DoS Attacks Targeting SIP Server and Improvements of Robustness

M.Voznak and J. Safarik

***Abstract*—**The paper describes the vulnerability of SIP servers to DoS attacks and methods for server protection. For each attack, this paper describes their impact on a SIP server, evaluation of the threat and the way in which they are executed. Attacks are described in detail, and a security precaution is made to prevent each of them. The proposed solution of the protection is based on a specific topology of an intrusion protection systems components consisting of a combination of Snort, SnortSam and Iptables applications, the solution was verified in experiments. The contribution of this paper includes the performed comparison of the DoS attacks' efficiency which were tested both without any protection and then with implemented Snort and SnortSam applications as proposed in our solution.

***Keywords*—**Protection against DoS, DoS attacks, IPS, Security, SIP.

I. Introduction

THIS paper deals with one of key issues of IP telephony regarding SIP server robustness against attacks and compares security risks inherent to various Denial–of–Service (DoS) attacks and addresses effective protection against them. We describe DoS attack types and the knowledge is used to test the robustness of the SIP proxy server.

As a result of the ever more widespread implementation of VoIP solutions, PSTN networks are likely to be completely replaced one day. A frequently implemented solution for telephony is a SIP server. Security was not the main goal in developing the application; it was actually rather on a sideline. Yet, security has become more and more important with its increasing popularity.

This situation is simplified by similarity of SIP protocol to HTTP and SMTP protocols, so potential hacker can use existing weakness of these protocols against SIP. One of the most used attacks is DoS (denying service completely or just particularly). On top of used attacks is because of its high efficiency and relatively simple feasibility. For our purpose, we decided to perform all experiments in a real used SIP server platform based on an open–source solution Asterisk.

II. Classification of DoS attacks

Denial of service can be achieved in several ways – flooding a server with malformed, damaged or useless packets as a result of which the server runs out of its resource capacity. The affected server is then unable to communicate with its regular users or process regular requests.

Security threats such as DoS almost do not affect the previous generation PSTN networks. This is due to their closed network topology originally designed to transfer voice information [1]. With the rising numbers of VoIP implementation, the situation is changing. And the users expect the same behavior from the new technology. We can divide DoS attacks into three general classes [2], [3]:

- Flooding attacks – targeting on server resources (CPU, memory or link capacity).
- Misuse attacks – the hacker uses a modified SIP message to cancel or redirect calls or misuses the service. These attacks typically affect a small group of users only.
- Unintentional attacks – the attacker targets the supporting services (DNS, call billing, etc.) in order to distort or restrict the service.

The impact of a DoS attack depends on the target. Targeting a particular client can lead to denying the service to this user only but when a SIP server is the target, no user can use VoIP. When a SIP server is attacked, the provider's reputation also suffers. As a result, the provider may lose some of his existing and potential customers [4].

In recent years, due to their increasing frequency, impact and complexity, DoS attacks became a major issue. But we need to distinguish between intentional and unintentional attacks [5]. As VoIP solutions have been developing fast, many server break–downs are caused by software bugs or bad configuration. Unintentional attacks, on the other hand, are for example instances of crowd frenzy when a high number of users are trying to communicate and the server cannot withstand such a high load (natural disasters, holidays, etc.). This state obviously passes very quickly, as more and more users are served by the server.

Manuscript received December 12, 2011. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 218086.

M. Voznak is with the VSB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava Poruba, Czech Republic (phone: +420 5969991699 ; e-mail: miroslav.voznak@vsb.cz).

J . Safarik is with the VSB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava Poruba, Czech Republic (phone: +420 5969991699 ; e-mail: jakub.safarik@vsb.cz).

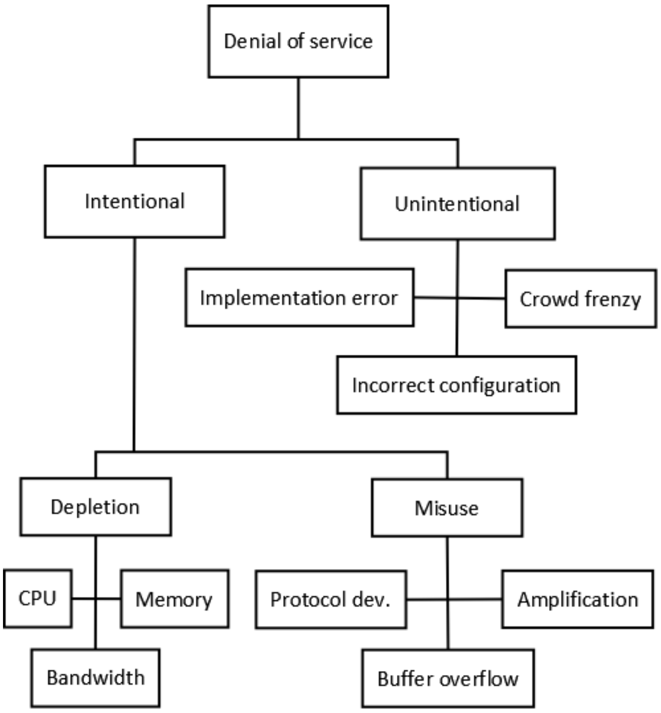

Fig. 1 DoS attacks classification

Identifying the distribution of maliciousness in the IP address space we can apply a Hilbert map of malicious activity created by Team Cymru which can be seen in Fig. 2 [6].

