Generic Security Services API authentication support for the Session Initiation Protocol

Lars Strand
Norwegian Computing Center
Oslo, Norway
lars.strand@nr.no

Josef Noll
UniK-University Graduate Center
Kjeller, Norway
josef.noll@unik.no

Wolfgang Leister
Norwegian Computing Center
Oslo, Norway
wolfgang.leister@nr.no

***Abstract*—The mandatory and most deployed authentication method used in the Session Initiaton Protocol, the Digest Access Authentication method, is weak. Other, more secure authentication methods have emerged, but have seen little adoption yet. In this paper, support for using a generic authentication method, the Generic Security Services API, is added to the Session Initiaton Protocol. When using this method, the Session Initation Protocol does not need to support nor implement other authentication methods, only use the provided API library. This enables the Session Initiation Protocol to transparently support and use more secure authentication methods in a unified and generic way. As the suggested method includes a modification of the Session Initiation Protocol, an initial deployment strategy towards the Generic Security Services API authentication methods is added. To negotiate an authentication service, we use the pseudo security mechanism Simple and Protected GSS-API Negotiation Mechanism.**

***Keywords*—VoIP, SIP, authentication, GSS-API, SPNEGO.**

I. Introduction

Voice over IP (VoIP) is taking over for the traditional Public Switched Telephony Network (PSTN). At the end of 2009, 29.1 % of the private market in Norway was using VoIP (not including mobile phones). There has been a steady increase in the number of VoIP users since 2002, as well as a decrease in PSTN [1]. With two billion users worldwide having access to the Internet by the end of 2010 [2], the VoIP growth potential is huge.

Several proprietary and non-proprietary VoIP protocols have been created, but the protocol pair Session Initiation Protocol (SIP) and Real-time Transport Protocol (RTP) is emerging as the industry standard. RTP transfers the media content (voice), while SIP handles the signaling, i.e., setup, modification and termination of sessions between two or more participants. SIP is an open standard developed by the Internet Engineering Task Force (IETF) and specified in RFC3261 [3]. Additional functionality is specified in numerous Request For Comments (RFC) standard documents, making the SIP standard large and complex [4].

PSTN is a mature and stable technology providing 99.999% uptime [5], and users will expect VoIP to perform at similar service level. But with an increasing number of VoIP users, VoIP will become a target for attackers looking for financial gain or mischief. A clear threat taxonomy is given by the "VoIP Security Alliance" [6] and is discussed by Keromytis [7]. Several vulnerabilities exist [8] and securing SIP based installation is far from trivial [9].

In the EUX2010sec research project [10], we revealed, in close collaboration with our project partners, that most VoIP installations only use the mandatory, but weak, Digest Access Authentication (DAA) method [11]. In two case studies, with a VoIP installation supporting 3000 and 4700 phones respectively, the public tender for those VoIP installations greatly emphasized security and authentication requirements; requirements that DAA does not cover adequately. We have replicated the project partner's VoIP installation in our VoIP lab [12]. An attack against authentication has been analyzed [13] and countered [14].

The main contribution of this paper is to present and add support for the Generic Security Services Application Program Interface (GSS-API) to SIP. Different security requirements may require different authentication mechanisms. Instead of adding support for many different authentication mechanisms in SIP, support for GSS-API will provide a generic interface that makes different authentication methods transparent to the SIP protocol. To negotiate the best available authentication service between two peers, the Simple and Protected GSS-API NEGOtiation (SPNEGO) mechanism is used on top of GSS-API.

The rest of the paper is organized as follows: a brief overview of other SIP authentication methods is given in Section II. The GSS-API and SPNEGO as a GSS-API mechanism are explained in Section III and how the GSS-API can be supported and implemented in SIP is shown in Section IV. The industry evolution uptake strategy is briefly discussed in Section V before future work and the conclusion are presented in Section VI.

II. Authentication in SIP

When SIP was designed, functionality — and not security — was the primary goal. The results today are a number of uncovered vulnerabilites and attacks [15], [16]. SIP supports a wide range of functionalities that can be utilized, ranging from mobile handsets to high-end servers, each with different security requirements. Different security requirements may use different authentication methods depending on the

 ISBN: 978-1-61208-123-6

usage and threat scenario. For example, a mobile handset may have different requirements for authentication than that authentication between SIP servers. Additional requirements like power consumption and computational power must also be considered. Thus, adding new security services to SIP to improve the security design and meet different security requirements can be challenging.

