Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 2580–2593
Computer Networks
www.elsevier.com/locate/comnet

A framework for protecting a SIP-based infrastructure against malformed message attacks

Dimitris Geneiatakis, Georgios Kambourakis *, Costas Lambrinoudakis, Tasos Dagiuklas, Stefanos Gritzalis

Laboratory of Information and Communication Systems Security, Department of Information and Communication Systems Engineering, University of the Aegean, Karlovassi, GR-83200 Samos, Greece

Received 26 January 2006; received in revised form 7 November 2006; accepted 13 November 2006
Available online 19 December 2006

Responsible Editor: Refik Molva

Abstract

This paper presents a framework that can be utilized for the protection of session initiation protocol (SIP)-based infrastructures from malformed message attacks. Its main characteristic is that it is lightweight and that it can be easily adapted to heterogeneous SIP implementations. The paper analyzes several real-life attacks on VoIP services and proposes a novel detection and protection mechanism that is validated through an experimental test-bed under different test scenarios. Furthermore, it is demonstrated that the employment of such a mechanism for the detection of malformed messages imposes negligible overheads in terms of the overall SIP system performance.

Keywords: Session initiation protocol; Malformed message attacks; Voice over IP security; Intrusion detection system

1. Introduction

Internet is susceptible to a plethora of attacks and undoubtedly it must be considered as a hostile environment by every critical real-time application such as Voice over IP (VoIP) telephony. Thus, the deployment of various VoIP services raises security challenges that have not been previously encountered in the Public Switched Telephone Network (PSTN), where it is true that the frequency of attacks is extremely low mainly due to its closed architecture. On the contrary, the open architecture of VoIP makes these services vulnerable not only to well known Internet attacks but also to more sophisticated attacks aiming to exploit vulnerabilities that may exist in the signaling or the voice transport of VoIP infrastructures. Researchers have made significant efforts in identifying security vulnerabilities that directly affect VoIP based infrastructures [1,2]. For instance, various flooding techniques could be utilized for attacking voice

* Corresponding author. Tel.: +30 22730 82247; fax: +30 22730 82009.
E-mail addresses: dgen@aegean.gr (D. Geneiatakis), gkamb@aegean.gr (G. Kambourakis), clam@aegean.gr (C. Lambrinoudakis), ntan@aegean.gr (T. Dagiuklas), sgritz@aegean.gr (S. Gritzalis).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2006.11.014

services or, alternatively, an attacker could employ specially malformed signaling or media packets.

For the latter attack, it is well known that both protocol implementations and network applications are often not fully compliant with the underlying standards (e.g. RFCs). As a result there are implementation errors that may pollute a network with incorrectly formed packets and lead to unstable conditions. Furthermore, standard protocol implementations usually focus on well-formed messages without considering any defense tactic against malformed messages. For this reason, once an attacker floods a VoIP target (e.g. a SIP proxy) with a number of malformed messages, the victim is unable to process them resulting to various undesired situations like crashing the VoIP server and creating Denial of Service (DoS) phenomena.

The term ''malformed message'' represents any type of invalid or non-standard message, skillfully formed by an attacker in order to exploit and eventually take advantage of any implementation gap or dysfunction might exist in the target system. As an example, numerous transport control protocol (TCP) common implementation problems are already documented in [8]. Specifically for Internet applications and services, various distinct types of malformed message attacks have been already launched [6,7]. It is therefore clear that malformed message attacks cannot be avoided in VoIP implementations and the corresponding signaling servers. Attackers will try to compromise the system by capitalizing on properly adapted malformed messages.

The security threats introduced by malformed message attacks are often poorly understood and require more research effort in order to effectively protect VoIP infrastructures against them. Security flaws caused by malformed messages in VoIP signaling protocols such as H.323 and session initiation protocol (SIP) implementations have been already published in [3–5]. More specifically, the PROTOS project [9] focuses on the identification of certain malformed input subclasses that can cause instability in the corresponding VoIP Signaling servers (e.g. a SIP proxy). Processing such messages in VoIP networks can, surprisingly, give access to an unauthorized user or drive the provided service to various unstable states and consequently cause DoS. However, such studies test and evaluate the robustness of the implementations without providing any solution for the prevention and protection of VoIP subsystems against this kind of attacks. More specifically, we are not aware of any research work addressing the detection or/and prevention, through some kind of practical mechanisms, of malformed message attacks in SIP realms.

This paper proposes a novel framework to protect SIP-based subsystems (e.g. SIP proxy) against malformed message attacks. The rest of the paper is organized as follows: Section 2 introduces various forms of SIP malformed message attacks, presenting practical examples by deploying them in two different SIP network subsystems namely SIP proxies and registration databases. Section 3 describes the proposed identification and prevention framework, while Section 4 evaluates the performance of the proposed solution in terms of the overheads introduced in the corresponding VoIP subsystems (SIP servers). Finally, Section 5 concludes the paper providing some pointers to future work.

2. Malformed messages and the SIP protocol

SIP is an application-layer signaling protocol for creating, modifying, and terminating multimedia sessions between one or more participants [10]. SIP messages can be either a request or an acknowledgment to a corresponding request, consisting of the header fields and optionally a message body. The overall structure of a typical well-formed SIP message, according to RFC 3261 [10], is as shown in Fig. 1.

A SIP-based multimedia connection between two users is established whenever the caller (e.g. User A) sends an INVITE message to the corresponding proxy, which in turn forwards it to User B (callee). The signaling flow procedure is depicted in Fig. 2.

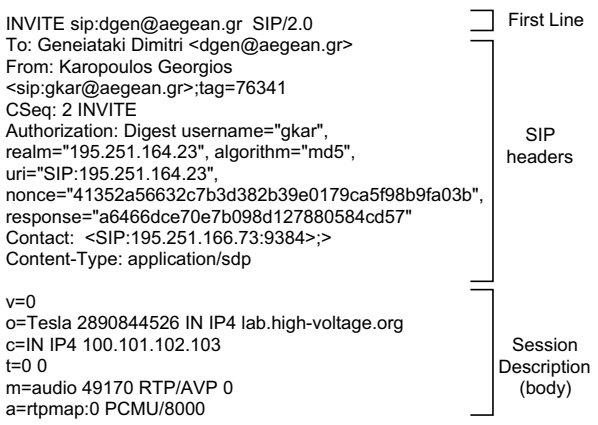

Fig. 1. A typical well formed INVITE message.

