Abusing SIP authentication

Humberto Abdelnur[1], Tigran Avanesov[1], Michael Rusinowitch[1] and Radu State[2]

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

[2]University of Luxembourg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg-Kirchberg
radu.state@uni.lu

***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

The Session Initiation Protocol (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 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.

We address in this paper one major vulnerability existing in the SIP specifications. This vulnerability can be used for massive toll frauds or caller impersonations. With all VoIP operators being vulnerable, the potential impact can be far reaching and important. We have developed a working prototype for practical assessment and formally proved it using automated constraint based verification. We have leveraged the AVIPSA tool for this purpose.

To analyze the SIP protocol security we have used AVISPA, which is a state-of-the-art tool for automatic verification of security protocols. 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.

Our paper is structured as follows: Section 2 gives a short introduction to SIP and VoIP. The modeling of a subset of some functionalities of SIP is presented in Section 3. We propose a mitigation in Section 4 and a validation approach in Section 5 and discuss relevant works in Section 6. We conclude the paper in Section 7 with conclusions and pointers to future works.

2. SIP Vulnerability

SIP is located at the application layer of the TCP/IP model [3] and it has been designed to be independent of the underlying transport layers. It is a decentralized protocol, where the intelligence is distributed through the entities that the network is composed of. Thus, different components have been defined in the SIP architecture playing different roles in a deployed network:

- *User Agents (UA)*: they are the end-user devices in a SIP network (e.g. software or hardware phones).
- *SIP Location Sever*: this server is used to store the current location addresses, features and other preferences of all the users from the domain. However, UAs do no directly interact with it but indirectly by means of proxies, redirect or registrar servers.
- *SIP Proxy Server*: used to forward requests on behalf of other SIP entities. It can not initiate a request by itself, but can offer additional services like for instance security, authentication and authorization.
- *SIP Registrar Server*: receives the request from a UA which wants to register in its SIP domain.
- *SIP Redirect Server*: used to indicate the location where the initial request has to be forwarded. It is mainly useful for mobility purposes.

In fact, the servers provide a vast range of extra functionalities just to facilitate the establishment of a session

between two SIP UAs. Only session signaling is considered by SIP, however it can be used in combination with other protocols to build a complete multimedia session.

SIP is a Request/Response protocol. In a normal session many requests can be generated and for each one several provisional responses are possible and only one final response is required.

Each request message of SIP, except CANCEL and ACK, can be challenged for authentication. Thus, the VoIP services can be protected against threats and attacks like impersonation, session teardown, fraud and others [13]. Authentication in SIP has also been leveraged on the design of HTTP authentication.

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, to be mounted such attacks 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.

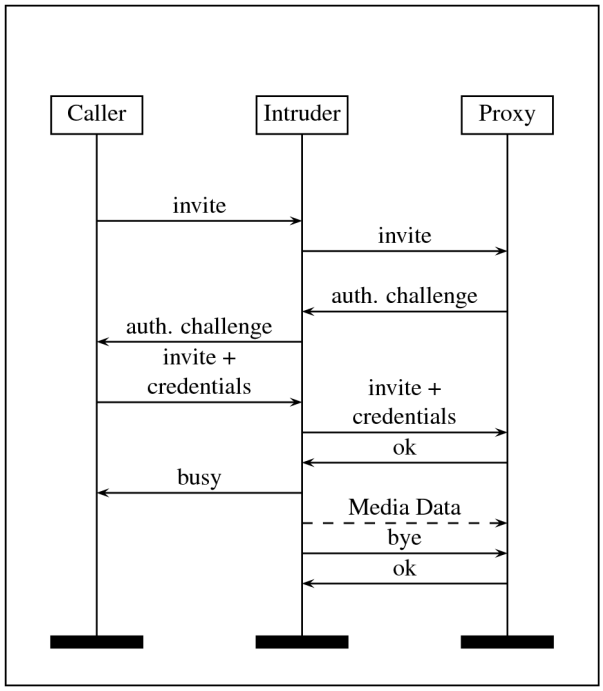

Figure 1. Authentication Attack

Secondly, the attacker cannot trigger the user to make such a call at a specific time: he has to wait for such an opportunity.

