Extending SIP Authentication to exploit user credentials stored in existing authentication Databases

Stefano Salsano[(1)], Andrea Polidoro[(1)], Luca Veltri[(2)]
[(1)] DIE, University of Rome "Tor Vergata"- Italy
[(2)]Dpt. Information Engineering, University of Parma - Italy
E-mail: stefano.salsano@uniroma2.it, andrea.polidoro@uniroma2.it, luca.veltri@unipr.it

Abstract: **The SIP protocol provides authentication and authorization of SIP requests through a challenge-response authentication scheme inherited by the HTTP protocol and named HTTP Digest Authentication. The current specification defines a particular algorithm for calculating the challenge response that uses the MD5 hash of a combination of user name, realm, and password. Unfortunately, a lot of authentication systems maintain the user credentials protected with a one-way function (usually a hash) in a way that is incompatible with the information required by the current HTTP Digest Authentication. Some examples are given by the mechanisms used for storing passwords by the Unix OS, LDAP servers, or other applications. In this paper, we propose to extends the original HTTP Digest Authentication by adding a new and flexible scheme that uses an arbitrary hash function and an arbitrary combination of various information such as user name, realm, password, salt, and/or other data. The proposed authentication scheme has been implemented within two testbeds in which a SIP UA authenticates itself with a remote proxy server (acting as authenticator) that uses respectively a LDAP server or a users' password file of a Joomla Content Management System.**

1. INTRODUCTION

The SIP (Session Initiation Protocol) [1] is the IETF standard for setting-up, maintaining, and tearing-down peer-to-peer sessions between two or more user agents (UAs). SIP is currently used as standard signaling protocol for VoIP (Voice-over-IP), Instant Messaging (IM), and conferencing applications, and has been adopted by the 3GPP (Third Generation Partnership Project) as standard signaling protocol for real-time applications within the IP Multimedia Subsystem (IMS) of the third generation mobile networks (UMTS) [3].

As specified in RFC3261 [1], SIP uses the HTTP Digest Authentication mechanism for authenticating and authorizing the users. The SIP servers that perform authentication/authorization usually access a database with the credential of the users. This database of credential can either store the clear text password (rarely used for security risks) or a non-revertible hash of the password, using a mechanism coherent with that used by the HTTP Digest Authentication.

The problem is that with other user authentication/authorization mechanisms different and non compatible mechanisms are used to store non-revertible hashes of the password. This prevents reusing an existing database of user credentials to offer SIP based services.

Examples of password storing mechanisms that are not compatible with the SIP Digest Authentication are those used by: i) LDAP server, ii) UNIX with shadow/password file, iii) Apache (and other systems) with htpasswd file, iv) SQL database.

Therefore at the current state of the art, if someone wants to offer SIP services to a community of users that is authenticated using one of the four mechanisms shown above, a different SIP specific database of credentials need to be realized. This may pose some logistic problems and create user discomfort.

For this reason, in this paper we propose a set of authentication mechanisms for SIP that extend the one specified in RFC 3261 [1]. These algorithms use specific hash algorithms and a combination of various data such as user name, real, password, salt, and/or other information, in order to interoperate with the storage of credentials in existing authentication/authorization mechanisms.

This work was driven by two specific application scenarios that we had to face. The first scenario was a Wireless VoIP application, to be realized on top of an already realized WiFi authentication infrastructure. The WiFi authentication infrastructure relied on LDAP servers for storing the password and a set of users was already in the LDAP database. The advantage of reusing the same authentication is evident.

The second scenario relates to a research project named "Simple Mobile Services" (founded by the EC) [2]. In this project we have developed a communication middleware based on SIP [10] to exchange messages among a set of mobile terminals and servers. In order to provide web based interaction a web portal is realized using the open source Content Management System (CMS) Joomla. Each user needs to be authenticated and authorized to access services on the Joomla web portal. At the same time the user's mobile terminal uses a SIP UA that needs to be authenticated in order to use the middleware facilities and be able to send/receive messages. It was again natural to desire a single, integrated handling of user credential, and in particular to re-

This work has been partially supported by the Italian Ministry for University and Research (MIUR) within the project PROFILES under the PRIN 2006 research program.

use the sophisticated user management system of a CMS like Joomla.
First in section 2 a brief description of the HTTP digest authentication used in SIP is provided, then in section 3 the proposed solution is described, with a discussion on how the solution has been chosen among some alternatives. Section 4 provides a description of the implementation, and finally some conclusions are drawn in section 5.