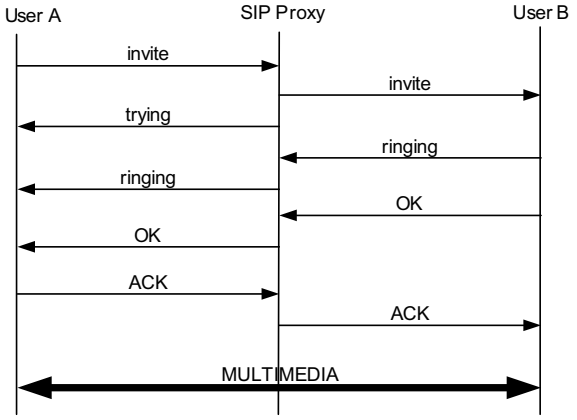

Fig. 2. SIP multimedia connection establishment.

Consequently, whenever a SIP request is received from a SIP proxy the first step is to parse the message. The parsing procedure is essential in order to represent the incoming request into a form that is appropriate for constructing the reply to the request.

Fig. 3 depicts the (initial) processing by SIP proxies (e.g. SER [11]) whenever they receive either a SIP request or a response. Although some SIP proxies' implementations, depending on the vendor, may slightly vary, the sequence described by steps 1 to 3 forms the general concept of the processing mechanism in a SIP-based server.

Generally, SIP parsers are being developed to receive and process well-formed messages; i.e. SIP messages conforming to the RFCs 3261 syntax [10]. However, an attacker, or even a poorly-implemented SIP client, is quite possible to generate and transmit various types of distorted messages [12], resulting to one of the following undesired situations:

- Denial of Service (DoS);
- Unstable operation;
- Unauthorized access.

These problems occur mainly because the SIP proxy parser is unable to successfully handle (e.g. drop) malformed messages. For instance, during the establishment of a multimedia SIP session (see Fig. 2), an attacker instead of sending a well-formed message could try various malformed message combinations to discover a security problem or flaw of the parser. Consider, for example, an attacker who instead of sending the expected well-formed INVITE message (see Fig. 1) he sends the malformed SIP INVITE message shown in Fig. 4. This message is invalid and cannot be generated under the standard SIP protocol syntax, due to the lack of a REQUEST-URI, which must always follow the SIP INVITE method [10]. The target of such a message is either a SIP proxy (if the parser of the proxy cannot handle null messages it may crash or it will generate null DNS requests forcing the underlying DNS service to perform time-consuming and unsuccessful host lookups) or the user's terminal (callee). More details for this kind of attack can be found in [12].

Another case that can be seen as a malformed message attack is that of SIP messages embedding SQL code in their authorization header as illustrated in Fig. 5.

The difference between the messages presented in Figs. 4 and 5, is that in the latter case the objective of the malicious user is the unauthorized modification of the SIPs proxy database (e.g. the registration

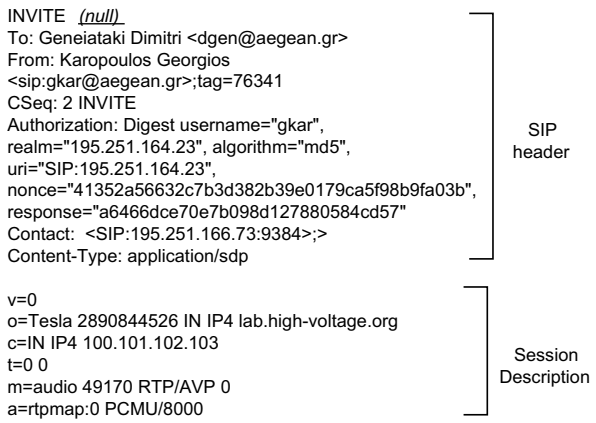

Fig. 4. Example of malformed SIP INVITE message.

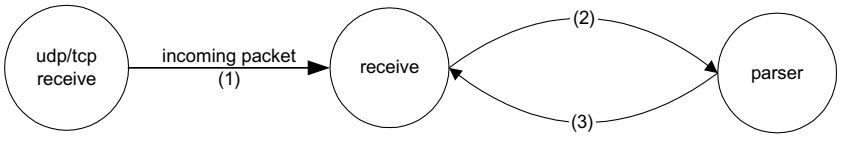

Fig. 3. Processing steps of SIP message in a SIP proxy server.

```
Authorization:Digest username="gkar';
    Update subscriber set first_name='malicious'
    where username='gkar'--",
    realm="195.251.164.23",  algorithm="md5",
    uri="sip:195.251.164.23",
    nonce="41352a56632c7b3d382b5f98b9fa03b",
    response="a6466dce70e7b098d127880584cd57
```

Fig. 5. Example of a malformed message that contains SQL code.

database); a detailed analysis of SQL injection attacks in SIP can be found in [13].

Even though the above attack categories have different targets, they are classified in the same type as they both violate SIPs, protocol specification trying to exploit a different vulnerability in the corresponding SIP infrastructure.

3. Proposed framework for protecting sip infrastructures against malformed messages

In order to have more agile and secure VoIP SIP-based services, capable to defend malformed message attacks, one has to deploy a defence suite with different types of prevention and detection mechanisms. This section introduces a complete security framework that deals with malformed messages in SIP implementations and aims at improving the availability, reliability and security level of the provided services.

3.1. General countermeasures and remedies

As already explained, the availability of VoIP subsystems can be reduced due to the fact that parsers in signaling servers, like SIP proxies, do not examine messages for illegal characters and text. Therefore, input validation procedures are necessary. The lack of any validation mechanism in the receiving process, as illustrated in Fig. 3, is responsible for several security flaws caused by processing such malformed messages. The employment of mechanisms for filtering malicious input, at the Internet application level, has thus been investigated by researchers [14]. Even state-of-the-art firewall technologies incorporate deep packet inspection methods [15] in order to check incoming data for malicious content. The same techniques can be applied to SIP architectures using the Middlebox Communication approach [16].

Moreover, according to RFC 3261 [10] the utilization of underlying security protocols like SSL, IPsec, S/MIME and HTTP Digest can substantially restrict or prevent the use of malformed messages, even though they introduce additional traffic and processing overhead to the corresponding SIP subsystems. Nevertheless, such security schemes, when utilized in SIP, require the installation of an end-to-end or layered Public Key Infrastructure (PKI) beforehand. A detailed analysis of SIP common security mechanisms can be found in [17].

All the aforementioned security protocols have, in some cases, proved to be ineffective. For example, as stated in [12], an attacker may utilize a SIP proxy from another realm to amplify the hazardous effects of his malformed messages. Consequently, although the SIP proxy may not crash it will forward the malformed message towards other proxies in the path and finally towards the end-user trying to cause a DoS. In addition, these mechanisms do not provide any real security against (malevolent) insiders, as it is well known that many security incidents originate from them. For example, consider the case where an insider creates a malformed message, signs it with his private key and then send the message to the corresponding SIP server. As the SIP server will successfully validate the signed message, the process will continue to the next stage that is the parsing of the incoming- malformed message. It must be stressed that such a scenario may crash the SIP server. Someone could claim that malformed message attacks can be repelled by simply blocking the sources that originate malformed packets. However, such a solution may cause a DoS to legitimate users if the attacker (a malevolent legitimate user who holds a legal certificate) has hijacked their connections (e.g. by spoofing their IP or MAC addresses). It is therefore clear that such attacks can be hardly defeated by conventional detection or prevention mechanisms like SSL, IPsec, S/MIME, etc. It is essential to emphasize that such threats cannot be ignored since many security incidents are caused by internal users.