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.

However, if the nonces are not correctly checked to be one time used, the third constraint could be relaxed since

> *"... in SIP, a server MAY check that the Request-URI in the Authorization header field value corresponds to a user for whom the server is willing to accept forwarded or direct requests, but it is not necessarily a failure if the two fields are not equivalent."* [9]

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 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 follows: 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 provide the destination that the attacker wants to call to, because, as specified by SIP [9] this information will be used to generate the message by the user agent.

The **Record-Route** header specifies that all outgoing messages from the user entity go directly to that entity.

3. SIP in AVISPA

To analyze the security properties of SIP protocol, we made use of AVISPA, a state-of-art tool for cryptographic protocol verification. As an intruder model, it relies on the well-known Dolev-Yao model [4]. This means that all the communication channels between agents (entities) are considered to be under the control of the intruder: the messages sent by an agent get immediately known to the intruder (this is a consequence of the eavesdropping capability of the intruder). Moreover, the intruder is able to drop the message from a channel and even forge another message and put it in any communication channel, i.e. send it to any agent (this is an active capability of the Dolev-Yao intruder). The deductive power of the intruder is quite natural: he can decompose a message (get every element from a list of values and decrypt encrypted messages under condition he possess the required keys) and compose messages (create a list from elements and produce an encryption with any key he knows). This model seems to give too much control to the intruder on the entire network. But the model is valid since it is possible to forge a Record-Route header, an intruder may force an agent to send his messages first to a compromised router (or to the intruder itself) and this will give an opportunity to realize an attack.

Protocols to be studied by the AVISPA tool have to be specified in HLPSL (standing for High Level Protocol Specification Language).

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

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

cryptographic primitives and their algebraic properties.