The Digest Access Authentication (DAA) method is the mandatory and most used authentication method used with SIP [17]. DAA is heavily influenced by HTTP digest authentication. It relies on a cryptographic verification of a plaintext password shared between client and server. The client computes a MD5 hash value using the password, a nonce value received from the server, and a few SIP header values. The server computes the same digest, which then is compared against the one received from the client. If these digests are identical, the client has proven its identity and is authenticated. Unfortunately, the DAA is considered weak and is vulnerable to a series of attacks [8], including registration hijacking [14].

A more secure authentication method can be achieved by using Secure MIME (S/MIME) [18]. There, the entire SIP message is encapsulated in a MIME body that is signed and optionally encrypted. When the S/MIME header is received, the receiver checks whether the sender's certificate is signed by a trusted authority. A client must support multiple root certificates since there is no consolidated root authority that is trusted by all clients. This and other certificate handling issues like revoking and renewing complicates the use of certificates. Industry support for S/MIME has been limited [19].

Two other authentication methods have emerged within the Internet Engineering Task Force (IETF):

1) The *Asserted Identity* [20] is intended to work within a trusted environment. An additional, unprotected SIP header is sent in clear that informs that the identity of the client has been checked. Since the SIP header is sent in clear rather than protected by cryptography methods, it can easily be removed by an attacker without any of the communicating peers noticing this.
2) The *SIP Strong Identity* [21] introduces a new SIP service, the "authentication service", which signs a hash over selected SIP header values, and includes the signature as a SIP header along with a URI that points to the sender's certificate. The receiver computes the same hash and compares the results. Using this method, only the client is authenticated. As above, an attacker can remove these headers without implications.

Note that both of these authentication methods rely on a successful DAA authentication to be applicable. These are also applied by the SIP servers rather than the clients themselves, and are thus only providing indirect authentication of the client since the server is authenticating on behalf of the client.

III. GSS-API WITH SPNEGO

The GSS-API [22] provides a generic interface for application layer protocols like SIP, with a layer of abstraction for different security services like authentication, integrity or confidentiality. With the GSS-API, an application does not need to support or implement every authentication method, but use the provided security API [23]. The GSS-API is developed by the IETF and has been scrutinized by security professionals over the years. It has been extensively tested, and is now classified as a mature standard by the IETF. Further extensions and improvements to GSS-API are done by IETF's "kitten" Working Group [24].

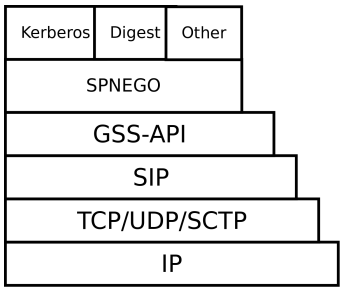

Figure 1: The GSS-API protocol stack with the SPNEGO negotiation mechanism and underlying security mechanisms.

The GSS-API is not a communication protocol in itself, but relies on the application to encapsulate, send, and extract data messages called "tokens" between the client and server. The tokens' content are opaque from the viewpoint of the calling application, and contain authentication data, or, once the authentication is complete, portion of data that the client and server want to sign or encrypt. The tokens are passed through the GSS-API to a range of underlying security mechanisms, ranging from secret-key cryptography, like Kerberos [25], to public-key cryptography, like the Simple Public-Key GSS-API Mechanism (SPKM) [26]. For an application, the use of the GSS-API becomes a standard interface to request authentication, integrity, and confidentiality services in a uniform way. However, GSS-API does not provide credentials needed by the underlying security mechanisms. Both server and client must aquire their respective credentials before GSS-API functions are called.

