VoIP Defender: Highly Scalable SIP-based Security Architecture

Jens Fiedler
Fraunhofer Institute for Open Communications Systems - FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
jens.fiedler@fokus.fraunhofer.de

Tomas Kupka
Fraunhofer Institute for Open Communications Systems - FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
tomas.kupka@fokus.fraunhofer.de

Sven Ehlert
Fraunhofer Institute for Open Communications Systems - FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
sven.ehlert@fokus.fraunhofer.de

Prof. Dr. Thomas Magedanz
Fraunhofer Institute for Open Communications Systems - FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
thomas.magedanz@fokus.fraunhofer.de

Dr. Dorgham Sisalem
TEKELEC Corp.
Am Borsigturm 11
13507 Berlin, Germany
sisalem@tekelec.com

ABSTRACT

VoIP services are becoming increasingly a big competition to existing telephony services (POTS / ISDN). The increasing number of customers using VoIP makes VoIP services a valuable target for attackers that want to bring down the service, take it over or simply abuse it to distribute their own content, like SPAM. Hence, the need arises to protect VoIP services from all kinds of attacks that target network bandwidth, server capacity or server architectural constrains. In this article we present *VoIP Defender*, a generic security architecture, called VoIP-Defender, to monitor, detect, analyze and counter attacks relevant for a SIP-based VoIP infrastructure. The VoIP-Defender is highly scalable and can be easily extended with new detection algorithms. Analysis and traffic control can be performed from the SIP layer down to the transport-, network- and MAC layer. VoIP Defender is designed to work fully transparent to clients and SIP servers, and can analyze and filter traffic in real time, which we demonstrate with measurements with our implementation.

1. INTRODUCTION

The Session Initiation Protocol (SIP) [1] has established itself as the de-facto standard for Voice-over-IP (VoIP) services. Several providers are already offering Internet Telephony services based on SIP. The popularity of SIP will likely rise even more with the advent of the IP Multimedia System (IMS) [2], the next-generation telephony network which is also based on SIP.

SIP is deployed in a client-server infrastructure where SIP clients (User Agent clients, UAC) contact a central SIP Proxy (i.e. a server) to manage ongoing sessions. It is a text based protocol designed to establish or terminate a session among two or more partners. The message format is derived from the HTTP protocol, with message headers and corresponding values, e.g. "`From: user@sip.org`" to denote the sender of a message.

As it is generally deployed in the open internet, SIP infrastructures can easily become a target for different attacks from the outside world, including Denial-of-Service attacks, Message Tampering, Call Hijacking and other threats [7] [8] [6].

In this paper we present *VoIP Defender*, an open security architecture that is designed to monitor the traffic flow between SIP servers and external users and proxies. The goal is to detect attacks directed at the protected SIP servers and provide a framework for attack prevention / mitigation. Our focus lies especially on high traffic flooding attacks that can easily overwhelm a proxy's resources in terms of CPU processing power, memory requirements or bandwidth capacity. Hence, VoIP Defender was designed with scalability in mind. As such, we presenent a layered solution with dedicated tasks (traffic monitoring, analysis, decision) with each layer optimized for scalability. With measurements in the FOKUS security testbed we demonstrate VoIP Defender's capabilities to process over 50,000 SIP msgs/s. The SIP traffic is forwarded to individual intelligent extension modules which provide monitoring, attack detection or attack prevention tasks.

In section 2 we explain why an architecture like this is needed.

Section 3 depicts related work. The proposed architecture is described in section 4. Performance measurements have been taken and are presented in section 5. Sections 6 and 7 summarize the work and give an outlook to future work.

