Abusing SIP authentication

Humberto Abdelnur Tigran Avanesov Michael Rusinowitch Radu State

INRIA, Nancy - Grand Est
Campus Scientifique - BP 239 - 54506
Vandoeuvre-lès-Nancy Cedex, France
{Humberto.Abdelnur, Tigran.Avanesov, Michael.Rusinowitch, Radu.State}@loria.fr

Abstract

The recent and massive deployment of Voice over IP infrastructures had raised the importance of the VoIP security and more precisely of the underlying signalisation protocol SIP. In this paper, we will present a new attack against the authentication mechanism of SIP. This attack allows to perform toll fraud and call hijacking. We will detail the formal specification method that allowed to detect this vulnerability, highlight a simple usage case and propose a mitigation technique.

Keywords: *Security threat, VoIP, SIP protocol, authentication, formal validation, AVISPA.*

1 Introduction

SIP is the IETF endorsed signaling protocol for VoIP. The developers of SIP leveraged well proven design concepts from HTTP to build a robust and multi-feature signaling protocol. The advance of highly dynamical services deployed over multimedia enabled networks and end user equipment had to be matched by an appropriate signaling protocol. At the basics, SIP allows to create, maintain and tear down a media session. The media session is represented by an RTP encoded audio/video data. The specific characteristics of this RTP flow are negotiated by SIP. In the simplest case, the call establishment with SIP has to be able to let the two communicating partners send RTP data between their two locations. However, in the more complex case, some additional features have to be supported. Call forwarding is the simplest feature that has to be supported. Renegotiating a media stream parameters (RTP) is also a minimum. For instance, in case of network congestion, another codec can be used. In order to support these features the so-called re-INVITE operation has to be used. The re-INVITE is issued during an already existing session and in order to avoid a call-hijacking attack, the receiver is allowed to challenge the sender to authenticate. Ironically, it is this security feature that can be abused to bypass the authentication mechanisms used in SIP network. We will show in this paper why the re-INVITE operation is a major threat to any SIP network and how a simple grandmaster attack is possible due to it.

To examine SIP protocol security we have used AVISPA tool. AVISPA is a push-button tool for the Automated Validation of Internet Security Protocols [1]. It provides a modular and expressive formal language, HLPSL (The High-Level Protocol Specification Language) for specifying protocols and their security properties, and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques. Experimental results, carried out on a large library of Internet security protocols, indicate that the AVISPA Tool is a state-of-the-art tool for automatic verification of security protocols.

2 SIP Vulnerability

When SIP is deployed without any underling cryptographic protection mechanism, the typical man in the middle and impersonation attacks between a caller and its proxy, (see Figure 1) are straightforward. However, these must are constrained by some important factors . Firstly, the attacker willing to impersonate the user has to be in the middle of the session path and be able to manipulate the session traffic. Secondly, the attacker cannot trigger the user to make such a call at a specific time. Finally, the attacker is restricted to use the generated response just to call the entity for which the user directed the call. In other words, the attacker is not able to call an entity of its choice.

During a testing process [2] carried out by us, we have discovered a scenario in which the user is reachable by the attacker and the latest can trigger the former

978-0-7695-3324-7/08 $25.00 © 2008 IEEE
DOI 10.1109/IAS.2008.29

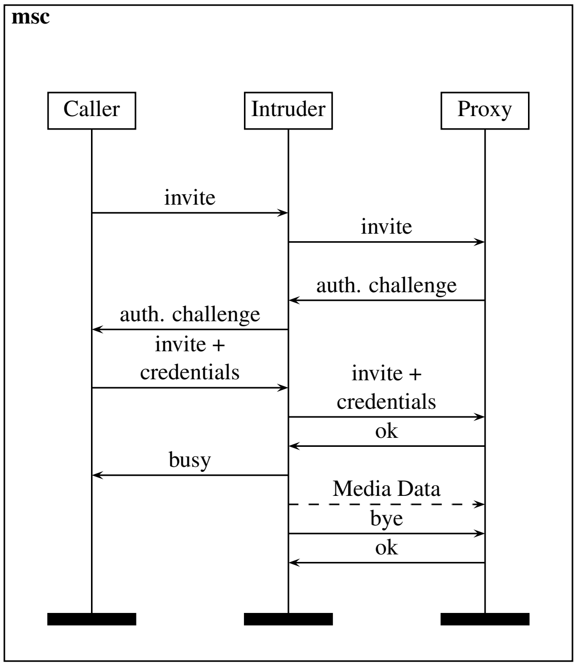

Figure 1. Authentication Attack

to generate an INVITE + credentials[1] (directed to any target destination). This allows an attacker to impersonate the user at the Proxy for any call. Therefore, the attacker can bypass the previous restrictions and make this attack a real security threat.