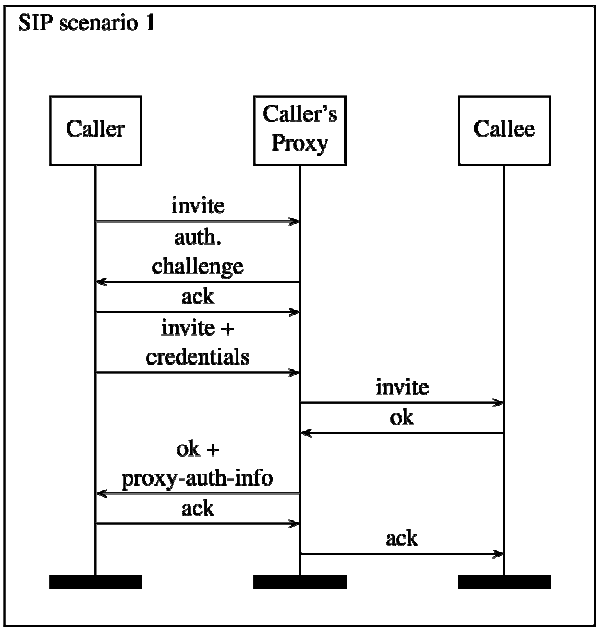

Figure 2. SIP scenario 1

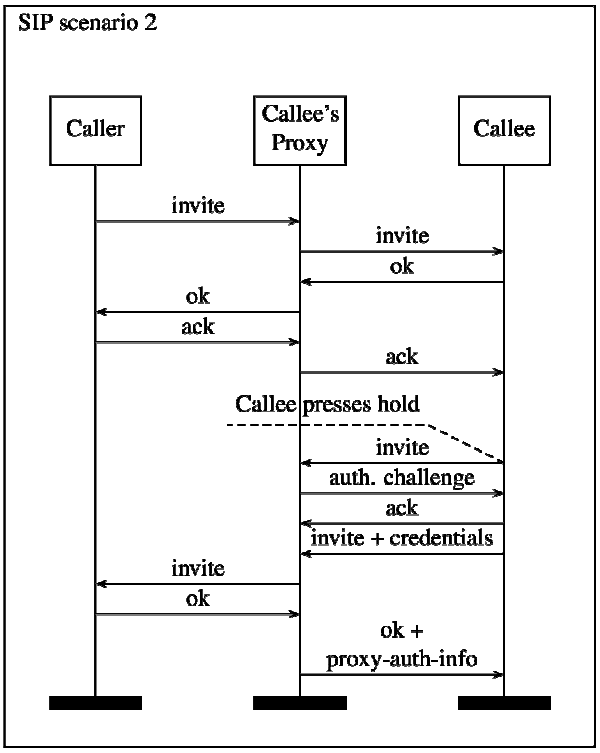

Figure 3. SIP scenario 2

These features make HLPSL well suited for specifying modern, industrial-scale protocols. For instance it has been applied to ZRTP flaws discovery in [7].

HLPSL is based on temporal logic. We specify a protocol by describing initial states and changes in states. A state is defined by an assignment of values to all the state variables. A change in states is describes by a transition predicate that relates the values of variable in current state with the values of variable in next or future state. We refer to the variable in next state as primed variables: for a variable `X`, `X` refers to the value in current state and `X'` to the value in the next state.

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, and then *Proxy* requests an authentication of *Caller*. The authentication procedure is based on a shared between *Proxy* and *Caller* secret, generated during a registration phase which is out of scope of this article. If simplify, *Proxy* sends a nonce and *Caller* should return hashed value of that nonce concatenated with the shared secret and some other data. Meanwhile, *Caller* invokes a reciprocal authentication. With his authentication response, he sends another nonce to *Proxy* and expects a correct answer (`proxy-auth-info`) by the similar algorithm.

The second case (shown in Figure 3): *Callee* is available only through *Proxy* where he is registered: any message sent to or sent by *Callee* comes first to the proxy. *Caller* establishes a conversation with *Callee*, and *Callee* puts it on hold by sending `invite` to *Caller* during the conversation. Having received this invite, *Proxy* can demand an authentication of *Callee*. The authentication steps here are similar to ones of scenario 1: *Calee* and his *Proxy* share a secret. If the response of *Callee* is valid, *Proxy* passes `invite` to *Caller*. In fact, the identity of authentication schemes in different scenarios admits an attack.

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*. Roles are independent processes: they have a name, receive information by parameters and contain local declarations. Basic roles are played by an agent whose name is received as parameter. The actions of a basic role are *transitions*, describing changes in their state depending on events or facts.

To describe both above scenarios in HLPSL we introduce three basic roles: `caller`, `callee`, and `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 which can execute both scenarios depending on received message, and similarly for `Callee` and `Proxy`).

We present now the declaration of basic roles and their (typed) 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 >;
- agent, text, channel(dy), (agent.text) set are the types.

In HLPSL variable names start with capital letters; constants, keywords and types start with lower-case letters.

3.2.1 *Composed roles*

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). This was done to simulate an execution of the protocol, i.e. to group several roles and share some parameters to execute the corresponding scenario.

```
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 (as more important one). A transition — is a rule that can be fired if the left-hand side (that is the part before "=|>") is satisfied.

3.2.2 *Role caller*

The first transition of the role concerning authentication is

```
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 the value of variable* State *equals to* 10 *and we receive on channel* RCV *a message equal to the concatenation of the values of variables* A, B, CallID, *constant* auth *and three more values that are to be assigned to the variables* Algorithm, Realm *and* Nonce".

The primed variable notation (e.g.: X') indicates that a new value is assigned to the variable. For example, a statement RCV(X) will mean that a message with value equal to the value of X (X was defined in this role before) is expected to be received via channel RCV, however statement RCV(X') would mean that any message (but with the same type as X, in the case of typed model) is expected to be received via channel RCV, and its value would be assigned to the variable X.