3.2. Detection scheme for malformed messages

The introduction of an appropriate detection mechanism for malformed messages in the existing VoIP infrastructure is considered vital for ensuring reliability and preventing DoS.

The main idea for the development of such a mechanism stems from the SIP syntax as described in the RFC 3261 [10]. More specifically, any message that does not comply to the RFC can be

characterized as malicious. Therefore, the detection mechanism for malformed message attacks can be effectively described through specific structures, known as "*attack signatures*", which consist of two parts based on the SIP syntax. The first part contributes to the identification of the malformed message; it is a *general signature* that can be applied to any SIP method. The second (optional) part specifies additional rules that can be applied to specific SIP methods as determined by the administrator of each SIP domain, according to the security policy of each VoIP provider. One important parameter that has been taken into account is that, unlike non-real time services, the detection mechanism should not introduce significant processing overhead. Otherwise, the interactivity between the involved parties will be jeopardized. An example of the *general signature* is depicted in Fig. 6.

The fist two lines imply that any SIP message:

- Must include a SIP_METHOD (e.g. INVITE, BYE), with a SIP or SIPS URI followed by the corresponding HEADERS.
- Optionally include a MESSAGE_BODY; its presence depends on the utilized SIP_METHOD.

In the section of additional rules it is noticed that:

- Any SIP message should not have the SIP_-METHOD and the MESSAGE_HEADER equal to NULL.
- The length of the SIP method and message body cannot be greater than a specific threshold.

The anticipated identification method has two major advantages:

- Since all SIP messages are based on the aforementioned RFC, it will be easier to embody a light SIP IDS mechanism in a slightly modified SIP protocol stack.
- Alternatively, it is also feasible to include this signature-based identification scheme in an existing open source IDS system (e.g. SNORT, PRELUDE) without making any modifications to the SIP stack. In this case, the only real requirement is to supply the appropriate signatures, even though, as presented in Section 3.3, there are specific limitations when the proposed identification system is applied to those systems.

There are certain circumstances of "well-structured" malicious messages that cannot be identified through the aforementioned rule. For these cases, special signatures for each distinct SIP-method are required. For instance, according to the SIP standard syntax, SIP INVITE messages must include at least one of some specific headers such as Call-ID or Content-Type. Consider the case where an incoming SIP INVITE does not include any of these headers. Such a message must be characterized as malicious and must be discarded prior it is handled by the parser. Otherwise, the parser will try (possibly indefinitely) to find and parse headers that do not exist. Fig. 7 describes the detection signature for a SIP INVITE message. It is stressed that the detection signature contains additional fields (MESSAGE HEADER) that are specific to the SIP INVITE message. Note that headers marked with a '*' character are both mandatory and unique fields. Consequently, if any of these headers is missing or it appears more than once at the incoming SIP message, then this message must be considered as malicious. Further, the line starting with the string INVITE_METHOD covers the case where the header (INVITE) is represented in HEX form.

In the same way it is possible to specify appropriate signatures for each distinct SIP method. For example, in contrast to SIP INVITE, the SIP REGISTER message does not require a message body. Furthermore, in special cases such as in the SIP-SQL injection attack [13], it is not sufficient to check

```
SIP_METHOD SIP-URI | SIPS-URI MESSAGE HEADER+
[MESSAGE_BODY]

additionall rules
SIP_METHOD!=NULL
MESSAGE_HEADER!=NULL
size_of(SIP_METHOD)>%constant% e.g 50 bytes
size_of(MESSAGE_BODY)>%constant%
```

Fig. 6. General detection signature for SIP.

```
INVITE_METHOD SIP-URI | SIPS-URI MESSAGE HEADER+
MESSAGE HEADER =Via | Max-Forwards | From* |To* | Call-Id*
                CSeq* | Contact* |User-agent
                   |Authorization |Event |Content-Length*
                |Content-type*|Record-Route
INVITE_METHOD="INVITE" | %x49.4E.56.49.54.45
MESSAGE_BODY
additionall rules
%Content-Length% >0
%Content-Length%==size_of(MESSAGE_BODY)
(*)mandatory fields
```

Fig. 7. Detection signature for SIP INVITE messages.

```
Authorization = Digest username=".+(';SQL-STATM COMMENT)"
                    realm=" Ipaddress" |
Authorization = Digest username=".+" realm="Ipaddress
                    (';SQL-STATM COMMENT)"  |
Authorization = Digest username=".+(';SQL-STATM COMMENT)"
                     realm=" Ipaddress (';SQL-STATM COMMENT)"
SQL-STATM= UPDATE | INSERT | UNION
COMMENT = "--|#"
UPDATE = SEE SQL 92 syntax
INSERT = SEE SQL 92 syntax
additional rules
size_of(Authorization)> %constant% e.g 100 bytes
```

Fig. 8. Defending SQL attack in SIP messages.

only the validity of the SIP syntax, but it as also necessary to examine the actual contents of the headers. More specifically, considering the situation of the SQL injection attack, the data of the Authorization Header must be also scanned and validated. In this context, the previously described signature must be extended to include some data validation fields specifically designed for the authorization header. The resulting signature will validate both the SIP syntax and the corresponding header contents (e.g. username, realm); regarding data validation, the username and realm are scanned in order to find out whether they contain or not SQL statements. This is illustrated in Fig. 8. It is stressed, that such a signature can be applied whenever a SIP message contains an authorization header.

3.3. Limitation of current intrusion detection systems

Classic IDS systems like SNORT usually attempt to describe all possible malicious messages by storing their signatures in a static signature database. Afterwards, every network stream is examined against all stored signatures or rules for possible matches. Such an approach lacks flexibility and scalability. On the contrary, the proposed detection mechanism adopts the opposite approach and thus any message that does not comply with the general grammar/syntax of the SIP standard must be discarded. This approach is considered far more efficient in recognizing malformed messages. In fact, identifying and archiving all possible variations of malformed messages using static signatures, is at least impractical.

For instance, consider an administrator who utilizes an IDS similar to SNORT for detecting SIP INVITE malformed messages. This implies that she must record in the signature database all possible combinations of the INVITE method with all corresponding headers that are not syntactically correct. Consequently the risk introduced by the development of independent static SNORT-style signatures is that some malformed messages (especially the new ones) may not be identified, while on the other hand some well-formed messages can be mistakenly characterized as malformed. Moreover, SNORT like IDSs are not usually able to identify ''logical errors'' e.g. those that combine information of First Line and the corresponding headers.