The synopsis is as follow: an attacker will issue a call directly to the victim, the victim answers and later on, puts the attacker on hold (transfers him to any other place or uses any other method which requires a re-INVITE). Once the attacker receives the re-INVITE specifying the ”On hold”, he will immediately request the victim to authenticate. This last authentication may be used by the attacker to impersonate the victim at its own proxy. Section 3 formalizes and describes in detail this attack.

Note, that to perform this attack, there are two headers in the INVITE message that are essential. The **Contact** header has to have the destination call that the attacker wants to call, because, as specified by SIP [7], this information will be used to generate the message by the user entity. The **Record-Route** header specifies that all outgoing messages from the user entity go directly to that entity.

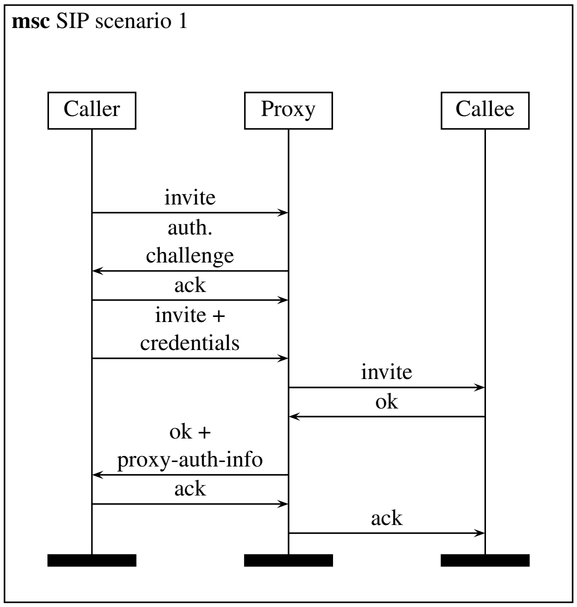

Figure 2. SIP scenario 1

3 SIP and HLPSL Specification

HLPSL is an expressive, modular, role-based, formal language that allows for the specification of control-flow patterns, data-structures, alternative intruder models, complex security properties, as well as different cryptographic primitives and their algebraic properties. These features make HLPSL well suited for specifying modern, industrial-scale protocols. For instance it has been applied to ZRTP flaws discovery in [5].

3.1 SIP scenarios

We have formalized two typical scenarios of SIP protocol. In both scenarios *Caller* wants to call *Callee*. The first one is shown in Figure 2: *Caller* is registered on *Proxy* and wants to call via his proxy, then *Proxy* requests an authentication of *Caller*.

The second case (shown in Figure 3): *Callee* is available only through *Proxy* where *Callee* is registered. *Callee* puts conversation on hold by sending `invite` during the conversation. Having received this invite, *Proxy* can demand an authentication of *Callee*.

3.2 SIP in HLPSL

In order to describe the protocol we should specify the actions of each kind of participant, i.e. the *basic roles*. To describe both above scenarios in HLPSL we introduce three basic roles: `caller`, `callee`, and