The right-hand side of the transition means:

"*then assign* 20 *to the variable* State *and send via channel* SND *the concatenation of values stored in* A, B, CallID *and constant* ack".

In other words, getAuth stands for getting a hash-function name (Algorithm), a realm value (Realm) and a nonce (Nonce), and as a response caller sends ack[2].

Another authentication related transition is the sndResponce. In this transition caller emits an authentication response to proxy right after sending ack. As no message is expected to be received to fire this transition, a special constant start is used (it is simply a stub, as receiving of message via channel is mandatory for left-hand side of any transition in HLPSL)

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

Here 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 the value* MdC'[3] *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.

The getProxyAuthInfo transition checks authentication credentials received from the proxy. If it is correct — sends ack signal:

```
getProxyAuthInfo.
```

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

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

```
      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')
```

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"*.

The witness-request pairs are used by AVISPA tool to detect authentication failures. If `request` is executed in one of the roles (by an agent who plays it), but corresponding `witness` was not executed, then it means, that the value whose authentication was requested was not produced by the expected agent. More information about HLPSL semantics can be found in user manual [1].

3.2.3 Role proxy

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 (store it into the variable `X'`) and to whom (store into the variable `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 using `in(X'.PasswdX', Keyring)` statement. Here we try to find value in set of pairs `Keyring`; as `X'` is already defined, so if there is a pair `X'.`something in `Keyring`, then the result of operator `in` will be true and 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 the 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 the callee and send a proxy authentication information to the 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.

3.2.4 Role environment

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`. Besides the authentication goals, it is also possible to define secrecy goals to check if specified value stays unrevealed by the intruder or by any other agent not allowed knowing it.

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)
```

3.2 MSC of the attack

When running the AVISPA tool on our HLPSL specification of SIP we get the message: "`UNSAFE`".

The tool automatically builds and displays the attack trace shown 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).

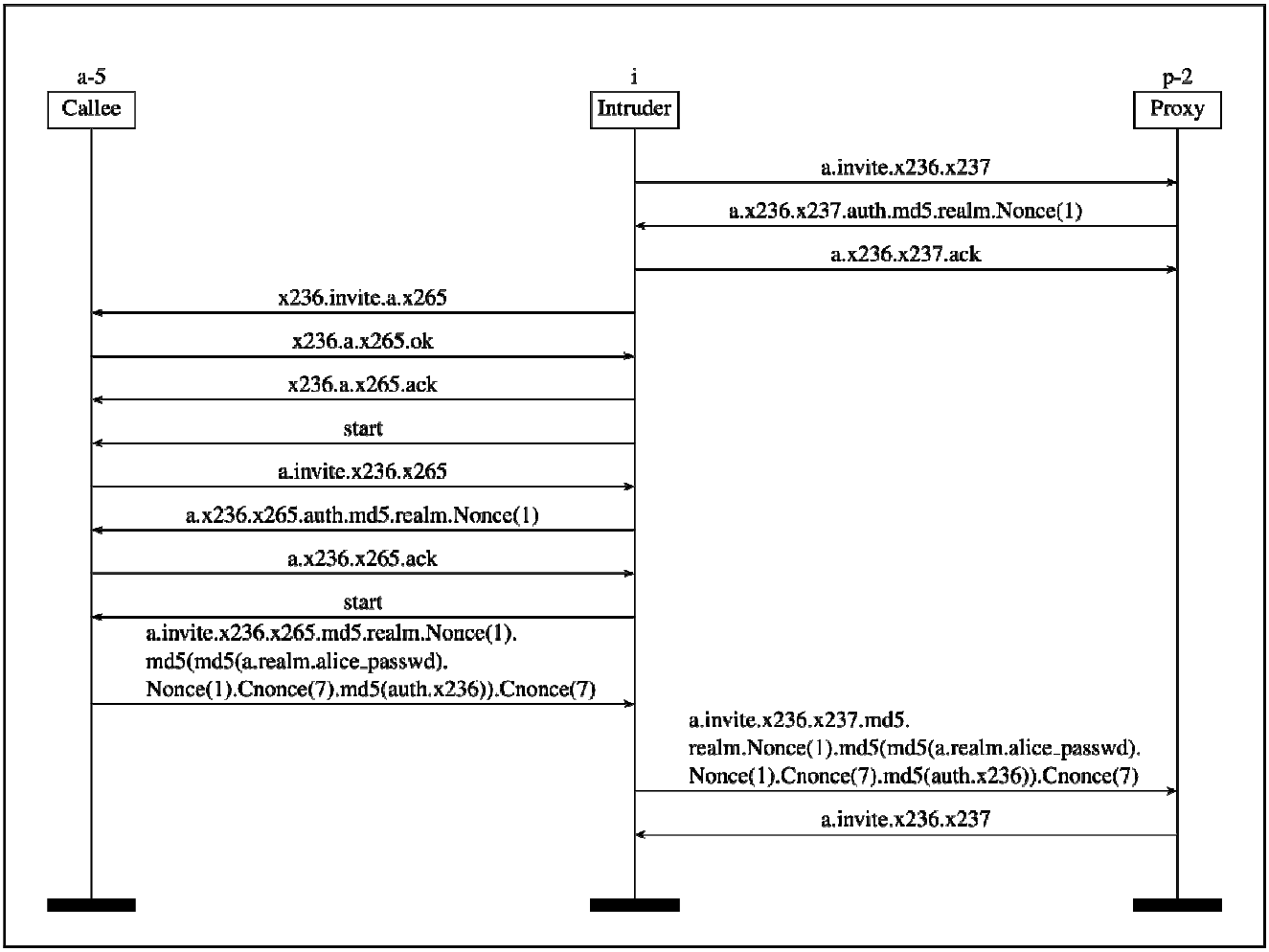

Figure 4. Attack trace

We can see that at first the intruder impersonates a caller `a` when speaking to proxy. After getting an authentication challenge from proxy and sending `ack`, the intruder starts another protocol session with `a`. Here the intruder impersonates a proxy for a 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)),
```