2. HTTP DIGEST AUTHENTICATION FOR SIP

SIP is a text-based protocol with a syntax very similar to HTTP and, like HTTP, it uses a client-server model based on request-response transactions. SIP requests are exchanged between a SIP user agent client (UAC) and a SIP user agent server (UAS), or between a UAC and an intermediate SIP proxy. The SIP authentication procedure specified in [1] is derived from HTTP Digest Authentication [4]. It is a general challenge-response mechanism in which a UAS authenticates a UAC based on a shared secret. The challenge response is computed through the use of a one-way function based on various user's credentials. The standard also specifies the use of a particular function (in turn based on the MD5 hash function) that requires that both the client and server knows the user secret (normally a password) or, at least, the hash of the concatenation of the user name, the realm, and the password.

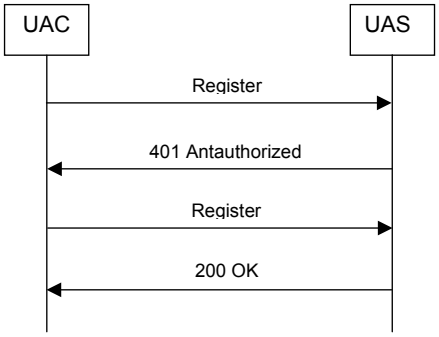

Figure 1 – The SIP digest authentication

A representation of this authentication procedure is given in Figure 1.
The challenge and response values are respectively inserted in the WWW-Authenticate header field (Proxy-Authenticate header for proxy authentication) of the response message sent by the UAS, and the Authorization header field (Proxy-Authorization header for proxy authentication) of the new request message sent by the UAC. According to [4] the response is computed as:

response = KD(H(A1), nonce:nc:cnonce:qop:H(method:uri))

where KD(secret,data) is a general two-parameters digest algorithm applied to "data" with secret "secret", H(data) is a digest algorithm applied to "data", and A1 is a quantity that should take into account user credentials.
As specified in [4], by default or in case of "algorithm=MD5", KD(,) H(), and A1 are respectively:

KD(secret, data) = MD5(secret:data))
H(data) = MD5(data)
A1 = username:realm:passwd

Particularly, the first term of KD(,) (indicated as "secret") and computed as H(A1) becomes:

secret = H(A1) = MD5(username:realm:passwd)

Note that both the requestor (the UAC) and the server do not need to know the user password, but rather this "secret" i.e. a digest function of username, realm and password itself: in order to correctly compute the response the UAC and the UAS need only to be able to calculate or retrieve from an archive the value MD5(username:realm:passwd).

3. NEW EXTENSIBLE AUTHENTICATION SCHEME

The HTTP Digest Authentication specified in [4] and described in the previous section requires that the response to the challenge, regardless of the selected algorithm, is in the form of:

response = KD(H(A1), nonce:nc:cnonce:qop:H(method:uri))

Since both the UAC and the UAS have to calculate such response value, they both have to be able to access user's credentials or, at least, the quantity H(A1). that in case of "algorithm=MD5" is:

H(A1) = MD5(username:realm:passwd)

Unfortunately several authentication systems are not compatible with this way of sending the user credentials in a digested form. At the contrary they use different hashing algorithms with different parameters. In this section we discuss some options on how to define authentication algorithms that are compatible with existing authentication systems and how to extend SIP signaling to this purpose.
Let us start from the definition of the algorithms. Assuming for the moment that we have no interest in changing the KD function, we can envisage two possible solutions:
a) we can define a different A1 and H(A1) function compatible with the specific authentication system (A1 is a formatted set of parameters that are taken into account by the H() function). For example A1 could be the concatenation passwd:salt and H(A1) could become:

H(A1) = MD5(passwd:salt)

b) we can reuse the definition of H(A1) and only replace the password parameter with a new derived pseudo-password A3 defined as A3=KP(password,other-params). In this case we are introducing a new function KP() with a new set of parameters that includes the user password. The value A1 becomes: A1=username:realm:A3.