[1] `http://voipsa.org/pipermail/voipsec_voipsa.org/2007-November/002475.html`

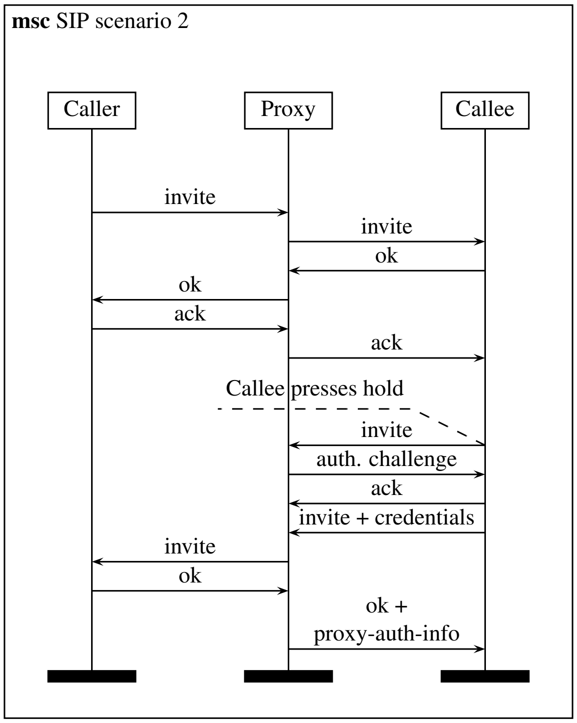

Figure 3. SIP scenario 2

proxy. Each role is obtained by merging corresponding roles from the two scenarios defined in 3.1 (this means *Caller* from Scenario 1 and *Caller* from Scenario 2 became one role and similarly for *Callee* and *Proxy*).

We present now the declaration of basic roles and their parameters in HLPSL:

```
role caller(A,B,P : agent, Apasswd :
    text,
    SND, RCV : channel (dy))
    ...
role proxy(P : agent,
    Keyring :  (agent.text) set,
    Realm :  text,
    SND, RCV : channel (dy))
    ...
role callee(B,P : agent, Bpasswd :
    text,
    SND, RCV : channel (dy))
    ...
```

Here we have: A, B, P - agents playing roles caller, callee and proxy respectively; Apasswd, Bpasswd - passwords of agent A and B respectively; RCV, SND are channels for sending and receiving messages and Keyring — set of pairs <username, password>.

In HLPSL variable names start with capital letters; constants, keywords and types start with lower-case letters. The primed variable notation (e.g.: X') indicates that a new value is assigned to the variable.

To execute several roles in parallel, the composition is presented in HLPSL. We defined two sessions (compositions): out_session (for a call by Scenario 1) and in_session (for a call by Scenario 2).

```
role in_session(A, B, P : agent,
    Keyring:  (agent.text) set, Bpasswd :
    text)
    ...
    composition
    caller(A,B,P, null, SA,RA) /\
    callee(B,P, Bpasswd, SB,RB) /\
    proxy(P, Keyring, realm, SP,RP)
end role
role out_session(A, B, P : agent,
    Keyring:  (agent.text) set, Apasswd :
    text)
    ...
    composition
    caller(A,B, P, Apasswd, SA,RA) /\
    callee(B,P, null, SB,RB) /\
    proxy( P, Keyring, realm, SP,RP)
end role
```

Symbol /\ denotes here a parallel execution.

Now we show the *transitions* of roles that are responsible for the authentication part. A transition — is a rule that can fire if the left-hand side[2] is satisfied. For role **Caller**:

```
getAuth.
    State=10 /\
    RCV(A.B.CallID.auth.
    Algorithm'.Realm'.Nonce') =|>
    State':=20 /\ SND(A.B.CallID.ack)
```

This transition is called getAuth and its left-hand side means: *"if value of variable State equals to 10 and we receive on channel RCV message equal to concatenation of values of variables A, B, CallID, constant auth and three more values that are to be assigned to variables Algorithm, Realm and Nonce"*; the right-hand side means: *"then assign 20 to the variable State and send via channel SND concatenation of values stored in A,B,CallID and constant ack"*. In other words, getAuth stands for getting hash-function name (Algorithm), realm value (Realm) and nonce (Nonce), and as a response *Caller* sends ack[3].

```
sndResponce.
    State=20 /\ RCV(start) =|> State':=30
    /\ Cnonce':=new() /\ MdC':=Algorithm(
    Algorithm(A.Realm.Apasswd).
    Nonce.Cnonce'.Algorithm(invite.B))
    /\ SND(A.invite.B.CallID.Algorithm.
    Realm.Nonce.MdC'.Cnonce') /\
```

[2] before "=|>"

[3] Notice: all the messages in our specification constantly contain *from*, *to* and *CallID* part. We omit mentions about this part

```
    witness(A,P, client_md,MdC')
```

In this transition caller emits an authentication response. caller generates a new nonce Cnonce and computes a message digest value MdC (exactly like shown in HLPSL specification above). Then caller sends this value together with others. The last line is an authentication event witness. Here it should be read as follows: *"agent A authenticates to agent P that value MdC'*[4] *is really generated by A for P"*. The constant client_md is used to identify unambiguously a pair witness-request. The authentication-related event *request* will be described later.

```
getProxyAuthInfo.
    State=30 /\ RCV (A.B.CallID.MdP'.ok)
    /\ MdP'=Algorithm(Algorithm(
    A.Realm.Apasswd).
    Nonce.Cnonce.Algorithm(B)) =|>
    State':=40 /\ SND (A.B.CallID.ack)
    /\ request(A,P, proxy_md, MdP')
```