3.4. Enhancing SIP servers' security against malformed messages

In order to protect SIP servers, either stateful or stateless, against malformed message attacks, a (pre)-filtering module should be employed for rejecting all non-well-formed SIP messages prior to forwarding them to the parser. As stated in RFC 3261 [10] the syntactical validity of any incoming SIP message must be checked prior parsing and, eventually, either putting it to the corresponding finish state machine (stateful operation), or statelesly forwarding it to the next network hop. At this point the RFC clearly notes: ''The request MUST be well-formed enough to be handled with a server transaction. Any components involved in the remainder of these Request Validation steps or the Request Forwarding section MUST be well-formed. Any other components, well-formed or not, SHOULD be ignored and remain unchanged when the message is forwarded.''

However the PROTOS research project [9] reveals that the validation checks that are employed in existing proxies are very limited and cannot effectively defend against malformed attacks. It is therefore critical that the module/filter examines the incoming messages prior to the SIP parser. If an incoming message matches any of the specified signatures it is instantly identified as malformed and it is discarded. At the same time the system maintains record of all rejected messages. When a specified threshold is violated (e.g. the system logs four or more malformed messages in 1 s) it activates an alarm to the operator's console. Fig. 9 illustrates the required modification to the SIP proxy architecture. Notice, that this module can be also embedded in a firewall capable to ''understand'' SIP traffic. However, this paper concentrates on enhancing SIP servers' behavior without modifying the general SIP architecture.

As already stated in Section 3.3 the development of a vast number of static signatures covering all possible combinations between methods and headers (SNORT alike IDSs) is considered impractical, inflexible and lacks scalability. Consequently, the heart of

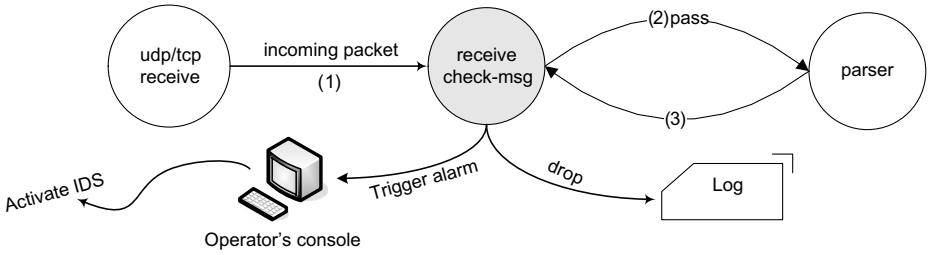

Fig. 9. Enhanced SIP proxy with message checking.

the proposed detection mechanism consists of rules – signatures based on regular expressions. These rules are applied in the opposite direction of the SNORT vision that is: ''any message which does not conform to the SIP standard is directly discarded''.

The proposed schema is divided in the following different inspection phases:

1. First Line inspection;
2. Header inspection;
3. Specialized inspection.

These three phases comprise the *general detection signature* form, that is, First Line and general Header inspection first (applied to *all* incoming SIP messages), followed by *method specific* inspection that is Header inspection (depending on the method used) and Specialized inspection. As far as the method specific inspection is concerned, the focus is on examining headers that couple with the corresponding SIP method, as well as on performing specialized data checking and validation in order to provide protection against SQL injection attacks [13]. Therefore, the first line of any incoming message is examined against the existing malformed rules. If it conforms to the standard SIP syntax then scanning continues to the second stage, which is the Header inspection. When a header is found to be malicious, the message is rejected and no further processing occurs. The message rejection event is recorded into a ''Bad-Transactions'' file. Provided that the header was not found to be malicious, the scanning process enters the third phase that is the Specialized inspection; an action that depends on the method that has been identified in the first stage. As mentioned in Section 3.2, SIP INVITE and SIP REGISTER methods do have some discrepancies. In particular, SIP REGISTER does not include a message body contrariwise to SIP INVITE that usually contains one. So a SIP REGISTER that includes a message body must be characterized as malicious.

Moreover, the Specialized inspection phase includes controls that utilize the combination of the corresponding method and header information. For example, when a SIP INVITE message is received the CSEQ header must be in the following form: *CSEQ: identification_number INVITE*. In this context, a malevolent user may send a *SIP* INVITE message in which the CSEQ header has the following syntax: *CSEQ: identification_number REGISTER*. Although this syntax, based on the RFC, can be characterized as well formed, the parser must discard it as it includes a logical error (it contains a REGISTER statement instead of INVITE).

Finally, specialized inspection performs data validation to shield against SQL injection attacks; the SIP message syntax is validated without any error but the authorization header may still contain data that will possibly try to cause a modification in the local users' database. In a nutshell, the anticipated procedure for the most utilized SIP methods (REGISTER and INVITE) is depicted in Fig. 10.

It must be stressed that when the first line and the common headers of an incoming message are examined then only the specialized rules related to the specific method that this message contains are applied; as opposed to SNORT based IDSs that apply all the rules.

3.5. Implementation issues

The specified signatures will be stored in a protected signature-database on each SIP server or user terminal. The rules will be regular expressions following the Perl Compiled Regular Expression (PCRE) syntax [18]. In order to avoid the introduction of overheads during the processing of incoming messages, regular expressions have been employed for the first line of the message, as all standard SIP parsers process this line, as well as for the most utilized headers (CSEQ, FROM, TO, VIA, Contact and Authorization). The first line representation is

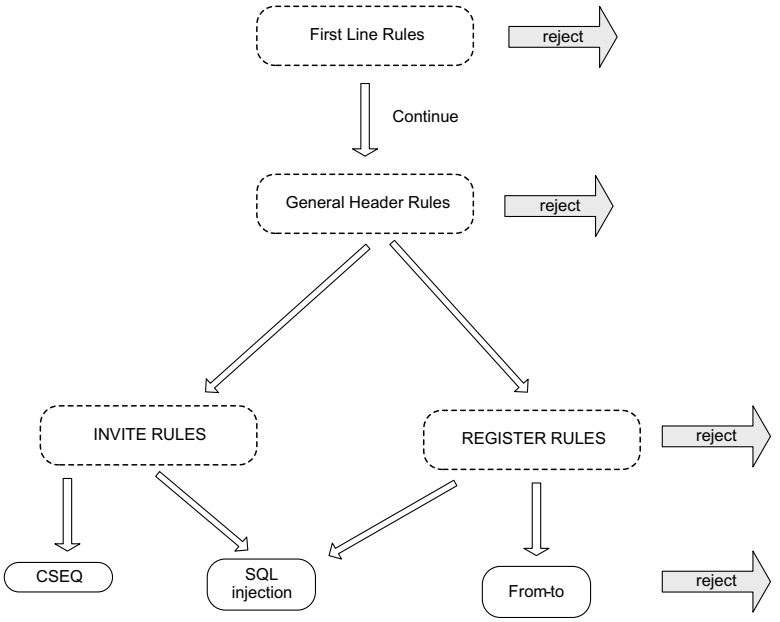

Fig. 10. General detection procedure for SIP malformed messages.

```
^\s*(INVITE|SUBSCRIBE|OPTIONS|CANCEL|ACK|REGISTER)\s+
((((\d{1,3}[.]){3,3}\d{1,3}(\:\d{1,5}))|((sip:){1}\s*\w+@(\w+[.])+\w+)|
((sip:){1}\s*(\w+[.])+\w+)))\s+(SIP[/]\d[.]\d)\s*
```

