Modern Applied Science; Vol. 8, No. 2; 2014
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Study on Security Issue in Open Source SIP Server

Muhammad Yeasir Arafat[1], Muhammad Morshed Alam[1] & Feroz Ahmed[1]

[1] Department of Electrical and Electronic Engineering, School of Engineering and Computer Science, Independent University, Bangladesh

Correspondence: Muhammad Yeasir Arafat, School of Engineering and Computer Science, Independent University, Bangladesh. Tel: 88-017-19301525. E-mail: yeasir08@yahoo.com

Received: January 7, 2014 Accepted: March 6, 2014 Online Published: March 11, 2014

doi:10.5539/mas.v8n2p124 URL: http://dx.doi.org/10.5539/mas.v8n2p124

Abstract

Session Initiation Protocol (SIP) is a core protocol for real-time communication networks, including voice over internet protocol (VoIP). In this paper, author's ensured security for asterisk based SIP server using packet filtering firewall tools know as iptables. Rules are applied at Linux iptables on the basis of respective port numbers, allowing and disallowing particular IP address or IP addresses with subnet. To protect the SIP server from external attack rules are applied at Linux iptables for the useful protocols likes TCP, UDP, RTP and ICMP. A popular simulation software or network protocol analyzer known as Wireshark is used to illustrate how the iptables rules worked that applied for above protocols and shows the changes before and after applying rules. This paper also shows the asterisk server monitoring using the Linux kernel log files and asterisk command line interface (CLI) that shows the successful, unsuccessful SSH login sessions and web based login (HTTPS) with IP addresses list, SIP register request to the SIP server. In this paper, our present approach helps to prevent the SIP server from unauthorized access and attacks.

Keywords: VoIP, SSH, SIP, SDP, UDP, TCP, RTP, ICMP

1. Introduction

VoIP may be the fastest growing technology that includes routing voice conversations over IP-based network. The flexibleness of the VoIP system convergence of voice and data networks brings with it additional security requirements. SIP servers are vital network elements that endow SIP endpoints to switch messages, register user position, and faultlessly move between networks. SIP servers permit network operators to determine routing and security principles, authenticate clients and control user locations. SIP server applications might take many forms, however the SIP standard defines three general kinds of server functionality that applies to all or any proxy, redirect and registrar servers. SIP server submissions may take numerous forms, but the SIP benchmark defines three general kinds of server functionality that applies to all proxy, redirect and registrar servers.

One of the very most common attacks is denial of service (DoS). After a session has been established with SIP the particular media transfer is transmitted with the true time transport protocol (RTP). In transport layer SIP uses user datagram protocol (UDP). Today it becomes challenge to provide extreme level security for the VoIP server. To ensure security for SIP based asterisk server a system administrator has to follow some steps such as change the default user name (root) and port (22) for remote SSH login to the SIP server, use packet filtering rules for SSH port (22) usually transport control protocol (TCP) packets based on the IP address at Linux iptables, as some open source PBX software provide web based login known as graphical user interface so apply packet filtering rules for HTTPS & TCP packets with port 443, allow SIP user agents according to IP block by applying rules at iptables for the port 5060 with UDP packets. To prevent internet control message protocol (ICMP) flood attack administrator can allow IP addresses with subnet for ICMP echo ping and trace route request to the SIP sever and block rest of the IP address of the world. Monitoring the Linux kernel log files which usually at */var* directory is very important because here administrator can see the all the remote Secure Shell (SSH) login, web based login to the SIP server with the host IP address list. Data log gives the list of registered and unregistered SIP peers with IP addresses and port number which is more important to see the unauthorized attack. Above things will discuss in details bellow with the network protocol analyzer Wireshark. Here Wireshark simulation will show the differences between before and after applying packet filtering rules to the Linux kernel.

Wireshark is a free and open-source packet analyzer. It has a rich and powerful feature set and runs on most computing platforms including Windows, OS X, Linux, and UNIX. Network professionals, security experts, developers, and educators around the world use it regularly. Wireshark is very similar to tcpdump, but has a graphical front-end, plus some integrated sorting and filtering. It is used for network troubleshooting, analysis, software and communications protocol development, options. Wireshark allows the user to put network interface controllers that support promiscuous mode into that mode, in order to see all traffic visible on that interface, not just traffic addressed to one of the interface's configured addresses and broadcast/multicast traffic.

In wireshark data can be captured "from the wire" from a live network connection or read from a file of already-captured packets. Live data can be read from a number of types of network, including Ethernet, IEEE 802.11, PPP, and loopback. Captured network data can be browsed via a GUI. Data display can be refined using a display filter. VoIP calls in the captured traffic can be detected. If encoded in a compatible encoding, the media flow can even be played.