For example, if we chose KP() as MD5(passwd:salt) the value H(A1) becomes:

H(A1) = MD5(username:realm:A3) = H(A1) =
= MD5(username:realm:MD5(passwd:salt))

The first solution (a) has the advantage that it may save some computation of crypto functions. The second option (b) has the advantage that it inherits all the security properties of the current MD5 solution specified for HTTP Digest Authentication. Moreover one could store in the client the derived password (i.e. the A3 value) instead of the password and maintain a compatibility with existing clients. The A3 value could be computed externally and then manually stored in the SIP client. Of course this is not our suggested solution, i.e. we believe that SIP stacks should be enhanced with the proposed solution so that SIP UA can natively support the new authentication method, anyway it was worth mentioning this possibility to reuse legacy clients in the short term.

Once that we have chosen which solution for the algorithm, we should discuss: 1) how to introduce the indication of this different authentication algorithms within SIP signaling; 2) how to convey the parameters that are needed by the new authentication algorithms. Again, we introduce two different options concerning issue 1):

i) introduce new algorithms specified as parameter "algorithm" within authentication headers. We could have for example algorithm=md5-ldap-sha1, algorithm=md5-crypt-des and so on.

ii) introduce a new authentication parameter called "pwd-algo" in order to indicate the chosen algorithm used to compute only the derived-passwd A3.

These two choices are not completely independent from the choice of the algorithm definition, as shown in the following table.

	i) introduce new values for the algorithm parameter	ii) introduce a new pwd-algo parameter
a) definition of a new A1 and H(A1)	Yes	No
b) reuse H(A1) with new A3 value in place of passwd	Possible	Suggested

If we define new A1 and H(A1) this should be signaled by introducing new values for the algorithm parameter. At the contrary, if we reuse H(A1) and introduce A3=KP() in place of the clear password, both options for the signaling are feasible, but we think that keeping the algorithm parameter unchanged (that specify A1, H(), and KD()) and introducing a new pwd-algo is preferable.

Now we finally need to discuss how to convey the additional parameter that may be needed by the different authentication mechanisms. We think that three options are possible:

1) introduce new parameters with specific names for each authentication algorithm;

2) introduce a generic parameter named pwd-param to carry algorithm specific parameters;

3) carry the new parameters added inside the nonce parameter. This is the approach that has been followed for example in the specification of the AKA authentication mechanism [5].

We believe that 1) is the worst solution as it may lead to add several new parameters. 2) and 3) are both feasible, where 2) is a cleaner approach that requires the definition of an additional parameter, while 3) has the advantage that does not require any new parameter.

Considering the various discussed options, a first possible solution is based on: a) reuse of H(A1) with a modified version of A1 in which the password value is simply replaced by A3 = KP(password,other-params), b) introduction of new "pwd-algo" parameter, and c) introduction of a new generic "pwd-param" parameter. Examples of the new pwd-algo parameter are:

pwd-algo=ssha
pwd-algo= crypt-md5
pwd-algo= crypt-apache

A second possible solution, more conservative from the point of view of SIP signaling is the use of A3 = KP(password,other-params) as above, but specifying the chosen algorithm in the existing "algorithm" parameter and carrying the algorithm specific parameters within the "nonce" parameter. Examples of the "algorithm" parameter are:

algorithm=md5-pwd-hashed
algorithm=sha1-pwd-salt-hashed

When we need to specify variants of the algorithm we think that a simple and efficient solution is to carry the name of the variant respectively into the pwd-param or nonce value.

3.1. FIRST SOLUTION

In the first solution we define two new parameters "pwd-algo" and "pwd-param" as follows:

pwd-algo = "pwd-algo" "=" ("plain" | "ssha" | "smd5" | "sha" | "md5" | "crypt-des" | crypt-algo | token)
crypt-algo = "crypt-" crypt-hash
crypt-hash = "des" | "md5" | "blowfish" | "apache" | token
pwd-param = "pwd-param" "=" (quoted-salt-value | quoted-string)
quoted-salt-value = LDQUOT salt-value RDQUOT
salt-value = <base-64 encoding of the salt value>

where "pwd-algo" specifies the function KP() used to compute the derived-password A3, while "pwd-param" has a completely opaque value, depending on the particular selected function KP, and indicates the values of the (eventual) parameters used in KP() computation (e.g. a salt value). A "pwd-algo=plain" value should indicate that none algorithm have to be used, and hence A3≡password.

