a mechanism that can be used to help by including a timestamp in the nonces, that combined with the protecting headers can prevent replay attacks.

Man in the Middle: attacks, this is where messages are altered by the attacker. One of the objectives of this draft is to allow a server to decide which headers it requires to be included in the credentials. Any header not chosen by the server will be vulnerable,

and known to a potential attacker. It should be impossible for an attacker to alter any of the headers the server deems relevant to protect. The weakness here is that a server may decide the Subject header is irrelevant, yet the recipient of the message might not appreciate an INVITE with an offensive subject injected by an intermediate proxy.

One important type of man in the middle attack is a bidding down attack. This is where the attacker removes stronger schemes, algorithms or quality of protection from a challenge. This attack then allows the attacker an increased ability to interfere with the session.

Masquerade attacks: this is where an attacker inserts itself into the signaling path and attempts to fool the client into providing credentials the attacker can use to create a false message.

The last two issues are addressed by both the client and server using the same set of headers for inclusion in the credentials and checking them. Any change to these headers will result in the credentials being invalid for the request.

Reliability of results is covered by the inclusion of the Authentication-Info header, which can provide authentication in the response to any authenticated request.

3.2. Unaddressed Threats

Privacy will be totally out of scope; both data and system usage are unprotected by RFC 2617 and will be ignored by this draft too. In order to protect privacy, encryption is required.

Denial of service this is also out of scope. Authentication inherently requires some level of additional work on behalf of the server and client. This additional load makes it easier to overwhelm the resources of the victim. That said stateless rejection of unauthenticated messages help prevent state loading denial of service attacks.

4. Syntax

RFC 2617 describes the Digest authentication scheme. This scheme is subject to the operation and limitations as described in RFC 2617. Namely, it relies on a shared secret between the client and server and provides no mechanism for distributing those secrets; it provides no ’hiding’ of the payload or headers. The change is purely to provide the same semantic integrity to SIP, as provided to HTTP by including the Method and URI in the credentials.

4.1. Specification of Generic Authentication Headers

4.1.1. WWW-Authenticate and Proxy-Authenticate Response Headers

These headers are defined in [3].

4.1.2. Authorization and Proxy-Authorization Request Headers

These headers are defined in [3].

4.1.3. Authentication-Info Response Header

UAS insert this header in responses to successfully authenticated requests in order to provide mutual authentication and the nonce to use in future requests in this dialog.

```
AuthenticationInfo = "Authentication-Info" HCOLON auth-info
auth-info          = scheme info-credentials
```

4.1.4. Proxy-Authentication-Info Response Header

Proxies place this header in the response in order to authenticate themselves with the UAC and the nonce to use in future requests in this dialogs.

```
ProxyAuthenticationInfo = "Proxy-Authentication-Info" HCOLON
                          proxy-auth-info
proxy-auth-info         = scheme info-credentials
```

4.1.5. UAS-Authenticate Header

The use of this header is described in Section 7.

```
UAS-Authenticate   = "UAS-Authenticate" HCOLON 1#uas-challenge
uas-challenge      = scheme target-param challenge
target-param       = target-realm-param | target-route-param
target-realm-param = "target" EQUAL target
target             = host
target-route-param = "route" EQUAL target-route
target-route       = Request-URI
```

target
  The target is a hostname or IP address indicating the domain or machine the authentication is targeted for.

target-route
  The target-route is uri indicating that any entity sending a request with this uri as the Request-URI is targeted for authentication.

4.1.6. UAS-Authorization Request Header

```
UAS-Authorization = "UAS-Authorization" HCOLON 1#uas-credentials
uas-credentials   = scheme target-ids credentials
target-ids        = target-param responder-param
responder-param   = "responder" EQUAL responder
responder         = sent-by
```

responder
The responder is hostname or network address optionally the port that matches the value which the proxy inserted in the Via.

4.1.7. UAS-Authentication-Info

This header is used by the UAS to authenticate itself to the proxy in the response. A response can contain multiple UAS-Authentication-Info headers targeted towards multiple proxies that have successfully authenticated themselves with the UAS.

```
UASAuthenticationInfo   = "UAS-Authentication-Info" HCOLON
                          1#uas-reverse-credentials
uas-reverse-credentials = scheme target-param info-credentials
```

4.2. New Response Codes

4.2.1. 492 Proxies Unauthorized

The request requires proxies to authenticate with the user agent server. This response is issued by user agent servers and registrars.