2. Common Attack in SIP Server

Since there is such a large difference between IPv4 and IPv6, they cannot communicate directly with each other. A system that is capable of handling IPv6 traffic can be made backward compatible, but an already deployed system that handles only IPv4 is not able to handle IPv6 datagram. This means that a major upgrade process would need to take place, involving hundreds of millions of machines, in order to make a complete transition to IPv6. This way is too expensive and time consuming and in any case will not happen overnight. The network world will most likely see a gradual transition to IPv6, where IPv6 will be integrated into the IPv4 world that exists today. There are different kinds of technologies which can be applied such as dual stack, tunnelling, and translation. At present, three transition mechanisms, dual stack, tunneling, and translation, are proposed to solve the problems due to the co-existence of IPv6 and IPv4. Over several transition techniques have been used and tested for the communications between different networks to ensure IPv4 and IPv6 interoperability. Therefore, to make decision on the best suited transition methods, it is really important to have an overview of the current IPv4 networks. In addition, enterprises must analyze needed functionalities, scalability, and securities in the corporation.

2.1 TCP SYN Flood Attacks

TCP encompasses a full handshake between sender and receiver before data packets are dispatched. The starting scheme drives a Synchronize (SYN) request in Figure 1. The obtaining system propels an acknowledgement (ACK) having its own SYN request. Then the dispatching system sends back its own ACK and connection can start between the two schemes. If the obtaining system is dispatched a SYNX package but does not obtain an ACKY+1 to the SYNY it sends back to the senders, the receiver will resend a new ACK + SYNY after some time has passed in Figure 1. The processor and memory resources at the obtaining scheme are booked for this TCP SYN demand until a timeout occurs. The TCP SYN attack uses three-way handshake involving the sending system and the receiving system by dispatching huge volumes of TCP SYN packets to the sufferer system with spoofed source IP addresses, so the casualty scheme answers to a non-requesting scheme with the ACK+SYN.

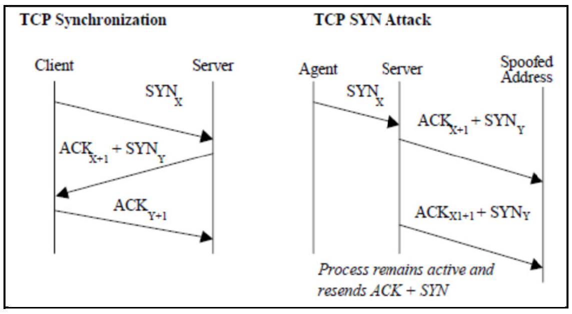


Figure 1. TCP synchronization and TCP SYN attack

When a large volume of SYN requests are being processed by a server and none of the ACK+SYN responses are returned, the server commences to operate out of processor and memory resources. Ultimately, if the amount of TCP SYN attack requests is huge and they continue over time the victim system will come to an end of resources and that will struggle to answer any legal users.

2.2 UDP Flood Attack

In UDP inundate attack attacker drives large amount of UDP packets to a sufferer system, because of which there is dispersion of the network and the decrease of accessible bandwidth for justifiable service requests to the sufferer system. A UDP inundate attack is possible when an attacker sends a UDP packet to a arbitrary port on the sufferer system. When the sufferer system obtains a UDP packet, it will choose what application is waiting on the destination port. When it realizes there is no application that is waiting on the port, it will assemble an ICMP package of place visited unreachable to the false source address. If ample UDP packets are delivered to ports of the sufferer, the systems will proceed down. UDP inundate attacks might also fill the bandwidth of connections located around the sufferer system that depending on the network architecture and line-speed. This can occasionally cause systems connected to a network near a sufferer system to know-how problems with their connectivity.

2.3 ICMP Attacks

Internet Control Message Protocol (ICMP) is an error reporting and analytic utility and it is considered as part of internet Protocol (IP) suite. Whereas this protocol is extremely imperative for ensuring accurate data allocation, it may be exploited by malicious users for conducting different denial of service (DoS) attacks. ICMP inundate attacks, which enables users to drive an echo packet to the host to test check if it's living or not. Expressly throughout a DDoS ICMP inundate attack the agents drive huge volumes of ICMP_ECHO_REPLY packets (ping) to the sufferer. These packets request reply from the sufferer and this consequences in saturation of the bandwidth of the sufferer's network connection. Throughout an ICMP inundate attack the source IP address may be spoofed.

