Computer Communications 31 (2008) 2443–2456

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Two layer Denial of Service prevention on SIP VoIP infrastructures

Sven Ehlert [a,*], Ge Zhang [a], Dimitris Geneiatakis [b], Georgios Kambourakis [b], Tasos Dagiuklas [c], Jiří Markl [d], Dorgham Sisalem [e]

[a] *Fraunhofer Institute FOKUS, Next Generation Network Infrastructures, Kaiserin-Augusta-Allee 31, Berlin 10589, Germany*
[b] *University of the Aegean, Greece*
[c] *TEI of Mesolonghi, Greece*
[d] *Nextsoft, Prague, Czech Republic*
[e] *Tekelec, Berlin, Germany*

ARTICLE INFO

Article history:
Received 11 April 2007
Received in revised form 10 March 2008
Accepted 10 March 2008
Available online 28 March 2008

Keywords:
VoIP
SIP
Denial of Service
Flooding protection
Security
Malformed messages
DNS cache

ABSTRACT

The emergence of Voice over IP (VoIP) has offered numerous advantages for end users and providers alike, but simultaneously has introduced security threats, vulnerabilities and attacks not previously encountered in networks with a closed architecture like the Public Switch Telephone Network (PSTN). In this paper we propose a two layer architecture to prevent Denial of Service attacks on VoIP systems based on the Session Initiation Protocol (SIP). The architecture is designed to handle different types of attacks, including request flooding, malformed message sending, and attacks on the underlying DNS system. The effectiveness of the prevention mechanisms have been tested both in the laboratory and on a real live VoIP provider network.

1. Introduction

Security threats are considered minimal in current circuit-switched networks as it is the case in the current Public Swithced Telephone Network (PSTN). This is achieved by using a closed networking environment dedicated to a single service (Voice). However, in an open environment such as the Internet, launching an attack on a telephony server is much simpler. This is due to the fact that Voice over IP (VoIP) services are based on standardized and open technologies (i.e., SIP, H.323, MEGACO) using servers reachable through the Internet, implemented in software and relied often on general purpose computing hardware. Therefore, such services can suffer from similar security threats as any other Internet service.

The Session Initiation Protocol (SIP) [1] has been adopted as the dominant signaling protocol to handle multimedia sessions at both the Internet and the 3G Realms [2]. In this paper we present an architecture to mitigate Denial of Service attacks on a SIP-based VoIP infrastructure. The proposed architecture is designed to detect effectively different SIP vulnerabilities (message flooding, malformed message sending, DNS blocking) through specialized detection modules without requiring modification in the core of the SIP architecture. Those modules have been implemented and assessed under various tests in laboratory and in a real-life VoIP network.

Flooding a server with malicious messages or even high-rate regular messages can have serious consequences on any service. The server is busy processing useless messages while lacking the processing power to reply to authentic user requests. In the worst case, a malformed message that exploits know security holes in an implementation might crash the whole system with a single packet (aka the infamous *Ping of Death* [3]).

We counter such attacks by providing a double-level security architecture. A first line *Bastion host* provides essential security checks against well-known TCP/IP related attacks and detects and prevents SIP message flooding against the host. In the second line of defence, we enhance the SIP proxy with additional security modules that provide specialized SIP related security features. We achieve this by providing a signature-based malformed message detection module that checks incoming messages and a specialized SIP-based DNS cache that is guaranteed to be non-blocking even under time-consuming operation requests.

* Corresponding author. Tel.: +49 30 3463 7378; fax: +49 30 3463 8000.
E-mail addresses: sven.ehlert@fokus.fraunhofer.de (S. Ehlert), ge.zhang@fokus.fraunhofer.de (G. Zhang), dgen@aegean.gr (D. Geneiatakis), gkamb@aegean.gr (G. Kambourakis), ntan@teimes.gr (T. Dagiuklas), jiri.markl@nextsoft.cz (J. Markl), dorgham.sisalem@tekelec.com (D. Sisalem).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.03.016

Laboratory and real-life testbed measurements prove the effectiveness of the modules for these kinds of attacks. Furthermore, we show that these modules introduce only a slight processing overhead, which does not affect service operation negatively.

The remaining of the paper is structured as follows. The next section provides some elemental information about SIP in VoIP environments, necessary for the analysis to follow. Section 3 exhibits its major security threats concerning SIP in the context of this paper. Section 4 introduces our defence architecture to repel and thwart attacks discussed in the previous section, while Section 5 presents and analyses the experimental results. The last section concludes the paper and gives pointers to future work.

1.1. Related works

Several researchers have proposed VoIP security solutions to detect and prevent VoIP-related attacks. Most of these solution focus on different detection strategies.

Sengar et al. [4] propose a detection framework based on Hellinger distance calculation. Another flooding detection algorithm based on Cumulative Sums is presented by Rebahi [5]. Both solutions are able to detect message flooding, but do not provide a mitigation solution.

Wu et al. [6] propose a cross-platform detection framework. Based on the correlation of SIP and RTP traces they detect misbehavioural patterns, e.g., malicious session termination.

Chen [7] proposes a concept for detecting DoS Attacks on SIP systems using a SIP state machine model. The system is outlined to detect unauthorized invalid message flooding.

Another online detection mechanism based on a Bayesian Model for SIP is proposed by Nassar et al. [8]. The system is able to detect different kinds of threats towards VoIP applications besides DoS, including SPIT and password cracking.

2. Background information

2.1. Voice over IP using the Session Initiation Protocol

SIP is an application-layer signaling protocol for creating, modifying, and terminating multimedia sessions among one or more participants [1]. It is a text based protocol designed to establish or terminate a session among two or more partners. The message format is similar to the HTTP protocol, with message headers and corresponding values, e.g., “From: user@sip.org” to denote the sender of a message. The destination of a SIP messages (Request-URI) is provided in the first line of the message, the *request line*. Fig. 2 illustrates a sample SIP-INVITE message. Additionally, several other message headers are dedicated to routing purposes in the network:

```
INVITE sip:dgen@aegean.gr  SIP/2.0                              ]  FIRST LINE
To: Geneiataki Dimitri <dgen@aegean.gr>                         ]
From: Karopoulos Georgios <sip:gkar@aegean.gr>                  ]
CSeq: 2 INVITE                                                  ]
Contact:  <SIP:195.251.166.73:9384>;>                           ]  HEADERS
CallId : 12345667@195.251.166.73                                ]
Content-Type: application/sdp                                   ]

v=0                                                             ]
o=Tesla 2890844526 IN IP4 lab.high-voltage.org                  ]
c=IN IP4 100.101.102.103                                        ]  MESSAGE
t=0 0                                                           ]  BODY
m=audio 49170 RTP/AVP 0                                         ]
a=rtpmap:0 PCMU/8000                                            ]
```

Fig. 2. A typical well-formed SIP-INVITE message.

To	Denotes the receiver of this SIP message. This is generally the public available address of the user (Address of Record).
From	Denotes the sender of the message.
Contact	The actual location where a user can be reached. This location can be different from the *From* URI.
Record-Route	Indicates that an intermediate proxy wants to receive further signalling traffic.
Route	Indicates a route that a new request is going to take.
Via	A list of all intermediate SIP entities that this messages has passed so far.

Further, various network entities compose a SIP network (see Fig. 1), such as *User Agents* (UAs) that generate or terminate SIP requests, *Registrars*, where users log in and announce their availability in the SIP network and *Proxies* that forward requests in the appropriate SIP networks. Several proxies can be deployed in a SIP infrastructure, e.g., outbound proxies that regulate routing outgoing traffic from one network to a foreign network and incoming proxies that handle all incoming SIP requests possibly enforcing additional security checks.

3. SIP security threats

3.1. Resources susceptible to DoS in SIP servers

Denial of Service (DoS) attacks aim at denying or degrading a legitimate user’s access to a service or network resource, or at bringing down the servers offering such services. According to a 2004 CSI/FBI survey report 17% of respondents detected DoS attacks directed against them, with the respondents indicating that DoS was the most costly cyberattack for them, even before theft of proprietary information [9]. To make things worse, attackers have developed tools to coordinate distributed attacks simultaneously

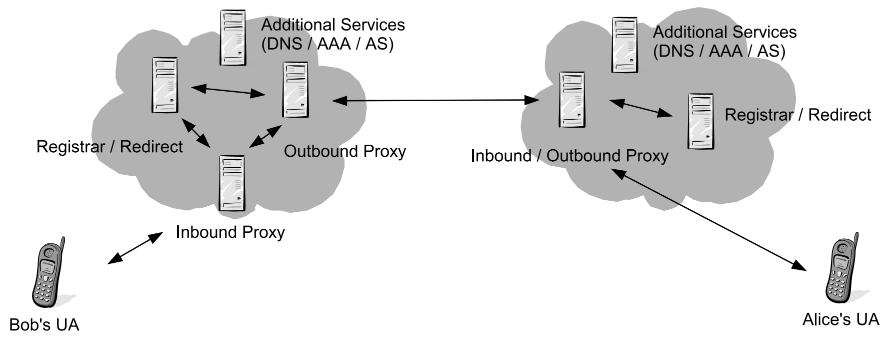

Fig. 1. SIP architecture schematic overview.

from different sources, which is also known as a *Distributed Denial of Service* (DDoS) attack.