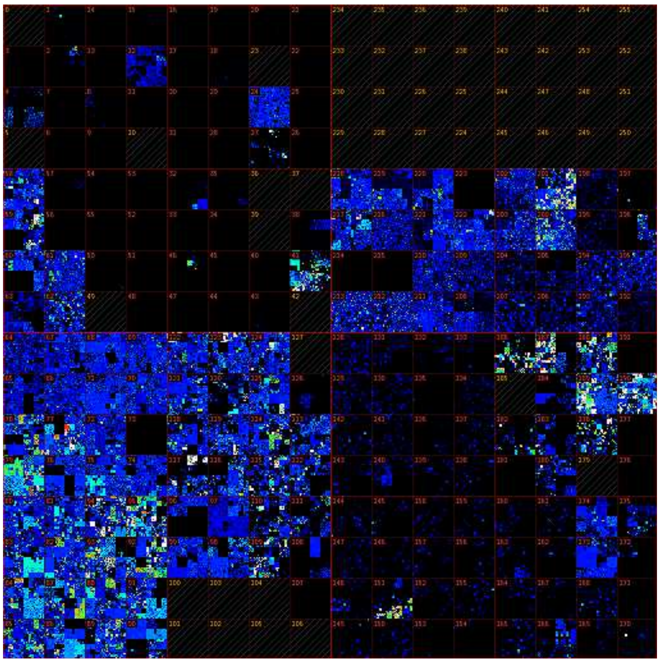

Fig. 2 Hilbert map of Internet malicious activity created by Team Cymru

The map shows the entire Internet address space, each pixel representing a block of 4096 IP addresses. Pixel color represents level of malicious activity produced by machines with IP addresses from the corresponding block, (heatmap scheme: black = none, white = highest). The cumulative distribution function activity computed for /8 network blocks was extracted directly from picture 2 by J. Stanek [7].

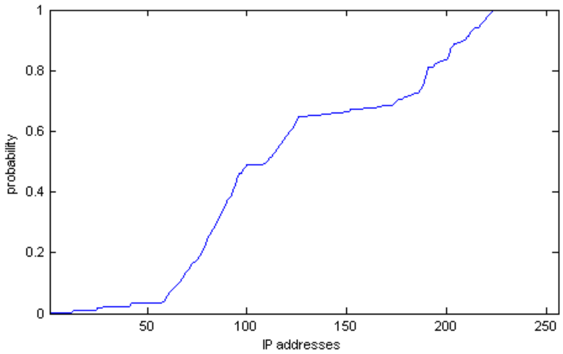

Fig. 3 Cumulative distribution function of the aggregate level of malicious activity

The distribution function depicted in Fig. 3 helps us in identifying IP addresses (or blocks of IP addresses) in /8 network blocks that are being used more often in attacks [6], [7].

A. Memory Depletion Attacks

When a server accepts a SIP message, it has to store small chunks of information. The time for which such information chunks are stored depends on the server mode – stateful or stateless. While the transaction is being performed, the information is kept in memory, and is deleted only once the transaction is closed or timed out. The most frequent attack is a TCP SYN flood attack. The server is flooded with packets with the SYN flag set. The server allocates the necessary resources and responds with packets with SYN+ACK flags. The attacker keeps on sending new packets with SYN flags, and does not respond to packets sent by the server. The server then quickly depletes all memory resources for a new TCP connection and starts refusing regular requests.

Another example of this type of attack is to send highly fragmented packets with certain parts intentionally omitted. The server attempts to request the missing parts and stores the received packets in memory. The useless information is then stored on the server until it is timed out.

B. CPU depletion

Another way to effectively limit the server's ability to process regular requests is CPU depletion. A higher load may be caused by a higher number of requests received or by receiving requests requiring additional complex calculations. The server can become flooded with ICMP packets but sending malformed REGISTER messages creates the same effect with a significantly smaller number of messages.

This is due to the fact that messages are analyzed after the server receives them. Even if the server is capable to process hundreds of regular messages, it can be easily forced to perform different calculations using malformed messages with bogus or invalid data or sent from nonexistent user accounts.