2. MOTIVATION

To secure a SIP-based network, a service provider has commonly two different options; Usage of a general IP-based *firewall solution* or a dedicated SIP *Session Border Controller* (SBC).
Traditional firewalls have the capabilities to filter IP traffic up to the interface transport level. Usually they lack proper intelligence for efficient message or stream content analysis. As such functionalities are out of scope for IP firewalls, scalability aspects are only of limited importance, as IP traffic can be filtered in realtime, even in gigabit networks.
A different approach is the use of a SBC [3] to hide and protect the internal network infrastructure. This reveals the existence of the defense entity (the SBC) to the attacker, and it can attack the SBC directly, if it knows an effective way to do this. Additional to its standard tasks, the SBC has to be extended with attack detection algorithms of currently unknown complexity, which introduces also a scalability problem to it.
Hence, protection options are limited to a potential scalable solution that lacks dedicated SIP features, or a SIP-aware solution that exposes itself directly to potential attackers and has unspecified scalability features.

3. RELATED WORK

Several researchers have prosed VoIP security solutions to detect and prevent VoIP-related attacks. However, most of them focus on algorithm intelligence and don't take performance into account.
Niccolini et al [5] propose a security architecture that performs well up to an incoming traffic load of about 850 messages per second. Their solution is tested on notebook computers and thus not directly comparable to our approach.
Schlegel et al propose a Spam over internet telephony (SPIT) prevention framework [12], which focuses only on specific types off attacks to VoIP services (SPIT).
Sengar et al [4] propose a detection framework based on Hellinger distance calculation. They argue that a processing speed of 500 messages per second should be sufficient, based on the fact that many current SIP proxies have a processing capacity of 100 messages per second. Clearly, such an approach would not be feasible within a real distributed DoS attack.

As the proposed architecture is intended to distribute SIP traffic according to SIP transaction and even session state, usual SIP load balancing technologies apply. A basic approach for load balancing is described in [10]. It proposes a single load balancer in front of multiple, partition based SIP servers. The load balancer distributes messages according e.g. to the registered user. In order to allow a more dynamic load balancing, a load-query mechanism is introduced in [9], which allows a better, more dynamic load distribution among the available servers by interchanging load metrics to learn the overall load situation and then to chose a lesser loaded server for new SIP transactions. Figure 1 depicts both approaches.

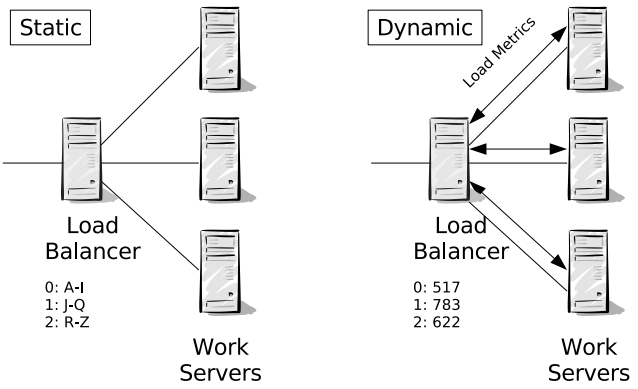

Figure 1: Static vs. Dynamic load balancing

4. VOIP DEFENDER ARCHITECTURE

In order to protect the Point of Service Provisioning (PoSP), all traffic addressed to it must be inspected and a decision must be taken, whether to pass messages or streams to it or not. Therefore, the basic idea is to insert an Point of Protection (PoP), as in classic IP firewall scenarios, but with extended features, tailored to SIP based VoIP signaling traffic. Features are scanning, analysis and filtering functionality, between the PoSP (e.g. SIP server or server farm) and the potentially offensive outer world, holding a mixture of legal users and malicious entities. In figure 2, the general

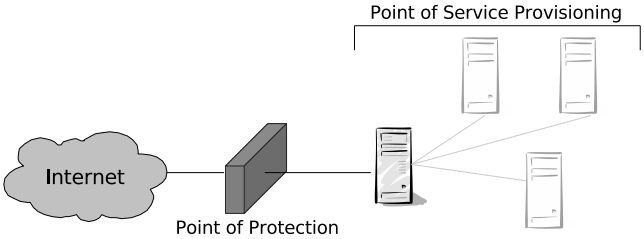

Figure 2: SIP server access

overview is depicted. As pointed out, the PoP must have substantial more features than a conventional IP firewall, as it is expected to recognize and counter also complex attacks on VoIP infrastructures.

