3GPP TSG SA WG3 Security — S3#23 **S3-020243**
14 - 17 May 2002
Victoria, Canada

SIP Working Group
INTERNET-DRAFT
<draft-ietf-sip-sec-agree-01.txt>
May 2002
Expires: December 2002

Jari Arkko
Vesa Torvinen
Gonzalo Camarillo
Ericsson
Tao Haukka
Nokia
Sanjoy Sen
Nortel Networks

Security Mechanism Agreement for SIP Sessions

Status of this memo

Abstract

SIP has a number of security mechanisms. Some of them have been built in to the SIP protocol, such as HTTP authentication or secure attachments. These mechanisms have even alternative algorithms and parameters. SIP does not currently provide any mechanism for selecting which security mechanisms to use over a connection. In particular, even if some mechanisms such as OPTIONS were used to make this selection, the selection would be vulnerable against the Bidding-Down attack. This document defines three header fields for negotiating the security mechanisms within SIP between a user agent client and its next hop SIP entity. A SIP entity applying this mechanism must always require some minimum security (i.e. integrity protection) from all communicating parties in order to secure the negotiation, but the negotiation can agree on which specific minimum security is used.

TABLE OF CONTENTS

1. 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 that algorithm development uncovers problems in old algorithms and produces new

ones. Furthermore, different mechanisms and algorithms are suitable for different situations. Typically, protocols also select other parameters beyond algorithms at the same time.

The purpose of this specification is to define a similar negotiation functionality in SIP [1]. SIP has some security functionality built-in such as HTTP Digest authentication [4], secure attachments such as S/MIME [5], and can also use underlying security protocols such as IPsec/IKE [2] or TLS [3]. Some of the built-in security functionality allows 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 (i.e., 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 utilized by SIP. Furthermore, it is likely that new methods will appear in the future, to complement the methods that exist today.

Chapter 2 shows that without a secured 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 or sometimes even impossible to know whether a SIP peer entity is truly unable to perform (e.g., Digest, TLS, or S/MIME) or if a MitM attack is in action. In small networks consisting of workstations and servers these issues are not very relevant, as the administrators can deploy appropriate software versions and set up policies for using exactly the right type of security. However, SIP will be deployed to hundreds of millions of small devices with little or no possibilities for coordinated security policies, let alone software upgrades, and this makes these issues much worse. This conclusion is also supported by the requirements from 3GPP [6].

Chapter 6 documents the proposed solution, and chapter 7 gives some demonstrative examples.

2. Problem Description

SIP has alternative security mechanisms such as HTTP authentication with integrity protection, lower layer security protocols, and 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.

Deployment of large number of SIP-based consumer devices such as 3GPP terminals requires all network devices to be able to accommodate past, current and future mechanisms; there is no possibility for instantaneous change since the new solutions are coming gradually in as new standards and product releases occur. It is sometimes even impossible 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, would be on 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. Assuming integrity protection 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. 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 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. If the attacker is able to break the weak security, the security method and/or the algorithm should not be used.

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]), 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.

3. Solution

3.1 Requirements

The solution to the SIP security negotiation problem should have the following properties:

(a) It allows the selection of security mechanisms, such as lower layer security protocols or HTTP digest. It also allows the selection of individual algorithms and parameters when the security functions

are integrated in SIP (such as in the case of HTTP authentication).

(b) It allows first-hop security negotiation.

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

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

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

Currently, SIP does not have any mechanism which fulfills all the requirements above. The basic SIP features such as OPTIONS and Require, Supported headers are capable of informing peers about various capabilities including security mechanisms. However, the straight forward use of these features can not guarantee a secured agreement. HTTP Digest algorithm lists [4] are not secure for picking among the digest integrity algorithms, as is described in the RFC itself. More seriously, they have no provisions for allowing

encryption to be negotiated. Hence, it would be hard to turn on possible future encryption schemes in a secure manner.