where **resp** is the actual authentication response.

Thus, the computed authentication response will be rejected if the method used for computing the response (i.e. the method value for `A2`) is different than the one in the challenged message.

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 value of variable `A2` is the same in both contexts. If different methods names are used for those contexts, then the generated authentication response is not suitable 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.

The proposed solution is simple and it should not require too many modifications in the complete protocol specification.

5. Validation

The proposed patch changes the Scenario 2 (see Figure 3) such that every `invite` appearing after comment “Callee presses hold” is changed to `reinvite`. So it is not difficult to change the HLPSL specification by 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 in the original version.

Tables 1 and 2 represent a comparative running time in seconds of AVISPA-tool with CL-Atse backend [1] and the result of safety analysis. These results depend on the roles

executed in parallel as well as on the values passed to instantiate them. One of the important parameters is a database of registered agents for the proxy: the `Keyring` parameter. So, for every table we presented two different values of this parameter, whereas every table corresponds to the different set of instantiated roles executed in parallel. The set of instantiated roles of the first table is a minimal one to discover this attack, i.e. to find the attack we need these two instantiated roles to be executed in parallel: an instance of `callee` and an instance of `proxy`, where name and password of the agent playing `callee` are known by the agent playing `proxy`. One can find, that these instances are included in the set of instantiated roles presented in Table 2, if look at definition of `in_session` and `out_session`. Column **sip** contains results for the original version of SIP and **sip'** — for the patched one.

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

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

Table 1. Comparison of results.

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

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

Table 2. Comparison of results.