Besides launching for example flooding attacks by generating a large number of useless and costless VoIP calls, attackers can exploit certain features of the employed underlying VoIP protocols to incur higher loads at the servers. This might involve issuing requests that must be authenticated, require database lookups by the VoIP servers or cause an overhead at the servers in terms of saved state information or incurred calculations.

Further, the VoIP infrastructure can be corrupted either by launching any kind of DoS attacks on a supplementary component or exploiting a known vulnerability in protocols, layers and services on top of which the VoIP infrastructure is based, including routing protocols or TCP. Nevertheless, the majority of DoS attacks is based on exhausting some of a server's resources and causing the server not to operate due to lack of resources. With SIP servers, there are three resources susceptible to a Denial of Service attack:

- *Memory*. A SIP server needs to copy each incoming request into its internal buffers to be able to process the message. The amount of buffered data and the time period the server is supposed to keep the buffered data varies depending on whether the server is working in a stateful or stateless mode. In any case, the server will at least need to maintain the buffered data while contacting another entity such as an AAA, DNS server or a database for example. Depending on the message type, the number of *Via* headers and the body of the message, the size of a SIP message might range from a few hundreds of bytes up to a few thousands. Further memory requirements depend on the operation mode.
  - *Stateless servers*. Stateless servers need only to maintain a copy of the received message while processing it. As soon as the destination to which a message is to be sent to is determined and the message sent out, the server can delete the buffered data.
  - *Stateful servers*. In general we can distinguish between two types of state in SIP:
    - *Transaction state*. This is the state that a server maintains between the start of a transaction, i.e., receiving a request and the end of the transaction, i.e., receiving a final reply for the request. A transaction stateful server needs to keep a copy of the received request as well as the forwarded request. Typically, transaction context consumes about 3 kilobytes (depending on message size, forking and memory management overhead) lasting about one to tens of seconds if user interaction is involved.
    - *Session state*. In some scenarios servers may need to maintain some information about the session throughout the lifetime of the session. This is especially the case for communication involving firewall or Network Address Translation (NAT) or for special accounting and security reasons as is the case for the 3GPP architecture.
- *CPU*. After receiving a SIP message, the SIP server needs to parse the message, do some processing and forward it. Depending on the content and type of the message and server policies the actual amount of CPU resources might vary. Whereas the CPU capacity of a well engineered and configured proxy should be able to process SIP messages up to link capacity, there are many server operations which make servers block. Such operations may be misused to quickly paralyze a server's operation.
- *Bandwidth*. This involves overloading the access links connecting a SIP server to the Internet to such a level as to cause congestion losses. By overloading the server's access links one could cause the loss of SIP messages which causes longer session setup times or even the failure of session setups. Protection of bandwidth is a general transport-layer issue unspecific to SIP, and affects other types of communication, too.

3.2. Denial of Service attacks in SIP

Despite the well-known attacks that the SIP architecture inherits due to the utilization of Internet technologies [3], there are also specialized attacks on the SIP protocol itself. Until now various researchers [10,6,11–13] have put great efforts in order to identify threats, vulnerabilities and possible attacks in VoIP subsystems. In this work we focus on the detection and prevention of the following major categories of attacks:

(1) Flooding attacks
(2) Malformed messages
(3) Irresolvable DNS attacks

In the following subsections these attacks are analyzed and evaluated. Later on we illustrate our solution to mitigate these described attacks.

3.3. SIP High bandwidth message flooding

Overwhelming a victim's capacities by flooding it with malicious traffic is the most basic and probably also the most difficult to handle DoS attack. The potential attacker can generate flooding attacks with SIP compliant messages (e.g., INVITE, REGISTER) to quickly exhaust the victim's resources. Different SIP proxy implementations vary in the processing speed of crucial tasks, including message parsing, verifying values of MD5 hashes in the authentication procedure, and additional communications with other servers like application, AAA, and DNS servers. Thus, a SIP proxy with slower request processing capabilities is naturally more predisposed for brute force attacks.

All SIP flooding attacks can be done from one source or can be distributed, similar to flooding attacks in TCP. In the case of a distributed flooding attack, the attacker employs a large number of (usually unaware) computers with different IP addresses to generate a higher-bandwidth stream of messages than it would possible from one single machine. Furthermore, attacks where source IP addresses in packets are spoofed to escape detection can be considered as a kind of distributed attack.

By using a fast stream of INVITE messages with different session identifiers such as *To*, *From* or *Call-Id* there is a possibility to exhaust the memory of the attacked proxy. In the case of a nonexistent recipient (invalid *To* header) the proxy will reply with a *404 Not Found* message. This is done immediately after the attempt to locate the user, in which case the state of the current transaction is maintained only for a short time. Hence, for memory exhaustion the attacking stream must be very high.

The memory can be blocked for a longer time by using an existing cooperative recipient. In this case the recipient can ignore INVITE messages and thus this messages are re-sent within a period of 32 s. The cooperative receiver can also reply with a provisional reply (like *180 Ringing* or *182 Queued*) and in this case the transaction state must be kept at the targeted proxy for at least 3 min.

Exhaustion of resources of SIP proxy can be also similarly caused by flooding it with a high count of REGISTER messages. This is especially possible in cases where the SIP proxy, Registrar server and Location server are located on a single machine and thus increasing processing load on the server.

3.4. Attacks using malformed messages

Generally, SIP parsers are developed to receive and process well-formed messages, as defined in RFC 3261 [1]. However, an attacker or even a poorly-implemented SIP client is able to send (either intentionally or unintentionally) various types of malformed messages [14] in order to induce undesired situations such as DoS, unstable operations and unauthorized access. In this context, it is highly likely that the attacker will try various malformed message combinations to discover a security problem/flaw within the victim subsystem. For example, the INVITE message illustrated in Fig. 3 is invalid and cannot be generated using standard SIP protocol syntax, due to the lack of a Request-URI, which is mandatory in an INVITE request.

Furthermore, the text based nature of SIP messages offers the opportunity for message tampering attacks in SIP telephony services, similarly to HTTP messages. This kind of attack is not only targeting at service corruption, but also in the downfall of additional database services. The attack can be triggered every time a SIP network entity (e.g., SIP UA, SIP Proxy) is asking for authentication. When this situation occurs, the User Agent (UA) on behalf of the authorized user computes the appropriate credentials based on the HTTP Digest mechanism [15]. The result of this computation (credentials) is included in the message *Authorization* header. Then the message in which the *Authorization* header is included will be forwarded to the corresponding proxy server, which has to authenticate the received message. Consequently, after receiving such a message the proxy recalculates the credentials using the user's password which is stored in the corresponding database. To accomplish this task, it generates an SQL statement according to the following syntax:

```
Select password from subscriber where username = ‘gkar’ and realm = ‘195.251.164.23’;
```

In case a malicious user attempts to launch an attack in the SIP architecture, exploiting SQL injection, he tampers the SIP message and inserts the malicious SQL code at the *Authorization* header (see Fig. 4). The candidate for injection message can be any SIP method, requiring authentication by a SIP server. The malicious code can be embodied either in the username or in realm fields in the *Authorization* header.

As soon as the proxy receives a SIP message with an infected *Authorization* header, it generates and executes the following SQL statement:

```
Select password from subscriber where username = ‘gkar’;
Update subscribe set first_name = ‘malicious’ where username = ‘gkar’–
```

As a result, albeit message authentication fails, due to the fact that the attacker does not know the legitimate user's password, the second SQL command (*Update*) manages to change gkar's first-name to *malicious*. It is also possible for a malicious user to attempt to employ similar SQL commands, aiming to make the database useless and cause a DoS to the provided VoIP service.

```
INVITE NULL                                              ] FIRST LINE
To: Geneiataki Dimitri <dgen@aegean.gr>                  ]
From: Karopoulos Georgios <sip:gkar@aegean.gr>           ]
CSeq: 2 INVITE                                           ] HEADERS
Contact:  <SIP:195.251.166.73:9384>;>                    ]
CallId : 12345667@195.251.166.73                         ]
Content-Type: application/sdp                            ]

v=0                                                      ]
o=Tesla 2890844526 IN IP4 lab.high-voltage.org           ]
c=IN IP4 100.101.102.103                                 ] MESSAGE
t=0 0                                                    ] BODY
m=audio 49170 RTP/AVP 0                                  ]
a=rtpmap:0 PCMU/8000                                     ]
```

