# Evaluation of Security and Countermeasures for a SIP-based VoIP Architecture

Marius HERCULEA, Tudor Mihai BLAGA, Virgil DOBROTA
*Technical University of Cluj-Napoca*
*Faculty of Electronics, Telecommunications and Information Technology*
*26 – 28 George Baritiu Street, 400027 Cluj-Napoca, ROMANIA*
*E-mail: marius_hm@student.utcluj.ro,{tudor.blaga,virgil.dobrota}@com.utcluj.ro*

## Abstract

*Evaluation tests were conducted on Asterisk IP-PBX and several SIP hardware and software clients from the security point of view. The tools involved allow scanning, enumerating, fingerprinting, UDP flooding, session and application hacking, traffic interception, eavesdropping, session teardown and RTP media injection/mixing. The responses were interpreted in regards to the capabilities of the targeted systems to make/receive calls, recover after the attacks, and provide debugging/error responses during the attacks. After testing for vulnerabilities, countermeasures were applied by encrypting the media stream, configuring a firewall and installing an Intrusion Detection/ Prevention System.*

## 1. Introduction

Voice over IP (VoIP), seen as an alternative to the traditional public-switched telephone network is emerging as a successful new trend in telecommunications. It is compatible with a variety of platforms (Linux, Windows, mobile devices). It uses a variety of protocols (SIP, RTP, SRTP, SCCP) that depend highly on the preexisting data networks and services (routers, DNS, DHCP, VLANs and so on).

As many new technologies, VoIP comes with security issues that add up to those inherited from the data architecture on which it is being built. The new protocol implementations have yet to undergo detailed security analysis. A rapidly increasing number of vendors that integrate Session Initiation Protocol [1] into their products have left the door opened for SIP-specific attacks [2]. Being aware of this, the only real solution is a well planned defense that extends the current security policy and integrates the new found concerns into the plan. The following chapters provide a classification of types of attack, and then present the architecture and the results from testing the SIP devices describing in the end the implemented countermeasures.

## 2. Types of attack

The first step in a successful attempt to hack the VoIP network is information gathering and profiling. After discovering SIP devices present in the network, by scanning, enumerating and fingerprinting, the door is open to a variety of security tests: UDP flooding, INVITE messages flooding for session hacking, traffic interception, eavesdropping, session teardown, specific attacks with REGISTER messages, RTP media injection/ mixing.

The information gathering phase can be done by detecting online hosts, and then enumerating the SIP based ones. Fingerprinting [3] is achieved by identifying a unique pattern of responses or behavior from each type of SIP device when receiving unusual requests (malformed, false or legitimate).

Flooding is a technique that allows an attacker to send a large number of packets which the target accepts and tries to process, delaying or dropping any legitimate traffic, thus resulting in denial of service (DoS). All SIP devices support UDP so flood attacks through this protocol have a high probability of occurrence. One of the many DoS methods involves creating false INVITE messages and sending them to a SIP device. Flooding is done with subsequent INVITE requests so the SIP target interprets each of them as independent call dialog initiation events. Because it's a SIP specific attack, the INVITE flooding qualifies as a session and application hacking method. Manipulating the SIP signaling can lead to some really dangerous forms of security breaches. An attacker can use carefully forged REGISTER messages and add

contacts for one user remove a registered user or substitute a legitimate contact with an illegal one. In SIP, BYE requests are sent between SIP phones to announce completion of the call. If the attacker sends a forged BYE request to a SIP phone previously engaged in a conversation, the result will be a session teardown.

SIP devices are also vulnerable to RTP media injection or mixing, resulting in new audio to be inserted into the conversation that is heard by the communicating users.

The SIP environment is also vulnerable to traffic interception. One could intercept a RTP stream from a conversation and extract the audio information (eavesdropping).

The approach for securing VoIP should be an offensive one. Testing the SIP environment with attack tools allows for early discovery of security issues, before a real attack could take place. The following table illustrates the tools [4] that can be used for each type of attack. All of them are available on the Internet for free download.

