An Empirical Study of Security of VoIP System

Ahmad Ghafarian
Dept. of Computer Science & Information Systems
University of North Georgia
Dahlonega, GA 30005, USA

Maria Dehghani
Department of Computer Engineering
Ferdowsi University of Mashhad
Mashhad, Iran

ABSTRACT

As VoIP (Voice-over-IP) Services are becoming more popular, various types of attacks against them are increasing. SIP (Session Initiation Protocol) is the main protocol that is used in VoIP. SIP is subject to various types of attacks including DoS (Denial-of-Service) attack. This paper reports our experiment of simulating VoIP system using existing open source tools and technology. The simulated VoIP system is used to demonstrate a normal VoIP communication, launching DoS flooding attacks against SIP and implementing a successful Snort-based Intrusion Detection System (IDS) capable of catching suspicious SIP messages. Additionally, we propose a new VoIP architecture, which is based on buffering all incoming messages from clients with the intention of processing the messages in the buffer before they are forwarded to the destination.

General Terms

Security, Voice over Internet Protocol, Softphone, Protocol.

Keywords

VoIP; DoS; denial of service attack; snort; IDS; intrusion detection system; SIP; session initiation protocol.

1. INTRODUCTION

VoIP [4] technology has become more widespread, due to the low cost. SIP (Session Initiation Protocol) [9] which forms the foundation of VoIP is based on free open-source Asterisk standards [9] in which the messages are transmitted in unencrypted clear text. Therefore, it is subject to various types of attacks. Common attacks against SIP include eavesdropping, connection hijacking, call fraud and DoS attack. DoS attack is the most damaging of all because it makes the service unavailable to legitimate users. Malicious users can use commonly available tools to launch attacks on VoIP system aiming to disrupt communications, gain free services, or to bring down the service [12]. According to the researchers in [8] there are three types of DoS attacks that can be launched against VoIP infrastructure, exploitation of implementation flaws, exploitation of application level syntactic vulnerabilities, and flooding of the SIP server. A flooding attack on a SIP server can target one or more of the three resources needed for operations, memory, CPU and bandwidth. To mitigate DoS attacks against SIP, implementation of network-based IDS (Intrusion Detection System) is a common practice [7]. Many researchers have proposed various architectures for implementing such an IDS using Snort [1], [2], [3], and [12]. Snort-based IDS can be configured to examine the incoming VoIP messages and trigger an alarm if the message seems to be suspicious.

This paper reports our experience simulating a VoIP system using existing open source technologies including Kali Linux Debian, Asterisk running CnetOS and Xlite Softphone. The simulated VoIP system is used to demonstrate normal VoIP communication as well as launching DoS flooding attack against SIP. To launch an attack we configure the system so that one client acts as the hacker and sends massive number of INVITE messages to the victim client and consequently brings down the system. To mitigate a DoS flooding attack, we use Snort to configure IDS and implement it to examine its effectiveness in the simulated VoIP environment. Additionally, we propose a new SIP architecture which is based on buffering all incoming messages at the SIP server with the intention of catching the adversaries in the buffer before they are forwarded to the destination.

The rest of the paper is organized as follows: Section 2 covers literature review, section 3 discusses background information, technology used appears in section 4, experiment details is presented in section 5, section 6 details our proposed architecture for implementing defense against DoS attack, and section 7 provides conclusion and further research.

2. LITERATURE REVIEW

Several researchers have proposed VoIP security solutions to detect and prevent DoS attacks against SIP protocol. Most of these solutions use Snort and focus on different architectures for detecting and preventing attacks. A Snort-based IDS proposal for detecting DoS attack on SIP protocol is presented in [6]. The authors use the knowledge of SIP traffic to create Snort rules for the detection of unusually high traffic of specific types. The researchers suggest that their proposed rules can also be used for the detection of various SIP network misconfigurations against unwanted and accidental overload. Voznak and Safarik [12] provide a comprehensive discussion of various types of DoS attack and compare the degree of damage they can do on VoIP system. The authors propose an IDS solution which is composed of Snort, SnortSam and Iptables applications and use of demilitarized zone (DMZ) in their proposed architecture.

Fan and Wan [3] have analyzed various ways to improve the detection mechanism's performance of DoS attack in SIP. Their approach is based on using balanced message numbers in their architecture. Ehlert S. et al [1] presented a method for detecting flooding attacks in SIP using dedicated IDS that examine all incoming traffic. The main contribution of their work is the communication with a firewall component to block offending traffic and thus keeping the service alive even under attack conditions. They have used state machine to model SIP proxy server’s transaction.

A design of effective defenses against SIP-specific DoS attacks can be found in [8]. The researchers address all four aspects that an effective solution against DoS attacks should cover namely, definition, detection, mitigation, and validation. Another proposed architecture to mitigate DoS attacks on a SIP-based VoIP infrastructure can be found in [2]. The researchers have designed architecture to detect various types of SIP vulnerabilities effectively, including message flooding, malformed message sending, and DNS blocking.