Paradoxically, enabled authentication on a server can trigger off more challenging operations, which makes it easier to deplete CPU resources. When a server uses certificates, the attacker may send a message with an invalid certificate. In the end, the server discovers that this certificate is invalid but the processing of the message had already consumed much of server resources.

C. Bandwidth depletion attacks

This type of attack does not consume resources of the physical server but rather the capacity of the link connecting the server to the network. When the link is not able to transfer regular packets, these are discarded before they can reach the SIP server. This is why it is not possible to distinguish between regular and malicious packets. Using the UDP protocol to transmit SIP messages makes the situation even worst. For obvious reasons, attackers use the stateless UDP protocol with the maximum packet size.

D. Misuse attacks and attacks on SIP features

In general, attacks of this type need only a small number of packets to achieve DoS. They use the weaknesses of the target to their benefit. We can divide these attacks into three subgroups: attacks against the operating system, implementation of TCP/IP stack and attacks using the SIP protocol. Below, we describe the attacks using the SIP protocol.

This attack attempts to deny the users the access to VoIP, i.e. users are the victims of these attacks. This attack does not necessarily affect all users. But from the provider's point of view, this attack is much more dangerous than the above described attacks. In order to be able to carry out this type of attack, the hacker has to be able to capture the network traffic, modify SIP messages or to disguise himself as a different user.

In the case of BYE attacks, one of the parties is convinced that the call was terminated. The attacker uses data captured from the SIP headers to create malicious BYE messages. CANCEL attacks are similar, except that they affect the calls before they are connected. The attacker sends a malicious CANCEL message, using the same sequential number as the INVITE message. Using the same sequential number causes that the malicious message does not have to be authenticated provided it arrives before the final answer from the legitimate user. The only protection against these attacks is to ensure an encrypted transfer of SIP messages.

E. Amplification attack

This is an instance of a distributed denial–of–service (DDoS) attack. The attacker sends packets to broadcast addresses in a specific sub–network with a spoofed source address (victim's IP address). The packet is delivered to all hosts in the sub–network and they respond back to the spoofed address.

The attacker does not need to infiltrate other hosts, he only uses them. Smurf attack and Fraggle attack are examples of this type of DDoS attack.

Another type is loop and forks scenarios. Loop scenario is based on resending request to the same location (server). The SIP specification provides header field named Max-Forwards, which work as well as TTL field in IP header. This field protects server from an endless looping of requests.

Fork attack use N other VoIP servers for amplification of attack. Each account is pointing to another account under other provider. When no stateful server is used, the overload easily consume resources due to message processing.

III. Technology Used

There are many possible ways to secure against DoS attacks. Due to features, performance and abilities of embedded systems; we choose to run the SIP server on this device. The server parameters are as follows.

- Single-core at 2,2 GHz
- 512 MB RAM
- cca 2 GB HDD

The protection mechanism should be a part of this solution. Attacks against the embedded systems are more dangerous due to their relatively lower performance which makes the attacks more efficient. We chose an IPS system, consisting of three applications.

A. Snort

The core of the entire IPS solution is IDS system Snort which detects malicious activity in the network [8]. The detection is based on signatures or detection of anomalies. The whole IDS system is modular, consisting of the following components:

- Packet decoder – Captures packets from network interfaces, prepare them to pre–processing.
- Pre–processor – Prepares or modifies packets before
- the processing (packet defragmentation, URI decoding, reassembling TCP streams, . . .).
- Detection engine – Responsible for attack detection.
- Logging and alerting system – Depending on detection engine, the packet may be used to log activity or generate an alert.
- Output modules – Or plugins, for adding another features.

B. SnortSam

This application operates on the client–server model. It allows Snort to dynamically intervene into IPtables rules. To ensure its proper operation, we need to first patch our Snort installation with a SnortSam plugin.

The client communicates with the Snort's sensor, sends commands to the server (when incident has been detected). The server listens on port 898, applying information from clients to IPtables rules. SnortSam messages are transferred as encrypted, based on preshared passwords which must be same on server and on client A whitelist of non–blockable IP addresses is also available.

The detected traffic is then blocked for some time. Once the attack is over and timed out, the blocked IP is allowed to communicate again. Thus, only malicious traffic that poses a threat to our server is blocked.

C. *IPtables*

An open–source firewall for Linux–based operation systems. It is used to block malicious traffic on a server. In our case, running at the same physical device as a VoIP server,

D. *Solutions limit*

The main limitation of the proposed securit6y mechanisms is dependent on the detection with Snort IDS. If there is no corresponding signature for detecting the attack, the attack cannot be blocked.