The getProxyAuthInfo transition checks authentication credentials received from the proxy. If it is correct — sends ack signal. Here we have the second authentication-related event request; it means *"A accepts the value MdP' and relies on the guarantee that agent P exists and agrees with A on this value"*

Now we show some **Proxy**'s transitions responsible for the authentication procedure with Caller:

```
getInviteSndAuth.
    State=11 /\ RCV(X'.invite.Y'.CallID')
    /\ in(X'.PasswdX', Keyring) =|>
    State':=21 /\ Nonce':=new()
    /\ SND(X'.Y'.CallID'.
    auth.md5.Realm.Nonce')
```

Transition getInviteSndAuth. At first we receive information about who wants to call (X') and to whom (Y'). The next condition tells that X' should be registered in this proxy, i.e. the pair X.password_of_X should belong to the keyring set. This is achieved by in(X'.PasswdX', Keyring) statement. X' is already defined, so if there is a pair X'.something in Keyring, then the value of "something" is assigned to variable PasswdX'.

```
checkAuth.
    State=31 /\ RCV(X.invite.Y.CallID.md5.
    Realm.Nonce.MdC'.Cnonce') /\
    MdC'=md5( md5(X.Realm.PasswdX).
    Nonce.Cnonce'.md5(invite.Y)) =|>
    State':=41 /\ SND(X.invite.Y.CallID)
    /\ request(P, X, client_md,MdC')
```

Transition checkAuth checks if received authentication credentials are right and if they are, sends invite to callee.

```
sndProxyAuthInfo.
    State=41 /\ RCV(X.Y.CallID.ok)
    =|> State':=51 /\
    MdP':=md5(md5(X.Realm.PasswdX)
    .Nonce.Cnonce.md5(Y)) /\ SND
    (X.Y.CallID.MdP'.ok) /\ witness(P,
    X, proxy_md,MdP')
```

Here we get ok from callee and send proxy authentication information to caller.

We skip the HLPSL-specification of the role callee and the rest of proxy's authentication related specification as it is very similar to the one described above.

There is a special role environment, that it is a top-level one (it is "called" from HLPSL file) where we declare agents and other constants, all the sessions to be executed simultaneously and where we define an initial intruder knowledge set using intruder_knowledge token. Here we initially let the intruder know the following constants: a, b, p, c, invite, try, ringing, ok, ack, auth, in other words he knows all agents names and all SIP methods.

To make AVISPA tool search for an attack, one should introduce a *goals* section to define security goals:

```
goal
    authentication_on proxy_md
    authentication_on client_md
    ...
end goal
```

For example, the first line is a command that makes AVISPA tool look for an authentication attack for the witness-request pair defined by constant proxy_md.

Now we can start AVISPA Tool. An attack is detected when the following role composition is presented in the top-role environment:

```
proxy(p, Keyring, realm, SP,RP) /\
callee(a,p, alice_passwd, SB,RB)
```

And Keyring contains the only pair a.alice_passwd. But we can also use two defined compositions (which include necessary roles):

```
out_session(a,c,p,Keyring,alice_passwd)
/\ in_session(b,a,p,Keyring,alice_passwd)
```

3.3 MSC of attack

When running AVISPA tool on our HLPSL specification of SIP we get the message: "UNSAFE". The tool automatically builds and displays the attack trace in Figure 4.

Here x236, x265 and x237 are variables that can take any values. Notation Cnonce(7) indicates an instance of a Cnonce variable (the parameter is used to distinguish one instance of the variable from another).