4.1 Requirements

We have defined strict requirements for the PoP to achieve the goal of fast, reliable and effective protection of the PoSP.
Transparency: The PoP must be completely invisible from the networking point of view. It must not be possible to gain knowledge about the existence of it by analyzing the network traffic. If the PoP is visible by some means, it can be identified and possibly circumvented or even attacked.

- *No IP routing.* If the PoP acts as a router it modifies IP packets, therefore its existence can be guessed from the outside.

- *No SIP proxying.* A very easy way to intercept SIP traffic is to act as a proxy and use the Record-Route feature from SIP to stay in the message path. This,

of course, announces the PoP to every VoIP user, regardless whether it is doing proxying or acts as Back-to-Back (B2B) User Agent [1]. Even here, the PoP can be circumvented by addressing the PoSP directly, e.g. by its IP address.

Line speed: All traffic must be dealt with in real time. This means that no buffering is allowed. Incoming packets must be sent out immediately after passing the filtering rules. Otherwise it would be possible to detect the PoP by analysis of network delays.
Scalability: The expected types of attacks also include flooding attacks, which feature a very high bandwidth towards the PoSP. In order to analyze this stream, the architecture must be capable to handle this stream. Therefore it must be possible to deploy new capacities into the PoP easily. DDoS attacks can easily reach into the gigabit per second area of bandwith.
Independence: To be as usable as possible, the PoP cannot rely on a special SIP proxy implementation. Hence, no direct communcation channel is established between the PoP and the PoSP. As a consequence, the PoP needs to implement a subset of the SIP logic to successfully follow the operation within the network (e.g. session state changes). As such, the PoP needs to monitor not only incoming traffic from the outside world, but also responses from the PoSP
Extensibility: As it is possible that new types of attacks are discovered, the architecture must be open to allow new algorithms for detection to be integrated fastly. New funcionality should be added or dropped without changing the PoP's architecture.

4.2 Components

The VoIP Defender architecture, as depicted in figure 3, consists of transport level load balancers (TLLB), filter/scanner nodes (FSN), Analyzers and a Decider. Additionally, a central administrative terminal allows user interaction with the architecture. These components are discussed in detail in the following sections.

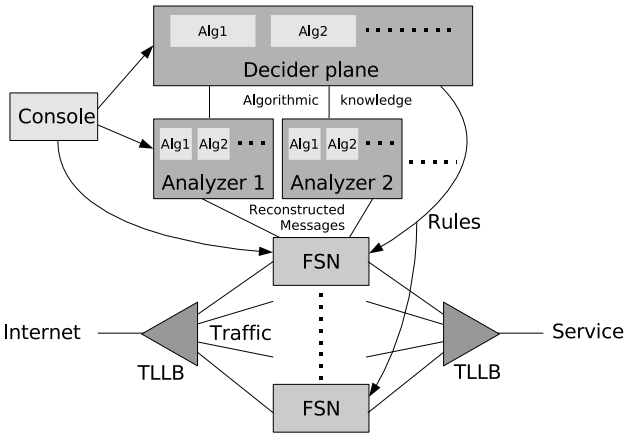

Figure 3: VoIP Defender architecture

4.2.1 Transport Level Load Balancer

The TLLB has the purpose to enable the usage of multiple FSNs, as these are expected to be exposed to high network and rule processing load. The TLLB distributes ethernet frames, which hold IP fragments, over multiple FSNs. Therefore it must be guaranteed, that frames belonging to the same

- IP packet
- UDP datagram
- TCP stream

are delivered to the same FSN, as they are expected to reassemble full SIP messages before applying rules. If e.g. IP packets for a single TCP stream arrive at different FSNs, none of them will be able to reconstruct the TCP stream and the contained SIP message. The TLLB also works in ethernet bridge mode, thus does not change any incoming frames. In figure 3 two TLLBs are depicted. This is due to the fact, that e.g. responses from the PoSP are sent back to the originator over the same TCP stream, as it was received on. Therefore returning TCP messages must pass the related FSN in order to keep the TCP state consistent.