2.4 Memory Depletion Attacks

When a server acknowledges a SIP message, it has to shop little chunks of information. The time for which such information portions are retained depends on the server mode – stateful or stateless. While the transaction is being performed the data is kept in memory and it is deleted only one time the transaction is closed or timed out. The most common attack is a TCP SYN inundates strike. The server is inundated with packets with the SYN flag set. The server assigns the essential resources and answers with packets with SYN+ACK flags. The attacker stays on sending new packets with SYN banners, and does not reply to packets dispatched by the server. The server rapidly depletes all memory resources for a new TCP connection and starts denying normal demands. Another case of this kind of attack would be to drive exceedingly fragmented packets with certain parts intentionally misplaced. The server endeavors to demand the missing parts and stores the obtained packets in memory. The futile information is then retained on the server until it is timed out.

2.5 CPU Depletion

An added way to competently limit the server's capability to method regular demands is central processing unit (CPU) depletion. A higher load might be the result of a higher amount of requests received or by receiving requests requiring extra complex calculations. The server can become inundated with ICMP packets but sending malformed REGISTER messages creates the similar effect with a considerably smaller amount of messages. This really is due to the undeniable fact that messages are studied following the server receives them. Even though the server has the capacity to process hundreds of regular messages, it may be simply forced to do different calculations using malformed messages with fake or invalid data or sent from nonexistent user accounts. Absurdly, enabled authentication on a server can punctual off more demanding procedures, which makes it simpler to decrease CPU resources. When a server uses certificate, the attacker may drive a note by having an invalid certificate. In the end, the server works out that this certificate is invalid nevertheless the processing of the message had already consumed much of server resources. The prospective is flooded with more messages than it may process at a given time. As SIP is a text-based protocol; it has to parse each incoming message. If SIP authentication info is supplied in the inundating message, it has to assess if the user is authorized to get access to the service (assimilate authentication). A specific case is once the target CPU will not extend its operation because it is awaiting input from other entities, such as for instance a database or the domain name system (DNS) service.

2.6 Bandwidth Depletion Attacks

This sort of attack does not consume resources of the physical server but alternatively the capability of the hyperlink connecting the server to the network. When the hyperlink is unable to transmit regular packets, they're unnecessary before they are able to arrive at the SIP server. For this reason it is not possible to differentiate between regular and malicious packets. Utilizing the UDP protocol to transmit SIP messages makes the condition even worst. For reasons, attackers utilize the stateless UDP protocol with the utmost packet size.

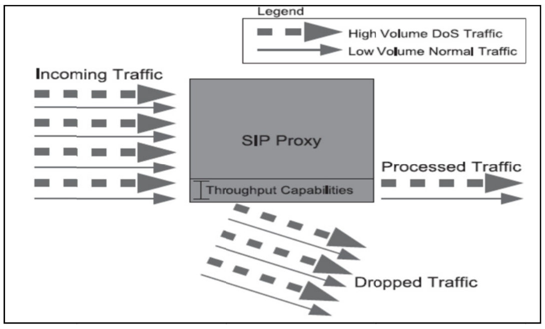


Figure 2. Overview of a DoS flooding attack

Because of the server's limited processing capabilities a allotment of normal demands cannot be processed in case a high load of malicious messages are targeted towards the server.

3. Proposed Solutions using Linux Operating System and IP tables

Linux is open-source operating systems, since it has the key advantageous features where users can modify the code. It is made in this method that it can operate on different types of hardware. The Linux kernel is issued under the general public license version 2 (GPLv2). IP firewalls support different types of filters. IP firewalls be adept of do filters and firewall policy by usernames, by group IDs and user profiles, by source and destination ports, by source host and destination hosts, by uniform resource locator (URL), by IP addresses, by packet ID flags, by protocols and also including filtering by media access control (MAC) address.

3.1 IP Tables

IPtables offers moderately higher speed and reliability compared to the other firewall tools. Being fully a Linux product, its integration with the functioning operating system (OS) is also more powerful and reliable. It keeps a stateful route of each connection passing through it and tries to anticipate feature actions. Its capability of filtering packets on the basis of MAC address makes it a appalling security system. It may filter out attacks using malformed packets and can restrict access from locally attached servers to other networks regardless of their valid IP addresses. Network address translation (NAT) with masquerading convenience of IPtables helps it to conceal internal network IP sub networks behind one or perhaps a small poll of external IP addresses. NAT and masquerading enables the firewall system to open and close ports on the gateway. The rate-limiting characteristic of IPtables can block attacks even from some forms of DoS attacks.