Another important factor is the delay between detection and attack blocking. Even if we have appropriate signature of attack, the following steps bring further delays (as in Figure 4). This delay can lead to a weakening of the entire security mechanism. Minimizing this delay is one of the important points in building a defensive solution.

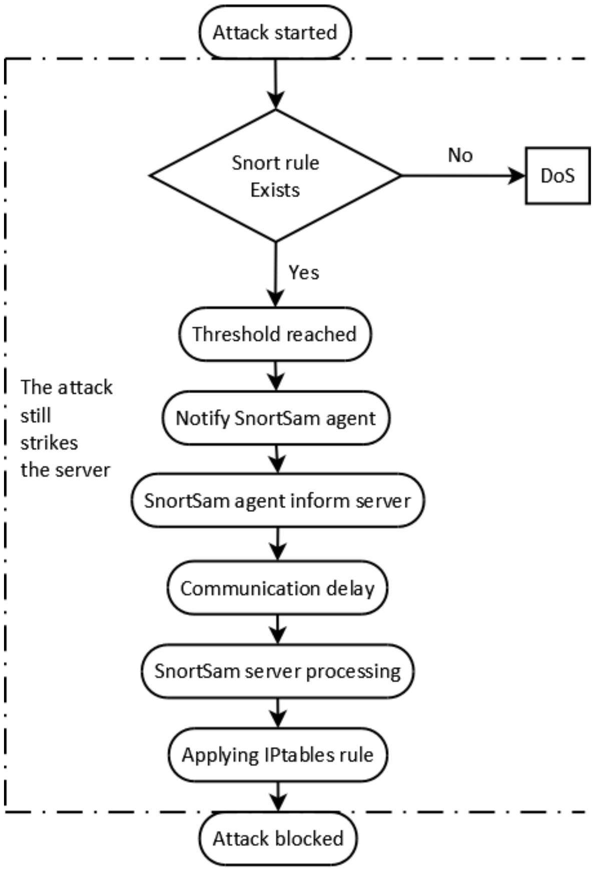

Fig. 4 The delay of a defense mechanism

IV. Results

We created a testing topology to measure DoS effectiveness and for further testing. It consists of SIP proxy server, hacker's PC and some endpoint devices. SIP proxy runs the Asterisk application, and the operation system implemented is Linux for servers – Ubuntu 10.04 LTS (server edition).

The malicious tools applied by hackers with the same OS as the SIP server are as follows [9]-[12]:

- Sipp (in repository named as sip–tester)
- inviteflood
- udpflood
- flood2
- juno
- hping, fping, nmap

A. *Attacks on server CPU using sipp*

The Sipp programme is primarily used to simulate calls and to carry out SIP proxy stress tests [13]. The application is an open source traffic generator that was designed specifically for testing purposes [14]. Sipp is capable of simulation of both UAC and UAS and can also generate both signaling and media traffic. The original source code of SIPp was written by Richard Gayraud and modified by Olivier Jacques, but thanks to its rapidly growing popularity, it quickly became a community tool and many authors joined its development [7].

At the beginning, the popularity of SIPp came from the fact, that it was quite easy to use. The original simple command line interface was intuitive and provided all the necessary functionality for generation of SIP testing communication. During the development, the CLI was additionally extended by usage of XML files specifying the SIP communication scenarios and these scenarios were extended by the possibility of insertion of data from external CSV files.

Another big advantage of SIPp is that it was (and still is) open source, written in C++. Therefore every user can easily implement any functionality according to his needs. Many patches for SIPp were created, extending its functionality in many ways. Thankfully, the maintainers of SIPp did not allow the tool to become a multi-branched unmanageable monster and restricted the way the patches were being adopted by the stable version of SIPp. The patches that did not make it to the stable version were made accessible in the project webpage for anyone to use or improve. Thanks to that, SIPp remained relatively simple and easy to use but there are plenty of extensions available [7], [15].

Sipp, with a simple upgrade of the call scenarios, can carry out malicious calls on SIP proxy [16]. These calls are intended to overload server's CPU. Figure 5 shows the impact of these attacks on the server. The attack scenario applied was the same for each attack. Sending malicious packets started in 10 s and continued for 60 s. Another 30 s shows the time for which the server is still inhibited by the attack.