Fig. 11. Rule for the first line inspection.

depicted in Fig. 11. The proposed signature embodies the most frequently used methods like INVITE, SUBSCRIBE, OPTIONS, CANCEL, ACK and REGISTER. If a local administrator needs to insert a new method she can easily update the signature with the name(s) of the desired SIP method(s).

Regarding header inspection, regular expressions (hereunder we provide each header and the corresponding detection signature) have been developed as presented below:

```
CSEQ:^\s*(CSeq)\s*\d+\b   ($utilized_
method)\b\s*$
Authorization:^\s*(Authorization:)\s*
((Digest\s+username[=]\s*[](\w|\s|[']|
[;])+[]\s*[,]))\s+(realm[=]\s*[]((\w+
[.])+\w+)[])(.*)\w|\S|\s)+(update|insert|
delete|union)(.*)
FROM:^\s*(From:)\s*([]*(\w+\s*\w*)*[]*\
s+((<)*(sip:)((\w+@    (\w+[.])+\w+)|
(((\d{1,3}[.]){3,3}\d{1,3})))(>)*))\s*
TO:^\s*(TO:)\s*([]*(\w+\s*\w*)*[]*\s+
((<)*(sip:)((\w+@(\w+[.])+\w+)| (((\d
{1,3}[.]){3,3}\d{1,3} )))(>)*))\s*
Via:^\s*(Via:)\s*(SIP[/]\s*\d[.]\d \s*
[/]\s*(\w+)\s+(\w+[.])+\w+\w((\s*[:]\
d+)*)(\s*[;] \s*\w+[=]\w+)*)\s*
Contact:^\s*(Contact:)\s*([]*(\w+\s*\
w*)*[]*\s+((<)*(sip:)((\w+@(\w+[.])+\
w+)|(((\d{1,3}[.]){3,3}\d{1,3}:\{1,5}))
(>)*)).\s*
```

The local administrator of the SIP network may easily update the header signature or insert a new signature in the header signature database. It must be noted that in some cases there is a correlation between the method, appearing in the First Line of the message, and the corresponding headers. In order to avoid logical errors, as described in Section 3.4, information that has been derived during the First Line inspection process is utilized. For instance the value of CSEQ depends on the corresponding SIP method. Moreover, the Authorization header is validated not only for its conformance with the RFC syntax, but also for the user data that it contains (see Section 3.4) in order to protect the corresponding database against unauthorized modification through SQL attacks.

4. Performance evaluation

Fig. 12 depicts the experimental test-bed that was set-up for testing the proposed pre-filtering and detection mechanism. It consists of two SIP

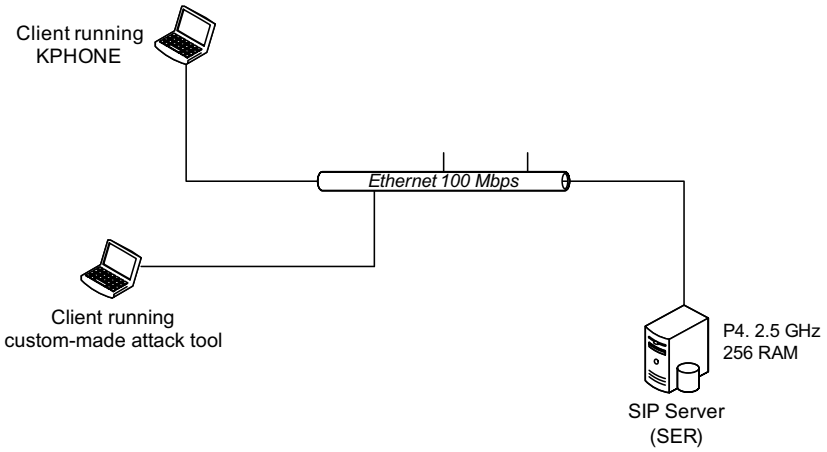

Fig. 12. Test-bed architecture for the malformed message IDS.

software clients and a SIP server. More specifically, the SIP software phone "KPHONE" [19] has been used, while for implementing the suggested protection scheme in a SIP proxy's core we utilized the open source SIP Express Router (SER) [11]. Moreover, two different attack tools have been used; the first one is the SIPBOMBER [20] while the second one is a custom-made tool used to create more sophisticated malformed messages meant to heavily stress the proposed mechanism. The SIPBOMBER tool utilizes tests contained in the PROTOS test suite [9]. Moreover, the SER [11] core engine has been modified accordingly in order to exploit the signature-database and consequently identify malformed message attacks as described in Section 3.4. The server machine uses a Pentium 4 processor clocked at 2.5 GHz and 256 MB of RAM while the local network is an Ethernet at 100 Mbps.

Our goal is to protect the parser from processing malformed messages and at the same time to estimate the processing overheads introduced by the proposed mechanism. In fact this overhead is vital for the practicality or not of the proposed IDS/IPS scheme.

All the processing times listed in this section reflect to the "worst case" scenarios, since the malicious messages that were constructed for the tests deliberately contain many malformed fields (errors). Furthermore, for the purpose of the tests, the embedded identification mechanism has been intentionally set up not to reject the message upon detection of the first malformed field but to continue until all inspections have been completed. All eight scenarios that have been employed for testing the proposed mechanism are described in Table 1. The malformed messages transmission rate for scenario

Table 1
Descriptions of the scenarios evaluated

Scenario number	Scenario description
Scenario 1 (S1)	This scenario utilizes the SIPBOMBER tool to create malformed messages as described in PROTOS suite
Scenario 2 (S2)	This scenario utilizes our custom-made tool to generate specific malformed messages that contain errors in one header only
Scenario 3 (S3)	This scenario utilizes our custom-made tool to generate specific malformed messages that contain errors in the first line only
Scenario 4 (S4)	This scenario utilizes KPHONE to generate well-formed messages
Scenario 5 (S5)	This scenario utilizes our custom-made tool to generate well-formed messages (register without authorization header)
Scenario 6 (S6)	This scenario utilizes our custom-made tool to generate malformed messages that contain errors in the following headers: From, To, Via, CSEQ
Scenario 7 (S7)	Same as the previous scenario with the addition of SQL injection malicious code in the authorization header
Scenario 8 (S8)	This scenario utilizes our custom-made tool to generate various well-formed messages (register, with authorization header)

1 was two requests per second, while for all other scenarios were twenty requests per second following a uniform distribution.