We can see that at first the intruder impersonates caller a when speaking to proxy. After getting an authentication challenge from proxy and sending ack, the

[4] in fact, this is a message digest value

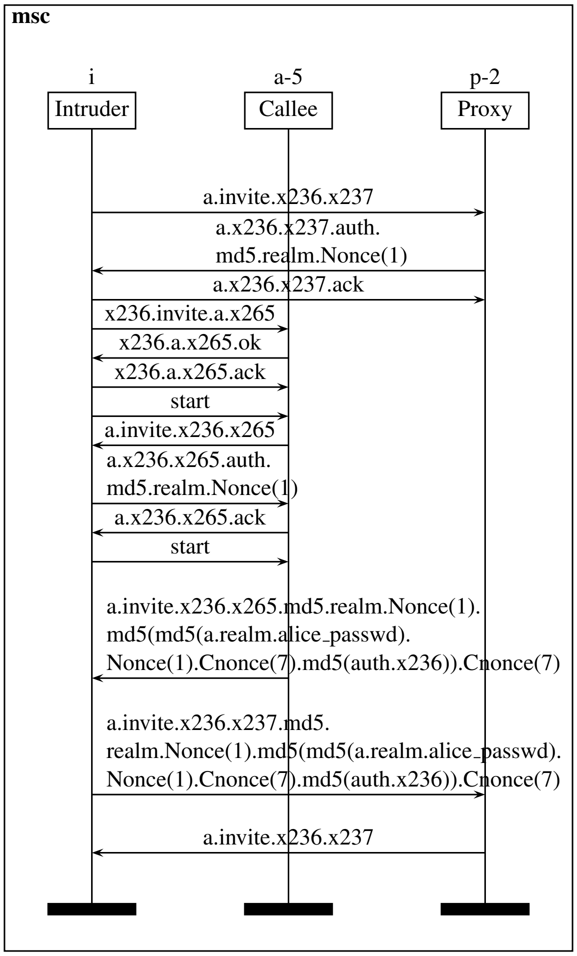

Figure 4. Attack trace

intruder starts a protocol execution with a. Here the intruder impersonates a proxy for callee a. Once the intruder gets the necessary authentication response from callee a, he reuses it (the only change is CallID value) to answer proxy's challenge he got at the beginning. Here we get a security violation on authentication, as an appropriate witness event was not generated to match the request event that is produced.

4 Mitigation

Authentication challenges in SIP are computed using pieces of information extracted from the authenticate message plus the username and shared secret. In the simplest case the authentication response is computed by:

```
A1   = username ":" realm ":" passwd
A2   = Method ":" Digest-URI
resp = MD5(MD5(A1) ":" nonce ":" MD5(A2))
```

Table 1. Benchmark

`proxy(p, Keyring, realm, SP,RP)`
`/\bob(a,p, alice_passwd, SB,RB)`

Keyring	**sip**	**sip'**
a.caller_passwd	0.07s, unsafe	0.06s, safe
a.caller_passwd, c.charley_passwd, i.i_passwd	0.07s, unsafe	0.07s, safe

where **resp** is the actual authentication response. Thus, the computed authentication responses will be rejected if the method of the message is different than the method used to generate the response.

However, the described attack abuses that restriction due to the fact that SIP defines an INVITE method which can be used in different contexts (i.e. for initiation of a session and renegotiation). Therefore, the variable $A2$ is the same in both contexts. If different methods names are used for those contexts, then the generated authentication response cannot be used for such an attack.

We propose a mitigation that consists in defining the re-INVITE method as a proper method with a new name: RE-INVITE. Note that computed authentication for such message will use the RE-INVITE method in the variable $A2$ rather than INVITE. Thus, it will generate an authentication token useful only for re-INVITEs messages. Our proposed solution is simple and it should not require to much modifications in the overall protocol.

Validation The proposed patch changes the scenario 2 (see fig. 3) such that every invite appearing after comment "Callee presses hold" is changed for reinvite. So it is not difficult to change HLPSL specification taking into account the proposed patch.

We ran AVISPA tool over "patched" HLPSL specification. For the patched version we got "safe" over the sessions which were unsafe for original version. Tables 1 and 2 represent a running time in seconds of AVISPA-tool with cl-atse backend and safety result depending on sessions and value of Keyring variable. Column **sip** represents the original version of SIP and **sip'** — the patched one.

5 Related Work

The comprehensive overview on VoIP security is the reference [11] addressing the operational and deployment aspects of VoIP security. The security mechanisms deployed in SIP are well described in [6] without covering the formal aspect of the security architecture.