4.2.2 Filter and Scanner Node

The FSN is probably the most important component for a good real time behaviour of the whole architecture. Its primary purpose is to fork incoming and outgoing traffic towards the analyzer nodes (scanning). Its second task is to apply filtering rules to the incoming traffic. Traffic from the PoSP is also forked, as input for the analyzer nodes, and another copy is sent out to the internet, where it is routed normally. Traffic forking is a requisite to gain high scalability in combination with intelligent detection algorithms. Otherwise, if traffic would be completely analyzed before forwardin it to the PoSP, high delays would be introduced, especially during flooding attacks.
As a platform for the FSN, a high end Linux PC has been chosen. The structure of the FSN and the distribution of the entities over kernel and user space are depicted in figure 4. Transparency is achieved by acting as an ethernet bridge,

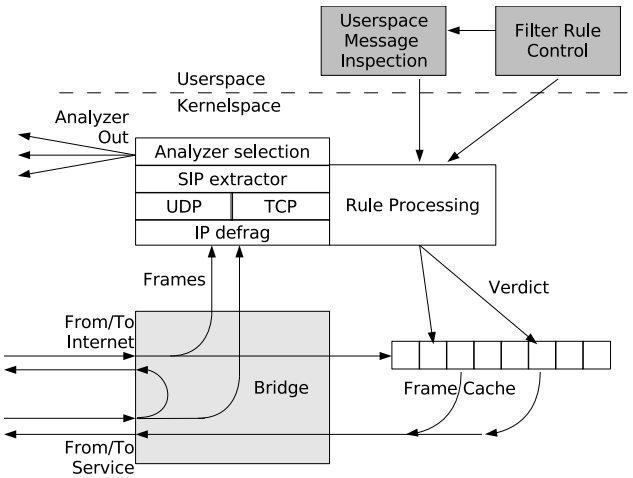

Figure 4: FSN Entities

which intercepts all traffic on the MAC layer and therefore does not change anything about the packets. If packets are allowed to leave the FSN, they leave exactly as they entered

	Kernel	User Space
IP/UDP/TCP/ICMP	yes	yes
regexp	yes	yes
scripts	no	yes

Table 1: Condition Types

it. This also includes all MAC layer information.
A packet, which has been received by the bridge is stored in kernel space. It is then replicated and one copy is send to the protocol reconstruction facility, another copy of the frame is stored in a frame cache, awaiting its verdict from the filter rule application.
As packets from the bridge are ethernet frames, IP defragmentation has to be performed in order to gain full IP packets. After this step, UDP and TCP streams are reconstructed. During this process, references to the involved frames in the kernel queue are stored along with the reconstructed streams. From them, the SIP messages are obtained, which are fed into rule processing and sent to an analyzer node as well, along with protocol meta data, like IP source addresses, number of fragments, timing information, etc.
The FSN is the second point in the system where scalability can be applied. Each FSN has multiple analyzer nodes, which it can feed with traffic. Analyzers rely on a consistent feed of SIP messages, therefore it must be granted that messages belonging to the same VoIP session are processed by the same Analyzer node. The FSN identifies the "Call-ID", "To" and "From" fields from each SIP message and extracts the tags to form a unique session identifier for matching messages to sessions. In order to chose an analyzer, it applies a hashing function to the Call-ID and the From-tag, computes a=*hash mod* n with n being the number of analyzer nodes available. The FSN here acts as a load balancer as described in section 3.
The extracted messages undergo inspection by the kernel space rule chains, as well as possible userspace decision. For this purpose, a dedicated kernel/userspace interface feeds a custom userspace daemon with messages, which then sends back verdicts.
Rules consist of conditions and an action to be taken, if the conditions are met. A condition can be any IP, UDP, TCP or ICMP property. Additionally, a condition may be a regular expression, which is applied to a SIP message. Therefore it is possible to decide about a message also by its content. A condition can also be a custom script, which gets executed by the user space application, inspecting the message flow. As an example, one could implement a time based message judge, which drops every *n-th* message, if a certain, script-defined, condition is met. Table 4.2.2 opposes the condition types and possible execution points. Kernel based rule processing includes filtering by IP, UDP, TCP and ICMP properties as well as applying regular expressions to the reconstructed SIP messages. The user space daemon can additionally execute custom scripts. The verdict of an applied rule is one of:

1. **Accept**: The message is harmless, pass it on
2. **Drop**: The message is malicious, drop it.
3. **Continue**: Apply further rules until one produces a final decision about the message.