Table 1. List of tools/types of attacks

| Tools \ Tests | Scan | Enumerate | Fingerprint | Intercept traffic/ Eavesdrop | UDP flood | INVITE flood | Session teardown | RTP media injection | RTP media mixing | REGISTER attacks |
|---|---|---|---|---|---|---|---|---|---|---|
| Cain&Abel | X | | | X | | | | | | |
| Inviteflood | | | | | | X | | | | |
| Netdiscover | X | | | | | | | | | |
| Nmap | X | X | X | | | | | | | |
| add_registration | | | | | | | | | | X |
| erase_registration | | | | | | | | | | X |
| reghighjack | | | | | | | | | | X |
| rtpinsertsound | | | | | | | | X | | |
| rtpmixsound | | | | | | | | | X | |
| SIPBomber | | | | | | X | X | | | X |
| SIP_rogue | | | | X | | | X | X | X | X |
| SIPSack | | X | X | | X | X | | | | X |
| SiVuS | | X | | | X | X | X | | | X |
| Udpflood | | | | | X | | | | | |
| Teardown | | | | | | | X | | | |
| Wireshark | | | | X | | | | | | |

## 3. VoIP architecture

The VoIP environment uses the following equipments: a PC with Fedora Core 8 operating system, running Asterisk IP PBX [5] version 1.4.18-trunk-r81432M; a GXV-3000 hardware videophone version 1.1, software 1.0.1.7; a laptop on a WindowsXPsp2 platform running the following softphones: X-lite v3.0, X-PRO v2.0, eyeBeam v1.5.7, PhonerLite v1.42, Snom360 v5.3, LynxPhone v0.6.0. The second laptop has Fedora Core 8 or BlackTrack v3 Linux distribution and WindowsXPsp2 as operating systems. It is used for security testing. A Linksys wireless WIP330 VoIP phone, the HTCx7500 pocket PC with Sjphone build 1.60.303c installed on a Windows Mobile 5 OS and a Linksys WRT54GR wireless router complete the implemented scenario. In this architecture (Figure 1) Asterisk acts as a VoIP Gateway between the PSTN, mobile service and SIP-based LAN.

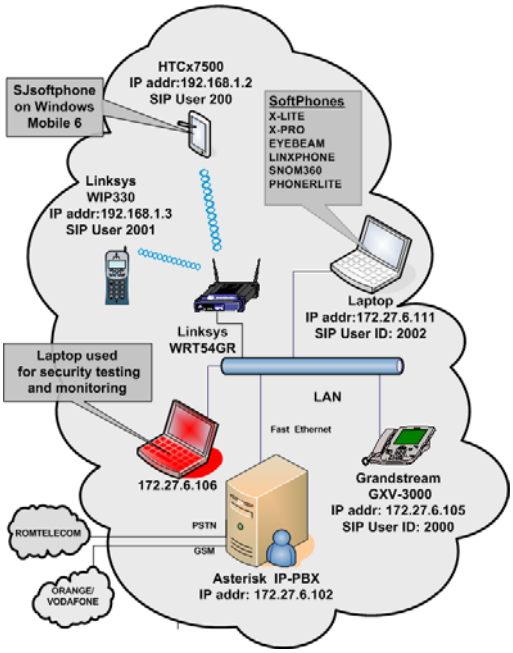


Figure 1: Architecture used for security testing

## 4. Attacking the VoIP environment

In the formentioned architecture, 6 types of tests were conducted: flooding trough UDP by targeting the softphones, hardphones and the Asterisk proxy; flooding SIP phones and Asterisk with INVITE messages; VoIP traffic interception, eavesdropping, REGISTER based attacks and session teardown with BYE messages. Other attacks where performed using the available VoIP environment that are not presented in this paper, due to the limited space.

### 4.1 DoS with UPD flood attacks

First type of tests involved flooding with UDP packets while the phone was in the "on-hook" state. In

the second test devices were flooded after a call was established. Using *udpflood* tool a total of *1000000* packets were sent through interface *eth1* from the source IP address *172.27.6.106* to the destination address *172.27.6.105* using source port *9* and destination port *5060*:

```
#udpflood eth1 172.27.6.106 172.27.6.105 9
5060 1000000
```

During the first type of test only LynxPhone and X-lite could establish a connection, but the audio signal was delayed and degraded. The WIP330 wireless VoIP phones reboots after sending approximately 400.000 packets. If we stop the attack before the 100.000 packets threshold, both the wireless SIP-phone and SJsoftphone installed on the HTCx7500 recover. In addition to this, HTCx7500 experienced low OS responses during the test.