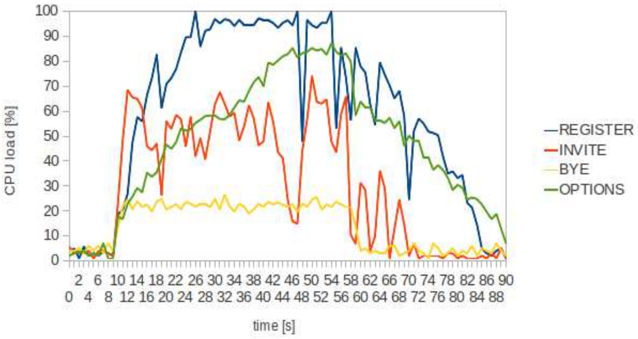

Fig. 5 The impact of different attacks on a server’s CPU load

To enable the comparison of the efficiency of individual malicious SIP messages, the messages had been sent to the SIP server with the same rate (250 messages per second). For instance register flood attack is run by following command:

```
sipp -sn uac 192.168.0.10:5060 –sf reg_notag.xml -m 100000
-r 250 -s 1003
```

Clearly, the most effective SIP messages to attack a SIP server are REGISTER and OPTIONS. In the first case, the endpoint could not register or make calls, though running calls was not affected (the RTP stream only between endpoints). Sipp attacking scenario is shown below.

```
<?xml version=”1.0” encoding="UTF—8”?>
<scenario name=”SIP DOS REGISTER”>
  <send>
    <![CDATA[
      REGISTER sip:[service]@192.168.0.10:5060 SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port]
      From: <sip:[service]@[local_ip]:[local_port]>
      To: <sip:[service]@[remote_ip]:[remote_port]>
      Call—ID: [call_id]
      Cseq: 1 REGISTER
      Contact: sip: [service]@[local_ip]:[local_port]
      Max—Forwards: 70
      Subject: Performance Test
      Content—Type: application/sdp
      Content—Length: O
    ]]>
  </send>
</scenario>
```

Fig. 6 Example of xml attack scenario

Paradoxically was the scenario in Figure 6 one of the most effective register attack scenario. It’s because of his simplicity.

When SIP proxy doesn’t have enough information in register message, it tries to find or guess missing data. This behavior leads to a further increase in computional load (regular SIP header is shown in fig. 7).

```
REGISTER sip:192.168.0.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.0.3:15068;branch=z9hG4bK-
d8754z-4bbaa5ad277e3c56-1---d8754z-;rport
Max-Forwards: 70
Contact:
<sip:1001@192.168.0.3:15068;rinstance=e7238becce042b76
>
To: "1001"<sip:1001@192.168.0.10>
From: "1001"<sip:1001@192.168.0.10>;tag=f4c01cef
Call-ID:
NDE5NGI0NTY0ZDk3NDNjNzg0ZGVlN2YyMjNmZmIxO
Dk.
CSeq: 2 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE,
REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
User-Agent: X-Lite 4 release 4.0 stamp 58832
Authorization: Digest
username="1001",realm="asterisk",nonce="5935b1a9",uri="si
p:192.168.0.10",response="79c8c8d3c37e2f026a2a7503d958
713a",algorithm=MD5
Content-Length: 0
```

Fig. 7: A regular SIP register header

OPTIONS flood caused merely a delay in request processing, yet the situation deteriorated as the attack continued. In the end, not a single endpoint was able to register or make calls. The relatively long time necessary for the server to recover (in both cases) was rather surprising.

The delay in connection was evident in the attack performed by means of INVITE messages. Some calls failed to be connected at all. The attack was performed by a non–existing source user.

Attacks performed by means of BYE, CANCEL and ACK messages returned almost the same results (the figure illustrates only the attack by means of the BYE and CANCEL message).

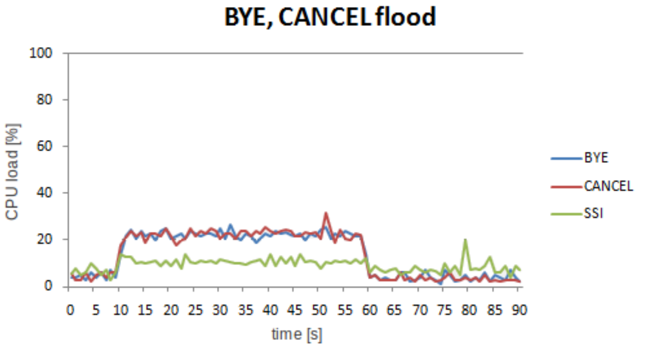

Fig. 8 Impact of BYE and CANCEL flood attacks

During the attack, no call or registration was affected. BYE and CANCEL were not sent to end a particular call.

Security precautions against all these attacks include Snort rules tracking the number of messages sent to the SIP server from a particular source address. The blocking rules were similar in most cases, like this Snort rule for blocking unwanted register flood.

```
alert udp $EXTERNAL_NET any ->
   $SIP_PROXY $SIP_PORT (msg:"SIP
   DoS attempt(registerflood)"; content:"REGISTER sip";
   detection_filter:track by_src, count 50, seconds 5;
   classtype:misc-attack; sid:1000001; rev:1; fwsam:src,
   10min;)
```

Fig. 9 The example of Snort rule.