To establish peer entity authentication, a security context is initialized and established. After the security context has been established, additional messages can be exchanged, that are integrity and, optionally, confidentially protected. To initiate and manage a security context, the peers use the *context-level* GSS-API calls. The client calls `GSS_Init_sec_context()` that produces a "output_token" that is passed to the server. The server then calls `GSS_Accept_sec_context()` with the received token as input. Depending on the underlying security mechanism, additional token exchanges may be required in the course of context establishment. If so, `GSS_S_CONTINUE_NEEDED` status is set and additional tokens are passed between the client and server until a security context is established, as depicted in Figure 4.

After a security context has been established, *per-message* GSS-API calls can be used to protect a message by adding

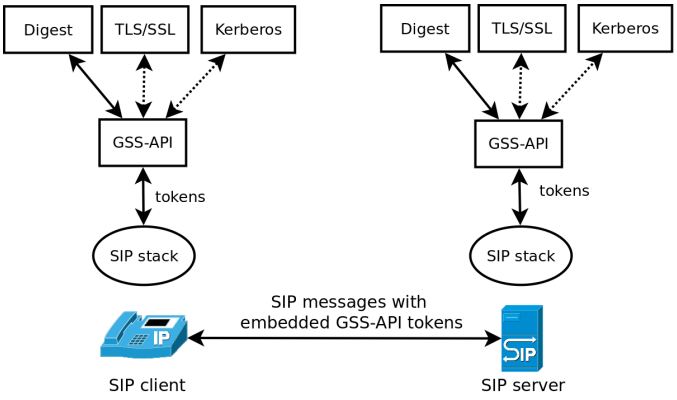

Figure 2: The GSS-API interface in SIP.

a Message Integrity Code (MIC) with `GSS_GetMIC()` and verifying the message with `GSS_VerifyMIC()`. To encrypt and decrypt messages, the peers can use `GSS_Wrap()` and `GSS_Unwrap()`. Thus, two different token types exist:

1) *Context-level tokens* are used when a context is established.
2) *Per-message tokens* are used after a context has been established, and are used to integrity or confidentiality protect data.

In addition to send and receive tokens, the application is responsible to distinguish between token types. This is necessary because different tokens types are sent by the application to different GSS-API functions. But since the tokens are opaque to the application, the application must use a method to distinguish between the token types. In our solution, we use explicit tagging of the token type that accompanies the token message.

SPNEGO [27] is a pseudo security mechanism that enables peers to negotiate a common set of one or more GSS-API security mechanisms. The GSS-API stack with SPNEGO is shown in Figure 1. The client sends a prioritized list of supported authentication mechanisms to the server. The server then chooses the preferred authentication method based on the received list from the client. The client initiates `GSS_Init_sec_context()` as with an ordinary GSS-API security mechanism, but requests that SPNEGO is used as the underlying GSS-API mechanism ("mech_type"). The SPNEGO handshake between client and server is communicated by sending and receiving tokens. After the handshake, the client and server initiate and set up a security context (authentication) using the agreed GSS-API security mechanism.

IV. GSS-API SUPPORT FOR SIP

Instead of adding numerous different authentication methods to SIP based on different security requirements, it is desirable to keep the changes to the SIP standard to a minimum. The industry might also be reluctant to adopt immature and non-standardized security services, like different authentication methods. Adding support for the GSS-API requires only one small change to the SIP standard, and will open up for a wide range of different authentication methods. In the following subsections, we outline how to include support for the GSS-API into the SIP authentication to replace the original weak DAA.

A. SIP authentication using DAA

When a SIP client is authenticated to a server using DAA, the authentication handshake data is encapsulated in the `WWW-Authenticate` header from server to client, and the `Authorization` header from client to server. We reuse these headers for GSS-API support, and instead of encapsulate DAA data, we send the GSS-API tokens. An example of both DAA `Authorization` header and the new `Authorization` header with GSS-API data is depicted in Figure 3.

During the initialization of a security context it is necessary to identify the underlying security mechanism to be used. The caller initiating the context indicates at the start of the token the security (authentication) mechanism to be used. The security mechanism is denoted by a unique Object Identifier (OID). For example, the OID for the Kerberos V5 mechanism is `1.2.840.113554.1.2.2`. However, there is no way for the initiating peer to know which security mechanism the receiving peer supports. If an unsupported "mech_type" is requested, the authentication fails. The GSS-API standard resolves this by recommending to manually standardizing on a fixed "mech_type" within a domain. Since SIP addresses are designed to be global [28], and not confined to a local domain, a GSS-API *negotiation* mechanism is required. The SPNEGO is such a GSS-API negotiation mechanism.