Fig. 3. Example of a malformed SIP-INVITE message.

```
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. 4. An example of an SQL-injected SIP message.

3.5. DNS attacks

A rather simple way to disturb server operation is to include unresolvable host names into a SIP message [16]. A SIP message can contain URIs in varying header fields, including *Via*, *Route*, *Record-Route*, and Request-URI. A SIP server encountering an unresolvable address in a header field (e.g., `Via: unresolbvable.domain.org`) has to wait for the resolver reply to continue operation. If the DNS subsystem knows about the domain, a timely answer might even arrive in case of an unresolvable address. However, often no answer can be provided until a timeout occurs at the DNS system. Depending on the implementation, the proxy might be blocked until the answer from the DNS arrives. We have witnessed through simulation, that in such cases a SIP server can be blocked for up to 5 s through one simple message (see Fig. 5).

4. The defense architecture

4.1. Architecture overview

We have defined and implemented an architecture to detect, evaluate and finally protect SIP-based infrastructures and services against the attacks mentioned in Section 3. Hereunder, we provide a short description of the proposed security architecture and its cardinal components (see Fig. 6).

- *Infrastructure entry bastion host*. This is the point of entry into the VoIP network. An Intrusion Detection System (Snort IDS) [17] extended for VoIP awareness is installed that filters clearly malicious messages and updates firewall settings accordingly. The entry bastion host includes the flooding Denial of Service prevention capabilities.

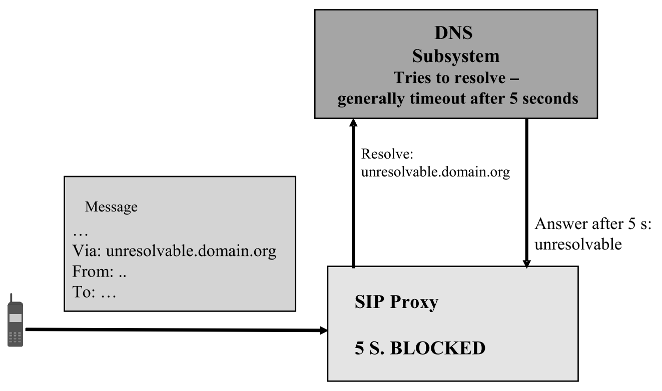

Fig. 5. DNS blocking attack by sending requests containing unresolvable domain names.

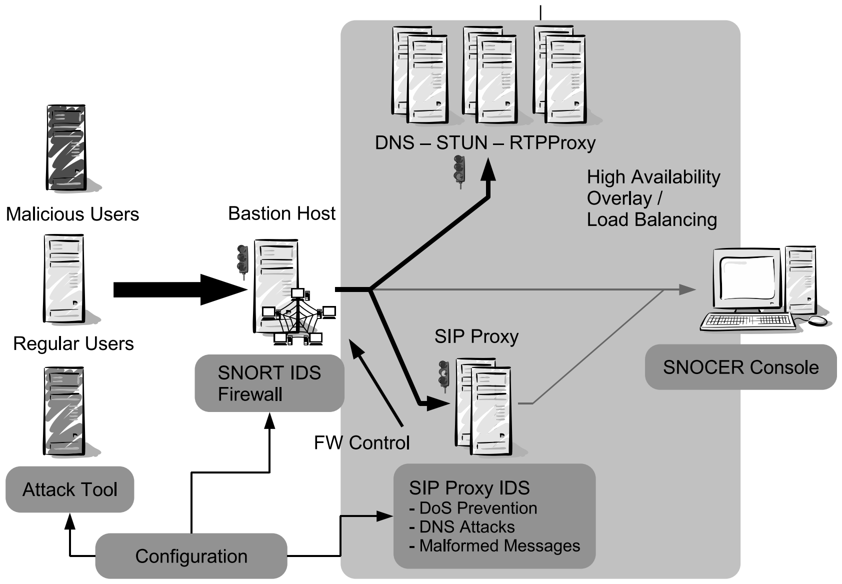

Fig. 6. Defined and implemented two layer defense architecture.

- *Enhanced protected SIP proxy*. One of the main targets of adversaries will be the SIP proxy itself, thus it needs additional protection from attacks. For this reason, a Deep Packet Inspection (DPI) module is deployed that scans incoming messages for malicious content. An enhanced caching daemon stores incoming DNS requests and prohibits server lockdown by unresolvable DNS requests. In case an attack is detected, a communication channel towards the bastion hosts allows dynamic updates of firewall settings.
- *Administrative console*. Every attack indication is logged and forwarded to the operator terminal where attacks can be further inspected and analyzed. This gives the opportunity to react immediately on new threats and allows fine-tuning of detection parameters to avoid unnecessary false alarms.
- *High availability overlay*. For reliability reasons an additional high availability overlay is installed. This consists of replicas of the important SIP network components (SIP proxy, RTP proxy) and a SIP specialized Load Balancer. We present details of these features in [18].

4.2. Flooding defense

For flooding detection we have installed the infrastructure entry bastion host. This is the entry point into the VoIP network. We have installed an Intrusion Detection System based on the Snort IDS [17] which we have extended for VoIP awareness.

The Snort IDS system is configured with a set of newly developed rules for detection of flooding attacks on SIP infrastructures. Detection is based mainly on packet payload inspection which is used to resolve message type (e.g., INVITE, REGISTER) and further on counting of particular messages arriving from a single source. If a pre-defined threshold is reached an alert is generated. As an example of a Snort rule for the detection of INVITE flooding attacks see Fig. 7. As can be seen in the rule example the content and depth directives are used for the detection of an "INVITE" string in the first six bytes of the packet payload. The directive threshold is used for the declaration of the threshold rule which in this case means that an alert is generated when within 60 s more than 100 INVITE packets directed to the SIP proxy are detected. Similar, additional flooding detection rules exist for other SIP messages (e.g., REGISTER or OPTIONS).

4.3. Malicious messages defense

Any message that either does not conform to or violate the SIP specification can cause security problems in any SIP subsystem, however, it is difficult to distinguish between all possible legal and illegal messages. Consequently, in order to effectively detect malicious messages we propose signatures based on the SIP grammar as defined in RFC 3261 [1]. Those signatures are composed of two different parts. The first part identifies malformed message that can be applied to any SIP message. Contrary, the second one specifies some optional rules that must be applied only for specific SIP methods (e.g., INVITE, CANCEL, etc.) and are defined by a SIP domain security policy. The general signature structure is illustrated in Fig. 8. More details about the signature structure can be found in [19].

```
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. 8. General structure of the SIP signature.

```
#Rule for alerting of INVITE flood attack:
alert ip $EXTERNAL_NET any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"INVITE message flooding"; content:"INVITE"; depth:6; \
threshold: type both , track by_src, count 100, seconds 60; \
sid:5000004; rev:1;)
```

Fig. 7. An example Snort rule for INVITE message flooding detection.

In a nutshell, there is always the possibility of input that might not have been considered properly when implementing the SIP stack installed in each SIP product. In order to protect SIP servers from malformed message attacks (by "well formed", we define all SIP messages conforming to RFC 3261 syntax), a pre-filtering module is employed for rejecting all non well-formed SIP messages prior to forwarding them to the SIP parser as illustrated in Fig. 9. If an incoming message matches any of the specified signatures it is instantly identified as malformed and discarded. At the same time the system maintains a record of all rejected messages which is available at the operator console.

4.4. DNS blocking defence

While different countermeasures exist [11] to reduce DNS dependency in a SIP network, in case of a real DNS attack advanced DNS caching is mandatory to successfully mitigate the attack. A DNS cache answers to DNS resolve requests from the SIP proxy. It saves the results of the latest DNS queries, and if the SIP proxy tries to resolve the same address a second time, the stored result in the cache can be returned instead of initiating another time-consuming query.

While different operating systems already provide DNS caches, for optimal usage in SIP they need extended functionality. For example a SIP entity generally uses additionally DNS records to locate other proxies, including NAPTR/SRV [20] records. A general operation system DNS cache does not consider such records for caching. Furthermore, a dedicated SIP DNS cache needs a specialized refresh strategy, as it should keep in its cache table preferably the addresses of known and frequently used proxies, while general OS DNS caches follow a most-recently-used (MRU) policy. To be effective against DNS attacks, a combination of the cache with a SIP proxy with non-blocking parallel message processing capabilities is suggested.

As an example assume a SIP proxy with x parallel message processing queues. Then, allow y message queues, where $x - y$ should be set reasonably low, e.g., to 2, to concurrently resolve addresses through the cache. If now more than y message queues need to resolve an address concurrently, this is an indication of an DNS attack underway. In this case the cache only answers requests from its stored content, and returns an unresolvable message for any request that cannot be answered directly from the cache, thus preventing proxy blocking.

Four traditional cache replacement policies exist, *First In First Out* (FIFO), *Least Recently Used* (LRU), and *Least Frequently Used* (LFU) are well-known cache replacement strategies for paging and web scenario [21,22]. Considering that the time cost of looking up different domain name maybe different, we investigate also a *Time Cost* (TC) policy. As a rule of thumb, all the cache records are arranged as a queue. The newest record is inserted into the head of cache while the oldest one is deleted from the tail of cache.