Scenarios 1–4 do not include the inspection procedures for authorization headers and consequently for SQL code injection attacks. This is not the case for the remaining scenarios that include all inspection procedures, as described in Section 3.4. Moreover, the header inspection procedures include not only the normal header validation but also some specialized scanning, like CSEQ header syntax, logical controls (see Sections 3.4 and 3.5), the existence of multiple headers that must be unique e.g. FROM, TO, etc.

The main objectives of the tests were to:

- Evaluate the robustness and effectiveness of the proposed detection and prevention mechanism.
- Determine the processing overheads introduced by the proposed mechanism.

As far as the first objective is concerned, all scenarios were realized without producing any wrong results; that is all malformed messages have been successfully detected while no well-formed messages were rejected. More specifically, no false alarms (either positive or negative) were observed during our tests. The only possibility for a message to be characterized as malicious while it is well formed is to introduce a tag in an existing header that is not described in the current signatures. For example, consider the case where a specific SIP product introduces a tag in the *FROM* header named as "uatag". The proposed mechanism will evaluate such a message as malformed. However, such UAs are not fully compliant with SIP RFC. Even though, the local administrator can on request update the mechanism's signature database to introduce this special tag as part of the well formed message format.

Furthermore, it has been demonstrated that the processing overheads introduced by the proposed mechanism are negligible. Figs. 13–16 present the processing overheads introduced for First line Header and Specialized inspection, respectively. Additional results are provided in Tables 2 and 3, which contain the time durations for average, maximum, minimum and standard deviation parameters.

More specifically, Figs. 13 and 14 and Table 2 illustrate the processing times and various statistical parameters for First line inspection, which in average remain under 35 μs. The fact that all graphs seem to have similar distribution is expected, since the only differentiation between the scenarios was the different length of the First Line and the variations in the malformed messages that were tested. The moderately high standard deviation times, especially for scenario 1, can be explained by the fact that the SIPBOMBER tool and partially our-custom made application, generate malformed packets of excessive length. In addition, some other services running on the SIP server might have instantly affected the recorded processing times.

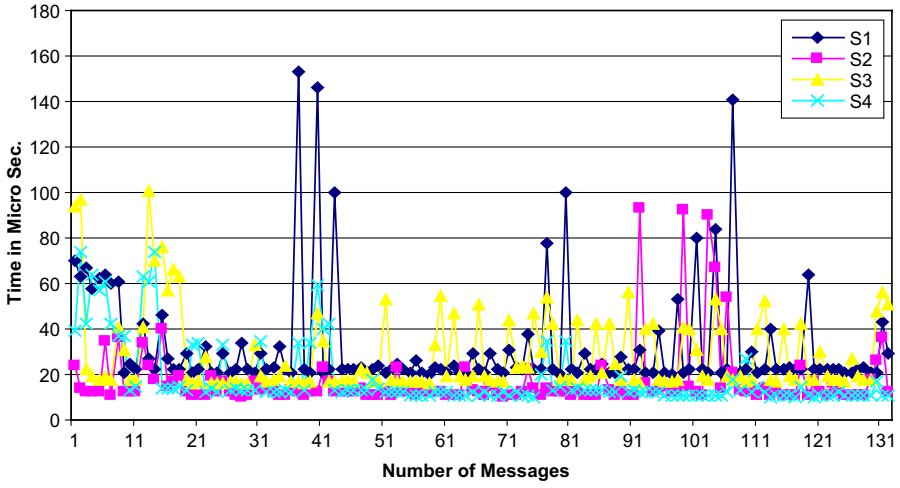

Fig. 13. First line inspection time overheads for Scenarios 1–4.

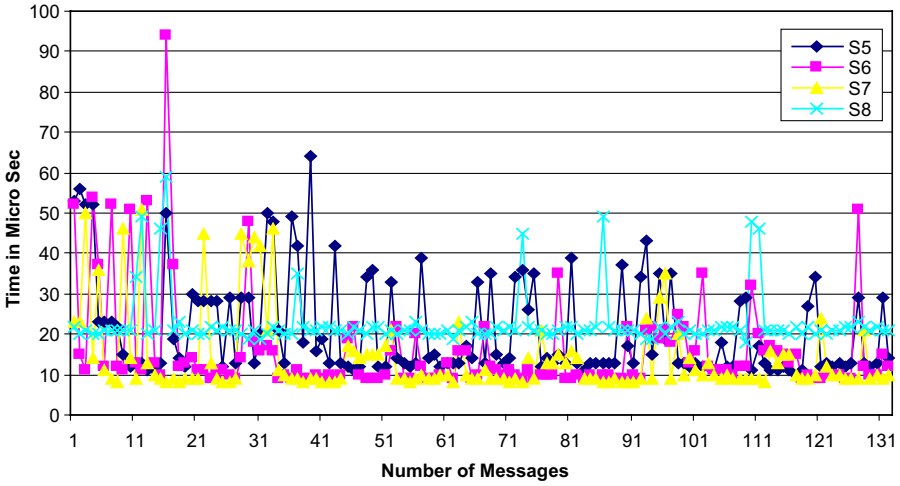

Fig. 14. First line inspection time overheads for Scenarios 5–8.

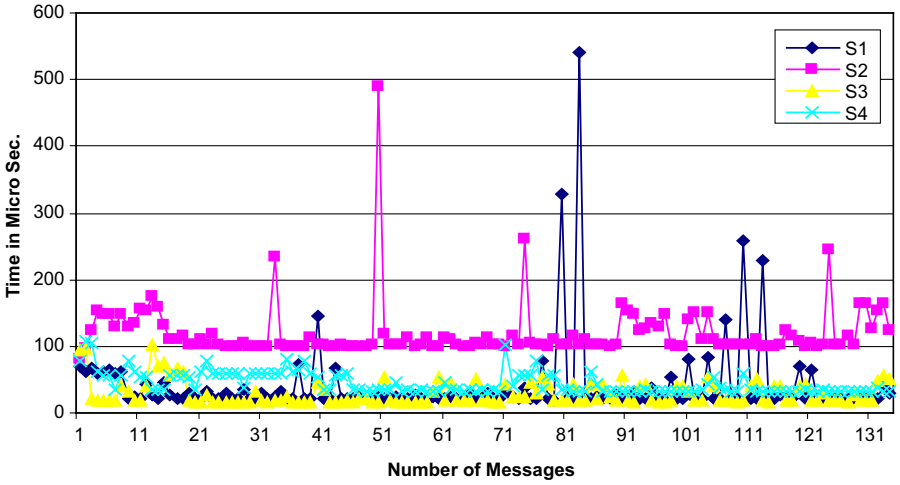

Fig. 15. Header and specialized inspection time overheads for Scenarios 1–4.

