3GPP TSG SA WG3 Security — S3#22 **S3-020099**

February 25th– February 28th, 2002

Bristol, UK

Agenda Item:	7.3
Source:	Ericsson
Title:	Security Mechanism Agreement for SIP Connections
Document for:	Information

S3 is kindly asked to review the updated version of the internet draft on "Security Mechanism Agreement for SIP Connections" (draft-arkko-sip-sec-agree-01.txt).

The main modifications introduced from the previous version are compiled in section 11 of the draft itself and can be summarized in …

```
11.  Modifications

     The -01 version of this draft introduced the following modifications:

        -  Reversed approach to make servers stateless

        -  Removed discussion of the use of this for Digest algorithm
           selection, since Enhanced Digest already has bidding-down
           protection

        -  Renamed org.iana.sip.digest to org.iana.sip.edigest and removed the
           parameters, as we can rely on Enhanced Digest to perform the
           algorithm selection.

        -  Removed agreements for full paths.
```

The plan is to submit the new version of the draft by the end of the week. Feedback from S3 would be much appreciated.

Network Working Group
INTERNET-DRAFT
<draft-arkko-sip-sec-agree-01.txt>
30 February 2002

Jari Arkko
Vesa Torvinen
Ericsson
Tao Haukka
Nokia
Sanjoy Sen
Lee Valerius
Nortel Networks

Security Mechanism Agreement for SIP Connections

1. Status of this Memo

2. Abstract

SIP has a number of security mechanisms for hop-by-hop and end-to-end protection. Some of the security mechanisms have been built in to the SIP protocol, such as HTTP authentication or secure attachments. In these mechanisms there are even alternative algorithms and parameters. Currently, HTTP authentication is known to be vulnerable to so called Bidding-Down attacks where a Man-In-The-Middle attacker simply modifies messages in a way that leads parties to believe the other side only supports weaker algorithms than they actually do. Also, currently it isn't possible to select which security mechanisms to use over a connection. In particular, even if some mechanisms such as OPTIONS or NEGOTIATE were used to make this selection, the selection would be again vulnerable against the Bidding-Down attack. On small networks configuration and software update methods are sufficient to deal with this type of attacks, but on large networks that evolve over time, the security implications are serious: either you deny connections from large amounts of older equipment, or risk losing all value of new algorithms through attacks that are trivial to the attackers.

3. Contents

4. Introduction

Traditionally, security protocols have included facilities to agree on the used mechanisms, algorithms, and other security parameters. The reason for this is that experience has shown algorithm development uncovers problems in old algorithms and produces new ones. Furthermore, different algorithms are suitable for different situations. Typically, protocols also select other parameters beyond algorithms at the same time.

The purpose of this paper is to study whether similar functionality is necessary in SIP [1]. SIP has some security functionality built-in such as different variants of HTTP authentication [4], secure attachments such as S/MIME, and can also use underlying security protocols such as IPSec/IKE [2], TLS [3]. Some of the built-in security functionality has also alternative algorithms and other parameters. While some work within the SIP Working Group has been looking towards reducing the number of recommended security solutions (e.g. recommend just one lower layer security protocol), we can not expect to cut down the number of items in the whole list to one. There will still be multiple security solutions in SIP. Furthermore, given that security work around SIP is in its early stages, it is likely that new methods will appear in the future, to complete the methods that exist today.

Chapter 5 shows that without a secure method to choose between security mechanisms and/or their parameters, SIP is vulnerable to certain attacks. As the HTTP authentication RFC [4] points out, authentication and integrity protection using multiple alternative methods and algorithms is vulnerable to Man-in-the-Middle (MITM) attacks. More seriously, it is hard to know if a SIP peer entity truly can't perform e.g. auth-int QOP in Digest, TLS, or S/MIME, or if a MITM attack is in progress. In small workstation networks these issues are not very relevant, but the deployment of hundreds of millions of small devices

with little or no possibilities for coordinated security policies, let alone software upgrades makes these issues much worse. This conclusion is supported by the requirements from 3GPP [7].

Chapter 6 outlines some possible solutions to these problems, and Chapter 7 documents our proposed solution.

5. The Problem

SIP has alternative security mechanisms such as HTTP authentication / integrity protection, lower layer security protocol(s), S/MIME. It is likely that their use will continue in the future. SIP security is developing, and is likely to see also new solutions in the future, for example along the introduction of SIP for new network access technologies. Future services may also bring with themselves different security requirements and methods.