Rules are uploaded to the FSN via a userspace daemon, which interacts with the kernel and the userspace rule message inspection daemon.

4.2.3 Analyzer

The Analyzer is the first layer of the intelligence of the detection architecture. It analyzes incoming traffic from the internet and outgoing traffic from the PoSP. This is necessary to keep track of the state of ongoing sessions.
An analyzer node includes the low level functions for each detection algorithm in the system. A low level function is the part of a detection algorithm, which must deal directly with the message flow. Low level functions are expected to produce a result, which can be used along with the results of the other analyzer nodes to decide about the start of an attack, its status and its end. Analyzer nodes are running in parallel to allow easy scaling of the analysis load, which depends on the number and complexity of the deployed detection algorithms, as well as on the expected network load situation. Figure 5 depicts the components of a single analyzer node.

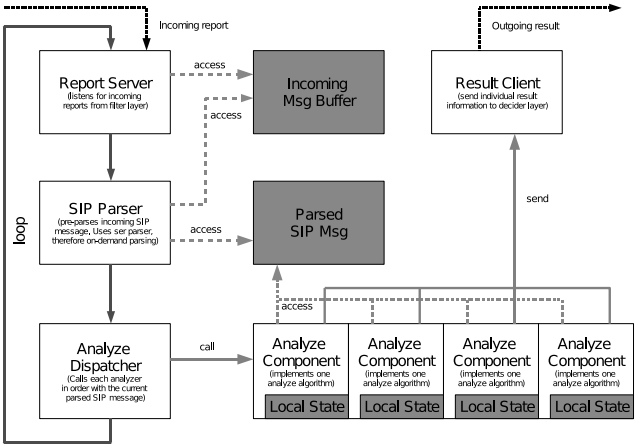

Figure 5: Analyzer architecture

Report Server: It receives reports from the FSNs, which contain SIP messages and meta information about them. These include source and destination IP addresses, transport protocol and number of IP fragments for the message. The received message is queued in the Incoming Message Buffer.
SIP Parser: Incoming message are centrally parsed as SIP messages by the SIP Parser, which generates data structures suitable for the processing needs of the installed detection algorithms. The SIP parser can be programmed by the detection algorithm about the SIP header fields to parse.
Analyze Dispatcher: IT presents each message to the first-layer parts of the installed analysis algorithms.
Analyze Components: They run tests on the message, keep statistics and states about transactions, sessions, etc. They are the parallel part of each algorithm, as they run on multiple Analyzer nodes. They can now easily apply their detection techniques and produce an algorithm specific output, describing the current attack situation, according to that specific algorithm.
Result Client: This entity sends analysis reports from the

Analyze Components to the upper detection layer (Decider). The reports are specific to each analysis algorithm and contructed by the corresponding Analyze Component.

4.2.4 Decider

The Decider is the upper layer of the intelligence of the PoSP architecture. It gathers the output from the analyzers and decides about the actual attack situation. It hosts an entity for each detection algorithm, which is capable to correlate the output of its specific analyzer bottom half function. The decider itself can also be scaled up, by deploying a dedicated decider node for each algorithm. Figure 6 shows the components of the decider node.