An implementation of two stages IDS in VoIP appears in [7]. In the first stage, the system applies knowledge-based techniques on the packets passing through the network and on the second stage the test is behavior-based on the packets that have passed the knowledge-based step. The authors have implemented a prototype of their IDS and show that the IDS is able to analyze the SIP protocol traffic. The researchers in [10] studied possible approaches for mounting DoS attacks against a SIP server. The researchers used knowledge of SIP to launch DoS attacks. They suggest that misbehaving implementations and misconfigurations are the most frequent cause of the DoS attack. Finally, testing several open-source and proprietary VoIP security tools and comparative results can be found in [5].

3. BACKGROUND INFORMATION

SIP is an application-layer signaling text-based protocol used for establishing or terminating a session between two or more partners using VoIP system. There are four SIP entities that cooperate with each other to establish a link between a caller and a callee: 1) *Proxy server* receives a request and then forwards it directly to the callee or to another server that can better perform the forwarding process. 2) *Redirect server* receives a request and informs the caller about the next hop server. The caller then contacts the next hop server directly. 3) *User Agent* is a logical entity that is responsible for generating and terminating SIP requests. 4) *Registrar server* is a database containing locations as well as user preferences as indicated by the user agents. According to RFC 3261 [14] there are six types of SIP messages: INVITE, ACK, BYE, CANCEL, REGISTER, and OPTIONS. Figure 1 illustrates a sample of SIP INVITE request. The INVITE contains the details of the type of session that is requested. It could be a simple voice session, a multimedia session such as a video conference, or a gaming session.

In Figure 1, Tesla sends an INVITE message to the Proxy Server. The server in turn either sends it directly to Marconi or checks the database for the next hop. Once the request reaches the destination, the equipment rings and Marconi sends 200 ok indicating that the acceptance of the call and the session initiation is then finalized by Tesla sending an ACK message back to Marconi. After finishing the session, Marconi ends it by sending a BYE message to Tesla.

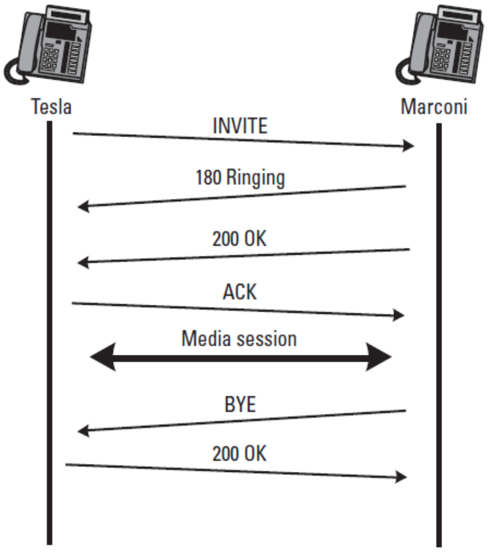


Figure 1 Sample SIP session, courtesy of Allen Johnson

4. TYPES OF DOS ATTACK ON SIP

Most SIP-based DoS attacks aim to utilize available resources such as memory, bandwidth, or CPU with the intention of bringing down the system. We address these types of DoS attacks briefly below.

4.1 Memory

In a VoIP system an incoming message is saved in SIP’s internal buffers for processing the message [10]. The amount of buffered data and the time period the server keeps the buffered data depends on whether the server works in a stateful or stateless mode. Message buffering in stateful SIP servers is a good target for DoS attacks. For example, when a hacker continuously issues INVITE messages with different session identifiers, the server stores state information at the SIP proxy server’s memory. If the recipient is non-responding the proxy will have to forward the message and keep state information in the buffer for at least 30 seconds or more. This situation causes a memory-based DoS attacks.

4.2 Bandwidth

SIP is an Internet-based technology which means there are access links that connect a SIP server to the Internet. If a malicious user overloads the server’s access links, there is a possibility of SIP messages being lost. This in turn may cause longer session setup times or even a failure of the message setup. This situation will lead to a DoS attack. Protection of bandwidth, however, is a transport-layer issue.

4.3 CPU

Upon receipt of a SIP message, the SIP server needs to do some processing before forwarding the message. The processing is done by the CPU of the server and it varies for different SIP architectures. Normally, a proxy server’s CPU should be able to process messages efficiently and quickly. However, there are many server operations which may block servers. Such operations can be misused to launch a DoS attack on SIP architecture causing the system to stop operating. Usually, a SIP proxy server with slower request processing capabilities is subject to this type of DoS attacks. Flooding the system with REGISTER messages will also cause exhaustion of SIP proxy CPU and thus increases the processing load on the server.