Deployment of large number of SIP-based consumer devices such as 3GPP terminals requires all network devices to be able to accommodate both current and future mechanisms; there is no possiblity for instantaneous change since the new solutions are coming gradually in as new standards and product releases occur. It isn’t even possible to upgrade some of the devices without getting completely new hardware.

So, the basic security problem that such a large SIP-based network must consider, how do security mechanisms get selected? It would be desirable to take advantage of new mechanisms as they become available in products.

Firstly, we need to know somehow what security should be applied, and preferably find this out without too many additional roundtrips.

Secondly, selection of security mechanisms MUST be secure. Traditionally, all security protocols use a secure form of negotiation. For instance, after establishing mutual keys through Diffie-Hellman, IKE sends hashes of the previously sent data -- including the offered crypto mechanisms. This allows the peers to detect if the initial, unprotected offers were tampered with.

The security implications of this are subtle, but do have a fundamental importance in building large networks that change over time. Given that the hashes are produced also using algorithms agreed in the first unprotected messages, one could ask what the difference in security really is. First, assuming hashing is mandatory and only secure algorithms are used, we still need to prevent MITM attackers from modifying other parameters, such as whether encryption is provided or not. Secondly, it turns out, however, that there indeed is still a difference even for hashes. Let us first assume two peers capable of using both strong and weak security. If the initial offers are not protected in any way, *any* attacker can easily "downgrade" the offers by removing the strong options. This would force the two peers to use weak security between them. But if the offers are protected in some way -- such as by hashing, or repeating them later when the selected security is really on -- the situation is different. It would not be sufficient for the attacker to modify a single message. Instead, the attacker would have to modify both the offer message, as well as the message that contains the hash/repetition. More importantly, the attacker would have to forge the weak hash / security that is present in the second message, and would have to do so in real time between the sent offers and the later messages. Otherwise, the peers would notice that the hash is incorrect.

In conclusion, the security difference is making a trivial attack possible versus demanding the attacker to break algorithms. An example of where this has a serious consequence is when a network is first deployed with integrity protection (such as HTTP Digest [4, 8]), and then later new devices are added that support also encryption (such as S/MIME [1]). In this situation, an insecure negotiation procedure allows attackers to trivially force even new devices to use only integrity protection.

It can be asked why the devices would be allowing both weak and strong security in the first place. The answer lies in understanding how networks are deployed, and in the logistical and economical problems in upgrading global networks instantanously. These issues are of particularly high relevance for networks with a large number of devices, such as the third generation mobile networks. Once millions or even hundreds of millions of devices have been sold to customers, it becomes impossible to replace them with new devices. Therefore, network equipment such as SIP proxies must continue to accept even the older equipement that are less capable in terms of security. Similarly, clients wishing to stay in contact regardless of who they call or where they are, have a need to allow both weaker and stronger mechanisms.

Therefore, we feel that in large networks it is necessary to include some security agreement mechanisms in SIP.

6. Alternative Solutions

Basic SIP features such as OPTIONS and Require, Supported headers are capable of informing peers about various capabilities including security mechanisms. However, the straightforward use of these features does not guarantee a secured agreement. (It might be possible to add some new behaviour rules for these headers to allow their use also in secure manner. However, it appears that in order to introduce security, the headers must be repeated under the selected security protection. In order for the repetition to be useful, either the server would have to be stateful, or the client must repeat the server's list. Stateful servers are not desireable and neither to the Supported or the Require header appears suitable for the client to describe server's capabalities. Hence, the use of these headers is not desireable.)

HTTP Digest algorithm lists [4] are not secure for picking among the digest integrity algorithms, as is described in the RFC itself. Enhanced HTTP Digest [8] corrects this problem. More seriously,

neither the original or Enhanced Digest has no provisions for allowing encryption to be negotiated. Hence, it would be hard to turn on possible future encryption schemes in a secure manner.

The SIP NEGOTIATE method [5] allows powerful negotiation of various kinds of parameters, including security mechanisms and algorithms. However, it does not allow for secure negotiation as is described in the Internet Draft itself.

The SIP Security Framework [6] also allows for the agreement about the used security mechanisms. However, it does not do this in a secure manner.