The second attack showed that eyeBeam, LynxPhone and X-lite continue to operate but audio quality was poor. WIP330 could establish a connection but no audio was present on both ends. All the SIP phones tested, except WIP330, resumed normal functionality after stopping the UDP-flood attacks.

Asterisk was tested with *udpflood* in two scenarios. First, a connection between two phones was attempted while running the tool against Asterisk; second the proxy was attacked when forwarding a RTP stream from one phone to another. The first attack had no effect on the call establishment between the two end-points, but the second one, resulted in the inability to hear any audio on both ends after the connection succeeded.

### 4.2 Flooding SIP phones with INVITE messages

GrandStream was first tested using i*nviteflood* tool with the following parameters: outbound interface *eth1*, source IP address *172.27.6.106*, destination IP address *172.27.6.105*, the number of packets to be sent *1000000*, and the destination port *5060*:

```
#inviteflood eth1 2000 172.27.6.106
172.27.6.105 1000000 –D 5060
```

During the attack the phone rings and any attempt to start a connection fails. The same tool was used for all the softphones and the wireless Linksys hardphone. Attacking X-lite, eyeBeam, LynxPhone Snom360 and SJsoftphone required knowing their random port numbers chosen for SIP signaling. Snom360 proved harder to flood because it uses a tag of the form *line=random_string* needed in the INVITE request. The WIP330 wireless SIP-phone reboots after sending 200.000 packets. Table 2 provides the obtained results.

Table 2: Summary of the results for SIP phones

| Phone | Interface usable? | Make/receive calls? | Recovered after attack? |
|---|---|---|---|
| GrandStream | Yes | No | Yes |
| X-Lite | No | No | No |
| eyeBeam | No | No | Yes |
| X-Pro | No | No | No |
| LynxPhone | Yes | No | No |
| PhonerLite | No | No | Yes |
| Snom360 | No | No | No |
| SJsoftphone | No | No | No |
| WIP330 | No | No | No |

### 4.3 Flooding Asterisk with INVITE messages

For the Asterisk proxy, tests were carried out in a specific order: sending INVITE messages for a nonexistent SIP phone; for a valid SIP phone and invalid IP domain address; a nonexistent SIP phone and invalid IP domain address; a valid SIP phone. Depending on the scenario, the proxy uses more or less resources to process and generate messages. Results for each attack are presented in Table 3. The related commands were:

```
#inviteflood eth1 6666 172.27.6.102
172.27.6.102 1000000
#inviteflood eth1 2000 193.226.2.31
172.27.6.102 1000000
#inviteflood eth1 6666 193.226.2.31
172.27.6.102 1000000
#inviteflood eth1 2002 172.27.6.102
172.27.6.102 1000000
```

Table 3. Results of the attacks

| Targeting the proxy with INVITEs of a: | Make/ Receive calls? | Code Response received from the proxy | Proxy Error displayed |
|---|---|---|---|
| Nonexistent phone | No | 404 Not Found, 500 Server Internal Error | Too many opened files |
| Valid phone and invalid IP Domain Address | No | 404 Not Found, 408 Request timed out, 500 Server Internal Error | Too many opened files |
| Nonexistent phone and invalid IP domain address | No | 404 Not Found, 500 Server Internal Error | Too many opened files |
| Valid phone | No | No response | Too many opened files |

### 4.4 VoIP interception and eavesdropping

The goal is to intercept the RTP packet flow between GrandStream GXV-3000 and eyeBeam running on a laptop, during a conversation. First, using the Cain&Abel tool [6], the two victims` MAC addresses were discovered. The same tool was then used to send an unsolicited ARP reply to the GrandStream phone, fooling it to think that the MAC address of the X-lite laptop has changed to the attacker`s MAC address. The same had to be done with the laptop which runs X-lite. Wireshark sniffs the intercepted RTP packets transmitted between the two end-points during a call and then processes them to extract the audio conversation both ways. This method is useful only when the attacker has access to the network.