4.4 Other Types of DoS Attacks

Flow tampering attacks is another type of DoS attack where an attacker target connections between users by introducing fake signaling messages into the communication channel. For example, a BYE message with the appropriate credentials can prematurely terminate the session. In order to launch a Flow Tampering attack the hacker needs to know the session parameters for these attacks to function correctly. This can be done by sniffing the network. It is also possible to tamper with the actual SIP message and cause DoS attack. In this case attackers can try to inject harmful content into a message, by entering meaningless or wrong information with the aim of creating a buffer overflow at the target.

5. TECHNOLOGY USED

The following is the list of software tools we used in our experiment.

- SIP server: Asterisk version 11.12.0 with CentOS 64 bit and 4 GB memory
- KALI Linux with Debian 7, 64 bit and 2 GB memory
- Two Windows 7, 64 bit with 2 GB memory
- Two Xlite Softphone version 4.7.0
- Snort 2.9.5.1
- Wireshark protocol analyzer version 1.12.0
- Mozilla Firefox 31.0
- VMware workstation 10.0 virtual machine.

6. EXPERIMENT DETAILS

Our experiment consists of the following four steps:

1. Simulation of VoIP system
2. Demonstration of normal VoIP communication
3. Launch of DoS attack
4. Implementation of Snort-based IDS

6.1 Simulation of VoIP system

In this subsection we describe the process of setting up the VoIP environment. For efficiency purposes we decided to install a SIP server on a virtual machine (VM). First we installed VMware Workstation 10 [13] on a physical machine running Windows 7 and created the VM. Then we installed a version of KALI Linux called Debian 7 on the VM. The Debian 7 is used as a platform for installation and configuration of SIP's Asterisk exchange server running CnetOS. We also installed Windows 7 on the VM to be used by one of the clients, Sara. Another physical machine with Windows 7 is used for the other client, Maria. In order for Maria and Sara to be able to use VoIP system and call each other, we simulated VoIP by installing and configuring one Xlite Softphone on each machine. To complete the process, we registered both Sara and Maria to the SIP server. At this stage, we have a complete VoIP network with two clients having access to SoftPhone and able to communicate with each other. Figure 2 shows Sara with id number 100 (pointed with red arrow on the left) and Maria with id number 200 (shown with red arrow on the right) ready to call or receive call from each other.

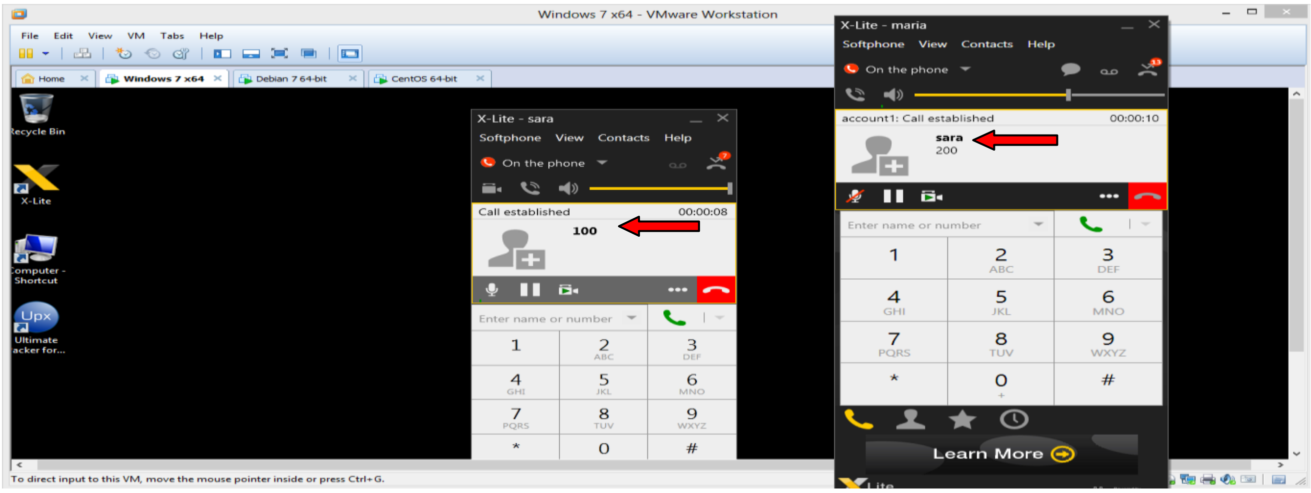


Figure 2 VoIP Configuration

6.2 Demonstration of Normal Communication

The two clients Sara and Maria shown in Figure 2 above, have used their Softphone to call each other successfully using VoIP. To demonstrate this, we installed WireShark software on VM to capture the fllow of network traffic during the communication. Figure 3 shows Maria sends an INVITE message to Sara, the shake hand takes place and the connection is established.