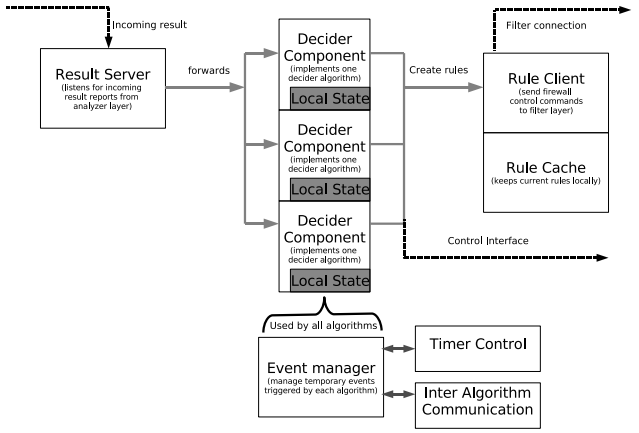

Figure 6: Decider architecture

Result Server: It receives the reports from the first layer parts of the detection algorithms. Reports wear an algorithm-ID, which allows the Result Server to trigger the corresponding algorithm specific decider module.
Decider Component: As in the Analyzer, the Decider also hosts algorithm specific modules. Each decider module matches a corresponding Analyze Component in the Analyzer. Decider modules are the upper layer part of each algorithm. Here each algorithm decides, whether the gathered data indicates an attack, its end or normal behavior. They also decide what measures need to be taken to counter it. Countering means to create rules for the FSN, which will block all malicious traffic to the PoSP matching the rules created by a decider module. In an attack situation, all traffic is still delivered to the analyzer nodes. Thus it is possible to decide, when the attack is over, so the created rules can be removed.
Rule Cache: The Rule Cache entity features static rules, which are created by human beings, or were taken from other systems, where they proved to suppress certain types of attacks (pre-set rules). The Rule Cache also remebers all custom rules, which have been set by the decider modules. This enables the Decider to update newly conencting FSNs with the actual set of rules in charge. Rules consist of a protocol specific condition, like addresses for IP, or port numbers for UDP/TCP, but can also be regular expressions for content matching, or even complex scripts for user space based filtering engines, as introduced in section 4.2.2
Rule Client: This entity is responsible for the communication with the FSNs. It informs them about new rules, or if one is deleted. It also upload the full rule set to newly connecting FSNs, e.g. when they return after a shutdown.
Event Manager: It queues and dispatches events from timers and messages sent by other entities to the single decider modules.
Timer Control: As detection algorithms may have the need for time based actions, a programmable Timer Control modules takes care of generating the appropriate timer events.
Inter-Algorithm Communication: Decider modules may depend their decision about an attack to the status of other detection algorithms. Therefor an algorithm can broadcast its idea of the state of an attack to other modules. As an example, it is thinkable to have a kind of super-decider, which gives an alert if at least two different algorithms have detected an attack.
As an example for Analyzer and Decider co-operation, take DoS detection with the CUSUM algorithm [11]. In short, CUSUM monitors incoming sources, and detects if within a certain timeframe a flow of unrecognized sources appear. For the implementation of CUSUM in VoIP Defender, only the source (IP address) and the reply type from the PsOP (Acknowledge / Deny) are of importance. The Analyzer module for the CUSUM alorithm only extracts this information from the stream of SIP messages and forwards only this essential information to the Decider. The actual CUSUM algorithm implementation is located at the Decider layer.

4.2.5 Console

The adminstrative console is a the central interaction point for the user. Here, individually extension modules can be dynamically started and stopped and the status of each component is delivered to the user. Monitoring, detection and prevention messages and alarms are also gathered at this point.

4.3 Interaction

In section 4 we introduced the components of the protection architecture and their tasks. In this section we discuss the information interchange behaviour, i.e. which component communicates with which other component and what information is interchanged.
TLLB: Traffic entering and leaving it consists of unmodified ethernet frames, carrying the PoSP's payload, the VoIP signalling traffic, probably SIP.
FSN: Traffic between the FSN and the TLLBs also consists of unmodified ethernet frames, carrying the PoSP's payload. Incoming traffic is potentially filtered (packets are dropped), outgoing traffic is not suppressed. The observed SIP traffic is sent over multiple TCP connections to the analyzers. along with the following meta information

- Timestamp when the first frame has been received
- Duration for reception of the whole SIP message
- IP information (source, destination, etc.)
- TCP information (ports, flags, etc.)