A self-describing security mechanism is a security mechanism that, when used, contains all necessary information about the method being used as well as all of its parameters such as algorithms.

A non-self-describing security mechanism is a security mechanism that, when used, requires that the use of the method or some of its parameters have been agreed beforehand.

Most security mechanisms used with SIP are self-describing. The use of HTTP digest, as well as the chosen algorithm is visible from the HTTP authentication headers. The use of S/MIME is indicated by the MIME headers, and the CMS structures inside S/MIME describe the used algorithms. TLS is run on a separate port in SIP, and where IPsec/IKE is used, IKE negotiates all the necessary parameters.

The only exception to this list is the use of manually keyed IPsec. IPsec headers do not contain information about the used algorithms. Furthermore, peers have to set up IPsec Security Associations before they can be used to receive traffic. In contrast S/MIME can be received even if no Security Association was in place, because the application can search for a Security Association (or create a new one) after having received a message that contains S/MIME.

In order to make it possible to negotiate both self-describing and non-self-describing security mechanisms, we need another requirement on the security agreement scheme:

(f) the security agreement scheme must allow both sides to decide on the desired security mechanism before it is actually used.

This decision can, and must, take place on both sides before we can

be sure that the negotiation has not been tampered by a man-in-the-middle. This tampering will be detected later.

3.2. Overview of Operations

The message flow below illustrates how the mechanism defined in this document works:

1. Client ----------client list---------> Server
2. Client <---------server list---------- Server
3. Client ------(turn on security)------- Server
4. Client ----------server list---------> Server
5. Client <---------ok or error---------- Server

Figure 1: Security negotiation message flow

Step 1: Clients wishing to use this specification can send a list of their supported security mechanisms along the first request to the server.

Step 2: Servers wishing to use this specification can challenge the client to perform the security agreement procedure. The security

mechanisms and parameters supported by the server are sent along in this challenge.

Step 3: The client then proceeds to select the highest-preference security mechanism they have in common and to turn on the selected security.

Step 4: The client contacts the server again, now using the selected security mechanism. The server's list of supported security mechanisms is returned as a response to the challenge.

Step 5: The server verifies its own list of security mechanisms in order to ensure that the original list had not been modified.

This procedure is stateless for servers (unless the used security mechanisms require the server to keep some state).

The client and the server lists are both static (i.e., they do not and cannot change based on the input from the other side). Nodes may, however, maintain several static lists, one for each interface, for example.

Between Steps 1 and 2, the server may set up a non-self-describing security mechanism if necessary. Note that with this type of security mechanisms, the server is necessarily stateful. The client would set up the non-self-describing security mechanism between Steps 2 and 4.

3.3. Syntax

We define three new SIP header fields, namely Security-Client, Security-Server and Security-Verify. Their BNF syntax is provided below:

```
security-client = "Security-Client" HCOLON
```

```
                    sec-mechanism *(COMMA sec-mechanism)
   security-server = "Security-Server" HCOLON
                    sec-mechanism *(COMMA sec-mechanism)
   security-verify = "Security-Verify" HCOLON
                    sec-mechanism *(COMMA sec-mechanism)
   sec-mechanism   = mechanism-name *(SEMI mech-parameters)
   mechanism-name  = ( "digest-integrity" / "tls" / "ipsec-ike" /
                     "ipsec-man" / "smime" / token )
   mech-parameters = ( preference / algorithm / extension )
   preference      = “q” EQUAL qvalue
   qvalue          = ( “0” [ “.” 0*3DIGIT ] )
                     / ( “1” [ “.” 0*3(“0”) ] )
   algorithm       = "alg" EQUAL token
   extension       = generic-param
```

Note that qvalue is already defined in the SIP BNF [1]. We have copied its definitions here for completeness.