- For the *FIFO* policy, all records except the newest and oldest one will be moved to the tail by one unit when new records enter queue.
- *LRU* policy is similar to FIFO, the only difference is that if one record of LRU cache is accessed, it will be moved to the head of queue immediately.
- With a *LFU* policy the DNS frequency usage is measured and records are ordered by their frequencies. The higher the frequency, the closer it is to the head of the queue.
- *TC* policy is similar to LFU, but it is ordered by the time cost of each DNS lookup request. Generally speaking, we do not wish to always request domain names from the DNS server which will take more time to look up. The more loop up time the record costs, the closer it is to the head of the queue.

5. Experiments

While we used different testbed setups for the actual measurements, the defense architecture and testbed share multiple common components. These include:

- *The protected SIP proxy*. We have used the SIP Express Router (ser) [23] for this task. SER is an open source SIP server which can act as SIP registrar, proxy or redirect server. In our test bed, all messages to or from a caller have to go through SER.
- *The attacking tool* capable of launching various types of attacks, including the aforementioned ones, to test the SIP infrastructure and to provide input to and evaluate any developed intrusion detection tool suitable for SIP. The tool is build on a black-box approach, so that the tester will not need to know technical issues with respect to the developed platform creating a security test. The tool is able to describe the attack simply by utilizing the appropriate interface and subsequently executing the corresponding tests [24].
- *Operator Interaction and Countermeasures*. Every attack indication is logged and forwarded to the operator terminal and stored in a database. There, attacks can be further inspected and analyzed. This gives the opportunity to react immediately on new threats and allows fine-tuning of detection parameters to avoid unnecessary false alarms. Outgoing Status and Detection messages have been distributed using the Prelude framework [25]. Additionally, when an attack is detected it can be blocked by the intrusion prevention part of the architecture. The prevention part is based on usage of the SnortSam tool [26], which we have enhanced to communicate with prelude. SnortSam unifies communication with several types of software and hardware firewalls such as IPtables, Checkpoint, Cisco PIX, etc., of which we used IPtables with IPset extension [27]. In case of an attack, the suspicious user will be blocked for a user-configurable amount of time. See Fig. 10 as an example setup for the flooding testing.

5.1. Flooding experiments

For flooding experiments, the testbed was prepared using IPtables with IPset extension firewall, Snort IDS, Sip Express Router, the Prelude framework including Prewikka web frontend and the

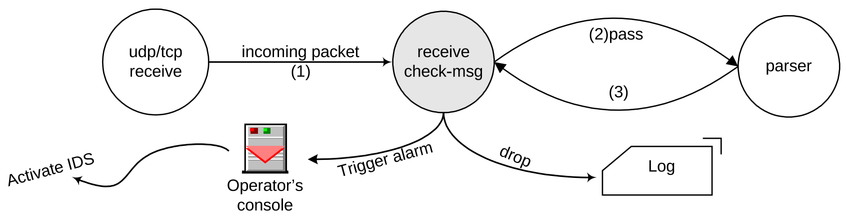

Fig. 9. Testbed setup for malicious message detection.

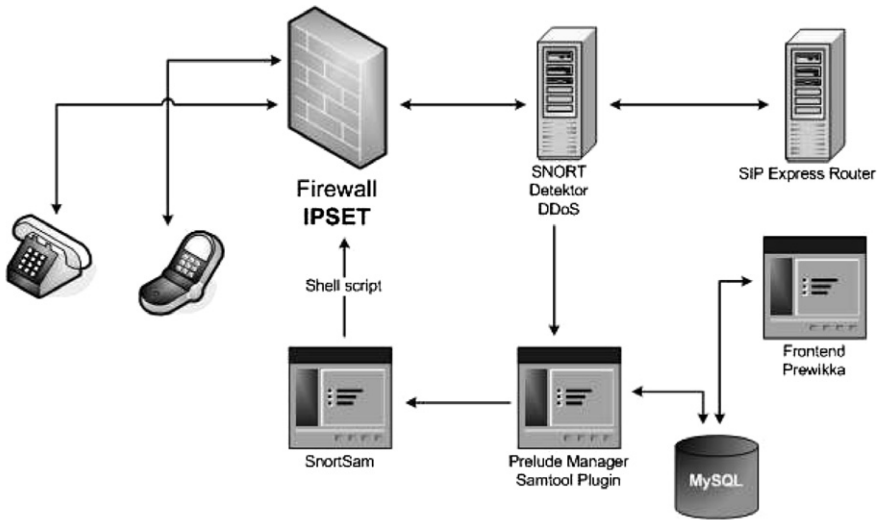

Fig. 10. SIP proxy with enhanced message checking.

attack generation machine. The structure of the testbed is depicted in Fig. 10. The Snort bastion system was configured with our defined SIP rules and the general Snort intrusion rules to detect non-SIP related attacks, thus firewall policies are defined be these Snort rules. All security tools (Snort, Prelude framework and its database, SnortSam + IPtables/IPpset firewall) were placed on the same machine (Bastion Host) during the test. The attack was generated on another machine and targeting the SIP proxy behind the firewall. The firewall machine had two interfaces. The measurements were done on both of those interfaces. The default timeout of IPset for removing blocked address from the firewall is 120 s, but it was changed to 2 s so every source address was blocked for 2 s after the occurrence of an alert.

5.1.1. Scenario: REGISTER flooding

In the first scenario the maximum REGISTER message flood was generated in order to test the defence architecture's mitigation possibilities. The attack originates from a single source IP address. Test results are depicted in Fig. 11, showing the flow on the ingress interface of the firewall host and the packet flow on the egress interface of the firewall host (which is on the side of SIP proxy). We can see every 2 s a peak in the outgoing traffic flow, whenever a blocking rules expired. However, a new alert is generated immediately in case of an ongoing attack, consequently a new blocking rule on the firewall is activated. We can see that the peaks appear in approximately 2 s intervals. Because there is a small delay between attack detection, actual alert occurrence and firewall reaction time, there is a limited flow of high-rate traffic reaching the proxy.

5.1.2. Scenario: increasing INVITE message flooding

In the second test scenario a continually increasing flow of INVITE messages was used to test the effectiveness of the architecture. The flow of INVITE message originates again from the same source IP address. Testbed parameters within the testbed are the same as in the previous case. In this case the INVITE

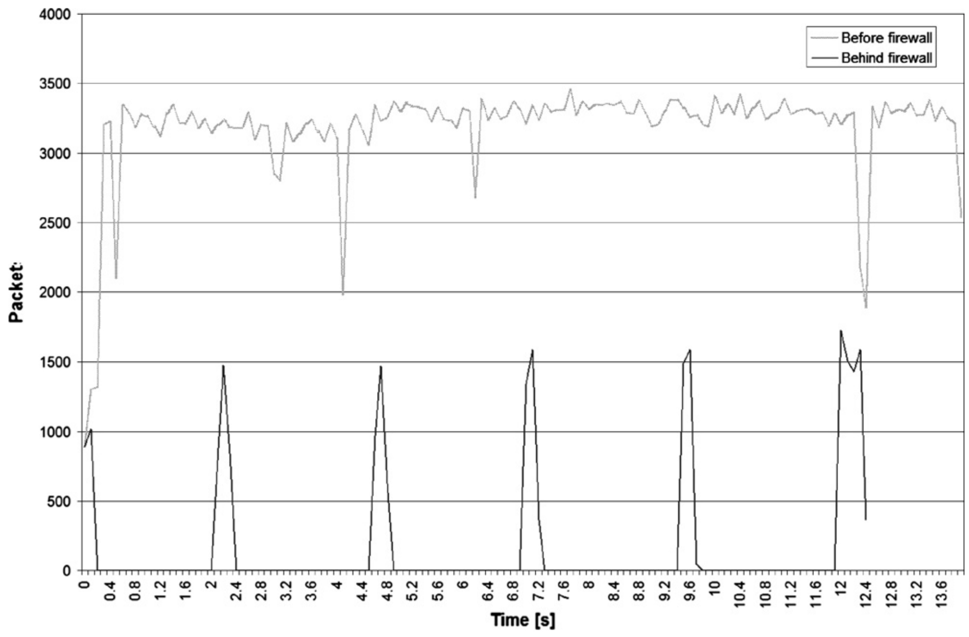

Fig. 11. Flooding defense architecture sub-components.