### 4.5 Registration addition, eraser, highjacking

Using the *add_registration* tool, a special REGISTER message was sent to the Asterisk registrar. The command required information about the outbound interface eth1, the target user 2002 ,the new contact, 2001, to be added, the IP address of the target domain, 172.27.6.102, and the IP address of the target proxy/registrar, 172.27.6.102:

```
./add_registration eth1 2002 2001
172.27.6.102 172.27.6.102
```

Prior to the registration addition the Asterisk showed the following peers as registered:

| Name/username | Host | Port | Status |
|---|---|---|---|
| 2002/2002 | 172.27.6.111 | 53020 | OK (3 ms) |
| 2001/2001 | 172.27.6.106 | 5060 | OK (3 ms) |
| 200/200 | (Unspecified) | 0 | UNKNOWN |
| 2000/2000 | 172.27.6.105 | 5060 | OK (1 ms) |

After the test, the resulted output for the “sip show peers” [7] command in Asterisk was:

| Name/username | Host | Port | Status |
|---|---|---|---|
| 2002/2002 | 172.27.6.106 | 5060 | OK (4 ms) |
| 2001/2001 | 172.27.6.106 | 5060 | OK (3 ms) |
| 200/200 | (Unspecified) | 0 | UNKNOWN |
| 2000/2000 | 172.27.6.105 | 5060 | OK (1 ms) |

Notice that the related host IP address for user 2002 changed to 172.27.6.106, the same address of the targeted user. If there is an inbound call for 2002, both devices with registered users 2002 and 2001 will ring.

A brief look at the SIP peers from Asterisk reveals that the 200 contact is not used. If we add the 200 contact at the 2002 user, then 2002 will be alerted for all the 200 inbound calls. Adding a 666 contact (which is invalid) to the 2002 user resulted in the inability to call the legitimate user. The tool can be used to add multiple contacts. After adding the 200 and the 2002 contacts to the 2001 one, any 2001 inbound calls were alerted on all the contacts.

*Erase_registration* allows sending REGISTER messages to Asterisk, which practically erases targeted contacts from the tables. The following command was used to erase the 2002 contact registered in the 172.27.6.102 domain located at the 172.27.6.102 IP address:

```
./erase_registration eth1 2002 172.27.6.102
172.27.6.102
```

The forged REGISTER message enables a registration deletion if its *Contact* field has the “*” parameter and the *Supported* field contains the “replaces” directive. The softphone of the related 2002 contact could not receive inbound calls, while outbound ones were permitted.

The *reg_highjack* command input required information about the domain IP, *172.27.6.102*, domain's SIP Registrar IP address, *172.27.6.102*, hijack contact *2000@172.27.6.105*, user to highjack *200*, and authentication password *1234:*

```
./reghijacker eth1 172.27.6.102 172.27.6.102
2000@172.27.6.105 out.txt -u 200 -p "1234"
```

The output messages were written into the out.txt file. This operation added the 200 contact to the 2000. Every time an inbound call is received for 200, the device registered with 2000 rings. The 200 user can still make calls but cannot be called.

### 4.8 Session teardown with BYE messages

First, the *From* Tag, *To* Tag and *Call ID* were extracted from an *OK* response using Wireshark. This information was needed to create a BYE message with the teardown tool:

```
#teardown eth1 2000 172.27.6.106 72.27.6.105
69d0e001f532@172.27.6.102 1482582073
as6e028a36
```

The source IP address is *172.27.6.106*, the destination IP address is *172.27.6.105* and targeted User Agent *2000@172.27.6.102*. The other entries where the tags discussed above: *Call ID*, *To Tag* and respectively *From Tag*. Sending the false BYE request made each SIP phone end the ongoing call.

## 5. Countermeasures

Based on the information obtained from the attacks, several measures were taken to ensure some, if not complete, protection against these issues. After applying the countermeasures, tests were performed for evaluating their efficiency using the same tools.

### 5.1 Encrypting the media stream

Secure RTP (SRTP) is a protocol that encrypts the media traffic between two communicating parties. It is used as a countermeasure to the privacy problem concerning eavesdropping. The SRTP parameters (encryption keys, authentication keys) are transmitted using INVITE signaling messages with SDP (Session Description Protocol). A module for SRTP capabilities can be added to Asterisk by installing a patch [8]. After applying it, the IP-PBX needs to be logically placed in the path of the media stream between communicating SIP clients, where it acts as intermediary node.