B. SIP authentication using GSS-API and SPNEGO

When using GSS-API with the SPNEGO mechanism, the number of SIP messages between client and server during authentication needs to be increased. During a DAA authentication, the client sends a `REGISTER` message to the server. The server, upon receiving a `REGISTER`, challenges the client with a nonce. The client then generates a digest response, a hash value computed over several SIP header values, the nonce, and a shared secret. The client then re-sends the `REGISTER` message with the digest response embedded. The message flow of a SIP DAA handshake is shown in the first four messages depicted in Figure 4.

In the following paragraphs, the numbers in parentheses refer to the numbers in Figure 4. When a client comes online and registers itself to a "location service" (SIP server), it does so by sending a SIP `REGISTER` message (1). We define the token type in the variable `ttype`. In the following messages, the `ttype` is set to "context" indicating that these tokens are *context-level tokens*. The first message (1) does not contain any `Authorization` header. The server responds with an empty `WWW-Authenticate` header (3):

```
REGISTER SIP/2.0
WWW-Authenticate: GSSAPI ttype="context"
  token=""
```

```
1.  REGISTER sip:CompanyA SIP/2.0
2.  Via: SIP/2.0/UDP 192.168.1.102;branch=z9hG4bK32F3EC44EB23347BFB0D488459C69E4E
3.  From: Alice <sip:alice@CompanyA>;tag=1234648905
4.  To: Alice <sip:alice@CompanyA>
5.  Contact: "Alice" <sip:alice@192.168.1.102:5060>
6.  Call-ID: 2B6449C74C10D4F95006A6C034E79E8E@CompanyA
7.  CSeq: 19481 REGISTER
8.  User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
9.  Authorization: Digest
    username="alice",realm="asterisk",nonce="3b7a1395",response="ccbde1c3c129b3dcaa14a4d5e35
    519d7",uri="sip:CompanyA",algorithm=MD5
10. Max-Forwards: 70
11. Expires: 3600
12. Content-Length: 0
```

```
1.  REGISTER sip:CompanyA SIP/2.0
2.  Via: SIP/2.0/UDP 192.168.1.102;branch=z9hG4bK32F3EC44EB23347BFB0D488459C69E4E
3.  From: Alice <sip:alice@CompanyA>;tag=1234648905
4.  To: Alice <sip:alice@CompanyA>
5.  Contact: "Alice" <sip:alice@192.168.1.102:5060>
6.  Call-ID: 2B6449C74C10D4F95006A6C034E79E8E@CompanyA
7.  CSeq: 19481 REGISTER
8.  User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
9.  Authorization: GSSAPI ttype="context"
    token="0401000B06092A864886F712010202DACD139402AAF44350CDE32"
10. Max-Forwards: 70
11. Expires: 3600
12. Content-Length: 0
```

Figure 3: A SIP `REGISTER` message with the original DAA `Authorization` header to the left, and the same header carrying GSS-API data to the right.

The client then calls `GSS_Init_sec_context()` with SPNEGO as underlying GSS-API mechanism to negotiate a common authentication mechanism (4). The GSS-API "mech_type" is set to SPNEGOs OID `1.3.6.1.5.5.2`. The token data might be in binary format, depending on the security mechanism used. Since the SIP headers are in ASCII string format, the token data is base64 encoded:

```
SIP/2.0 401 Unauthorized
Authorization: GSSAPI ttype="context"
  token="0401000B06092A864886F7120..."
```

The server retrieves the GSS-API data, the token, and passes this to the SPNEGO GSS-API mechanism. In this first initial token, the client embeds authentication data for its first preferred authentication mechanism. This way, should the server accept the clients preferred mechanism, we avoid an extra SIP message round trip. If the client's preferred method was accepted by the server, the server passes the relevant authentication data to the selected authentication mechanism in a `401` SIP message (5). The selected authentication method continues to pass tokens between client and server as many times as necessary to complete the authentication (6-7-N) and establish a security context. Once the security context is established, it sends a `200 OK` SIP message (N+2). Should the server have some last GSS-API data to be communicated to the client to complete the security context, it can be carried in a `WWW-Authenticate` header embedded in the `200 OK` message:

```
SIP/2.0 200 OK
WWW-Authenticate: GSSAPI ttype="context"
  token="dd02c7c2232759874e1c20558701..."
```