4.3. Specification of SIP Digest Headers

4.3.1. Challenge Construction

The following BNF is used in the same context as RFC 2617

```
challenge        = "Digest" digest-challenge

digest-challenge = 1#(realm | domain | nonce | opaque | stale |
                   algorithm | qop-options | auth-params |
                   header-options | generic-param)
realm            = "realm" EQUAL realm-value
realm-value      = quoted-string          ; See Appendix C of [2]
domain           = "domain" EQUAL domain-value
domain-value     = LDQUOT SIP-URL RDQUOT
nonce            = "nonce" EQUAL nonce-value
```

```
nonce-value      = quoted-string
opaque           = "opaque" EQUAL quoted-string
stale            = "stale" EQUAL ("true" | "false")
algorithm        = "algorithm" EQUAL ("MD5" | token )
qop-options      = "qop" EQUAL LDQUOT 1#qop-value RDQUOT
qop-value        = "auth" | "auth-int" | "auth-extd-int" |
                   "auth-hdr-int" | token
auth-params      = token EQUAL quoted-string
header-options   = "header" EQUAL LDQUOT 1#header-value RDQUOT
header-value     = "To" | "From" | "Contact" | "Expires"
                   | "CSeq" | "CallID" | extension-header
extension-header = token
generic-param    = token EQUAL (token | quoted-string)
```

realm
A string that can be rendered for the end user to provide the context with which to authenticate itself.

domain
A quoted SIP URL that corresponds to the request uri of the request as it arrives at the server. For registration in the domain example.com this would be sip:example.com.

nonce
A server-specified data string ideally uniquely generated for each 401/407/492 response. Care should be taken generating this value to ensure it is unique and other parties should not be able to predict its value. (See Section 6.)

opaque
A string of data specified by the server, which should be returned by the client unchanged in any credentials generated in response to this challenge.

stale
A flag indicating if the previous request from the client was rejected because the nonce was stale and the client should retry with the new nonce. If this value is anything other than "true", or is not present, the username and/or password are invalid.

algorithm
A string containing the hashing algorithm used for the authentication. Currently this is only MD5 and should be assumed to be so by default. [5]

qop-options
A quoted string containing the "quality of protection" options supported by the server. The value "auth" indicates authentication; the value "auth-int" indicates authentication with message body integrity protection; the value "auth-extd-int" indicates authentication with complete message integrity protection; the value "auth-hdr-int" indicates authentication and integrity protection of the message body and various headers.

auth-params
This is included for future extensions unknown values should be ignored.

header-options
This directive specifies which header the server requires the client to include in its credentials. It is a quoted comma separated list of headers. Unknown values mentioned should be assumed to be the names of headers in a SIP request that will be included in the credentials the same way as known values, using the capitalisation in the header-value. This directive MUST be present if qop-options contains the value "auth-hdr-int".

generic-param
This is included to specifically allow extensibility, unknown parameters SHOULD be ignored.

4.3.2. Credentials Construction

```
credentials       = "Digest" digest-response
digest-response   = 1# ( username | realm | nonce | digest-uri |
                    response | algorithm | cnonce | opaque |
                    message-qop | nonce-count | auth-param |
                    header-options | generic-param)
username          = "username" EQUAL username-value
username-value    = quoted-string
digest-uri        = "uri" EQUAL DQUOT SIP-URL DQUOT
response          = "response" EQUAL request-digest
request-digest    = LDQUOT 32LHEX RDQUOT
cnonce            = "cnonce" EQUAL cnonce-value
cnonce-value      = quoted-string
message-qop       = "qop" EQUAL qop-value
nonce-count       = "nc" EQUAL nc-value
nc-value          = 8LHEX
```

username
The user’s name in the realm in which it is trying to authenticate itself.

digest-uri
This is the Request-URI the client is using to send the request. This may have changed in transit so is included here.

sip-response
This is a string of 32 hex digits computed as defined below, which proves the user knows a password and the headers covered in header-options have not been altered.

cnonce
This is a compulsory value returned by the client to the server for prevention of chosen plaintext attacks.

message-qop
This value contains the qop value chosen from the list in the challenge that was used to create the credentials.

nonce-count
The nc-value is a hexadecimal count of the number of requests that the client has sent using this nonce.

header-options
This is a quoted comma separated list of the headers that the client encoded into the header-list used to calculate S2. Its value MUST be a superset of the values in the WWW-Authenticate header. i.e. The client can add but not remove headers from the list, this can be used by the client to negotiate up the protection it provides to the integrity of messages.