Deep inspection overheads and major statistical metrics for headers scanning, SQL injection, etc, are presented in Figs. 15 and 16 and Table 3. In average, all processing times remain under 120 μs with the exception of those for scenarios 6 and 7. The increased overhead for these scenarios is due to the fact that they deliberately contain more than one error in a single SIP Message. In particular, malformed messages for scenario 6 included four header errors, while scenario 7 deployed messages that had the same number of header errors and additionally one of them included SQL malicious code. However, these two scenarios (S6, S7) have been employed only for the "worst case" demonstration and do not have any practical use, since in a real-environment scanning will be aborted and the message will be rejected as soon as the first malformed field is detected.

Clearly, the maximum processing overhead of 120 μs, which has been introduced by the proposed mechanism, is insignificant. Furthermore, the overheads introduced in the scenarios that involve malformed messages are approximately the same with those in scenarios that employ well-formed messages (scenarios 4, 5 and 8). Another important comparative metric is the time required for estab-

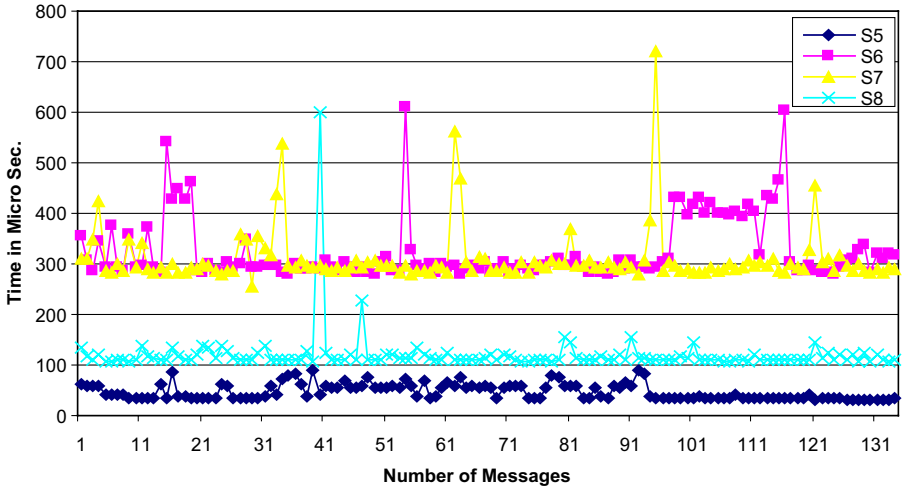

Fig. 16. Header and specialized inspection time overheads for Scenarios 5–8.

Table 2
First line inspection time overheads

Parameter	S1	S2	S3	S4	S5	S6	S7	S8
Max	153.00	93.00	101.00	74.00	64.00	94.00	51.00	59.00
Min	20.00	10.00	16.00	10.00	10.00	09.00	08.00	18.00
Average	32.08	16.71	28.59	18.37	21.07	15.86	14.07	22.63
St. Dev.	24.01	13.98	17.66	14.25	12.33	12.45	09.92	06.59

Table 3
Statistical parameters for header and specialized inspection

Parameter	S1	S2	S3	S4	S5	S6	S7	S8
Max	541.00	489.00	101.00	107.00	88.00	657.00	719.00	600.00
Min	20.00	80.00	16.00	31.00	31.00	280.00	255.00	107.00
Average	40.10	118.98	28.77	44.26	46.84	324.54	309.23	119.96
St. Dev.	60.32	42.54	17.59	16.31	15.32	64.38	56.36	44.03

lishing a SIP connection, which normally is more than 1 s. Moreover, the introduction of a new header that requires deep inspection, such as authorization checking in the SQL injection code case, results in an extra overhead of about 80 μs.

Moreover, in order to test the overall reliability of a SIP server (i.e. SER), that incorporates the proposed mechanism, a flooding attack with malformed messages has been launched. Without the proposed protection mechanism the SIP server crashed only after a few seconds, as opposed to the enhanced SIP server that discarded all malicious traffic and continued to operate smoothly.

Finally, it should be noted that the footprint of SER has been modified by only 0.005% (footprint without the malformed module is 1.205 kbytes, while with the introduction of the malformed module it becomes 1.212 kbytes). Combined with the evaluation results it is clear that the introduction of such a mechanism does not affect significantly the proxy performance while it improves its reliability and availability.

5. Conclusions and future work

VoIP systems gradually become more and more popular as their user base increases fast and the associated services gain in acceptance. In this context, SIP seems to overwhelm other standards mainly due to the fact that it has been adopted by various standardization organizations (e.g. IETF, ETSI, 3GPP) as the protocol for both wireline and wireless

world in the Next Generation Networks era. Meanwhile, various kinds of attacks against those sensitive real-time systems are reported, stemming mainly from VoIP open nature inherited by the Internet. It is beyond doubt that malicious users will try to expose and finally exploit any vulnerability in SIP systems as well as in any VoIP subsystem, aiming to decrease their availability and trustworthiness.

In this paper SIP malformed message attacks have been analyzed. We have presented two different aspects of the attack against different SIP subsystems (i.e. SIP servers and users' registration databases), exploiting implementation "errors", sending malformed messages and launching an SQL injection attack correspondingly. Through experimentation, the proposed solution has been evaluated in terms of robustness and processing overhead, considering well-respected SIP products (server and client software). The derived times have demonstrated that the proposed solution is robust, flexible, feasible to implement and above all secure. Moreover, the proposed scheme and its associated signatures database can be easily applied to other VoIP signaling protocols (e.g. H.323), firewalls or other open source IDS products, enabling them to defend against malformed messages attacks.

Currently, as a statement of direction, we are working:

- On performance issues of the proposed mechanism in user terminals' SIP parsers, focusing on mobile devices (e.g. handhelds) that have limited processing capabilities, and of course, restricted power reserves.
- On the selection of tools, like bison/flex, for improving the efficiency of the proposed mechanism while keeping performance overheads minimal.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments. This work has been performed in the framework of the IST-2004-005892 project SNOCER, (www.snocer.org), which is funded by the European Union.

References

[1] VOIPSA, VoIP Security and Privacy Threat Taxonomy, October 2005. <http://www.voipsa.org/Activities/taxonomy.php>.

[2] D. Sisalem, S. Ehlert, D. Geneiatakis, G. Kambourakis, T. Dagiuklas, J. Markl, M. Rokos, O. Botron, J. Rodriguez, J. Liu, Towards a Secure and Reliable VoIP, Deliverable 2.1, May 2005. <http://www.snocer.org>.