Where the limit for messages was exceeded, the blocking rule was activated on the firewall by snortsam server. ¨

The time of blocking malicious traffic was set to 10 minutes. After this interval is the blocked IP allowed to communicate with server. Snortsam automatically extend the time of blocking, when the attack is still detected. Whole solution than really effective against actual attacks and can simply converge to non-blocking state without human intervention when the attack ends.

The CPU load with the activated IPS system was about 9% during all these attacks (fig. 10).

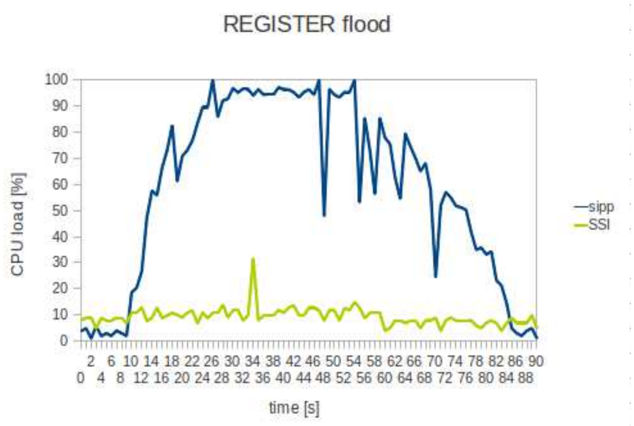

Figure 10: The impact of an attack with (SSI) and without the protection.

The attacker could be sending all the above mentioned malicious messages at a higher rate. In this way each malicious message can consume up to 100% of the server's CPU. Just to compare, the INVITE messages need 10 times higher rate than the REGISTER messages to consume a similar load of the affected machines CPU. The INVITE messages can also send the inviteflood application and create a situation very similar to the flooding with UDP packets (the same is true for any attack with a high rate of packets sent). Differences between applications can be seen in Figure 11.

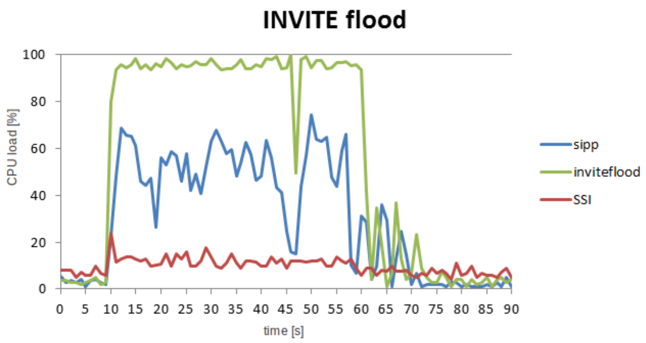

Fig. 11 Inviteflood using a variety of applications

B. *Link flooding attacks*

Unlike the above mentioned attacks, udpflood only floods the target destination with useless UDP packets. These packets contain a sequence from 1 to 9, followed by zeros. The packet size is 1400 bytes, and the tool can spoof the source address. The syntax of *udpflood* coomand is following:

```
./udpflood <source_ip> <destination_ip>
    <source_port> <dest_port> <number_of_packets>
```

The CPU load is very low during the attack but all communication with the server is blocked due to a high volume of traffic. This traffic was in tests about 10.9 MB/s, which is almost the bandwidth of the fast Ethernet line (100Mb/s) between tested computers. Another delay brings virtualization of whole testing concept and upper layer protocols.

Blocking the traffic on server's interface is useless as the link would still be flooded. There is no efficient protection to be applied on the server, it is only possible to eliminate the impact of such an attack.

C. *TCP SYN flood attack*

The last type of attack against SIP proxy tested was to flood it with TCP SYN flag set packets. We used flood2 and juno applications. The Juno tool is especially dangerous as it can be easily upgraded to spoof the source address and ports.

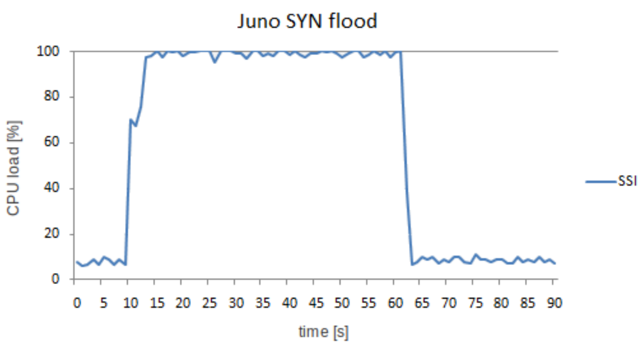

Figure 12: The server's CPU load during TCP/SYN flood attack

When the attack was launched, the connection with the server was lost almost instantly. Detecting this attack is simple but surprisingly useless. Even with an active firewall rule, Snort still analyzes the malicious traffic and the server's CPU load approaches 100%, due to small computing power of test machine.