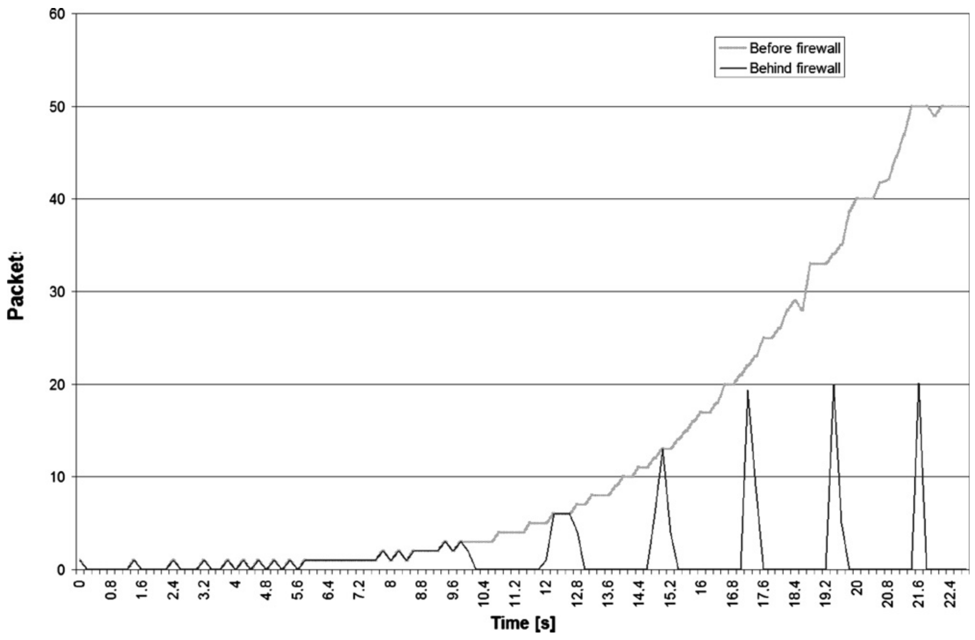

Fig. 12. Defence against REGISTER message flooding.

flood rule generates an appropriate alert and evokes the blocking policy on the firewall. The measurements results are visible in (Fig. 12). Again, we see the firewall rule in effect independently of the input rate.

5.2. Malicious message identification

Fig. 13 depicts the real testbed topology employed in order to test the proposed pre-filtering and detection mechanism in a real environment. Specifically, the attack tool has been utilized to create the necessary malicious traffic, while in all scenarios the target proxy was fed with normal traffic that was copied from a real VoIP SIP-based network and "passed" to the testbed environment.

Here, we provide real scenarios results and testimonials showing that the proposed prevention and detection malformed message scheme is robust and effective. Whenever a message is identified as malformed, an alarm is raised. Fig. 9 illustrates the required modification in the SIP proxy architecture (e.g., SIP Express Router (SER)).

The specified signatures are stored in a protected signature-database on the SIP proxy. In order to implement the rules we utilized the Perl Compiled Regular Expression (PCRE) syntax [28]. 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*) based on the general structure illustrated in Fig. 8.

The first line representation is depicted in Fig. 14. The proposed signature embodies the most frequently used methods including INVITE, SUBSCRIBE, OPTIONS, CANCEL, ACK and REGISTER. Considering the case where a local administrator requires to insert a new method the signature can easily be updated with the name(s) of the desired SIP method(s).

Regarding header inspection, regular expressions for the most prominent headers have been developed (see Fig. 15). Furthermore the developed signatures do not only identify non-conformance messages but also detect attacks embodying malicious code like the SQL injection example presented in Section 3 (see the *Authorization* header in Fig. 15).

The local administrator of the SIP network may easily update the header signature as well as insert a new signature in the header signature database. Furthermore, in some cases there is a correlation between the method, appearing in the First Line of the message, and the corresponding headers, which is used in order to avoid logical errors. For example in an INVITE request the *CSeq* header should have the following form `CSeq: INVITE` *sequence number*. In case that the "INVITE" substring does not exist in the *CSeq* header, this request would be considered as malformed.

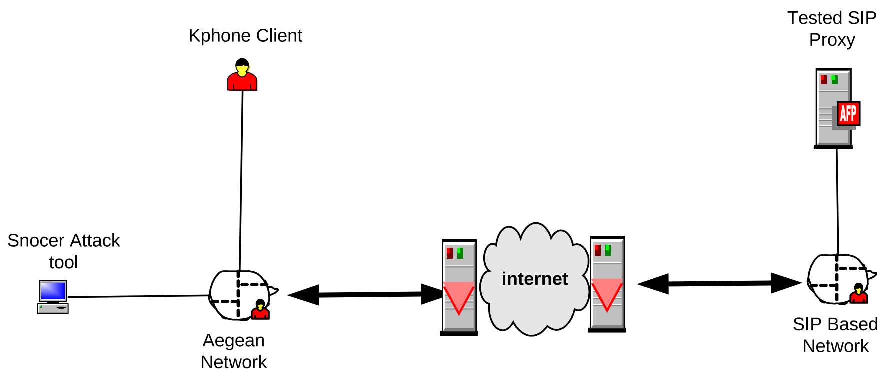

Fig. 13. Flooding by an increasing rate of INVITE messages.

```
^\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. 14. SIP signature for malicious message detection (message first line).

The employment of the malformed detection/prevention mechanism in the real-life network will produce useful conclusions about its applicability. Therefore, for evaluation purposes we have developed the scenarios illustrated in Table 1. Particularly, for scenarios 1–3 the real-life SIP-based network was attacked with various malformed messages. On the contrary, in scenario 4 only real traffic is utilized. Moreover, the average received messages from the SIP Proxy, for scenarios 1–4 was 21, 114, 34 and 16 messages per second, respectively. The main goals of the SIP-based real testbed are the following:

- 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, real scenarios results have shown a small number of false alarms, which is absolutely normal. The false alarms that have been produced are mainly false positives. The main reason of these alerts is based on SIP client inconsistency with RFC 3261, which clearly means that existing SIP clients are not fully conforming to RFC 3261 message syntax as supposed by the authors in the proposed signature scheme. However, after a testing period of the malformed detection mechanism, the administrator of any specific realm can eliminate these alarms by modifying the signatures appropriately. The probability that a well-formed message is falsely declared as a malicious is very small regardless of the SIP proxy being attacked or not. Table 2 presents the false alarms (false positives) that were produced during testing.

Regarding performance results, we have demonstrated that the processing overhead introduced by the malformed detection tool is not significant. As mentioned in Section 4.3, detection/prevention consists of two distinct phases: (a) the first line and (b) the header inspection phase. Thus, in Fig. 16 a sample of the logged measurements, focusing on the processing overheads introduced from the first line inspection is presented, while in Fig. 17 we illustrate the probability density function (PDF) of the corresponding delay. Additionally, Table 3 shows the corresponding standard statistical metrics like time durations for average, maximum, minimum and standard deviation.

Observing Table 3 and Fig. 16 one could easily realize that the introduced delay for the first line inspection under the different situations is negligible, mainly for normal traffic as illustrated from the results of Scenario 4. Moreover, it must be noted that this situation is probably the common case as the system operates under normal traffic most of the time, while attack incidents are generally rare. In the case where the SIP proxy is under attack it seems that additional delay has been introduced due to the fact that the system is more stressed (which of course is absolutely logical). On the other hand, the well-formed traffic seems to be unaffected as the average delay produced is about 15 μs, while in case the system is not under attack this additional overhead is insignificant at near to 7 μs.

Besides, as depicted in Fig. 17, the probability of introduced delay to be between the period of 7 and 18 μs is greater than 0.85, in the case of attack scenarios (scenario 1–3). On the other hand, in case the system is not under attack (scenario 4) – which is the most common one - the probability that the introduced delay is between 5 and 15 ms is greater than 0.95.

Concerning the delay introduced in the header inspection phase, similar to the first line inspection discussion, Fig. 18 presents the processing overheads, while Fig. 19 illustrates the probability density function of the overhead, respectively. Table 4 contains time durations for average, maximum, minimum and standard deviation parameters. Specifically, the average delay introduced as illustrated in Table 4 varies between 95 and 130 μs. The fact that the plots seem to have similar distribution is normal since the only differentiation between these scenarios were the different length either in the first line or the headers 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 in this attack phase we generated malformed packets of excessive length that exist in PROTOS tests [14]. The same variation has also been spotted in laboratory tests [19]. However, the average delay in scenario 1 is about 130 μs and thus is considered insignificant. Furthermore, the introduced average delay (for the headline checking) when the SIP proxy is under attack is less than 100 μs.

As depicted in Fig. 19, the probability of introduced delay to be in the range between 90 and 120 μs is greater than 0.9. In addition, in case the SIP proxy is not under an attack the probability for the introduced delay to be lesser than 120 μs is more than 0.95).

Summarizing the above results we are able to conclude that the processing overhead introduced by the malformed detection/prevention mechanism is minimal – less than 100 μs – in the case that the SIP proxy is not under an attack. Even though in the case of various malformed attacks launched against the SIP proxy the introduced delay in the well-formed messages is also insignificant. On the top of that, the malformed detection/prevention mechanism shows to be robust, scalable, feasible and practical to implement in a real SIP-based system.

5.3. DNS attacks

5.3.1. Testbed

The DNS testbed consists of different components. The main component is our protected SIP proxy as described earlier. The SIP proxy can be configured to have different parallel processing queues *n*. The proxy is connected to our deployed DNS caching solution. We use our developed attack tool generator to launch messages containing unresolvable domain names. The tool generates continuously hard-to-resolve SIP messages every *i* seconds. Finally, we have deployed regular user agents in form of the SIPp [29] message generating and processing tool. The full testing setup can be seen in Fig. 20.