If the client's preferred mechanism is not the server's most preferred mechanism, the server outputs a negotiation token and sends it to the client embedded in a new `401` SIP message (5). The client processes the received SIP message and passes the authentication data to the correct authentication mechanism. The GSS-API then continues as described in the previous paragraph.

V. Evolution strategy

Industrial uptake of new ideas and protocols requires an evolution strategy, especially in the telecom world where standards are used to serve more than four billion people. TCP/IP, IPv6, and UMTS are all examples of technologies where the evolutionary path was not clearly identified, and the technological uptake was significantly delayed.

Authentication for SIP-based services will not only be limited to mobile handsets and SIP authentication servers, it will be used for sensors and devices in the future Internet of Things (IoT). Thus the basic requirement of an advanced authentication scheme is modularity and flexibility: both of these are provided by the suggested approach. Support for the GSS-API will extend the SIP protocol with an improved security mechanism that offers more flexibility in different scenarios.

An industrial uptake will first be envisaged for mobile devices such as smartphones or tabs, where SIP clients can be provided with the new functionality. Discussions with industrial actors are ongoing to ensure the compatibility on the server side. After this initial phase, an extension of the approach is envisaged including access to services in a sensor network.

VI. Conclusion and future work

Since the only mandatory and widely deployed Digest Access Authentication method in SIP is weak, other more secure authentication methods are desired. In this paper, we have added support for GSS-API in SIP, as well as for the SPNEGO mechanism that is used to negotiate the preferred GSS-API security mechanism supported by both client and server. The required change to the SIP protocol has been kept to a minimum, and the authentication header from DAA has been reused to prevent adding additional SIP headers to the standard.

Different VoIP installations have different security requirements that may require different security services. We have shown that the use of the GSS-API provides SIP with a wide range of different authentication methods in a uniform and standardized way. Different authentication methods can be used depending on the different security requirements for each SIP installation. This adds to the flexibility of SIP, like adding a new authentication method, without requiring further changes to the SIP standard, once the GSS-API is supported.

In our earlier work, we have shown that the DAA is weak [14]. Therefore, we want to replace DAA with support for a better, more robust authentication scheme that authentication methods like GSS-API supports. This implies that we *replace*