The parameters described by the BNF above have the following semantics:

Mechanism-name: It identifies the security mechanism supported by the client, when it appears in a security-client header fields, or by the server, when it appears in a security-server header field. This specification defines six values:

- "tls" for TLS [3].
- "digest-integrity" for HTTP Digest [4] using additional integrity protection (i.e., the qop parameter) for the Security-Verify header field.
- "ipsec-ike" for IPsec with IKE [2].
- "ipsec-man" for manually keyed IPsec without IKE.
- "smime" for S/MIME [5].

Preference: The "q" value indicates a relative preference for the particular mechanism. The higher the value the more preferred the mechanism is.

Algorithm: An optional algorithm field for those security mechanisms which are not self-describing or which are vulnerable for bidding-down attacks (e.g., HTTP Digest). In the case of HTTP Digest, the same rules apply as defined in [4] for the "algorithm" field in HTTP Digest.

3.4. Protocol Operation

This section deals with the protocol details involved in the negotiation between a user agent client and its next-hop SIP entity. Throughout the text the next-hop SIP entity is referred to as the first-hop proxy or outbound proxy. However, the reader should bear in mind that a user agent server can also be the next-hop for a user agent client in the absence of proxies. Note as well that a proxy can also have an outbound proxy.

3.4.1 Client Initiated

A client wishing to establish some type of security with its first-hop proxy SHOULD add a Security-Client header field to a request addressed to this proxy (i.e., the destination of the request is the first-hop proxy). This header field contains a list of all the security mechanisms that the client supports. The client SHOULD NOT add preference parameters to this list. The client MUST also add a Require header field with the value "sec-agree" to its request.

The Security-Client header field is used by the server to include any necessary information in its response. For example, if digest-integrity is the chosen mechanism, the server includes a WWW-Authenticate header in the response. If S/MIME is chosen, the appropriate certificate is included. If the security mechanisms supported by the client do not need any further information to be established (e.g., TLS) the client MAY choose not to include the Security-Client header field in its request.

A server receiving a request that contains a Require header field with the value "sec-agree" MUST challenge the client with a 494 (Security Agreement Required) response. The server MUST add a

Security-Server header field to this response listing the security mechanisms that the server supports. The server MUST add its list to the response even if there are no common security mechanisms in the client's and server's lists. The server’s list MUST NOT depend on the contents of the client's list.

The server MUST compare the list received in the Security-Client header field with the list to be sent in the Security-Server header field. When the client receives this response, it will choose the common security mechanism with the higher preference value. Therefore, the server MUST add the necessary information so that the client can initiate that mechanism (e.g., a WWW-Authenticate header field for digest-integrity).

When the client receives a response with a Security-Server header field, it SHOULD choose the security mechanism in the server’s list with the highest "q" value among all the mechanisms that are known to the client. Then, it MUST initiate that particular security mechanism as described in Section 3.5. This initiation may be carried out without involving any SIP message exchange (e.g., establishing a TLS connection).

All the subsequent SIP requests sent by the client SHOULD make use of the security mechanism initiated in the previous step. These requests MUST contain a Security-Verify header field that mirrors the server’s list received previously in the Security-Server header field. This request MAY use SIP loose routing mechanism (i.e., Route header fields) to traverse the proxy, but its final destination may be different than the proxy. In this case, the request SHOULD NOT include a Require header field with the value "sec-agree".

For example, the first request was an OPTIONS request directly addressed to the proxy and the second request is an INVITE that will traverse the proxy but that is addressed to a real user (see

example in section 4.1).

The server MUST check that the security mechanisms listed in the Security-Verify header field of incoming requests correspond to its static list of supported security mechanisms. The server can proceed processing a particular request if, and only if, the list was not modified. If modification of the list is detected, the server MUST challenge the client with a 494 (Security Agreement Required). This response MUST include a challenge with server's unmodified list of supported security mechanisms.

Once the security has been negotiated between two SIP entities, the same SIP entities MAY use the same security when communicating with each other in different SIP roles. For example, if a UAC and its outbound proxy negotiate some security, they may try to use the same security for incoming requests (i.e., the UA will be acting as a UAS).