7. Proposed Solution

In our opinion, the optimal solution to the SIP security negotiation problem has the following properties:

(a) It allows the selection of security mechanisms, such as lower layer security protocols or secure attachments. It also allows the selection of individual algorithms and parameters where the security functions are integrated in SIP (such as in the case of HTTP authentication or secure attachments).

(b) It allows both end-to-end and hop-by-hop negotiation.

(c) It is secure, i.e. prevents bidding down attacks.

(d) It is capable of running without additional roundtrips. This is important in the cellular environment, where an additional roundtrip could cost 1000 to 1500 ms for the call set up delay.

(e) It does not introduce any additional state to servers and proxies.

7.1. Design

We propose a scheme - a bit like the one in the framework draft [6] - where security features are represented as regular option tags in SIP. If there will ever be any features that require parameters such as key lengths, the option tags can be associated with an optional value field. The clients MUST announce a list of supported option tags in their first request. The servers MUST use this information in preparing their response, such as including a challenge if the first commonly supported mechanism is Enhanced Digest. It isn't necessary, however, for the server to remember the clients preferences beyond the response.

The servers MUST announce a list of supported option tags in their first response. This list MUST NOT depend on the contents of the list sent by the client in the first message. Typically, the server's list of supported option tags is static. In the client’s second request, the client MUST return the server’s list.

The client makes the selection of the used security mechanism based on its own preferences and the server’s list. The client MUST start to use the selected security mechanism from the second request message.

The security of the agreement comes from the client’s repetition of the server’s list of option tags in the second request message. The server can then proceed to verify that the list has not been modified. If a modification is detected, the server returns on error or disconnects. The server MUST send a positive answer if and only if the list was not modified. The server does not need to memorize the lists it has sent in earlier responses, provided that the set of security mechanisms supported by the server is constant, which seems like a reasonable assumption.

Attackers could also try to modify the repeated list in the second request from the client. However, if the selected security mechanism uses encryption this may not be possible, and if it uses integrity protection any modifications will be detected by the server. In order to ensure this, all clients that implement this specification MUST select Enhanced Digest [8], S/MIME, TLS, IPsec, or any stronger methed for the protection of the second request.

Attackers could also try to modify the client's list of security mechanisms in the first message. This would either be revealed to the participants, because of unexpected challenges in the server’s first response, or would have no effect because the client picks its own security method only based on its local information and the server’s static list.

The client’s first protected request can be a real request such as INVITE, as the server MUST check the correctness of the lists before it proceeds to execute the requested operation.

Our approch explicitly lists the recipients of the security method agreement. This is intended to allow a negotiation of the first-hop security mechanism while at the same time running e.g. a REGISTER with Digest authentication to a server some hops further away.

This approach could also be trivially extended to support security agreeent over a full path. However, since the sips: URI scheme already solves the most pressing issue in that area we have chosen to not support this.

7.2. Header descriptions

The Security-Method header indicates who wants security towards whom, and what kind of security. The syntax of this header is as follows:

```
  "Security-Method:" to-uri "," from-uri "," mechlist

  Where

    to-uri = uri
    from-uri = uri
    mechlist = mechopts *( ";" mechopts )
    mechopts = mechtag *( "," mechtag )
    mechtag = option-tag ["=" token *( ":" token )]
```

The meaning of these fields is as follows:

- The "to-uri" indicates the desired receiver of the information. The value of this field should be a SIP URI. When sent by a client, the value would typically (but not necessarily) contain just the host and port number parts.

- The "from-uri" indicates the sender of the security agreement information. The value of this is also a SIP URI. When sent by a client, the value would typically (but not necessarily) include a username part.

- The "mechlist" represents a list of security mechanisms, all of which must be supported simultaneously on the same connection (such as both Digest TLS).

- The "mechopts" represents a list of alternative security mechanisms. Inside one "mechlist" entry we can have multiple alternative mechanisms and algorithms. For instance, the the list "org.iana.sip.edigest, org.iana.sip.smime; org.iana.sip.ike" would represent the requirement that one must run simultaneously IPsec/IKE and either HTTP Digest or S/MIME.