 ISBN: 978-1-61208-123-6

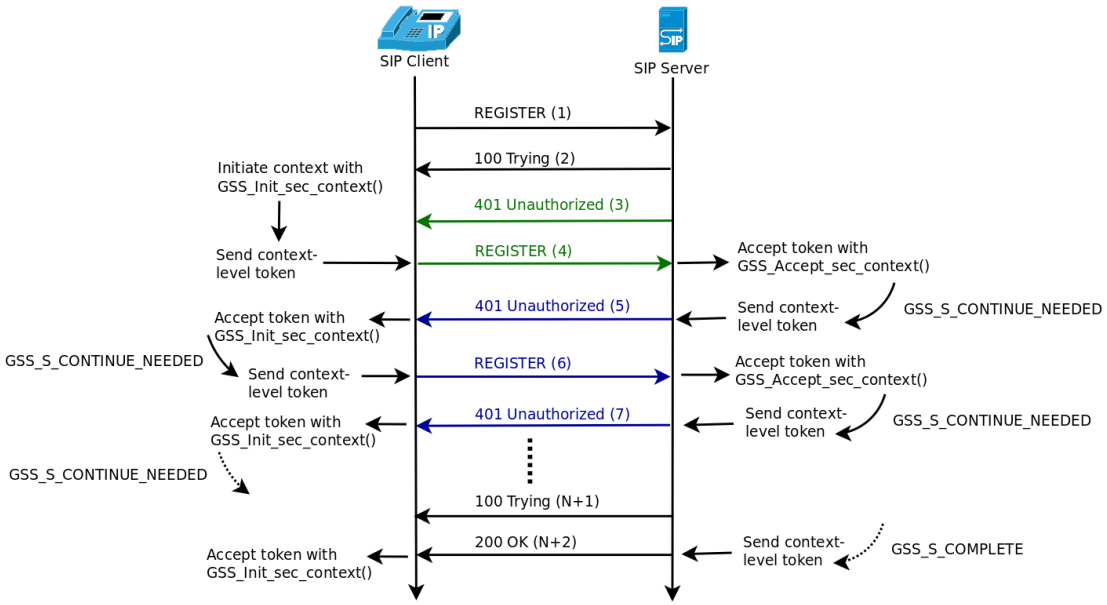

Figure 4: SIP `REGISTER` message flow with GSS-API security context establishment (authentication).

the original DAA header content with GSS-API data content. However, since the GSS-API only is an interface to underlying security mechanisms, the use of the GSS-API does not in itself provide any security service. Thus, the security of the GSS-API is no stronger than the weakest security mechanism acceptable to the client and server using the GSS-API. So, if the underlying GSS-API authentication mechanism does not protect relevant SIP headers, it might be as vulnerable to the attack shown previously with the DAA. We still need to examine what kind of SIP message integrity protection is offered by the different GSS-API authentication mechanisms.

The credential acquisition between peers must also be completed before initiating the GSS-API. Also, if the SP-NEGO negotiation is not integrity-protected, the negotiation is vulnerable to a man-in-the-middle "down-grade" attack. An attacker can intercept and modify the negotiation messages so that the least favorable authentication method is used.

Future work will look into different GSS-API security mechanisms and their implications for SIP, including overhead evaluation benchmark. Implementing a proof of concept with GSS-API support for SIP is also desired. We plan to co-operate with the IETF and the "kitten" WG to further elaborate the GSS-API for SIP. Further challenges are the Simple Authentication and Security Layer (SASL) [29] for SIP, and a comparison of SASL with GSS-API, as well as the support for using GSS-API mechanisms within SASL [30].

Acknowledgment

This research is funded by the EUX2010SEC project in the VERDIKT framework of the Norwegian Research Council (Norges Forskningsråd, project 180054). The authors would like to thank Trenton Schulz and the anonymous reviewers for valuable comments on earlier drafts of this paper.

References

[1] "Det norske markedet for elektroniske kommunikasjonstjenester 2009 (The Norwegian market for electronic communication services 2009)," Post- og teletilsynet (The Norwegian Post and Telecommunications Authority), 2010. [Online]. Available: http://www.npt.no/ikbViewer/Content/119027/Ekomrapport_2009_.pdf 10. Jan 2011

[2] Telecommunication Development Sector (ITU-D), "The world in 2010," ITU-T ICT facts and figures, 2010.

[3] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261 (Proposed Standard), Internet Engineering Task Force, Jun. 2002, updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt 10. Jan 2011

[4] H. Sinnreich and A. B. Johnston, Internet communications using SIP: Delivering VoIP and multimedia services with Session Initiation Protocol, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., August 2006.

[5] D. Kuhn, "Sources of failure in the public switched telephone network," Computer, vol. 30, pp. 31–36, 1997.

[6] VoIPSA, "VoIP security and privacy threat taxonomy," Public Realease 1.0, Oct. 2005. [Online]. Available: http://voipsa.org/Activities/VOIPSA_Threat_Taxonomy_0.1.pdf 1. Nov 2011

[7] A. D. Keromytis, "Voice over IP: Risks, Threats and Vulnerabilities," in Proceedings of the Cyber Infrastructure Protection (CIP) Conference, New York, June 2009.

[8] H. Dwivedi, Hacking VoIP: Protocols, Attacks, and Countermeasures, 1st ed. No Starch Press, Mar. 2009.

[9] A. D. Keromytis, "Voice-over-IP security: Research and practice," IEEE Security & Privacy Magazine, vol. 8, no. 2, pp. 76–78, 2010.