Some "pwd-algo" values have been already defined in order to support the main hash methods used by some authentication systems such as in LDAP, Unix, Apache, etc.

For example, in case of LDAP server, possible values for these parameters are:

```
pwd-algo=smd5
pwd-param="123456"
pwd-algo=crypt-des
pwd-param="fzwhEV6E"
```

Examples of use of such new two parameters within a SIP transaction are:

```
INVITE sip:bob@neverland.net SIP/2.0
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
[…]

SIP/2.0 401 Unauthorized
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
WWW-Authenticate: Digest realm="example.com",
      nonce="cc5a61b2954e03541847f227102f",
      qop="auth", algorithm=MD5, pwd-algo=crypt-md5,
      pwd-param="fzwhEV6E"
[…]

INVITE sip:bob@neverland.net SIP/2.0
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
Authorization: Digest username="alice", realm="example.com",
   nonce="cc5a61b2954e03541847f227102f",
   pwd-algo=crypt-md5, pwd-param="MD5-fzwhEV6E"
   response="587410ee2dc5edd9bbe9370ddc1fA3a1",
   uri="sip:bob@neverland.net", qop=auth, nc=00000001
   cnonce="226827CAD1C949A18B17FD71EC68"
[…]

SIP/2.0 200 OK
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
[…]
```

3.2. SECOND SOLUTION

The second solution does not require any new authentication parameters since both the selected function KP() (used to generate the new "password" value A3 used in A1) and the eventual parameters of KP() are indicated respectively in the standard algorithm and nonce authentication parameters.
According with this solution, the "algorithm" and "nonce" parameters defined in RFC 3261 [1] can be extended as follows:

```
algorithm = "algorithm" EQUAL ( algorithm-value | compound-algorithm)
algorithm-value = "MD5" | "MD5-sess" | token
compound-algorithm = algorithm-value "–" algorithm-sub
algorithm-sub = "pwd-hashed" | "pwd-salt-hashed" | token
nonce = "nonce" EQUAL (nonce-value | compound-nonce )
compound-nonce = LDQUOT compound-nonce-value RDQUOT
compound-nonce-value = ( hash-algo | salted-hash-algo ) ":" nonce-value
hash-algo = "sha" | "md5" | token
salted-hash-algo = (hash-algo | "crypt-des") "-" salt-value
salt-value = <base-64 encoding of the salt value>
```

where "compound-algorithm" is the algorithm name that completely specifies the KD(), H(), A1, and KP() functions for the proper authentication scheme. It is formed by the basic H() algorithm name (e.g. "MD5") followed by the KP() algorithm name (e.g. "pwd-hashed").
In case of KP() function requires additional parameters such as sub-algorithm type, salt, etc, their values are included within the nonce parameter, as specified above (e.g. nonce="crypt-des-fzwhEV6E:cc5a61b2954e03541847f2").
For example, in case of LDAP server, possible values for these parameters are:

```
algorithm=MD5-pwd-hashed
nonce="sha: cc5a61b2954e03541847f2"
nonce="md5: cc5a61b2954e03541847f2"
algorithm=MD5-pwd-salt-hashed
nonce="crypt-des-fzwhEV6E: cc5a61b2954e03541847f2"
nonce="md5-fzwhEV6E: cc5a61b2954e03541847f2"
nonce="sha-fzwhEV6E: cc5a61b2954e03541847f2"
```

Examples of use of such parameters within a SIP transaction are:

```
INVITE sip:bob@neverland.net SIP/2.0
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
[…]

SIP/2.0 401 Unauthorized
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
WWW-Authenticate: Digest realm="example.com",
   nonce="crypt-des-fzwhEV6E:cc5a61b2954e03541847f2",
   qop="auth", algorithm=MD5-crypt
[…]

INVITE sip:bob@neverland.net SIP/2.0
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
Authorization: Digest username="alice", realm="example.com",
   nonce="crypt-des-fzwhEV6E:cc5a61b2954e03541847f2",
   response="587410ee2dc5edd9bbe9370ddc1fA3a1",
   uri="sip:bob@neverland.net", qop=auth, nc=00000001
   cnonce="226827CAD1C949A18B17FD71EC68"
[…]

SIP/2.0 200 OK
To: sip:bob@neverland.net
From: sip:alice@wonderland.net
[…]
```

We believe that this second solution is to be preferred to the one presented in the previous sub-section, as it does not require addition of new parameters. This is the same approach that has been followed when defining the SIP-AKA authentication mechanism [5]. We have posted the proposed solution as an internet-draft [7].