The "mechtag" represent one individual mechanism. The "option-tag" syntax is used for these in order to facilitiate the easy addition of new mechanisms. All option tags starting with "org.iana.sip." MUST be documented in Internet Drafts or RFCs. The initial list of standardized option-tags is presented below:

```
  org.iana.sip.edigest: Extended HTTP Digest authentication
  org.iana.sip.tls: TLS
  org.iana.sip.smime: S/MIME
  org.iana.sip.ike: IPsec/IKE
```

The optional "token" parameters associated with an "option-tag" can be used to assign parameter values to certain options. This may be useful to select algorithms, key lengths, or other similar parameters in mechanisms integrated to SIP. No such parameters are defined for the four above mechanisms, however.

Multiple instances of the same header field can appear in SIP messages. Typically, the client inserts its own Security-Method header when it sends a request, and the server/proxy adds its own response. The parameters are in all cases set in an appropriate manner to indicate in the "to-uri" paremeter the party who inserted the header.

8.   Examples

8.1. Selecting Between New and Old Mechanisms

In this example we demonstrate the use of the framework for securing the first hop using some security mechanism, without knowing beforehand which methods the server supports. We assume that the client is not willing to reveal any information on what it intends to do, so it uses OPTIONS in the first message that is sent in the clear. The example starts by a client sending a message to the server, indicating that it is of the new variant that supports TLS in Step 1. In Step 2, the server responds that with it own list of security mechanisms -- Enhanced Digest and TLS in this case -- and the peers start only common security service i.e. TLS at Step 3. In Step 4, the client resends the server’s Security-Method header, which the server verifies, and responds with 200 OK.

```
  1. Client -> Server:

     OPTIONS server SIP/2.0
     Security-Method: sip:client sip:server org.iana.sip.tls

  2. Server -> Client:

     200 OK
     Security-Method: sip:server sip:client org.iana.sip.edigest,
                      org.iana.sip.tls

  3. Security handshake at a lower layer i.e. TLS

  4. Client -> Server:

     INVITE server SIP/2.0
     Security-Method: sip:server sip:client org.iana.sip.edigest,
                      org.iana.sip.tls

  5. Server -> Client:

     200 OK
```

In the example we have omitted the returned values of Security-Method in replies for clarity. Typically in SIP the servers do not remove header fields as they answer, they only add new headers.

If this example was run without Security-Method in Step 2, the client would not know what kind of security the other one supports, and would be forced to error-prone trials.

More seriously, if the Security-Method was omitted in Step 4, the whole process would be prone for MITM attacks. An attacker could spoof "ICMP Port Unreachable" message on the trials, or remove the stronger security option from the header in Step 1, therefore substantially reducing the security.

8.2. 3GPP Terminals

This example attempts to show that the 3GPP requirements on being able to use lightweight security methods over the cellular interface and secure agreement on algorithms in these methods can be achieved using our method.

In 3GPP networks, the clients make REGISTER operation in their first message, in order to inform the home network that they are at a particular location. Due to the properties of 3GPP radio interfaces, it is necessary to optimize the number of roundtrips needed in the whole process. Therefore, we try to parallelize the tasks. It should be noted that the same functionality could be achieved using additional OPTIONS messages. We assume that 3GPP uses Enhanced HTTP Digest authentication to protect signaling in the first hop. As Enhanced Digest can securely negotiate the used algorithms it is not necessary to use this method for that. However, as Enhanced Digest does not provide confidentiality, it may be necessary to upgrade to the use of TLS or S/MIME in future terminals.

The example starts by an old version client coming to a new area and learning the address of the local proxy. The proxy is of a newer version, so it supports multiple security mechanisms. The client also knows its home server address. We assume that some trust has already been established between the client and the home, and between the client and the proxy. Perhaps this trust is in the form of the nodes belonging under the same PKI, or having distributed shared secrets beforehand.

In Step 1 the client contacts the proxy using a REGISTER message. (We omit communications with the home server in this discussion, but the proxy forwards the messages onwards.) In Step 2, the proxy responds indicating that it is of the new variant that supports Enhanced Digest, S/MIME, and TLS for the protection of the first hop. In Step 3, the client selects the first method is supports (enhanced digest in this case), the protection is turned on and the client sends the next round of REGISTER messages to the server. This includes the repetition of the original security capabilities of the server. The server verifies this list, and in Step 4 it responds with a 200 OK.