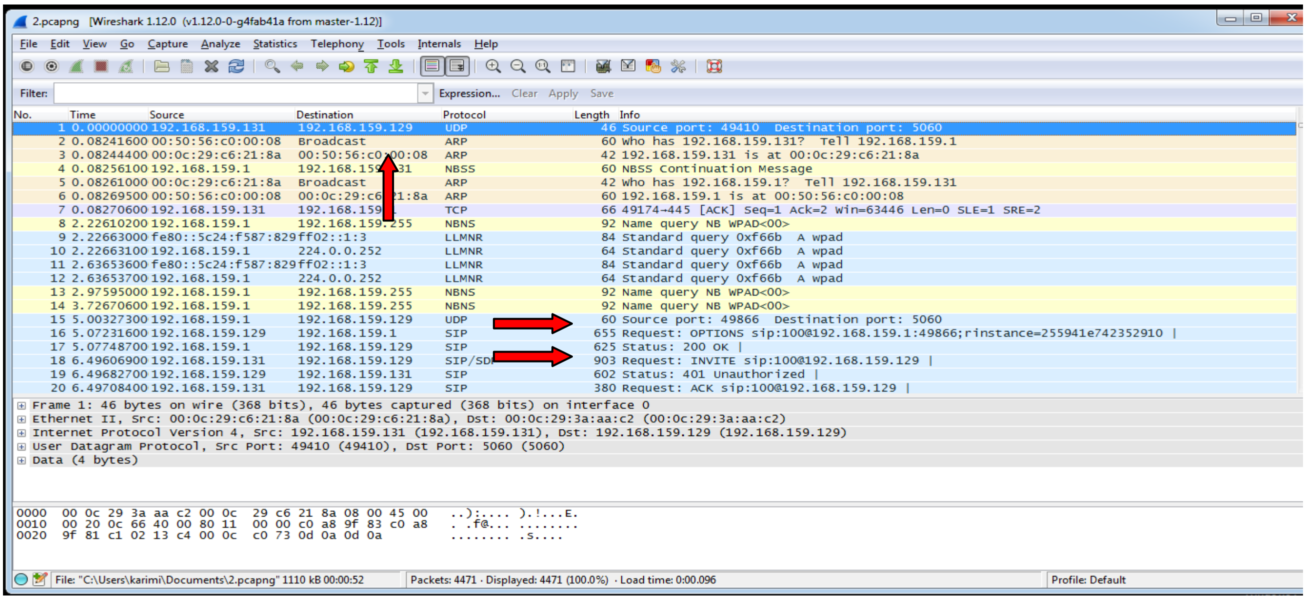
Figure 3 shows Maria with IP address 192.168.159.131 calls Sara whose IP address is 192.168.159.129 by sending an INVITE SIP 192.168.159.129 message (this is shown by the top red arrow on the Figure). Sara sends the 200 ok message to Maria indicating that she accepts the call and Maria sends an ACK message to her (these are shown by the two red arrows at the bottom of the Figure). The session initiation begins and the two start communicating with each other. We noticed that the IP addresses for both Sara and SIP are the same. This is because they are both on a VM machine. Other details of their communications can be viewed from captured traffic using Wireshark.

6.3 Launching DoS Attack

We implemented a Dos attack by sending a INVITE flood message 3,000,000 times from Kali Linux against SIP server. During the time that the SIP server is flooded, the SIP gateway prevented users from making phone calls resulting in a DoS attack. By using this attack the server will become busy processing these flooding messages and none of the SoftPhones could place a call. Any attempt to make a call generated the message, "Failed to establish a connection". In the normal SIP initiation, there would be messages like 100 ok, 180 ringing and 200 ok (see Figure 1) and also we can listen to the phone call by decoding it and play it back. However, when the DoS attack occurs none of these 100 ok, 180 and 200 ok messages are generated and you cannot listen to the phone call (see Figure 4). This is an example of CPU-based DoS attack because SIP server keeps processing incoming messages without any resolution. This network traffic was also captured by Wireshark (see Figure 4). The Figure shows repeated INVITE messages from source IP address (Maria): 192.168.159.132 to the destination (Sara): IP address 192.168.159.129. Repeated INVITE flood messages are depicted by red arrows.

6.4 Implementation of Snort-based IDS

Snort [11] is a network-base IDS capable of sniffing VoIP network traffic packets, monitoring them in real time and trigger alarms when suspicious conditions are met. Snort can be configured by using different rules. In this work, we used Snort and configured it to detect DoS attacks against SIP flooding attack. For the purpose of this research, we used four rules to configure Snort. The first rule is to set Snort to trigger an alarm when continued requests come from the same IP address and are destined to the same destination. The other three rules are related to external network addressee. In Snort it is possible to filter the external IP addresses to a subset of external IP addresses. However, we did not use any filtering and set the rules to monitor all external network traffic. Snort was also configured to create log files which can be analyzed offline. In addition, a special filter for Snort called Wirshnork was setup so that we can open and analyze Snort log files. Figure 5 shows Snort implementation of IDS to detect and prevent DoS flooding attacks against SIP server. Maria continually sending INVITE flood messages but Snort IDS generates “who has the IP address” warning message. This is shown in the highlighted area of the Figure 5.