4.3.3. Info Credentials construction

The implications of changing the nonce do not generally apply to SIP and one-time nonces are useful, without hindering performance. However, there are some environments in which more static security contexts are needed, and consequently the implications of changing the nonce should be considered.

```
info-credentials = 1#(nextnonce | message-qop |
                   response-auth | cnonce | nonce-count |
                   realm | [opaque] | [generic-param])
nextnonce        = "nextnonce" EQUAL nonce-value
response-auth    = "rspauth" EQUAL response-digest
response-digest  = LDQUOT *LHEX RDQUOT
```

nextnonce
This is the nonce that should be used to generate the next credentials corresponding to this info-credentials.

message-qop
Indicates the "quality of protection" options applied to the response by the server. The values here correspond directly to their equivalents in credentials. The server SHOULD use the same value for the message-qop directive in the response as was sent in the credentials of the corresponding request.

response-auth
This value computed as defined below and provides authentication in the responses.

cnonce
This value is copied from the credentials into the corresponding info-credentials.

opaque
This value if present MUST be copied into any credentials using the nextnonce value specified in these info-credentials.

5. Digest Calculation

The method of calculating the request-digest and response-digest are as follows, if the qop-value is present

```
request-digest = LDQUOT < KD( H(A1), unq(nonce-value)
                                     ":" nc-value
                                     ":" unq(cnonce-value)
                                     ":" unq(qop-value)
                                     ":" H(A2)
                              ) RDQUOT
```

is used, otherwise,

```
request-digest = LDQUOT < KD( H(A1), unq(nonce-value) ":" H(A2)) >
                 RDQUOT
```

```
response-digest = request-digest
```

is used where KD, H, A1 and unq are as defined in [3].

5.1. Computing A2

If the qop-value is unspecified, "auth" or "auth-int" the value from [3] is used. If the qop-value is "auth-extd-int", then A2 is:

```
A2 = H(<entire message excluding the credential>)
```

If the qop-value is "auth-hdr-int", then A2 is:

```
A2            = (Method | Status-Code)":" digest-uri-value
                ":" entity-body ":" header-list

header-list   = *headers
headers       = (to-addr | from-addr | contact-value
                | cseq-value | callid-value |expires-value
                | other-value)
to-addr       = To eol-marker
from-addr     = From eol-marker
contact-value = Contact eol-marker
cseq-value    = CSeq eol-marker
callid-value  = Call-ID eol-marker
expires-value = Expires eol-marker
other-value   = field-name ":" [field-value] eol-marker
eol-marker    = <CR> <LF>
```

headers
These are the relevant headers converted to canonical form [2]

with the addition that Contact headers always include the < and > delimiters even if the display name is empty. i.e.

1 No short form header fields.
2 Header field names are capitalised as shown in [2].
3 No white space between the header name and the colon.
4 A single white space after the colon.
5 Line termination with a CRLF.
6 No line folding.
7 No comma separated lists of header values, each must appear as a separate header.
8 Only a single white space between tokens, between tokens and quoted strings, and between quoted strings. No space after the last token.
9 No LWS between tokens and separators, except as described above.
10 The To, From and Contact header fields always include < and > delimiters even if the display-name is empty.

The other-value is left in for the inclusion of headers the author omitted (only long form headers should be used). It should also be noted Authentication-Info, Authorization, Max-Forwards, Proxy-Authorization, Proxy-Authentication-Info, Record-Route, Route, UAS-Authorization and Via headers MUST NOT appear in this list, as they are altered or added/removed in the course of normal signaling.

6. Nonce Generation

Traditionally nonces have contained no meaning for the client, however, in order to prevent bid-down attacks this draft will recommend a format. This format is designed to allow a server to encode the type of protection required. This means

6.1. Syntax

The following definition will replace nonce-value

```
nonce-value      = LDQUOT "(" 1#auth-type ")"
                   trad-nonce-value RDQUOT
auth-type        = auth-algorithms | digest-auth-type | token
auth-algorithms  = "MD5" | "AKA" "SHA1"
digest-auth-type = "dauth" | "dauth-int" | "dauth-extd-int" |
                   "dauth-hdr-int"
trad-nonce-value = *(qdtext |quoted-pair)
```

auth-algorithm
These are the algorithms used by the Digest scheme to produce the digest.

digest-auth-type
These are the protection encodings for Digest authentication. Each value corresponds to a qop-value prefixed with the letter d.