1. Client -> Proxy:

```
REGISTER server SIP/2.0
Security-Method: sip:client sip:proxy org.iana.sip.edigest
```

2. Proxy -> Client:

```
401 Authentication Required
(Some end-to-end authentication headers)
(Enhanced digest challenge to the client from the proxy)
Security-Method: sip:proxy sip:client org.iana.sip.smime
                 org.iana.sip.tls org.iana.sip.edigest
```

3. Client -> Proxy:

```
      REGISTER server SIP/2.0
      (Some end-to-end authentication headers)
      (Enhanced digest response to the proxy)
      Security-Method: sip:proxy sip:client org.iana.sip.smime
                       org.iana.sip.tls org.iana.sip.edigest
```

4. Proxy -> Client:

```
      200 OK
      (Some end-to-end authentication headers)
      (Some enhanced digest headers from the proxy)
```

As in the previous example, if this was run without Security-Method in Step 2, the client would not know what kind of algorithms the server supports. In this example we demonstrate also the need for the client to send its own mechanism list in Step 1. If this wasn’t known to the proxy when it responds in Step 2, it could not have provided a suitable Enhanced Digest challenge because at that point the proxy would not have known if the client supports that.

As in the previous example, removing the repetition of the Security-Method header in Step 3 would open the system to MITM attacks.

9. Security Considerations

This draft is about making it possible to select between various SIP security mechanisms in a secure manner. In particular, the method presented here allow current networks using e.g. Digest later securely upgrade to e.g. S/MIME without requiring a simultaneous modification in all equipment.

The method presented in this draft is secure only if the weakest proposed mechanism offers at least integrity protection. Therefore, we recommend that at leat Enhanced HTTP authentication SHOULD be used in conjunction with our approach.

10. Conclusions

The presented methods appear to secure the selection between different security mechanisms. This is important for deployments in large networks. The authors seek comments on the proposed approach, and encourage security analysis of both current SIP and the proposal.

11. Modifications

The -01 version of this draft introduced the following modifications:

- Reversed approach to make servers stateless

- Removed discussion of the use of this for Digest algorithm selection, since Enhanced Digest already has bidding-down protection

- Renamed org.iana.sip.digest to org.iana.sip.edigest and removed the parameters, as we can rely on Enhanced Digest to perform the algorithm selection.

- Removed agreements for full paths.

12. Acknowledgments

The authors wish to thank Rolf Blom, Hugh Shieh, Gunther Horn, Krister Boman, David Castellanos-Zamora, Aki Niemi, Valtteri Niemi, and members of the 3GPP SA3 group for interesting discussions in this problem space.

13. References

[1] Handley, M., Schulzrinne, H, Schooler, E. and Rosenberg, J., "SIP: Session Initiation Protocol", Work In Progress, draft-ietf-sip-rfc2543bis-08.txt, IETF, February 2002.

[2] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[3] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

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

[5] S. Parameswar, B. Stucker, "The SIP Negotiate Method", Work In Progress, draft-spbs-sip-negotiate-00.txt, IETF, August 2001.

[6] M. Thomas, "SIP Security Framework", draft-thomas-sip-sec-framework-00.txt. Work In Progress, IETF, July 2001.

[7] M. Garcia, D. Mills, G. Bajko, G. Mayer, F. Derome, H. Shieh, A. Allen, S. Chotai, K. Drage, J. Bharatia, "3GPP requirements on SIP", draft-garcia-sipping-3gpp-reqs-00.txt. Work In Progress, IETF, October 2001.

[8] J. Undery, S. Sen, V. Torvinen, "SIP Digest Authentication: Extensions to HTTP Digest Authentication", draft-undery-sip-auth-00.txt. Work In Progress, IETF, January 2002.

14. Author's Address

Jari Arkko, Vesa Torvinen
Ericsson
02420 Jorvas
Finland
EMail: Jari.Arkko@ericsson.com, Vesa.Torvinen@ericsson.fi

Tao Haukka
Nokia
Finland

EMail: Tao.Haukka@nokia.com

Sanjoy Sen
Nortel Networks
2735-B Glenville Drive
Richardson, TX 75082, USA
EMail: sanjoy@nortelnetworks.com

Lee Valerius
Nortel Networks
2201 Lakeside Blvd
Richards, TX 75082, USA
EMail: valerius@nortelnetworks.com