The user of a UA MAY be informed about the results of the security mechanism negotiation. The user MAY decline to accept a particular security mechanism, and abort further SIP communications with the peer.

3.4.2 Server Initiated

A server decides to use the security negotiation described in this document based on local policy. A server that decides to use this negotiation MUST challenge requests regardless of the presence or the absence of any Require or Supported header fields in incoming requests.

A server that by policy requires the use of this specification and receives a request that does not have the sec-agree option tag in a Require or Supported header field MUST return a 421 (Extension Required) response. If the request had the sec-agree option tag in a Supported header field, it MUST return a 494 (Security Agreement Required) response. In both situation the server MUST also include in the response a Security-Server header field listing its capabilities and a Require header field with an option-tag 'sec-agree' in it. All the Via header field entries in the response except the topmost value MUST be removed.

Clients that support the extension defined in this document MAY add a Supported header field with a value of "sec-agree". In addition to this, clients SHOULD add a Security-Client header field so that they can save a round trip in case the server decides to challenge the request.

3.5. Security mechanism initiation

Once the client chooses a security mechanism from the list received in the Security-Server header field from the server, it initiates that mechanism. Different mechanisms require different initiation procedures.

If TLS is chosen, the client MUST contact the server using the host

part of the Request-URI in the first request to the server as the destination of the connection (note that this may involve using standard SIP DNS procedures to locate a server). If this connection attempt fails, the security agreement procedure MUST be considered to have failed, and MUST be terminated.

If digest-integrity is chosen, the 494 (Security Agreement Required) response will contain an HTTP authentication challenge. The client MUST use the qos parameter possibly together with some variant of MIME tunneling so that the Security-Verify header field in the request is integrity protected in the MIME body. Note that digest alone would not fulfill the minimum security requirements of this specification.

To use "ipsec-ike", the client attempts to establish an IKE connection to the host part of the Request-URI in the first request to the server. If the IKE connection attempt fails, the agreement procedure MUST be considered to have failed, and MUST be terminated.

Note that "ipsec-man" will only work if the communicating SIP entities know which keys and other parameters to use. It is outside

the scope of this specification to describe how this information can be made known to the peers.

In both IPsec-based mechanisms, it is expected that appropriate policy entries for protecting SIP have been configured or will be created before attempting to use the security agreement procedure, and that SIP communications use port numbers and addresses according to these policy entries. It is outside the scope of this specification to describe how this information can be made known to the peers, but it could be typically configured at the same time as the IKE credentials or manual SAs have been entered.

To use S/MIME, the client MUST construct its request using S/MIME. The client may have received the server's certificate in an S/MIME body in the 494 (Security Agreement Required) response.

3.6. Duration of Security Associations

Once a security mechanism has been negotiated, both the server and the client need to know until when it can be used. All the mechanisms described in this document have a different way to signal the end of a security association. When TLS is used, the termination of the connection indicates that a new negotiation is needed. IKE negotiates the duration of a security association. If the credentials provided by a client using digest-integrity are not longer valid, the server will re-challenge the client. It is assumed that when IPsec-man is used, the same out-of-band mechanism used to distribute keys is used to define the duration of the security association.

3.7. Summary of Header Field Use

The header fields defined in this document may be used to negotiate the security mechanisms between a UAC and other SIP entities including UAS, proxy, and registrar. Information about the use of

headers in relation to SIP methods and proxy processing is summarized in Table 1.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Security-Client	R	ard	-	o	-	o	o	o
Security-Server	401,407,421,494		-	o	-	o	o	o
Security-Verify	R	ard	-	o	-	o	o	o

Header field	where	proxy	SUB	NOT	PRK	IFO	UPD	MSG
Security-Client	R	ard	o	o	-	o	o	o
Security-Server	401,407,421,494		o	o	-	o	o	o
Security-Verify	R	ard	o	o	-	o	o	o

Table 1: Summary of header usage.