The traffic bandwidth sent out by each FSN is at least the sum of incoming and outgoing traffic, plus the meta information.
Analyzer: It receives the FSN's SIP traffic plus meta information. Each Analyzer is connected to at least one decider node over a TCP connection. The information exchanged

is specific for the installed algorithms. The protocol just allows dispatching of the algorithms.
Decider: The decider node itself has an open TCP connection to each FSN. Rules, consisting of conditions and resulting verdict, are text based encoded and sent to each FSN as soon as a decider module installs a rule. The same is for rule removal. Protocol operations are:

- Install/remove rules
- Set a default policy
- Query rules
- Wildcard delete rules

Console: The central console is connect to each other layer through a telnet based command interface. Commands can be issued from a command line or generated through a GUI.

5. MEASUREMENTS

Measurements at the FSN have been recorded in order to see needs for additional optimization or scalability, as this is the point where massive load in terms of network traffic (SIP stream) and processing power (rule application) is expected. All measurements have been taken at one single FSN node.

5.1 Testbed setup

VoIP Defender exists as a prototype implementation in the FOKUS security testbed. Performance measurements have been done here in a private network, to make sure that only the generated messages are processed and no traffic disturbs other attached networks. The setup is depicted in figure 7.

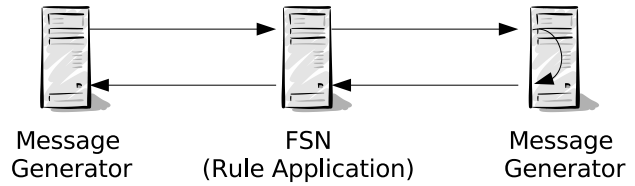

Figure 7: Testbed setup

The message generator can generate UDP messages of preset contents and measure the delay between sending them out and receiving them back from the target machine. For ICMP measurements, the standard UNIX command `ping` has been used, which uses ICMP echo request and echo responses (message type 0 and 8) for delay measurement. All machines are connected by gigabit ethernet with switches between them.

5.2 RTT

Table 2 lists round trip times for ICMP echo/response (`ping`) and UDP ping cycles. The first column shows the number and type of rule entries in the FSN. Times have been measured without any FSN in place as reference, as well as with different numbers of installed rules at two different sending speeds. The rules have been chosen not to trigger ever, thus it was made sure that for each packet, the whole list of rules had to be applied. This test has shown, that even high numbers of rules only marginally influence the delay of network frames through the FSN. Opposing that, the number and complexity of regular expression, applied to UDP packet content, raises latency dramatically, as regular expression processing consumes much CPU power.

Setup	1 Packet/s		10 Packets/s	
	ICMP ping	UDP ping	ICMP ping	UDP ping
no FSN	0.2	0.28	0.19	0.25
0	0.31	0.33	0.26	0.34
10/IP	0.28	0.36	0.31	0.29
100/IP	0.31	0.4	0.34	0.39
1000/IP	0.4	0.51	0.37	0.47
5000/IP	0.54	0.55	0.51	0.54
10000/IP	1.03	0.92	0.95	0.81
17000/IP	1.7	1.64	1.6	1.52
2500/IP&UDP	0.47	0.59	0.47	0.57
5000/IP&UDP	0.55	0.91	0.53	0.94
	regexp = ".*23[a|b].*"			
10/regexp	n.a.	107.03	n.a.	112.73
100/regexp	n.a.	1131.4	n.a.	-
	regexp = "abc[a|b].*"			
10/regexp	n.a.	0.73	n.a.	0.8
100/regexp	n.a.	2.92	n.a.	2.8

Table 2: Measured round trip times (in ms)

5.3 Throughput

In table 3 the recorded throughput is listed along with the observed CPU load at the FSN. The data source sends out a 170 Mbit/s SIP stream, which passes the FSN on its way to the SIP server.
It turns out that simple, IP properties based rules are processed much faster than regular expressions. Therefore, regular expression influence throughput much more than IP based rules, as they require much more CPU power.