```
CSEQ: ^\s*(CSeq:)\s*\d+\s+\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+((\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}):\d{1,5}))(>)*)).+\s
```

Fig. 15. SIP signature for malicious message detection (header fields).

Table 1
Descriptions of the employed real scenarios

Scenario number	Scenario-description
Scenario 1	This scenario utilizes the PROTOS test to create malformed messages
Scenario 2	This scenario utilizes specific malformed messages that contain errors in the first line only
Scenario 3	This scenario utilizes specific malformed messages that contain errors in one header
Scenario 4	This scenario utilizes only real-life traffic

Table 2
Malicious message false alarm alerts

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Processed Message	42036	310000	40968	56308
False Positives	246	198	274	209
Probability	0.005852	0.000639	0.00668	0.003712

5.3.2. Scenario

Within our testbed, we use SIPp to simulate multiple SIP UA that perform regular SIP background traffic consisting of REGISTER and INVITE requests with successive OK responses. We registered 20 simulated UAs at 10 external SIP proxies, with two users per external proxy. One user belongs to the *caller* group while the other user belongs to the *callee* group. The caller group was configured to use our local SIP proxy as an outbound proxy, while the callee group directly registered with the external proxies. See Table 5 for the list of contacted external providers.

Table 3
Statistical parameters for first line inspection

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Max	291.00	54.00	65.00	57.00
Min	4.00	5.00	5.00	5.00
Average	15.07	10.63	8.09	7.57
St. Dev.	13.65	2.85	4.91	5.47

We generated 1000 REGISTER messaged plus 1000 INVITE from the caller group destined to the callee group, which in turn accept every incoming INVITE request. Within 1 s, the caller group generates 20 messages. We counted the number of positive replies from our local proxy to the caller UA group, which should be in the optimal case 2000.

The used attacker was configured to send continuously messages to the local SIP proxy with hard to resolve DNS names within, thus blocking it. Under attack, the number of positive replies from the normal traffic model will be lower than 2000. The effectiveness of the attack is highly dependent on the configured parallel processing queues at the proxy server (n), thus we performed measurements with different values of n.

We created three different scenarios to vary the initial setup:

(1) The caller group calls each other user in the callee group in sequence. Each callee is called once. After every callee has been called, the sequence starts again.
(2) The same as above, with one callee called 11 times in a row, followed by the other users only one time.
(3) Like scenario 2, but destinations are selected randomly with one callee having a higher probability to be called than the other ones (55%, the same as in scenario 2).

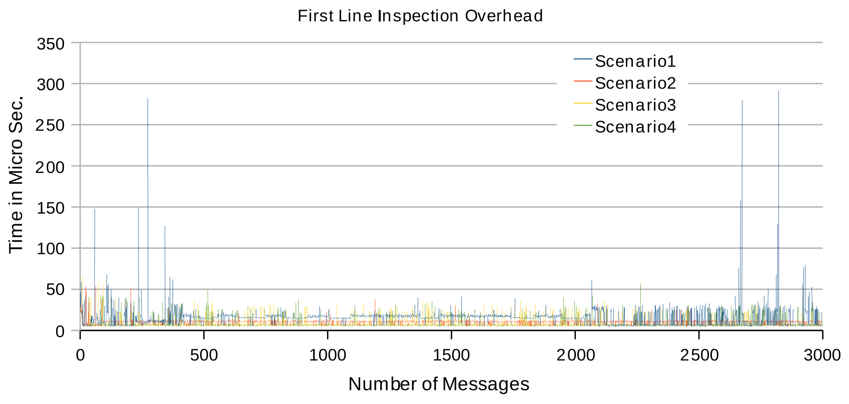

Fig. 16. First line inspection overhead time for scenarios 1–4.

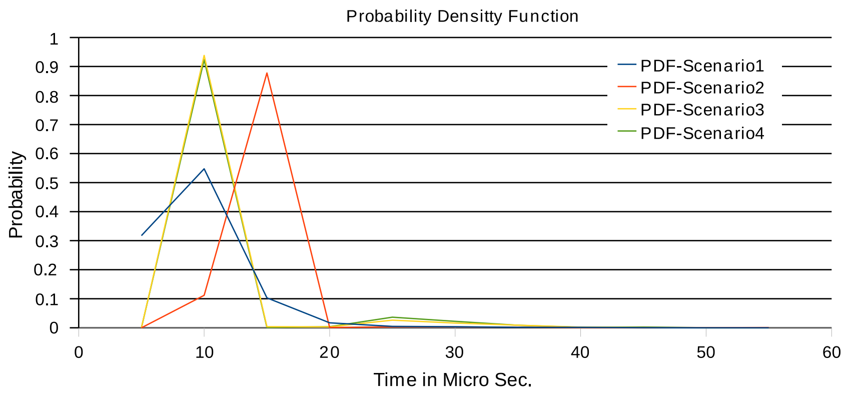

Fig. 17. Probability density function for introduced delay for scenarios 1–4.

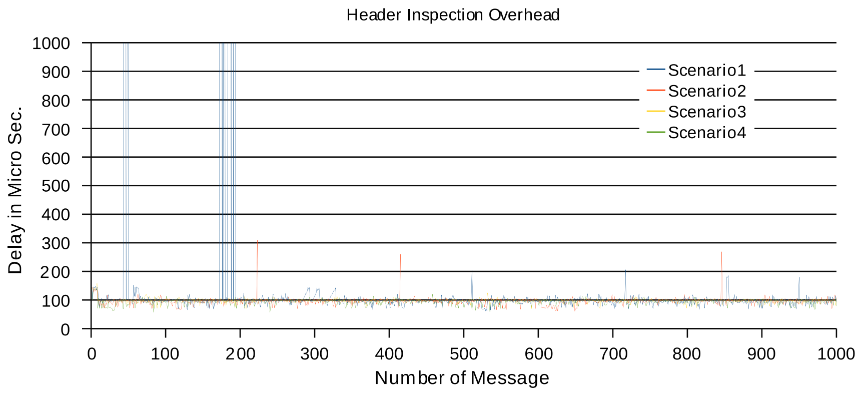

Fig. 18. Header inspection overhead time for scenarios 1–4.

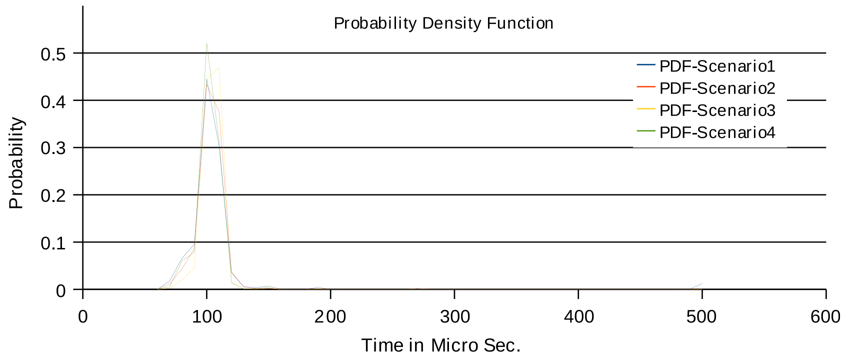

Fig. 19. Probability density function in header inspection for scenarios 1–4.

Table 4
Statistical parameters for header inspection

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Max	3395.00	447.00	497.00	196.00
Min	61.00	61.00	64.00	57.00
Average	130.47	97.59	98.37	95.81
St. Dev.	303.70	15.25	9.62	8.94

Table 5
Used providers

Provider name	SIP Proxy URL
Iptel	iptel.org
Bluesip	bluesip.net
Sipphone	proxy01.sipphone.com
FWD	fwd.pulver.com
Voiptalk	voiptalk.org
Nikotel	calamar0.nikotel.com
Voipuser	sip.voipuser.com
Sipgate	sipgate.co.uk
Irepublics	voiz.irepublics.com
Sipnumber	sipnumber.net

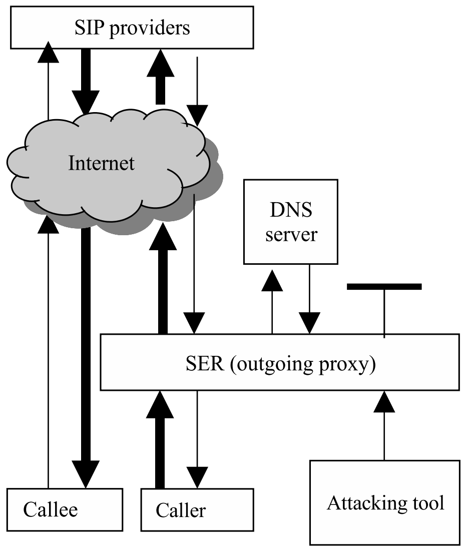

Fig. 20. DNS blocking testbed scenario.

5.3.3. Results

To prove the attack's effectiveness, we have run the attack on our testbed with no DNS cache installed. The results can be seen in Figs. 21–23, showing the number of processed requests at the SIP proxy in respect to the number of parallel message queues n and the attacking interval i in s.

From the figures we can see that independently of the selected scenario a successful attack can easily be launched to a SIP proxy. Even with 64 parallel messaging queues it is possible to reduce the proxy's performance by around 50%, i.e., from 2000 generated requests only 1000 are successfully processed, while the others are not processed during the blocking attack. This outcome can be achieved easily be sending 10 DNS attack messages per second. From the attacker's side this even does not require the utilization of distributed attacking hosts (DDoS), as even a very low performance machine can create this amount of traffic.

For less powerful proxies, i.e., proxies with fewer parallel processing queues this attack is even more effective with les-

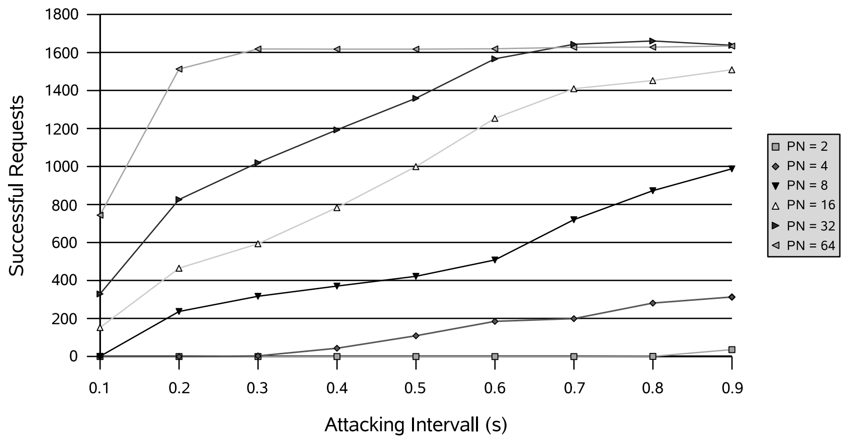

Fig. 21. Scenario 1, no cache.

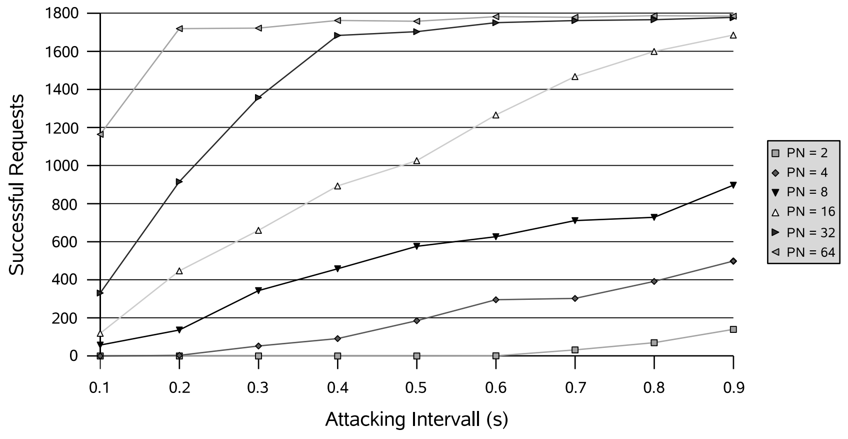

Fig. 22. Scenario 2, no cache.

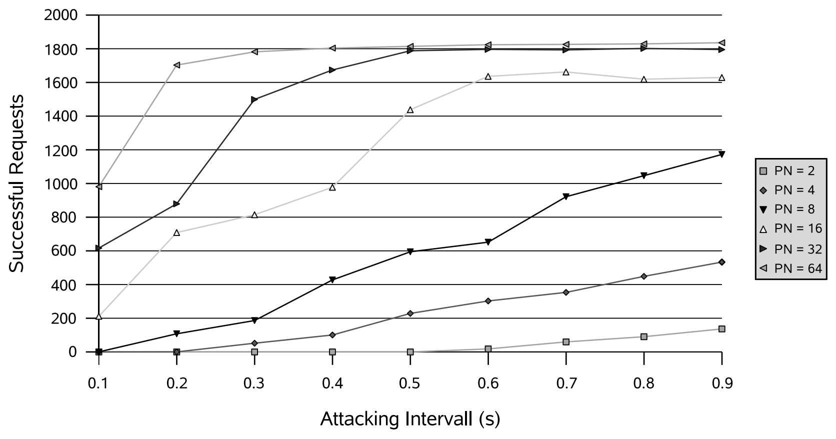

Fig. 23. Scenario 3, no cache.