The "where" column describes the request and response types in which

the header field may be used. The header may not appear in other types of SIP messages. Values in the where column are:

- R: Header field may appear in requests.

- 401, 407 etc.: A numerical value or range indicates response codes with which the header field can be used.

The "proxy" column describes the operations a proxy may perform on a header field:

- a: A proxy can add or concatenate the header field if not present.

- r: A proxy must be able to read the header field, and thus this header field cannot be encrypted.

- d: A proxy can delete a header field value.

The next six columns relate to the presence of a header field in a method:

- o: The header field is optional.

4. Backwards Compatibility

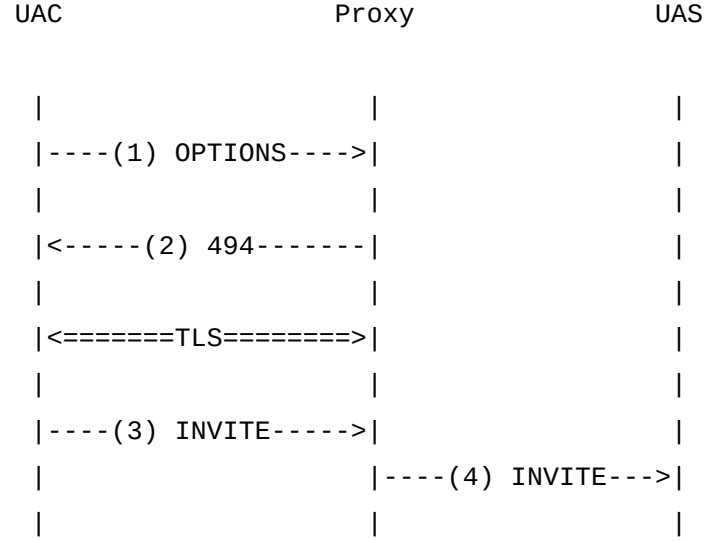
A server that, by local policy, decides to use the negotiation mechanism defined in this document, will not accept requests from clients that do not support this extension. This obviously breaks interoperability with every plain SIP client. Therefore, this extension should only be used in closed environments where it is ensured somehow that every client implements this extension.

5. Examples

The following examples illustrate the use of the mechanism defined above.

5.1. Client Initiated

A UA negotiates the security mechanism to be used with its outbound proxy without knowing beforehand which mechanisms the proxy supports.



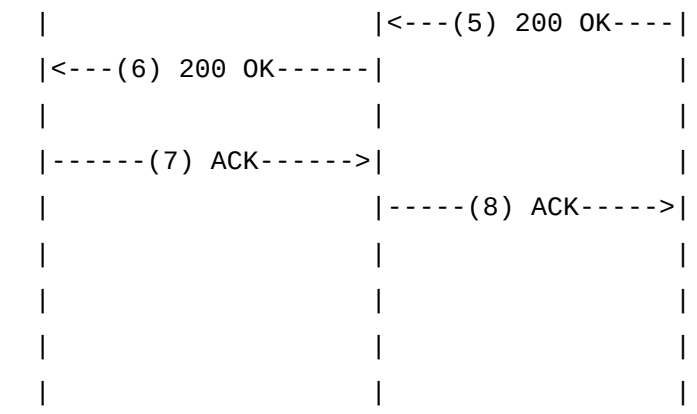


Figure 2: Negotiation initiated by the client

The UAC sends an OPTIONS request to its outbound proxy, indicating that it is able to negotiate security mechanisms and that it supports TLS and digest-integrity (Step 1 of figure 1). The outbound proxy challenges the UAC with its own list of security mechanisms – IPsec and TLS (Step 2 of figure 1). The only common security mechanism is TLS, so they establish a TLS connection between them (Step 3 of figure 1). When the connection is successfully established, the UAC sends an INVITE over the TLS connection just established (Step 4 of figure 1). This INVITE contains the server’s security list. The server verifies it, and since it matches its static list, it processes the INVITE and forwards it to the next hop.