[10] "Research project: EUX2010SEC – Enterprise Unified Exchange Security"." [Online]. Available: http://www.nr.no/pages/dart/project_flyer_eux2010sec 1. Nov 2011

[11] L. Fritsch, A.-K. Groven, L. Strand, W. Leister, and A. M. Hagalisletto, "A Holistic Approach to Open Source VoIP Security: Results from the EUX2010SEC Project," International Journal on Advances in Security, no. 2&3, pp. 129–141, 2009.

[12] L. Strand, "VoIP lab as a research tool in the EUX2010SEC project," Norwegian Computing Center, Department of Applied Research in Information Technology, Tech. Rep. DART/08/10, April 2010.

[13] A. M. Hagalisletto and L. Strand, "Formal modeling of authentication in SIP registration," in Second International Conference on Emerging

 ISBN: 978-1-61208-123-6

Security Information, Systems and Technologies SECURWARE '08. IEEE Computer Society, August 2008, pp. 16–21.

[14] L. Strand and W. Leister, "Improving SIP authentication," in Accepted for publication in The Tenth International Conference on Networks (ICN 2011), Jan 2011.

[15] A. M. Hagalisletto and L. Strand, "Designing attacks on sip call set-up," International Journal of Applied Cryptography, vol. 2, no. 1, pp. 13–22(10), July 2010. [Online]. Available: http://inderscience.metapress.com/link.asp?id=jh437k6747064307 10. Jan 2011

[16] A. D. Keromytis, "A Survey of Voice Over IP Security Research," in Proceeding of the 5th International Conference on Information Systems Security (ICISS), December 2009, pp. 1 – 17.

[17] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617 (Draft Standard), Internet Engineering Task Force, Jun. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2617.txt 10. Jan 2011

[18] J. Peterson, "S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP)," RFC 3853 (Proposed Standard), Internet Engineering Task Force, Jul. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3853.txt 10. Jan 2011

[19] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend, and H. Schulzrinne, SIP Security. WileyBlackwell, Mar. 2009.

[20] C. Jennings, J. Peterson, and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks," RFC 3325 (Informational), Internet Engineering Task Force, Nov. 2002, updated by RFC 5876. [Online]. Available: http://www.ietf.org/rfc/rfc3325.txt 10. Jan 2011

[21] J. Peterson and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)," RFC 4474 (Proposed Standard), Internet Engineering Task Force, Aug. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4474.txt 10. Jan 2011

[22] J. Linn, "Generic Security Service Application Program Interface Version 2, Update 1," RFC 2743 (Proposed Standard), Internet Engineering Task Force, Jan. 2000, updated by RFC 5554. [Online]. Available: http://www.ietf.org/rfc/rfc2743.txt 10. Jan 2011

[23] D. Todorov, Mechanics of User Identification and Authentication: Fundamentals of Identity Management, 1st ed. Auerbach Publication, Jun. 2007.

[24] "IETF Common Authentication Technology Next Generation (kitten)." [Online]. Available: http://datatracker.ietf.org/wg/kitten/charter/ 1. Nov 2011

[25] L. Zhu, K. Jaganathan, and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2," RFC 4121 (Proposed Standard), Internet Engineering Task Force, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4121.txt 10. Jan 2011

[26] C. Adams, "The Simple Public-Key GSS-API Mechanism (SPKM)," RFC 2025 (Proposed Standard), Internet Engineering Task Force, Oct. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc2025.txt 10. Jan 2011

[27] L. Zhu, P. Leach, K. Jaganathan, and W. Ingersoll, "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism," RFC 4178 (Proposed Standard), Internet Engineering Task Force, Oct. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4178.txt 10. Jan 2011

[28] L. Strand and W. Leister, "A Survey of SIP Peering," in NATO ASI - Architects of secure Networks (ASIGE10), May 2010.

[29] A. Melnikov and K. Zeilenga, "Simple Authentication and Security Layer (SASL)," RFC 4422 (Proposed Standard), Internet Engineering Task Force, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4422.txt 10. Jan 2011

[30] S. Josefsson and N. Williams, "Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family," RFC 5801 (Proposed Standard), Internet Engineering Task Force, Jul. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5801.txt 10. Jan 2011

 ISBN: 978-1-61208-123-6