4. IMPLEMENTATION

The proposed SIP authentication mechanisms have been also implemented and integrated into two application scenarios.
The implementation has been based on the open source MjSip stack [6]. MjSip is a complete Java-based implementation of the complete layered SIP stack

architecture as defined by RFC 3261 [1]. Both the support on JavaSE and JavaME (running on J2ME/CLDC1.1/MIDP2.0) is available [6]. We have made available the source code of the implemented authentication mechanisms under GPL, see [11]. We have implemented both a Java SE version of the code and a J2ME version that can run on most mobile terminal that supports Java.

The first application scenario concerns about a VoIP application for mobile users that shares the authentication service with an already available Wi-Fi distributed infrastructure, that in turn is based on captive-portal access control and authentication functions implemented through a remote LDAP server. More detail on the complete wireless infrastructure can be found in [9]. Due to the characteristics of the password storing scheme implemented within the LDAP server, the implementation of the VoIP application required a new authentication mechanism as described in this document. Consequently we have modified both the mobile SIP UA and the SIP proxy server according with the solution proposed in section 3.1. The resulting implementation allows a mobile UA running on a laptop or smarthphone with WiFi interface to authenticate with the SIP Proxy server using the same user's credentials shared with the WiFi system.

The second application scenario is in the context of the Simple Mobile Services research project [2] where we integrated SIP authentication with the user credential Database stored by Joomla Content Management System. The SIP UAs authenticate with a remote proxy/registrar server (acting as authenticator) that use Joomla CMS password database. The Simple Mobile Service project implemented a open source platform for mobile service execution and creation. The platform is composed of a SIP based communication middleware (SMILE) [9], a J2ME client application named MOVE (Mobile Open and Very Easy), a set of server side components and a web based portal. The implementation is up and running. You can register to the system using the Joomla based portal [1]. Then you can download the MOVE client (a J2ME midlet suite) and configure it by inserting your user name and password in the configuration midlet. The client will use SIP with HTTP Digest, extended with our modification and its credentials will be retrieved by the SIP server on the mySQL database managed by Joomla.

5. CONCLUSIONS

In this paper we have proposed extensions to SIP authentication mechanism in order to interoperate with existing databases of user credentials. We have analyzed four types of commonly used user credential storage systems. Based on the requirements posed by these systems, we have analyzed different solutions and finally we have selected a preferred one. The chosen solution has been implemented and it has already been integrated in two different application scenarios. Finally some implementation metrics have been derived.

The chosen solution described in section 3.2 has a very limited impact on the SIP protocol, as no new parameters have to be defined. Only new values for the parameter that specifies the type of authentication mechanisms are needed.

REFERENCES

[1] J. Rosenberg et al., "SIP: Session Initiation Protocol", IETF Request for Comments RFC 3261, June 2002.

[2] The IST-Simple Mobile Service (IST-SMS) Project. http://www.ist-sms-org

[3] 3GPP TS 24.229: "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)".

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

[5] A. Niemi, J. Arkko, V. Torvine, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", IETF Request for Comments RFC 3310, September 2002

[6] MjSip project, http://www.mjsip.org/

[7] L. Veltri, S. Salsano, A. Polidoro, "HTTP digest authentication using alternate password storage schemes" <draft-veltri-sip-alt-auth-00>

[8] Joomla CMS project, http://www.joomla.org/

[9] D. Severina, M. Brunato, A. Ordine, L. Veltri, "UniWireless: a Distributed Open Access Network", WMASH 2006, Los Angeles, USA, pp. 30 – 36, September 29, 2006

[10] S. Salsano, G. Bartolomeo, C. Trubiani, and N.Blefari, "SMILE, a Simple Middleware Independent LayEr for distributed mobile applications", IEEE WCNC 2008, Las Vegas, USA, 31 March – 3 April, 2008.

[11] Implementation of hash-based SIP authentication, http://www.mjsip.org/projects/sip-alt-auth

[12] Portal of the Simple Mobile Service Community http://netgroup.uniroma2.it/SMS-community/