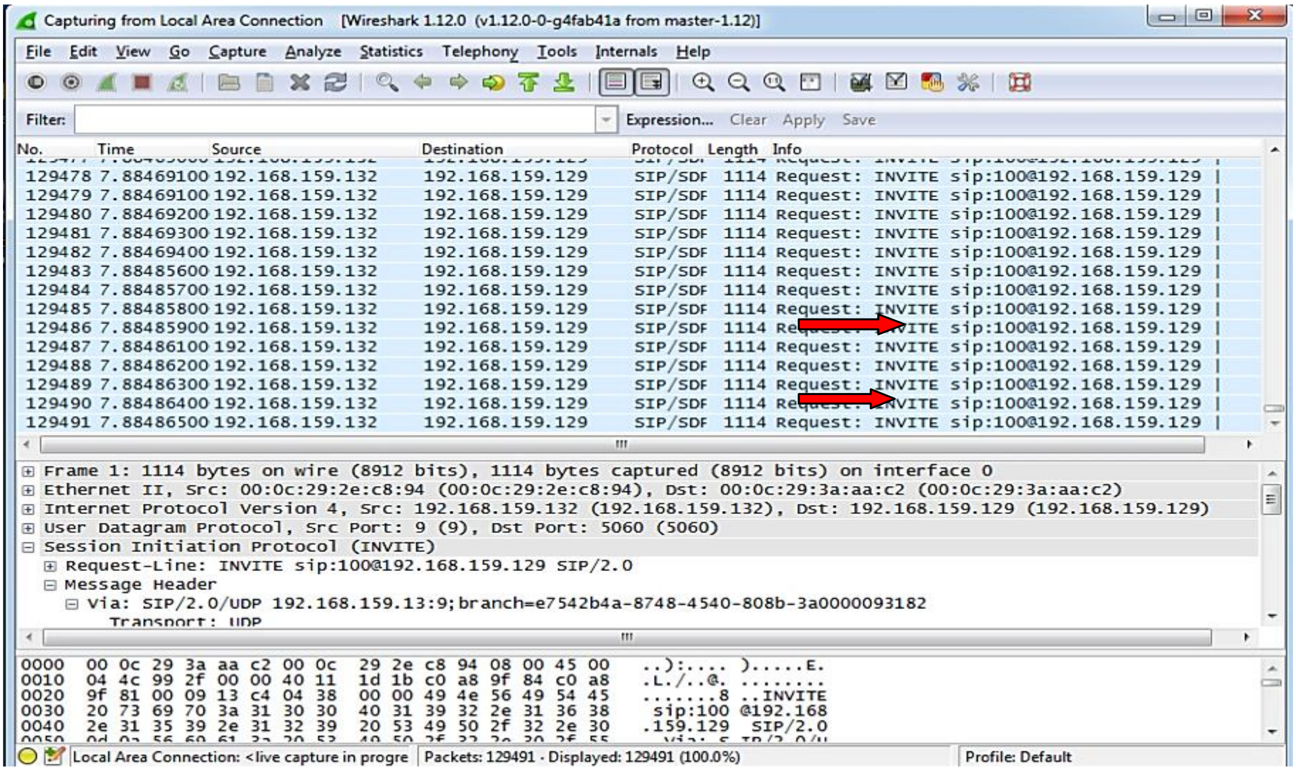

2.pcapng [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time       Source              Destination         Protocol  Length Info
    1 0.00000000 192.168.159.131     192.168.159.129     UDP           46 Source port: 49410  Destination port: 5060
    2 0.08241600 00:50:56:c0:00:08   Broadcast           ARP           60 Who has 192.168.159.131?  Tell 192.168.159.1
    3 0.08244400 00:0c:29:c6:21:8a   00:50:56:c0:00:08   ARP           42 192.168.159.131 is at 00:0c:29:c6:21:8a
    4 0.08256100 192.168.159.1       192.168.159.131     NBSS          60 NBSS Continuation Message
    5 0.08261000 00:0c:29:c6:21:8a   Broadcast           ARP           42 Who has 192.168.159.1?  Tell 192.168.159.131
    6 0.08269500 00:50:56:c0:00:08   00:0c:29:c6:21:8a   ARP           60 192.168.159.1 is at 00:50:56:c0:00:08
    7 0.08270600 192.168.159.131     192.168.159.1       TCP           66 49174→445 [ACK] Seq=1 Ack=2 Win=63446 Len=0 SLE=1 SRE=2
    8 2.22610200 192.168.159.1       192.168.159.255     NBNS          92 Name query NB WPAD<00>
    9 2.22663000 fe80::5c24:f587:829ff02::1:3          LLMNR         84 Standard query 0xf66b  A wpad
   10 2.22663100 192.168.159.1       224.0.0.252         LLMNR         64 Standard query 0xf66b  A wpad
   11 2.63653600 fe80::5c24:f587:829ff02::1:3          LLMNR         84 Standard query 0xf66b  A wpad
   12 2.63653700 192.168.159.1       224.0.0.252         LLMNR         64 Standard query 0xf66b  A wpad
   13 2.97595000 192.168.159.1       192.168.159.255     NBNS          92 Name query NB WPAD<00>
   14 3.72670600 192.168.159.1       192.168.159.255     NBNS          92 Name query NB WPAD<00>
   15 5.00327300 192.168.159.1       192.168.159.129     UDP           60 Source port: 49866  Destination port: 5060
   16 5.07231600 192.168.159.129     192.168.159.1       SIP          655 Request: OPTIONS sip:100@192.168.159.1:49866;rinstance=255941e742352910 |
   17 5.07748700 192.168.159.1       192.168.159.129     SIP          625 Status: 200 OK |
   18 6.49606900 192.168.159.131     192.168.159.129     SIP/SDP      903 Request: INVITE sip:100@192.168.159.129 |
   19 6.49682700 192.168.159.129     192.168.159.131     SIP          602 Status: 401 Unauthorized |
   20 6.49708400 192.168.159.131     192.168.159.129     SIP          380 Request: ACK sip:100@192.168.159.129 |
⊞ Frame 1: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface 0
⊞ Ethernet II, Src: 00:0c:29:c6:21:8a (00:0c:29:c6:21:8a), Dst: 00:0c:29:3a:aa:c2 (00:0c:29:3a:aa:c2)
⊞ Internet Protocol Version 4, Src: 192.168.159.131 (192.168.159.131), Dst: 192.168.159.129 (192.168.159.129)
⊞ User Datagram Protocol, Src Port: 49410 (49410), Dst Port: 5060 (5060)
⊞ Data (4 bytes)