A possible implementation of trad-nonce-value is;

```
trad-nonce-value = time-stamp "-" H(time-stamp ":" request-uri ":"
                   private-key)
```

where, the time-stamp is a non repeating value or time stamp, the request-uri is the Request URI from the request and the private-key is to ensure the nonce was generated by an entity that knows the private-key.

7. Authenticating Proxies with the UAS

7.1. User Agent Server Behaviour

When a UAS receives a request via a number of proxies, the UAS MAY authenticate some of the proxies before the request is processed. If no matching credentials (in the UAS-Authorization header field) are provided in the request, the UAS can challenge the proxies to provide credentials by rejecting the request with a 492 (Proxies Unauthorized) status code containing one or more UAS-Authenticate headers.

The UAS-Authenticate response-header field MUST be included in 492 (Proxies Unauthorized) response messages. If one or more proxies fail to authenticate themselves in a request containing relevant UAS-Authenticate headers the UAS MUST respond with 403 Forbidden.

Once the UAC is successfully authenticated, the UAS can do mutual authentication using the Authentication-Info header in the response. Similarly, once the proxies are successfully authenticated, the UAS can do mutual authentication using UAS-Authentication-Info header in the response.

7.2. User Agent Client Behaviour

When a UAC receives an unauthorised response (i.e. 401, 407 or 492) containing UAS-Authenticate headers it MUST, if it is able, re-originate the request with copies of the UAS-Authenticate headers.

If a UAC receives a response containing UAS-Authenticate headers within a dialog it MUST, if it is able, include a copy of the UAS-Authenticate headers within the next request it sends within that dialog.

7.3 Proxy Behaviour

When a proxy receives a request containing a UAS-Authenticate header it SHOULD check the headers to see if it matches. Matches are possible in two ways; firstly the proxy may be responsible for a domain mentioned within a "target" parameter; secondly it may be proxying the request with a Request-URI that matches a "route" parameter. Proxies MUST, if able to do so, include a UAS-Authorization header with credentials for all matching UAS-Authenticate headers.

If a stateful proxy receives a 492 and determines that it contains a single UAS-Authenticate header targeted solely at itself, it MAY resubmit the request to the UAS with a UAS-Authorization header containing the credential as a separate branch.

After the UAC is authenticated by the proxy, the proxy may use the Proxy-Authentication-Info header in the response to perform mutual authentication with the UAC.

8. Examples

8.1. UAC to UAS mutual authentication

The following example shows the how a client should respond to a request to authenticate its REGISTER request.

```
SIP/2.0 401 Unauthorised
WWW-Authenticate: Digest realm="Vacuity Proxy",
  nonce="(dauth-hdr-int)9912345-0123456789abcdef0123456789abcdef",
  algorithm=MD5, qop="auth-hdr-int"
  header="To, From, Expires, Made-Up-Header"
...
```

The UA then decides that for REGISTER requests it should protect the Contact header so adds it to the list of header-options. It is

assumed the UA will prompt the client for the username and password for the realm "Vacuity Proxy" before responding with,

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 10.0.0.10
Authorization: Digest username="auser", realm="Vacuity Proxy",
  nonce="(dauth-hdr-int)9912345-0123456789abcdef0123456789abcdef",
  uri="sip:example.com", response=fedcba9876543210fedcba9876543210,
  algorithm=MD5, cnonce="9912390-12345678123456781234567812345678",
  qop=auth-hdr-int, nc=00000001,
  header="To, From, Expires, Contact, Made-Up-Header"
To: A User <sip:auser@example.com>
From: A User <sip:auser@example.com;user=ip>;tag=1234
Call-ID: 12321@a.example.com
CSeq: 2 REGISTER
Contact: sip:10.0.0.10:5061, sip:10.0.0.37
m: sip:auser@home.example.com;example=madeup
```

In the above example the header-list used will be