D. Assessment of results

The performed tests clearly indicate that SIP proxy is rather vulnerable to DoS attacks. As the server runs on a limited physical machine, only very basic protection mechanisms against certain DoS attacks can be implemented.

TABLE 1: IMPACT OF ATTACKS ON THE SERVER

Attack type	VoIP server CPU impact	Threat	Communication with VoIP server
REGISTER flood	High CPU usage at a small rates	Very high	Impossible
INVITE flood	CPU load based on rates	High	Impossible
ACK,BYE, CNL flood	Low CPU load	Low	Very limited, delayed
OPTIONS flood	Gradually increasing CPU load	Very low	Impossible
UDP flood	Low CPU load, line overloaded	Very high	Impossible
TCP SYN flood	High CPU load, all TCP conn. lost	Very high	Impossible

This system consists of the following applications: Snort, SnortSam and Iptables. The tests proved that the analysis of the server's traffic does not significantly affect server's performance (except for TCP SYN flood attack).

The most dangerous attacks include flooding with REGISTER, INVITE and OPTIONS messages, link bandwidth depletion using udpflood and TCP SYN flood attack. The attacks using malicious ACK, BYE or CANCEL messages are harmless at lower rates, with the same impact as udpflood at higher rates. No effective protection to be applied directly on the server exists against certain attacks. In this case, a more secure network topology is the only solution (Figure 13).

The main change in this topology is the inclusion of a demilitarized zone (DMZ). It is located between two firewalls (inner and outer). The purpose of this zone is to separate the safe inner part from the rather dangerous outer part of the network. Both firewalls run SnortSam agents so rules can be dynamically applied on both machines.

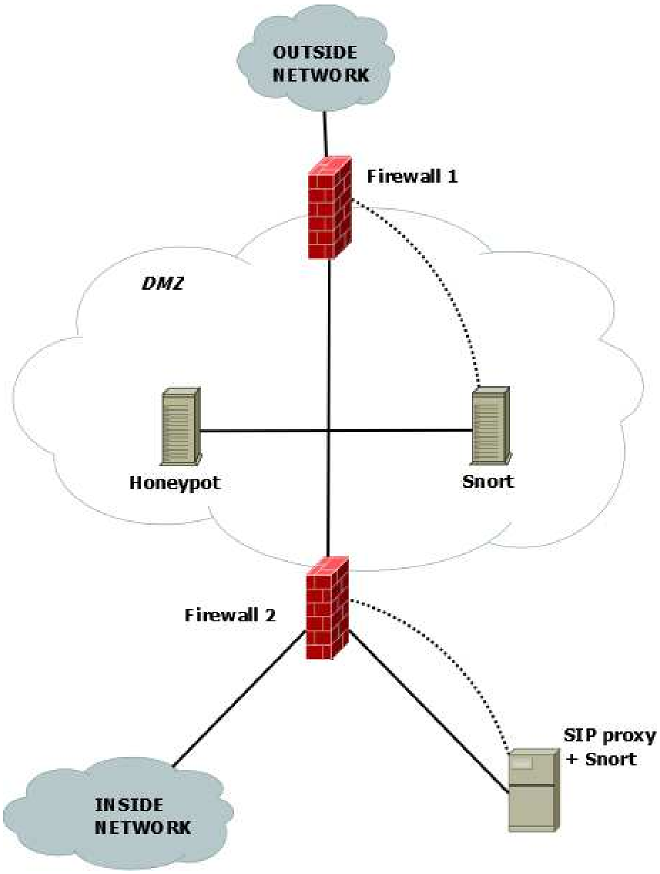

Fig. 13: The proposal of a safer topology.

The inner firewall (marked as Firewall 2) serves to protect the SIP server against the attacks from inside of the network. All traffic to the SIP server has to pass through at least one firewall. The safe inner network should be implemented as a matter of course. The potential attack from inside of the network would affect many users. Using encryption, VoIP VLANs and methods such as ARP inspection and DHCP snooping should provide an adequate response to possible security breaches. The implementation of a QoS mechanism should further reinforce the protection.

A honeypot located in the DMZ is an inspiration for further security precaution to be implemented.

Another security measure is the use of encrypted transmission of VoIP protocols. But if we do not have full support for encryption (both signalization and voice stream) on endpoint equipment, encryption cannot be deployed.