0000  00 0c 29 3a aa c2 00 0c  29 c6 21 8a 08 00 45 00   ..):.... ).!...E.
0010  00 20 0c 66 40 00 80 11  00 00 c0 a8 9f 83 c0 a8   . .f@... ........
0020  9f 81 c1 02 13 c4 00 0c  c0 73 0d 0a 0d 0a         ........ .s....

File: "C:\Users\karimi\Documents\2.pcapng" 1110 kB 00:00:52 | Packets: 4471 · Displayed: 4471 (100.0%) · Load time: 0:00.096 | Profile: Default
```

Figure 3 Demonstration of normal VoIP Communication

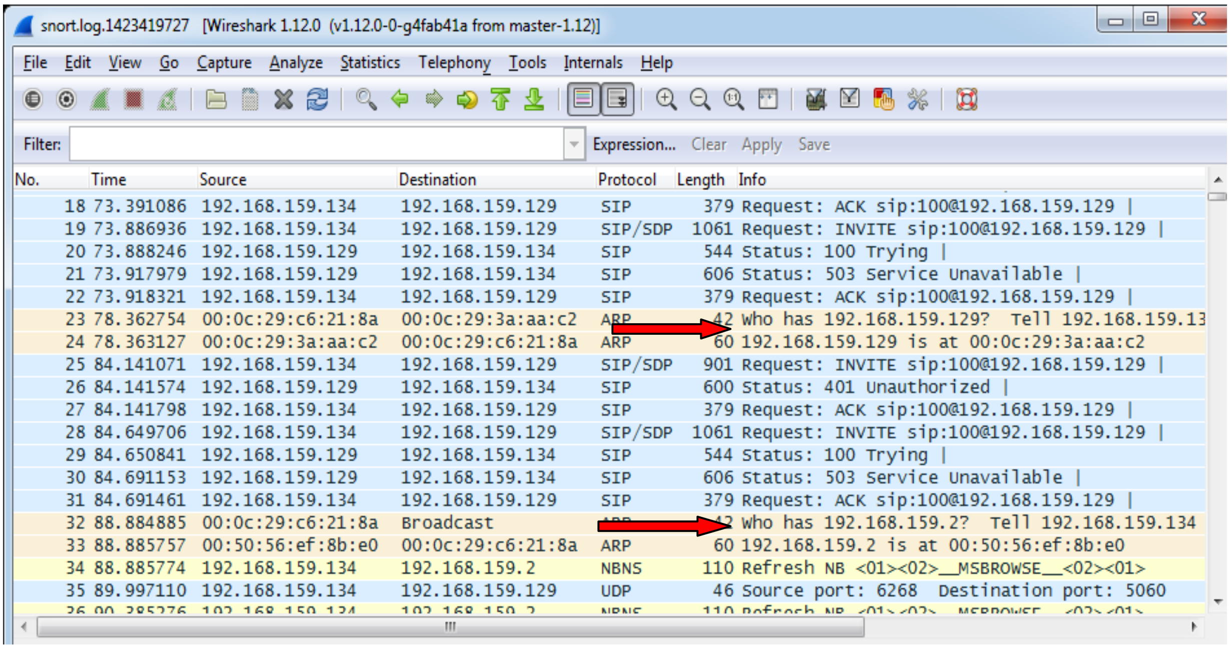

Capturing from Local Area Connection [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.    Time       Source           Destination      Protocol Length Info
129478 7.88469100 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129479 7.88469100 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129480 7.88469200 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129481 7.88469300 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129482 7.88469400 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129483 7.88485600 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129484 7.88485700 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129485 7.88485800 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129486 7.88485900 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129487 7.88486100 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129488 7.88486200 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129489 7.88486300 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129490 7.88486400 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
129491 7.88486500 192.168.159.132  192.168.159.129  SIP/SDP  1114 Request: INVITE sip:100@192.168.159.129 |
⊞ Frame 1: 1114 bytes on wire (8912 bits), 1114 bytes captured (8912 bits) on interface 0
⊞ Ethernet II, Src: 00:0c:29:2e:c8:94 (00:0c:29:2e:c8:94), Dst: 00:0c:29:3a:aa:c2 (00:0c:29:3a:aa:c2)
⊞ Internet Protocol Version 4, Src: 192.168.159.132 (192.168.159.132), Dst: 192.168.159.129 (192.168.159.129)
⊞ User Datagram Protocol, Src Port: 9 (9), Dst Port: 5060 (5060)
⊟ Session Initiation Protocol (INVITE)
  ⊞ Request-Line: INVITE sip:100@192.168.159.129 SIP/2.0
  ⊟ Message Header
    ⊟ Via: SIP/2.0/UDP 192.168.159.13:9;branch=e7542b4a-8748-4540-808b-3a0000093182
        Transport: UDP

0000  00 0c 29 3a aa c2 00 0c  29 2e c8 94 08 00 45 00   ..):.... ).....E.
0010  04 4c 99 2f 00 00 40 11  1d 1b c0 a8 9f 84 c0 a8   .L./..@. ........
0020  9f 81 00 09 13 c4 04 38  00 00 49 4e 56 49 54 45   .......8 ..INVITE
0030  20 73 69 70 3a 31 30 30  40 31 39 32 2e 31 36 38    sip:100 @192.168
0040  2e 31 35 39 2e 31 32 39  20 53 49 50 2f 32 2e 30   .159.129  SIP/2.0

Local Area Connection: <live capture in progre | Packets: 129491 · Displayed: 129491 (100.0%) | Profile: Default
```