ser messages. E.g., with $n = 4$, three attack messages per second are sufficient to achieve a full Denial of Service and with $n = 16$, 10 attack messages per second can reduce the proxy's performance by around 90% (only 200 out of 2000 messages processed).

From the figures we can deduce that even very powerful proxies can be rendered unoperational with low attacking resources.

In the next step we installed our DNS caching solution and ran the setup again. Note, as we have proven that an attack can be successful independently of the number of parallel processing queues, we have run this setup exemplarily with $i = 4$. We have implemented four different caching strategies for the DNS cache, as described in Section 3.5. The DNS cache was configured to keep 50% of the maximum possible number of entries in its internal database. Figs. 24–26 show the performance of a the developed DNS cache.

From the figures it can be seen that an attack caching strategies significantly increase the performance of the SIP proxy. However, there are differences in performance based on the type of caching strategy.

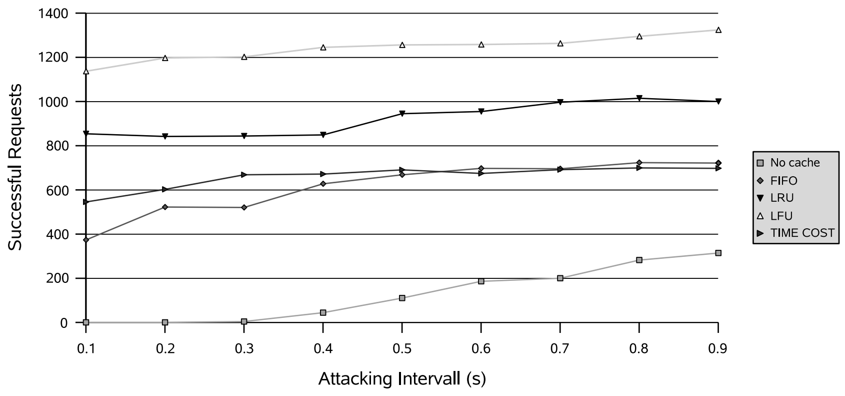

Fig. 24. Scenario 1, with cache.

The most successful caching strategies are Least Frequently Used (LFU) in all scenarios. However, only in sceanrio 1 there we can see a major difference to the second successful strategy, which is Least Recently Used (LRU). Both in scenarios 2 and 3 LRU caching is nearly as effective as LFU. Both First-In-First-Out (FIFO) and Time Cost (TC) Caching strategies yield lesser performance.

In absolute terms, there's also a difference in efficiency of the cache based on the sceanrio. While LFU reaches around 60% efficiency (1200 of 2000 requests are processed) in sceanrio 1, this values increases to around 70% in scenario 2 and reaches around 80% in sceanrio 3.

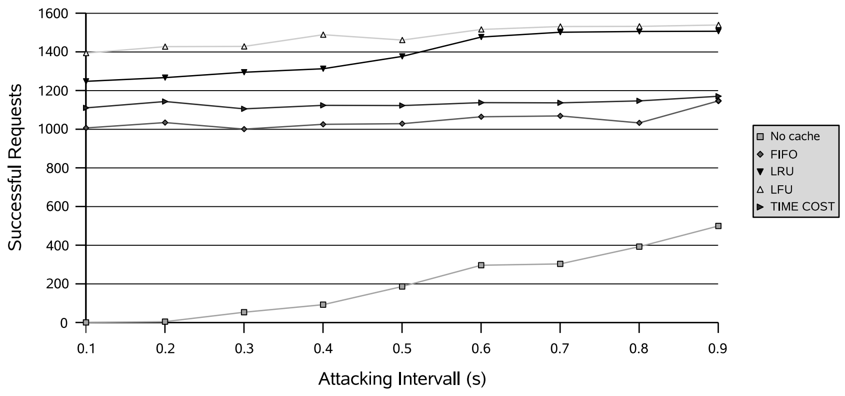

Fig. 25. Scenario 2, with cache.

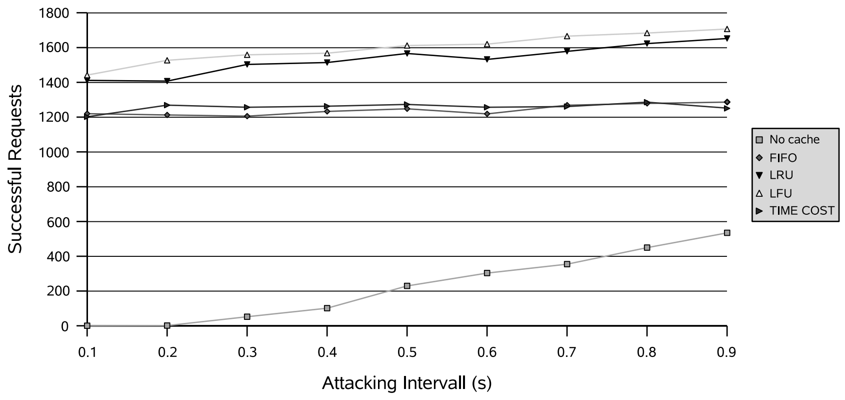

Fig. 26. Scenario 3, with cache.

Table 6
Distribution of cached vs. uncached requests within testruns of about 120,000 requests

Testrun	1	2	3	4	5	6	7	8	9	10
# Cached (%)	99.98	99.97	99.99	99.91	99.96	99.89	99.96	99.94	99.98	99.86
# Uncached (%)	0.01	0.02	0.01	0.08	0.03	0.1	0.03	0.05	0.01	0.13

Additionally, in comparison to the testrun without cache, we can see that the attacking interval has a considerably lower impact on efficiency.

The performance drop-down of LRU in scenario one can be explained by its special setup. As is this setup each callees are called strictly in order, the cache will not be able to keep the last entry in its database until it is called again. In the two other scenarios, because of its emphasis on calling one callee more often than the other ones, performance of LFU can be increased. This fact also explains why there is an efficiency increase in the latter two scenarios.

5.3.4. Real-life testing

We have installed the DNS cache into a real-life testbed architecture and have run it against user-generated traffic from the platform. While within in this architecture customers have the choice to call subscribers within the same service as well as customers in foreign domains (e.g., iptel.org, vortel.net). However, only seldom users do the latter. Hence, DNS resolving requests are focussed on a very close range of DNS names. In this scenario, a DNS cache can have a huge impact on performance, as it will have nearly every time all necessary domain mappings inside its cache.

We have run 10 different measurements of system performance at different times over a period of approximately one month. With each testrun we have run the prototype to answer about 120,000 requests generated from SIP messages. Table 6 shows the distribution of requests that have been directly answered from the cache with regard to those requests that needed to be resolved from the DNS systems, as they have been new domain names. As can be seen from the table, nearly all requests benefit from cache operation.

6. Conclusion

In this work we have presented measurements from our developed two layer security architecture prototype to provide Denial of Service attacks protection for VoIP systems.

The Denial of Service prevention mechanisms have been tested in both laboratory and real-traffic environments. Simulations have been specifically designed to test attacks based on message flooding, malformed message usage, and crafted DNS requests.

The results of the developed IDS/IPS infrastructure against different kinds of flooding attacks proved to be effective in both laboratory and real-life traffic environments. Even though the configuration parameters for the real-life environment are still being tuned, we can say that the addition of the IDS system in the network has been an informative resource to know more about the traffic moved, the events that occur in the network, detect malfunctioning services and devices and of course for detecting attacks against the network.

In relation to the malformed messages detection tool it has been demonstrated that the proposed detection and prevention mechanism is very robust and effective, and that the processing overheads that are introduced, even for the well-formed messages, to the system are minimal.