```
"To: A User <sip:auser@example.com>
From: A User <sip:auser@example.com;user=ip>;tag=1234
Expires:
Contact: <sip:10.0.0.10:5061>
Contact: <sip:10.0.0.37>
Contact: <sip:auser@home.example.com>;example=madeup
Made-Up-Header:
"
```

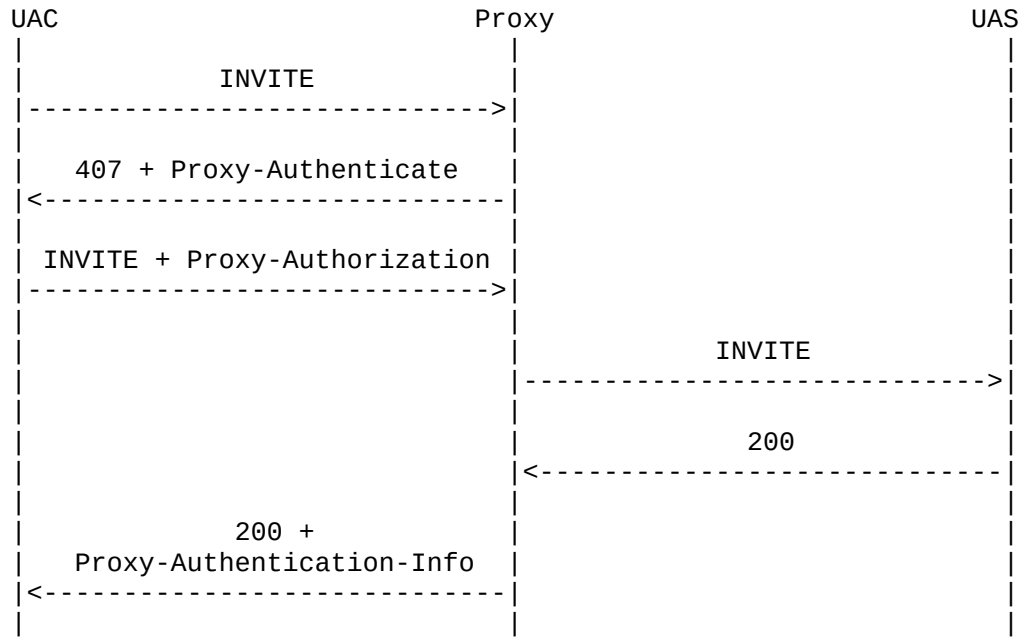
The headers inserted in the same order as the appear in the header-option, empty or missing headers are represented as empty headers, as shown by Expires in this example all lines are terminated with a carriage return line feed. An empty header consists of the long header name followed by a colon and a CRLF pair, without any space. Note that the Contact values appear in the list in the same order as they appear in the message, and that the client added them to the list of headers to protect. Also noteworthy is the positioning of the Authorization header before all the headers we are encoding.

In the 200 response to the REGISTER the registrar includes.