The tests conducted revealed that only PhonerLite and Snom360 could be successfully used with SRTP. EyeBeam has SRTP capability but it needs a secure channel for communication (TLS) to enable encrypted media usage. GrandStream GXV-3000 includes a Secure RTP feature but was unable to successfully admit SRTP media flow. Figure 2 reveals the difference between an intercepted RTP stream (green) and a SRTP one (blue) when audio extraction was attempted.

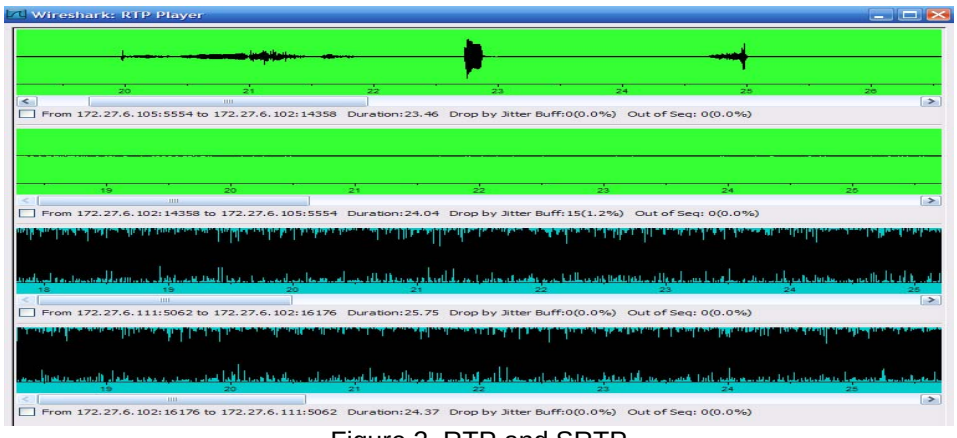

Figure 2. RTP and SRTP

The *iptables* firewall script contained rules for filtering unwanted traffic and accepting the VoIP related packets:

```
$IPTABLES -A INPUT -p udp -m state --
state NEW -m --dport 5060:5070 -j QUEUE
```

### 5.2 Intrusion detection and prevention

The security of any PC can be enhanced by using a firewall. In the case of Linux OS the preferred tool *iptables*. This firewall can be configured using a set of rules/commands resulting in flexible way of filtering the unwanted traffic. Another approach is using an Intrusion Detection/ Prevention System (IDS/IPS). Snort is currently the most powerful application of its kind available for free [9]. Intrusion Detection Systems inspects (reads) the packets and decides, based on sets of rules, whether or not they are malicious. If a packet matches a rule then an alert is generated. The Intrusion Prevention System can manipulate or drop those packets or it can also perform IPS alerting tasks.

The proposed solution includes a firewall and IPS/IDS to provide a more secure VoIP system (as in Figure 3). The first phase is instructing *iptables* to filter the unwanted traffic. Then a mechanism that takes packets out of the stack for queuing to userspace is used. Snort inspects these packets and depending on the verdict will drop or reinject them back into the kernel and generate alerts.

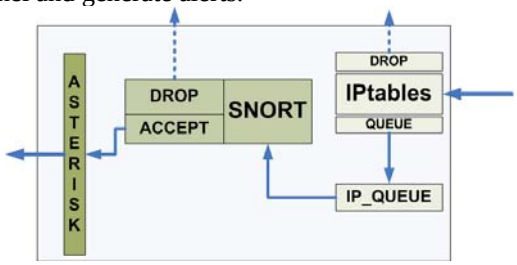

Figure 3. Snort and iptables

```
$IPTABLES -A INPUT -p udp -m state --
state NEW -m udp --dport 1024:5059 -j QUEUE
$IPTABLES -A INPUT -p udp -m state --
state NEW -m udp --dport 5071:65535 -j QUEUE
```