Figure 4 DoS attack on SIP protocol



snort.log.1423419727 [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]

```
No.     Time       Source             Destination        Protocol Length Info
     18 73.391086  192.168.159.134    192.168.159.129    SIP       379 Request: ACK sip:100@192.168.159.129 |
     19 73.886936  192.168.159.134    192.168.159.129    SIP/SDP  1061 Request: INVITE sip:100@192.168.159.129 |
     20 73.888246  192.168.159.129    192.168.159.134    SIP       544 Status: 100 Trying |
     21 73.917979  192.168.159.129    192.168.159.134    SIP       606 Status: 503 Service Unavailable |
     22 73.918321  192.168.159.134    192.168.159.129    SIP       379 Request: ACK sip:100@192.168.159.129 |
     23 78.362754  00:0c:29:c6:21:8a  00:0c:29:3a:aa:c2  ARP        42 Who has 192.168.159.129?  Tell 192.168.159.13
     24 78.363127  00:0c:29:3a:aa:c2  00:0c:29:c6:21:8a  ARP        60 192.168.159.129 is at 00:0c:29:3a:aa:c2
     25 84.141071  192.168.159.134    192.168.159.129    SIP/SDP   901 Request: INVITE sip:100@192.168.159.129 |
     26 84.141574  192.168.159.129    192.168.159.134    SIP       600 Status: 401 Unauthorized |
     27 84.141798  192.168.159.134    192.168.159.129    SIP       379 Request: ACK sip:100@192.168.159.129 |
     28 84.649706  192.168.159.134    192.168.159.129    SIP/SDP  1061 Request: INVITE sip:100@192.168.159.129 |
     29 84.650841  192.168.159.129    192.168.159.134    SIP       544 Status: 100 Trying |
     30 84.691153  192.168.159.129    192.168.159.134    SIP       606 Status: 503 Service Unavailable |
     31 84.691461  192.168.159.134    192.168.159.129    SIP       379 Request: ACK sip:100@192.168.159.129 |
     32 88.884885  00:0c:29:c6:21:8a  Broadcast          ARP        42 Who has 192.168.159.2?  Tell 192.168.159.134
     33 88.885757  00:50:56:ef:8b:e0  00:0c:29:c6:21:8a  ARP        60 192.168.159.2 is at 00:50:56:ef:8b:e0
     34 88.885774  192.168.159.134    192.168.159.2      NBNS      110 Refresh NB <01><02>__MSBROWSE__<02><01>
     35 89.997110  192.168.159.134    192.168.159.129    UDP        46 Source port: 6268  Destination port: 5060
     36 90.385276  192.168.159.134    192.168.159.2      NBNS      110 Refresh NB <01><02>__MSBROWSE__<02><01>
```