```
Authentication-Info: nextnonce="(dauth-int-hdr)12345-q1w2e3r4",
  qop=auth-hdr-int, rspauth="deadbeefdeadbeefdeadbeefdeadbeef",
  nc=00000001
```

This header provides the next nonce for the UAC to use and authenticates the response. It may be worth noting that once a user has been authenticated it is possible to provide nonces statefully although the nonce best practice rules should still be applied.

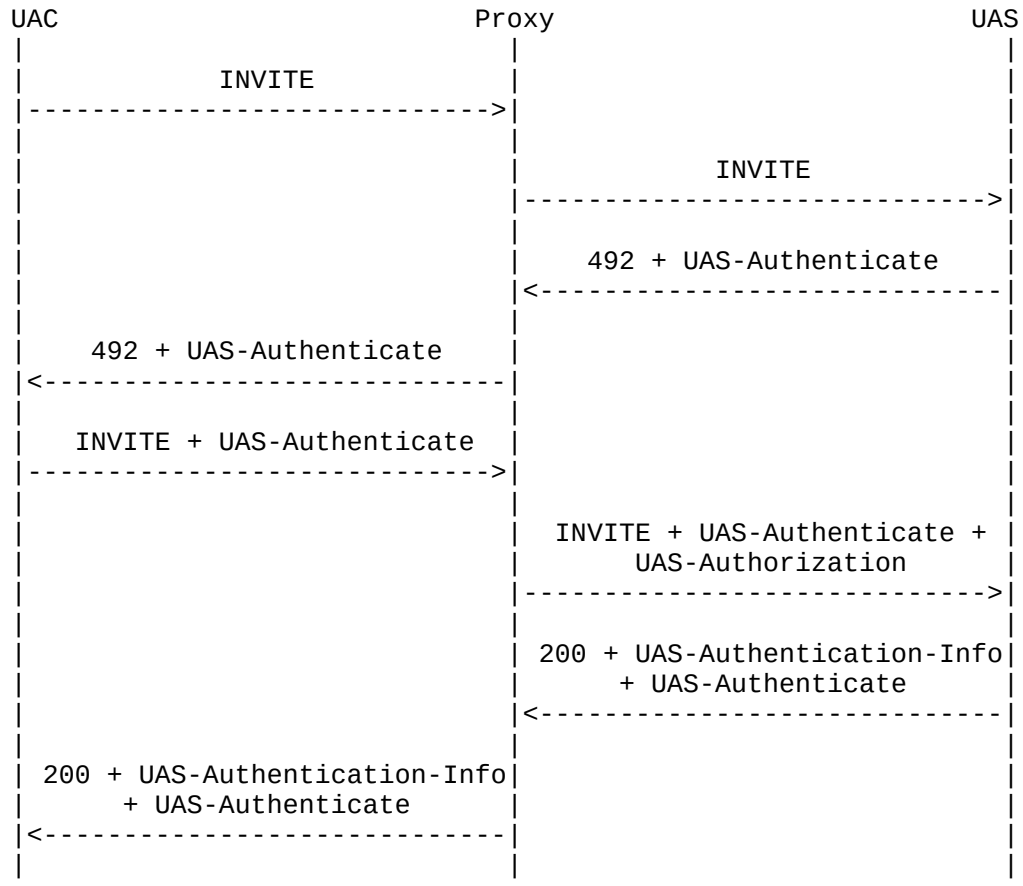
8.2. Proxy to UAC mutual authentication



The above diagram shows shows a call flow where the proxy challenges a caller and then authenticates itself in the response to the authenticated request. In this instance as a single hop is involved the qop value of auth-extd-int used so the entire message was integrity protected.

8.3. Proxy to UAS mutual authentication

The following diagram shows how the UAS-Authenticate header is used to provide proxy to UAS authentication. When the UAS receives the request it returns a 492 which is propagated back to the UAC. If the UAC is aware of the 492 response it then copies the UAC-Authenticate headers into the resubmitted request. When the request with the UAS-Authenticate headers arrive at the proxy it adds UAS-Authorization headers for all the challenges targeted at it. When the UAS receives request it ensures it has received all the UAS-Authorization headers it was expecting. (Note a UAS-Authenticate header may produce more than one UAS-Authorization header if more than one proxy match.) The UAS then populates UAS-Authentication-Info headers for all the proxies it wishes to mutually authenticate with. Proxies can the check for UAS-Authentication-Info headers in the response.



9. Security Considerations

The purpose of this draft is security. Items ruled out side of the scope of this document are privacy, resistance to denial of service, trustworthiness of results. The rationale for these is included in section 4

The security of this draft relies mainly on the headers chosen by the server for inclusion in the digest. If this selection is poor a false sense of security obtained; although if a client wishes the set can be increased to cover all the relevant headers.

9.1. Security Considerations Missing From RFC 2617

RFC 2617 [3] has a remarkably thorough security considerations section, however, in the author’s opinion an important consideration is missed. In the WWW-Authenticate header the qop directive can contain a list of schemes supported. It is possible for an attacker to downgrade the security on offer by removing auth-int if present so the body of the message isn’t included in the protection, or simply remove the qop parameter entirely.

10. References

1 Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

2 Handley, M., Schulzrinne, H, Schooler, E. and Rosenberg, J., "SIP: Session Initiation Protocol", work in progress, currently, draft-ietf-sip-rfc2543bis-03.txt, May 2001.

3 Franks, J. et al, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

4 Bellovin, S., "Report of the IAB Security Architecture Workshop", RFC 2316, April 1998.

5 Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

11. Acknowledgments

**** TODO ****

12. Author’s Addresses

James Undery
Ubiquity Software Corporation Ltd.
Ubiquity House
Langstone Park
Newport, UK
Email: jundery@ubiquity.net

Sanjoy Sen
Nortel Networks
Email: sanjoy@nortelnetworks.com

Vesa Torvinen
Oy LM Ericsson Ab
Email: vesa.torvinen@ericsson.fi

Full Copyright Statement