Table 2. Benchmark 2

```
out_session(a,c,p,Keyring,alice_passwd)
/\in_session(b,a,p,Keyring,alice_passwd)
```

Keyring	sip	sip'
a.caller_passwd	0.39s, unsafe	11.81s, safe
a.caller_passwd, c.charley_passwd, i.i_passwd	0.40s, unsafe	13785.07s, safe

Many works have been dedicated to analysis and testing of VoIP protocols, but dealing either with the PSTN interconnection as in [9], or [8]. Most of the performed work has addressed the prevention of SPAM over Internet Telephony (SPIT) attacks [3] as well as mitigating denial of service ones (DOS) [10]. Very few of them did address the cryptographic analysis of the protocol itself. Among the very few which did, most of them are based on human-analysis of the protocol. As of today, very few works address the formal specification and analysis of security properties. Among them — a thorough study of ZRTP (VoIP media transport layer protocol) using AVISPA tool [5], has allowed to find a new authentication attack. There are two families of potential attacks that can be performed against SIP. The first class of attacks is possible when no cryptographic protections are used in the SIP deployment. For instance, a DoS attack on SIP protocol has also been exhibited using a Petri nets modelling in [12] — using faked BYE message and showed an established conversation can be turned down prematurely [4]. Until now, the authentication and authorization mechanisms in SIP were considered sound, and such that only denial of service and brute force attacks were possible. We have disclosed some attacks against specific implementations (*CVE-2007-5468, CVE-2007-5469*), where cryptographic tokens could be reused or even fixed, but these were due to software implementation flaws and not really SIP specification level vulnerabilities. Our paper is the first to show a structural flaw in the SIP authentication mechanism itself due to the feature interaction in SIP.

6 Conclusion

We have presented in this paper a new attack against the SIP authentication mechanisms. This attack is extremely dangerous since SIP is worldwide deployed and no solution to mitigate this attack exists. We have confirmed this vulnerability using AVISPA tool and shown that extending SIP with one more operation can mitigate this attack. The solution has been automatically validated by the tool. It is a follow-up activity to fully specify SIP and completely analyze its behavior, but many scalability issues must be solved to achieve this task.

Acknowledgements The work presented in this paper was partially supported by the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" (www.avantssar.eu).

References

[1] Avispa project. http://www.avispa-project.org.

[2] H. Abdelnur, R. State, and O. Festor. "KiF: A stateful SIP Fuzzer". In *Proceedings of Principles, Systems and Applications of IP Telecommunications, IPTComm*, pages 47–56, New-York, NY, USA, JUL 2007. ACM Press.

[3] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, K. Ehlert, and D. Sisalem. Survey of security vulnerabilities in session initiation protocol. *Communications Surveys & Tutorials, IEEE*, 8(3):68–81, 3rd. Qtr. 2006.

[4] D. Geneiatakis and C. Lambrinoudakis. An ontology description for sip security flaws. *Comput. Commun.*, 30(6):1367–1374, 2007.

[5] P. Gupta and V. Shmatikov. "Security Analysis of Voice-over-IP Protocols". In *20th IEEE Computer Security Foundations Symposium (CSF)*, pages 49–63, Venice, Italy, JUL 2007. IEEE Computer Society, 2007.

[6] A. B. Johnston and D. M. Piscitello. *Understanding Voice over Ip Security (Artech House Telecommunications Library)*. Artech House, Inc., Norwood, MA, USA, 2006.

[7] H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. "SIP: Session Initiation Protocol". http://www.ietf.org/rfc/rfc3261.txt, June 2002.

[8] H. Sengar, R. Dantu, and D. Wijesekera. Securing voip and pstn from integrated signaling network vulnerabilities. *VoIP Management and Security, 2006. 1st IEEE Workshop on*, pages 1–7, 3 April 2006.

[9] H. Sengar, R. Dantu, D. Wijesekera, and S. Jajodia. "SS7 over IP: Signaling internetworking vulnerabilities". In *IEEE Network, Vol. 20, No. 6*, pages 32–41, November 2006.

[10] D. Sisalem, J. Kuthan, and S. Ehlert. Denial of service attacks targeting a sip voip infrastructure: attack scenarios and prevention mechanisms. *Network, IEEE*, 20(5):26–31, Sept.-Oct. 2006.

[11] P. Thermos and A. Takanen. *Securing VoIP Networks: Threats, Vulnerabilities, and Countermeasures*. Addison-Wesley Professional, 2007.

[12] H. Wan, G. Su, and H. Ma. "SIP for Mobile Networks and Security Model". In *International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1809–1812, Shanghai, China, September 2007. IEEE Computer Society, 2007.