6. Related work

The comprehensive overview on VoIP security is the reference [14] addressing the operational and deployment aspects of VoIP security. The security mechanisms deployed in SIP are well described in [8] without covering the formal aspect of the security architecture. As of today, the security of VoIP (and SIP in particular) is still in the infancy. Although strong authentication and authorization mechanisms have been defined, the operational deployment is lagging behind. Simple man in middle attacks against SIP infrastructures not using cryptographic authentication mechanisms, have been known for a while and are easy to implement. Ironically, the security mechanisms that have been proposed to counter these vulnerabilities induce more serous threats, because the non repudiation properties are a side effect of the SIP authentication method. To our knowledge, no other previous works have identified such a vulnerability, where a feature interaction in the SIP is combined with serious attack against the SIP.

Many works have been dedicated to analysis and testing of VoIP protocols, but dealing rather with the PSTN interconnection as in [11], or [10]. Most of the performed work has addressed the prevention of SPAM over Internet Telephony (SPIT) attacks [5] as well as mitigating denial of service ones (DoS) [12]. 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 [7], 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 [15] — using faked BYE message and showed an established conversation can be turned down prematurely [6].

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. The second class of attacks concerns poor implementations and/or efficiency driven fallacies in the authentication process. 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.

7. Conclusions

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 as of today. With SIP being the de-facto standard signalization protocol in VoIP, the consequences are far reaching. These consequences of this attack can range from toll fraud, caller impersonation and up to massive deflected denial of service attacks. The essential weakness comes from a feature interaction between routing information and authentication logics. This weakness cannot be avoided, unless the SIP standard is changed. This attack is practical on any VoIP domain and the expected impact quite dangerous. We have developed a prototype and evaluated in laboratory conditions the applicability of the attack.

We have formally confirmed this vulnerability using the AVISPA tool and shown that extending SIP with one more operation can mitigate this attack. Our solution has been automatically validated by AVISPA. 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.

Acknowledgment

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] Humberto Abdelnur, Radu State, and Olivier 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] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989.

[4] D. Dolev and A. Yao. On the security of public key protocols. 29(2):198–208, Mar 1983.

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

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

[7] 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.

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

[9] 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.

[10] 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.

[11] Hemant Sengar, Ram Dantu, Duminda Wijesekera, and Sushil Jajodia. "SS7 over IP: Signaling internetworking vulnerabilities". In *IEEE Network, Vol. 20, No. 6*, pages 32–41, November 2006.

[12] 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.

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

[14] Peter Thermos and Ari Takanen. *Securing VoIP Networks: Threats, Vulnerabilities, and Countermeasures*. Addison-Wesley Professional, 2007.

[15] 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.

Author Biographies

Humberto J. Abdelnur is a Research Engineer at the Nancy research of INRIA, the French National Institute for Research in Computer Science and Control. He received his MSc in Computer Science from National University of Cordoba (U.N.C.), Argentina, in 2005 and his doctorate on vulnerability assessment from the Université Henry Poincaré in Nancy in 2009. His current research and interests are concerned with Software Reliability, Fuzzing, Intrusion Detection and Network Fingerprinting.

Tigran Avanesov is a PhD student of Henri Poincaré University in Nancy, France. He received his master degree on computer science in 2005 from National Taras Shevchenko University of Kyiv, Ukraine. His main current research interests are security analysis of cryptographic protocols using symbolic methods and automated web services composition.

Michael Rusinowitch is Senior Researcher at Nancy research center of INRIA, The French National Institute for Research in Computer Science and Control. He received his doctorate on automated deduction from Université Henry Poincaré in Nancy in 1987. He has been working on rewrite systems, theorem-proving and automated verification. His current research interests include security analysis of protocols and web services. He has published more than 100 articles in international journals and conferences.

Radu State is senior researcher with the University of Luxembourg. Before joining the University of Luxembourg, he was a senior researcher at INRIA, Nancy. He holds a PhD from the university Henri Poincare in Nancy (2001) and MSc in engineering from the Johns Hopkins University (1998). His research interests include VoIP security, security assessment and security monitoring. He has published more then 60 articles in international journals and conferences.