Figure 5 Demonstration of Snort catching DoS attack

7. THE PROPOSED ARCHITECTURE

In general, a source client sends messages to the server first and then the server passes them to another route for the destination. In our proposed method, any incoming SIP message first goes to a buffer which is specifically designed for this purpose. In our proposed method, corresponding to any of the SIP messages INVITE, BYE, etc there would be a variable that counts the number of messages arrived and saved at the buffer. For example, for INVITE message the counter would be IC and will be incremented every time a message arrives. The server then checks the value of IC; if it is zero then it passes the INVITE request to the next hop and increment IC. Otherwise, it detects the INVITE message as a DoS attack and issues warning. We can use any data structure to implement the buffer. In this approach, each client should have its own buffer and it becomes empty after the message is forwarded to the next hop. Most of the previous solutions somehow change the SIP structure but our proposed method doesn't need any change to SIP architecture and we believe it can work with most existing SIP applications. Partial implementation of the buffer is shown below. The result of the implementation of this method will be presented in another paper.

```
IBuffer[ ];
IC =0;
String message =”empty”;
If (message ==INVITE )
then if (IC ==0) forward it to the next hop
        and IC= IC-1;
Once the session is established IC = 0;
```

8. CONCLUSION

In this experiment we used open-source software technology to simulate the VoIP environment. We first tested the simulated VoIP system to make sure it functioned properly. We then used the VoIP technology and launched CPU-based DoS flood attack by launching huge number (3, 000,000) of INVITE messages against the server. The results demonstrate that a DoS attack against SIP is easy to launch and is very effective in bringing down the SIP server. There is variety of DoS attacks that can be launched. A DoS attack on the server makes the system potentially unavailable to the legitimate users. In order to mitigate DoS flood attacks, we examined the effectiveness of Snort-based IDS against DoS attacks by setting up and configuring four Snort rules. Our experiment shows that with proper configuration of Snort we can implement a warning system so that the server can trigger an alarm. However, in most cases an analyzer like Wireshark is needed to analyze the network traffic in real-time or capture the traffic and save it in a log file for later analysis.

We also proposed a new architecture by implementing a buffer. The buffer would be a temporary place for message evaluation before being processed. The advantage of this approach is that messages will be interrogated before being processed. If the message is not a potential threat then it will be processed, otherwise it will be dropped. The implementation of this new architecture will appear in another paper.

9. ACKNOWLEDGMENT

This research was supported by the 2015 UNG Presidential Semester Scholar award. The first author would like to thank all the individuals who were involved in the establishment and implementation of the award.

10. REFERENCES

[1] S. Ehlert, Y. Rebahi, and T. Magedanz, ‘Intrusion Detection System for Denial-of-Service flooding attacks in SIP communication networks’, Int. J. Security and Networks, vol. 4, no. 3, pp. 189–200, 2009.

[2] S. Ehlert, G. Zhang, and D. Geneiatakis, "Two layer Denial of Service prevention on SIP VoIP infrastructure”, Computer Communications, vol 31, pp. 2443-2456, 2008.

[3] Z. Fan, and X. Wan, “The Design and Realization of SIP DoS attack Detection Plugin Based on Balanced Message Number Principle”, Proceedings of ICCTA, pp. 780-784, 2009.

[4] B. Goode, “Voice over Internet protocol (VoIP)”, Proceedings of the IEEE ,vol 90 , Issue 9 , 2002.

[5] S. McGann, and D. C. Sicker. “An Analysis of Security Threats and Tools in SIP-Based VoIP Systems”, University of Colorado Boulder, 2005.

[6] J. Markl, J. Dočkal, “Deployment of Snort IDS in SIP based VoIP environments”, Security and Protection of Information 2007.

[7] S. Niccolini, R. G. Garroppo, S. Giordano , G. Risi, and S. Ventura, “SIP Intrusion Detection and Prevention: Recommendations and Prototype Implementation”, IEEE, vol 5, no 6, 2006.

[8] G. Ormazabal, S. Nagpal, E.Yardeni, and H. Schulzrinne. “Secure SIP: A Scalable Prevention Mechanism for DoS Attacks on SIP Based VoIP Systems.” H. Schulzrinne, R. State, and S. Niccolini (Eds.): IPTComm 2008, LNCS 5310, pp. 107–132, 2008.

[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Spark, M. Handley, and E. Schooler, Session Initiation Protocol, no. RFC 3261, 2002.

[10] D. Sisalem, and J. Kuthan “Denial of Service Attacks and SIP Infrastructure: Attack Scenarios and Prevention Mechanisms”, Network IEEE, Vol 20, Issue 5, pp. 26-31, 2006.

[11] Snort, www.snort.org

[12] M. Voznak, and J. Safarik, "SIP proxy robustness against DoS attacks", Proceedings of the Applied Computing Conference, pp. 223-227, 2011.

[13] VMware.com. virtualization for desktop, available at VMware.com.