Setup	Throughput	Average CPU Load
no rules	170Mbit/s	0.00
100 rules IP	170Mbit/s	0.50
1000 rules IP	130Mbit/s	1.06
1 regexp rule	170Mbit/s	0.03
10 regexp rules	110Mbit/s	0.96
20 regexp rules	70Mbit/s	1.11

Table 3: The Filter Node's Throughput

6. SUMMARY

In the scope of this work, the highly scalable, reliable and efficient VoIP Defender architecture for detecting and preventing attacks on VoIP services has been introduced. Specialized components play together in a scalable way. The architecture is open for extensions in hardware and software, so new detection and prevention algorithms can be specified, implemented and deployed fastly. In case of high traffic, new analyzer nodes and FSNs can be installed to deal with higher loads in terms of algorithmic processing load, filter rule complexity or message traffic.

Measurements show, that simple IP properties based rule application is feasible even for many rules, while regular expression matching is possible, but their number and complexity influences latency and throughput. Thus the use of regular expressions is an expensive resource.

7. FUTURE WORK

The framework, which has been introduced in this work, offers a large variety of functionalities for detection algorithms, like SIP parsing, rule management and scalability features. As now a framework for deploying detection algorithms exist, the focus will be on the development and enhancement of such detection algorithms.
Measurements have shown, that content related matching can be improved. A possible way to do this could be to implement specific SIP related content matchers as userspace filter daemons, as an appropriate interface has been designed and realized.
The architecture consists of many nodes (FSNs, Analyzers, Decider). Currently, when scaling up the system, all components have to be stopped, reconfigured and restarted to reflect the new setup. Therefore it would be desirable to add the feature of a runtime reconfiguration, so that new components can be added on the fly. New components could announce themselves in a kind of a broadcast to the system, which then reconfigures itself and add the new component. The same should be for nodes which fail. They should be taken out of the system by automatic downscaling.

8. REFERENCES

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A.R. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "*SIP: Session Initiation Protocol,*" RFC 3261, IETF, June 2002

[2] 3GPP, TS 23.228 "*IP Multimedia Subsystem (IMS),*" December 2006

[3] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, M. Bhatia, "*Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments,*" http://www.ietf.org/internet-drafts/draft-camarillo-sipping-sbc-funcs-05.txt, IETF, October 2006

[4] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia, "*Fast Detection of Denial of Service Attacks on IP Telephony,*" Proceedings of IEEE IWQoS'2006 , New Haven, CT, June 2006.

[5] S. Niccolini, R. G. Garroppo, S. Giordano, G. Risi, S. Ventura, "*SIP Intrusion Detection and Prevention: Recommendations and Prototype Implementation*", 1st IEEE Workshop on VoIP Management and Security, Vancouver, Canada, Apr 2006.

[6] D. Sisalem, J. Kuthan, S. Ehlert, "*Denial of Service Attacks Targeting a SIP VoIP Infrastructure - Attack Scenarios and Prevention Mechanisms*", IEEE Networks Magazine, Vol 20, No. 5, 2006

[7] S. Vuong, Y. Bai, "*A survey of VoIP intrusions and intrusion detection systems,*" 6th International Conference on Advanced Communication Technology, 2004

[8] D. Geneiatakis; T. Dagiouklas; S. Ehlert; G. Kambourakis; C. Lambrinoudakis; D. Sisalem and S. Gritzalis, "*Survey of Security Vulnerrabilities in SIP", IEEE Communications Tutorials and Surveys*", Vol. 8, No. 3, October 2006

[9] G. Kambourakis, D. Geneiatakis, T. Dagiuklas, C. Lambrinoudakis, S. Gritzalis, "*Towards Effective SIP load balancing: the SNOCER approach,*" 3rd Annual VoIP Security Workshop, June 2006, Berlin, Germany, ACM Press

[10] K. Singh, H. Schulzrinne, "*Failover and Load Sharing in SIP Telephony,*" in International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Philadelphia, PA, July 2005.

[11] B. Reynolds, D. Ghosal, "*Secure IP Telephony using Multi-layered Protection*", 10th Annual Network and Distributed System Security Symposium, San Diego, California, Feb 2003

[12] R. Schlegel, S. Niccolini, S. Tartarelli, M. Brunner, "*SPam over Internet Telephony (SPIT) Prevention Framework,*" Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE, Vol., Iss., Nov. 2006