After the configuration of *iptables*, the next phase was to design a set of rules, for Snort, based on the types of attacks tested on the SIP devices. There were two sets of rules: ones that only generate alerts when malicious traffic is discovered and those that drop the detected packets. Alerting was mainly used for the traffic from the internal network. This was done as a measure of false positives mitigation (where legal packets can wrongfully be tagged as unwanted traffic and eliminated). The DROP rules were applied for traffic from the external network as well as for the messages that could be identified with no doubt as being malicious:

```
#Rule for alerting UDP flood attack:
alert udp any any -> $SIP_PROXY_IP any
(msg:"UDP message flooding directed to SIP
proxy"; content:!"INVITE";
content:!"REGISTER"; threshold: type both,
track by_src, count 300, seconds 10;
sid:5000007; rev:1;)

# Rules for SIP INVITE flooding
drop ip $EXTERNAL_NET any -> $HOME_NET 5060
(msg:"VOIP INVITE Message Flood";
content:"INVITE"; depth:6; threshold: type
threshold, track by_src, count 300, seconds
10;classtype:attempted-dos; sid:2003192;
rev:1;)

alert ip $HOME_NET any -> $SIP_PROXY_IP 5060
(msg:"VOIP INVITE Message Flood";
content:"INVITE"; depth:6; threshold: type
threshold , track by_src, count 300, seconds
10; classtype:attempted-dos; sid:2003193;
rev:1;)

#Rules for register erase and highjack
drop ip $HOME_NET any -> $HOME_NET 5060
(msg:"VOIP REGISTER Message Unauthorized
Registration Erase/Highjack Attempt";
content:"REGISTER"; depth:8;
content:"Contact: *"; threshold: type
threshold, track by_src, count 1, seconds 10;
sid:2003198; rev:1;)

drop ip $EXTERNAL_NET any -> $HOME_NET 5060
(msg:"VOIP REGISTER Message Unauthorized
Registration Erase/Highjack Attempt";
content:"REGISTER"; depth:8; content:"Contact:
*"; threshold: type threshold, track by_src,
count 1, seconds 10; sid:2003199; rev:1;)
```

The secured system was then tested. Results indicate that UDP flooding is alerted successfully from both internal and external network. While alerts were generated, flooding with INVITEs from the internal network was still possible, but attacks from the external network were successfully stopped. Issues involving forged REGISTER, BYE messages proved to be handled harder. Snort successfully blocks attacks that involved erasing or highjacking contacts using false REGISTER messages. Attacks like registration addition and session teardown where not detected by Snort.

## 6. Conclusions and future work

The security evaluation stage revealed that the most dangerous attacks are at the application layer, with SIP messages. The INVITE flood proved devastating for the SIP clients as well as for Asterisk resulting in DoS. Testing the SIP based network has other benefits as it provides information about which of the terminals are most reliable during the attacks. We recommend GrandStream as a hardware phone, and eyeBeam for the software phones. WIP330 proved the most unreliable SIP phone from our list, in case of attack. Using SRTP for the media traffic improved the privacy issue but did not solve it completely as an attacker can sniff the SIP messages used to relay the SRTP encryption parameters. These parameters are sent in clear-text. With this information and the SRTP stream, the attacker can eventually decrypt and extract the audio information from the media stream. A secure channel of communication for signaling (like TLS) is needed. Snort offered an improvement to security effectively alerting or blocking some types of attacks, while a number of issues still remain unresolved (problems like false positives). Testes revealed that illegal SIP messages are hard to detect due to their resemblance with the legitimate ones. The solution proposed manages to raise the level of security for the implemented SIP architecture, but there is still work to be done as none of the technologies, tools or methods used offered a complete solution.

## 7. References


[1] G.Camarillo, *SIP Demystified*, McGraw-Hill, August 2002, pp. 89-152

[2] D.Endler, M.Collier, *Hacking Exposed VoIP*, McGraw-Hill/Osborne, September 2007

[3] H.Yan, K.Sripanidkulchai, *"Information Leak Vulnerabilities in SIP Implementations*", *IEEE Network Magazine*, October 2006

[4] D.Endler, M.Collier, *Hacking Exposed VoIP: VoIP Security Secrets&Solutions*, 2006-2008
*http://www.hackingvoip.com/sec_tools.html*

[5] J.V.Meggelen, J.Smith, *"Asterisk the Future of Telephony*", O`Reilly Media, September 2007

[6] M.Montoro, 2001-2008, OXID.IT, *http://www.oxid.it/cain.html*

[7] Digium, 2008, Asteriskguru, *http://www.asteriskguru.com/tutorials/cli_cmd_14.html*

[8] J. Mikima, 2008, Asterisk and Secure RTP, *http://bugs.digium.com/view.php?id=5413*

[9] SOURCEfire, 2008, Snort.org Intrusion Detection/ Prevention System, *http://www.snort.org/dl/*