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

More seriously, if the Security-verify 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.

```
(1) OPTIONS proxy.example.com
    Security-Client: tls;q=0.1
    Security-Client: digest-integrity;q=0.2
    Require: sec-agree

(2) 494 (Security Agreement Required)
    Security-Server: ipsec-ike;q=0.1
    Security-Server: tls;q=0.2

(3) INVITE proxy.example.com
    Security-Verify: ipsec-ike;q=0.1
    Security-Verify: tls;q=0.2
    Route: callee@domain.com
```

The 200 OK response for the INVITE and the ACK are also sent over the TLS connection. The ACK (7) will contain the same Security-Verify header field as the INVITE (3).

5.2. Server Initiated

In this example of figure 3 the client sends an INVITE towards the callee using an outbound proxy. This INVITE does not contain any Require header field.

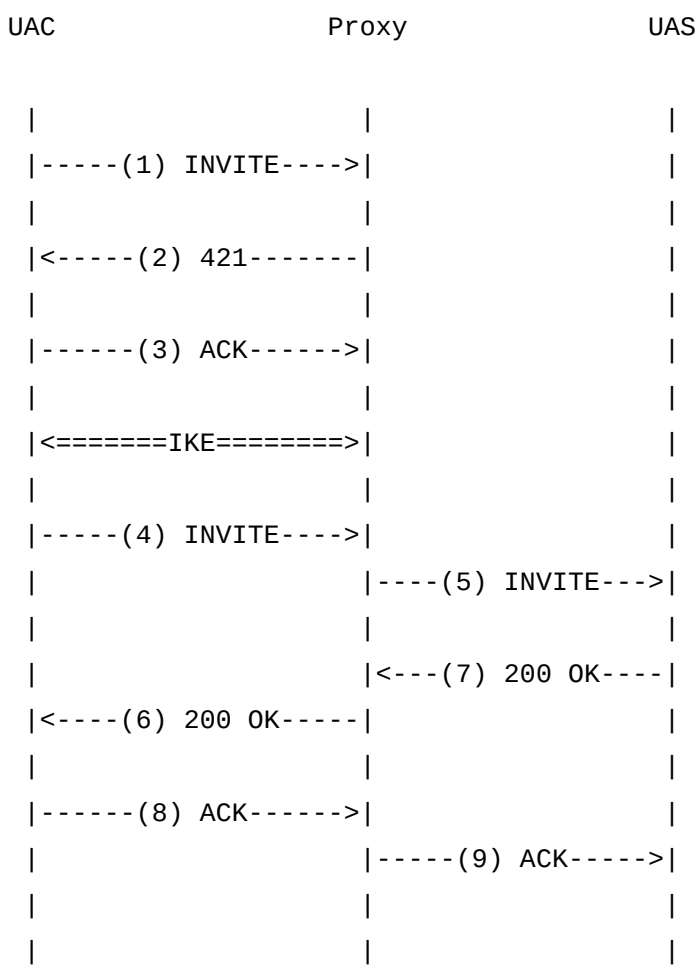


Figure 3: Server initiated security negotiation

The proxy, following its local policy, challenges the INVITE. It returns a 421 (Extension Required) with a Security-Server header field that lists IPsec-IKE and TLS. Since the UAC supports IPsec-IKE it performs the key exchange and establishes a security association with the proxy. The second INVITE (4) and the ACK (8) contain a

Security-Verify header field that mirrors the Security-Server header field received in the 421. The INVITE (4), the 200 OK (6) and the ACK (8) are sent using the security association that has been established.

6. Security Considerations

This specification 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, for instance, Digest, to be securely upgraded to, for instance, IPsec without requiring a simultaneous modification in all equipment. The method presented in this specification is secure only if the weakest proposed mechanism offers at least integrity protection.