The DNS cache module also presents very good results in both laboratory and real-life traffic environments. With the tests realized we have been able to see that the proxy was considerably shorter blocked by DNS requests with the cache than it was without it. That permits the SIP proxy to process more messages in time and reduce the risk of dropping legitimate messages.

The developed solution gives enhanced SIP security protection with specialized Denial of Service attacks. Currently, there have not been noticed a substantial increase in attack traffic within VoIP provider networks. However, as we have shown that attacks are both easy to mount while still providing substantial damage to the system, their usage will increase. In the future we are considering different DoS attacks as well as investigating a new threat for VoIP networks: VoIP Spam.

Acknowledgments

This work has been partly financed through the EU project SNOCER, http://www.snocer.org.

We would like to thank VozTelecom (www.voztelecom.com) for providing the real-life testbed.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Spark, M. Handley, E. Schooler, Session Initiation Protocol, 2002, RFC 3261.
[2] M. Garcia-Martin, Input 3rd-Generation Partnership Project (3GPP) Release 5 Requirements on the Session Initiation Protocol (SIP), 2005, RFC 4083.
[3] J. Mirkovic, S. Dietrich, D. Dittrich, P. Reiher, Internet Denial of Service: Attack and Defense Mechanisms, Prentice Hall, Upper Saddle River, NJ, 2005.
[4] H. Sengar, D. Wijesekera, H. Wang, S. Jajodia, Fast detection of denial of service attacks on IP telephone, in: Proceedings of IEEE IWQoSG2006, New Haven, CT, 2006.
[5] Y. Rebahi, Change-point detection for voice over IP denial of service attacks, in: 15. ITG/GI – Fachtagung Kommunikation in Verteilten Systemen, February 2007.
[6] Y.-S. Wu, S. Bagchi, S. Garg, N. Singh, T. Tsai, SCIDIVE: a stateful and cross protocol intrusion detection architecture for Voice-over-IP environments, in: 2004 International Conference on Dependable Systems and Networks, 2004.
[7] E. Chen, Detecting DoS attacks on SIP Systems, in: 1st IEEE Workshop on VoIP Management and Security, 2006.
[8] M. Nassar, R. State, O. Festor, Intrusion detection mechanisms for VoIP applications, in: 3rd Annual VoIP Security Workshop, Berlin, Germany, 2006.
[9] 2004 CSI/FBI Computer Crime and Security Survey, Technical Reports, 2004.
[10] S. Vuong, Y. Bai, A survey of VoIP intrusions and intrusion detection systems, in: 6th International Conference on Advanced Communication Technology, 2004.
[11] D. Sisalem, J. Kuthan, S. Ehlert, Denial of service attacks targeting a SIP VoIP infrastructure – attack scenarios and prevention mechanisms, in: IEEE Network, vol. 20, No. 5 (special issue on securing VoIP), 2006.
[12] D. Geneiatakis, A. Dagiouklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, S. Ehlert, D. Sisalem, Survey of security vulnerabilities in Session Initiation Protocol, IEEE Communications Surveys and Tutorials (2006).
[13] VoIP Security and Privacy Threat Taxonomy. Available from: <http://www.voipsa.org>.
[14] C. Wieser, M. Laakso, M. Schulzrinne, PROTOS: security testing of SIP implementations. Available from: <http://www.compose.labri.fr/documentation/sip/Documentation/Papers/Security/Pape/462.pdf>.
[15] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, HTTP Authentication: Basic and Digest Access Authentication (1999) (RFC 2617).
[16] G. Zhang, S. Ehlert, T. Magedanz, D. Sisalem, Denial of service attack and prevention on SIP VoIP infrastructures using DNS flooding, Principles, Systems and Applications of IP Telecommunications (IPTComm 2007) (July) (2007).
[17] M. Roesch, Snort – lightweight intrusion detection for networks, in: 13th USENIX LISA Conference, 1999.
[18] G. Kambourakis, J. Fiedler, D. Geneiatakis, S. Ehlert, T. Dagiuklas, High availability for SIP based infrastructures: solutions and real-time measurements performance evaluation, Elsevier Computer and Electrical Engineering Journal, submitted for publication.
[19] D. Geneiatakis, G. Kambourakis, C. Lambrinoudakis, T. Dagiuklas, S. Gritzalis, A framework for protecting a SIP-based infrastructure against malformed message attacks, Computer Networks, Elsevier, Amsterdam, 2006.
[20] J. Rosenberg, H. Schulzrinne, Session Initiation Protocol (SIP): Locating SIP Servers (2002) (RFC 3263).
[21] A. Silberschatz, P. Galvin, Operating System Concepts, seventh ed., Wiley, Hoboken, NJ, 2005.
[22] C. Aggarwal, J. Wolf, P. Yu, Caching on the world wide web, Transactions on Knowledge and Data Engineering 11 (1) (1999).
[23] Y. Rebahi, D. Sisalem, J. Kuthan, A. Pelinescu-Oncicul, B. Iancu, J. Janak, D.C. Mierla, The SIP Express Router - An Open Source SIP Platform, in: Evolute Workshop, Guildford, UK, 2003. Available from: <http://www.iptel.org/ser>.
[24] D. Sisalem, S. Ehlert, D. Geneiatakis, G. Kambourakis, T. Dagiuklas, J. Markl, M. Rokos, O. Bontron, J. Rodriguez, J. Liu, Towards a secure and reliable VoIP infrastructure, Technical Reports, 2005. Available from: <http://www.snocer.org>.
[25] Prelude IDS Communication System. Available from: <http://www.prelude-ids.org>.
[26] SnortSam, Firewall Control Plugin for the Snort IDS system. Available from: <http://www.snortsam.org>.
[27] IPSet Firewall. Available from: <http://www.ipset.netfilter.org/>.
[28] Perl compatible regular expressions. Available from: <http://www.pcre.org>.
[29] SIPp – SIP Traffic Generator. Available from: <http://www.sipp.sourceforge.net>.