[3] Asterisk SIP Implementation Issue, August 2003. <http://www.atstake.com/research/advisories/2003/a090403-1.txt>.

[4] CERT® Advisory CA-2004-01, Multiple H.323 Message Vulnerabilities, April 2004. <http://www.cert.org/advisories/CA-2004-01.html>.

[5] CERT® Advisory CA-2003-06, Multiple Vulnerabilities in Implementations of the Session Initiation Protocol (SIP), February 2003. <http://www.cert.org/advisories/CA-2003-06.html>.

[6] CERT-In Advisory CIAD-2003-09, Buffer Overrun in RPC Interface Could Allow Code Execution and Denial of Service, August 2003.

[7] J. Fontana, Exchange Server 5.5 Bug Could Be Exploited for Attacks, November 2000. <http://www.pcworld.com/resource/article/0,aid,33882,00.asp>.

[8] V. Paxson, M. Auman, S. Dawson, W. Fenner, J. Griner, J. Heavens, K. Labey, J. Semke, B. Volt, Known TCP implementation problems, RFC 2525, March 1999.

[9] C. Wieser, M. Laakso, H. Schulzrinne, Security Testing of SIP Implementations, 2003. <http://compose.labri.fr/documentation/sip/Documentation/Papers/Security/Papers/462.pdf>.

[10] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Spark, M. Handley, E. Schooler, Session Initiation Protocol, RFC 3261, June 2002.

[11] SIP Express Router. <http://www.iptel.org/ser>.

[12] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, S. Gritzalis, A framework for detecting malformed messages in SIP networks, in: Proceedings of 14th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN), Chania, Crete, Greece, September 2005.

[13] D. Geneiatakis, G. Kambourakis, C. Lambrinoudakis, T. Dagiuklas, S. Gritzalis, SIP message tampering: THE SQL code INJECTION attack, in: Proceedings of 13th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2005), Split, Croatia, September 2005.

[14] D. Scott, R. Sharp, Abstracting application-level web security, in: Proc. 11th Int'l World Wide Web Conf, May 2002, ACM Press, New York, 2002, pp. 396–407.

[15] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, J. Lockwood, Deep packet inspection using parallel bloom filters, in: Proceedings 11th Symposium of High Performance Interconnects (HOTI'03), 2003, pp 44–71.

[16] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, A. Rayan, Middlebox communication architecture and framework, IETF, RFC 3303, August 2002.

[17] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, S. Gritzalis, SIP security mechanisms: a state-of-the-art review, in: Proceedings of the Fifth International Network Conference (INC 2005), Samos, Greece, July 2005.

[18] Perl Compatible Regular Expressions. <http://www.pcre.org>.

[19] KPhone A Voice over Internet Phone. <http://www.wirlab.net/kphone/>.

[20] SIPBOMBER. <http://www.metalinkltd.com/downloads.php>.

Dimitrios Geneiatakis was born in Athens, Greece, in 1981. He received the Diploma in information and communication systems in 2003, and the M.Sc. in security of information and communication systems in 2005, both from the department of Information and Communications Systems Engineering of the University of Aegean, Greece. His current research interests are in the areas of Security mechanisms in Internet telephony, Smart Cards and Network Security and he has several publications in the above areas. He is a member of the Technical Chamber of Greece.

Georgios Kambourakis was born in Samos, Greece, in 1970. He received the Diploma in Applied Informatics from the Athens University of Economics and Business (AUEB) in 1993 and the Ph.D. in information and communication systems engineering from the department of Information and Communications Systems Engineering of the University of Aegean (UoA). He also holds a M.Ed. from the Hellenic Open University. Currently, he is a Lecturer in the Department of Information and Communication Systems Engineering of the University of the Aegean, Greece. His research interests are in the fields of Mobile and ad-hoc networks security, VoIP security, security protocols, Public Key Infrastructure and mLearning and he has several publications in the above areas. He has been involved in several national and EU funded R&D projects in the areas of Information and Communication Systems Security. He is a reviewer of several IEEE and other international journals and has served as a technical program committee member in several conferences. He is a member of the Greek Computer Society.

Costas Lambrinoudakis was born in Greece in 1963. He holds a B.Sc. (Electrical and Electronic Engineering) degree from the University of Salford (UK), an M.Sc. (Control Systems) and a Ph.D. (Computer Science) degree form the University of London (UK). Currently he is an Assistant Professor at the Department of Information and Communication Systems of the University of the Aegean. His current research interests include: Information Systems Security, Smart Cards and Computer Architectures. He is an author of several refereed papers in international scientific journals and conference proceedings. He has participated in many national and EU funded R&D Projects. He has served on program and organizing committees of national and international conferences on Informatics and he is a reviewer for several scientific journals.

Tasos Dagiuklas was born in Patras, Greece in 1967. He received the Engineering Degree from the University of Patras-Greece in 1989, the M.Sc. from the University of Manchester – UK in 1991 and the Ph.D. from the University of Essex – UK in 1995, all in Electrical Engineering. Currently, he is employed as Assistant Professor at the Department of Telecommunications and Network Systems, Technical Educational Institute of Mesolonghi. Past Positions include teaching Staff at the University of Aegean, Department of Information and Communications Systems Engineering, Greece, senior posts at telecommunication companies INTRACOM and OTE, Greece and British Telcom Labs, Ipswich, UK. He has been involved in several EC R&D Research Projects (ACTS, IST, TEN-TELECOM, CRAFT) in the fields of All-IP network and next generation services. He will be the conference general organizer of the international conference, Mobile Multimedia 2007 (Mobimedia 2007). His research interests include All-IP Networks, systems beyond 3G and multimedia services over fixed-mobile networks. He has published more than 60 papers at international journals and conferences in the above fields. He is a member of IEEE, IEE and Technical Chamber of Greece.

Stefanos Gritzalis holds a B.Sc. in Physics, an M.Sc. in Electronic Automation, and a PhD in Informatics all from the University of Athens, Greece. Currently he is an Associate Professor, the Head of the Department of Information and Communication Systems Engineering, University of the Aegean, Greece and the Director of the Laboratory of Information and Communication Systems Security (Info-Sec-Lab). He has been involved in several national and EU funded R&D projects in the areas of Information and Communication Systems Security. His published scientific work includes several books on Information and Communication Technologies topics, and more than 130 journal and national and international conference papers. The focus of these publications is on Information and Communication Systems Security. He has served on program and organising committees of national and international conferences on Informatics and is an editorial advisory board member and reviewer for several scientific journals. He was a Member of the Board (Secretary General, Treasurer) of the Greek Computer Society. He is a member of the ACM and the IEEE. Since 2006 he is a member of the "IEEE Communications and Information Security Technical Committee" of the IEEE Communications Society.