Attackers could try to modify the server’s list of security mechanisms in the first response. This would be revealed to the server when the client returns the received list using the security.

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.

Finally, attackers could try to modify the client's list of security mechanisms in the first message. The client selects the security mechanism based on its own knowledge of its own capabilities and the server's list, hence the client's choice would be unaffected by any such modification. However, the server's choice could still be affected as described below:

- If the modification affected the server's choice, the server and client would end up choosing different security mechanisms in Step 3 or 4 of figure 1. Since they would be unable to communicate to each other, this would be detected as a potential attack. The client would either retry or give up in this situation.

- If the modification did not affect the server's choice, there's no effect.

All clients that implement this specification MUST select HTTP Digest with integrity, TLS, IPsec, or any stronger method for the protection of the second request. If HTTP Digest is used alone, the security agreement headers MUST be protected. This can be done with HTTP Digest if combined with MIME/SIP tunneling, for example.

7. IANA Considerations

This specification defines the 'sec-agree' SIP option tag which should be registered in IANA.

This specification also defines a new SIP status code, 494 (Security Agreement Failed), which should be registered in IANA.

8. Modifications

The draft-sip-sec-agree-01.txt version of this specification introduced the following modifications:

- Scope narrowed down to first-hop negotiation.

- Fixed syntax of header fields.

The draft-sip-sec-agree-00.txt version of this specification introduced the following modifications:

- Many editorial changes, restructuring and clarifications.

- Motivation section has been shortened since this is now a WG item.

- Clarified that the solution requires always some base level of security (i.e. integrity) in order to work. Even ’the weak security’ must not be broken.

- Text related to alternative solutions shortened and moved to a new place.

- New rules for possible error and special cases has been added, (e.g., for the case in which an non-adjacent SIP entities try to negotiate hop-by-hop security mechanisms).

- Syntax of the header redesigned. Wanted to get rid of the semantics related to the relative position of a header component in the header e.g., first parameters defines the 'from-uri', second the 'to-uri', and third the first supported security mechanism). The option tags are now used to identify the Security Agreement extension, not the individual security mechanisms.

- The semantics of the header slightly changed: the AND operator between the indivicual mechanisms is removed because it is really need with HTTP Digest only. And even in this case, the negotiation is not needed beforehand if some underlying security is used.

- Options for HTTP Digest algorithms and manually keyed IPsec added.

- Explicit rules were added to all mechanisms on how they should be used, such as TLS to be run on port 5061 etc.

- References to Enhanced HTTP Digest removed.

- Example related to 3GPP generalized.

The draft-arkko-sip-sec-agree-01.txt version of this specification 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.

- Simplified syntax

9. Acknowledgments

The authors wish to thank Lee Valerius, Rolf Blom, James Undery, Jonathan Rosenberg, Hugh Shieh, Gunther Horn, Krister Boman, David Castellanos-Zamora, Aki Niemi, Miguel Garcia, Valtteri Niemi, Martin Euchner, Eric Rescorla and members of the 3GPP SA3 group for interesting discussions in this problem space.

10. Normative References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler "SIP: Session Initiation Protocol", Work In Progress, draft-ietf-sip-rfc2543bis-09.txt, IETF, February 2002.

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

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

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

[5] B. Ramsdell and Ed, "S/MIME version 3 message specification," RFC 2633, IETF, June 1999.

11. Non-Normative References

[6] 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.

12. Authors’s Addresses

Jari Arkko
Ericsson
02420 Jorvas
Finland
EMail: Jari.Arkko@ericsson.com

Vesa Torvinen
Ericsson

02420 Jorvas
Finland
EMail: Vesa.Torvinen@ericsson.fi

Gonzalo Camarillo
Ericsson
02420 Jorvas
Finland
EMail: Gonzalo.Camarillo@ericsson.com

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