V. CONCLUSION

DoS attacks can be carried out in many different ways. We tested their efficiency in practice and documented the results. This article maps the most frequently used attacks of today and evaluates the risk inherent to each of them. The resulting solution is an IPS system based on the Snort application. This application is combined with two other – SnortSam and

Iptables. The disadvantages of this solution include the delay between the detection and response (typically where the firewall is not on the same physical machine). If the attacker eliminates the IDS system, the whole protection system is useless. The impact of certain attacks can only be reduced by implementing changes to the topology. In this paper, we propose to reinforce the topology's security by introducing a demilitarised zone. The argument here is the impact of udpflood and a TCP SYN flood attacks. The paper also mentions other security precautions which help to enhance the server endurance against attacks in general.

As a result of attacks, the high computing capacity can be significantly reduced. This can be prevented by using parallel computing and a link with high capacity (Etherchannel, optical cables, . . .). However, such measures can increase the cost of the proposed solution slightly. The solution proposed in this article should ensure a basic level of protection suitable for small and middle–size offices or detached workplaces requiring their own VoIP solution.

The contribution of this paper includes the performed comparison of the DoS attacks' efficiency. It was tested both without any protection and then with implemented Snort and SnortSam applications as proposed in our solution.

References

[1] B. Johnston, *SIP: Understanding the Session Initiation Protocol*.London: Artech house, 2009.

[2] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend, H. Schulzrinne, *SIP SECURITY*. Wiley, 2009.

[3] M. Li, M. Li, X. Jiang, "DDoS Attacks Detection Model and its Application," *WSEAS TRANSACTIONS on COMPUTERS*, Issue 8, Volume 7, August 2008, pp. 1159-1168.

[4] D. Sisalem, J. Kuthan, T. Elhert, Denial od Service Attacks Targeting SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms. *IEEE Network*, 2006.

[5] S. Malliga, A, Tamilarasi, "A proposal for new marking scheme with its performance evaluation for IP Traceback," *WSEAS TRANSACTIONS on COMPUTER RESEARCH*, Issue 4, Volume 3, April 2008, pp. 259-272.

[6] Hilbert map of malicious activity on the Internet, Available : http://www.team-cymru.org/Monitoring/Malevolence/maps.html (URL)

[7] J. Stanek, "DoS and DDoS attacks on the SIP protocol," In *Diploma thesis*, Charles University in Prague, Faculty of Mathematics and Physics, Prague, 2010.

[8] R. Rehman, *Intrusion Detection Systems with Snort*. New Jersey: Prentice Hall PTR, 2003.

[9] M. Voznak, F. Rezac, K. Tomala, SIP Penetration Test System, In *Proceedings 34th International Conference on Telecommunications and Signal Processing*, Baden near Vienna, Austria, 2010.

[10] M. Voznak, F. Rezac, "Web-based IP telephony penetration system evaluating level of protection from attacks and threats," *WSEAS Transactions on Communications*, Volume 10, Issue 2, February 2011, Pages 66-76.

[11] D. Endler, M. Collier, *Hacking Exposed VoIP*. McGraw–Hill Osborne Media, 2009.

[12] M. Voznak, J. Rozhon, "SIP back to back user agent benchmarking," In *Proceedings 6th International Conference on Wireless and Mobile Communications*, art. no. 5629122, Valencia, 2010, pp. 92-96.

[13] M. Voznak, J. Rozhon, "SIP infrastructure performance testing," In *Proceedings 9th WSEAS International Conference on Telecommunications and Informatics*, Catania, 2010, pp. 153-158.

[14] SIPp - Open Source tool, Available: http://sipp.sourceforge.net/ (URL)

[15] M. Voznak, F. Rezac, "SIP threats detection system," In *Proceedings International Conference on Data Networks, Communications, Computers*, Faro, 2010, pp. 125-130.

[16] M. Voznak, F. Rezac, "Threats to voice over IP communications systems," *WSEAS Transactions on Computers*, Volume 9, Issue 11, November 2010, Pages 1348-1358

Miroslav Voznak holds position as an associate professor with Department of Telecommunications, Faculty of Electrical Engineering and Computer Science (FEECS) VSB-Technical University of Ostrava, Czech Republic. He received his M.S. and Ph.D. degrees in telecommunications, dissertation thesis "Voice traffic optimization with regard to speech quality in network with VoIP technology" from the Technical University of Ostrava, in 1995 and 2002, respectively. The topics of his research include next generation networks, IP telephony, speech quality and network security. He is a member of the editorial boards of several journals and conference committees of international scientific conferences, a member of IEEE, the Czech Higher Education Development Fund Council for technical fields and the Scientific board of FEECS in Ostrava. He has been involved in research activities of CESNET association since 1999.

Jakub Safarik received his M.S. degree in telecommunications from VSB – Technical University of Ostrava, Czech Republic, in 2011 and he continues in studying Ph.D. degree at the same university. His research is focused on IP telephony, computer networks and network security. He has been involved in research activities of CESNET association since 2011.