(i) Firewall Rules

Different firewalls usually supply distinct direct reasoning with distinct parameters. But some absolutely vital components are universal to all. Each of them permits an action to be distinct allowing or banning specific network traffic. Also, those all allow checking for most critical elements in packets like IP addresses, ports and protocol. IP Tables is a command line tool for writing and executing of Firewall rules. One of the very important functionalities of IP Tables firewall is stateful inspection (S.I), which routinely opens only the ports necessary for internal packets to access the Internet. It only allows transfer of packets which are defined in firewall rules and which are section of established connections.

(ii) Firewall Chains

IP Tables group policy in chains. Different network packets are route by different chains:

- Incoming traffic – packets for firewall (INPUT chain).
- Forwarding traffic – incoming packets for another appliance (FORWARD chain).
- Outgoing traffic – packets generated by firewall (OUTPUT chain).

Forms of tables have been in IP tables are shown below:

- Filter table- This table principle is used for packet filtering in IP network.
- NAT table- This table policy is not used for packet filtering however it slightly offers NAT/PAT capabilities and IP masquerading.

- Mangle table- This table is employed to change packet fields and is also used to mark packets for later filtering.

The available targets or actions for policy in filter table chains are-

- ACCEPT- This target is requested those packets that are reliable and can be accepted through the firewall.
- REJECT- This target rules is requested those packets that are to be fallen but an ICMP reply will be provided for the inventor for information on what occurring to the packet.
- DROP- This target is placed on those packets that are to be just fallen without dispatching any ICMP reply like it never reached firewall.

(iii) Firewall Rules Parameters

Each rule recognizes explicit kind of network traffic. In alignment to permit this identification parameter for identification of specific network packets must be set for each direct.

Different types of parameters are:

- IP addresses – It can be destination or source IP address also, is often as just one IP, network IP block or IP range.
- Ports - It can be destination or source port also, is often as just one port, port range or port array.
- Protocol – It can be submitted to TCP, UDP, and ICMP or all together.
- Interface – It can be incoming or outgoing interface.
- TTL (Time to Live) field residing in the IP headers.
- ToS (Type of Service) field residing in the IP headers.
- Length of packet.
- MAC source address.
- Syn flag – identification of new connection.
- ICMP type.

4. Result and Analysis

4.1 Basic Security for the Linux Server

Apply rules for remote Secure Shell (SSH) login to the Linux kernel such as changing the default username, password and port number. Do not use the default username root and port 22 as the SSH port. To ensure the primary level security, a system administrator should change these default user name and port number of SSH form the Linux kernel just following the bellow steps:

Add a New Username and Password: To add and give a strong password for a new user has to give the command:

```
[root@l3ippbx ~]# adduser morshed
[root@l3ippbx ~]# passwd morshed
Changing password for user morshed.
New UNIX password: ********
Retype new UNIX password: *******
passwd: all authentication tokens updated successfully.
```

Now to ensure more security for a sever change the default SSH port 22. To do this administrator has to change the directory of the linux as given bellow:

```
root@l3ippbx ssh]# cd /etc/ssh
```

“etc” is a essential directory of linux kernel which provides the host specific system configuration files and has to move to the SSH directory. Here administrator looking for a file which name is `sshd_config`, inside this file administrator can add the login permission of the new user here it is “morshed” and also can change the SSH port 10 instead of default port 22 by using the following command-

```
[root@l3ippbx ssh]# ls -la | grep sshd
-rw-------  1 root root   3331 Oct 12 12:40 sshd_config
```

```
[root@l3ippbx ssh]# vi sshd_config
$OpenBSD: sshd_config,v 1.73 2005/12/06 22:38:28 reyk Exp $
allowUsers morshed
port 10
[root@l3ippbx ssh]# /etc/init.d/sshd restart
Stopping sshd:                                         [OK]
Starting sshd:                                         [OK]
```

Now SSH remote login default username root is disabled and Linux kernel allow only user "morshed" which is created earlier for the remote SSH login and port 10 will be used instead of 22 with the hostname or IP address of the asterisk server. Then the user "morshed" can login to the "root" via bellow command with root password:

```
[morshed@l3ippbx ~]$ su -
Password:
[root@l3ippbx ~]#
```

So this may help something administrator to provide multilevel password protection to his server and unauthorized remote sessions to his server from different hosts. To monitor the authorized sessions, administrator can check the log file of the Linux and asterisk regularly. From the log file we could see hundreds of failed register attempts from the several ip addresses in internet. This log may supply vital information to fight hackers because it shows the scale of the issue, the accounts they are attempting to crack, nature of attack and also attacker IP address.

4.2 Log Monitoring

The useful variable directory "`var`" which contains the entire log file of Linux kernel and also asterisk by using following command:

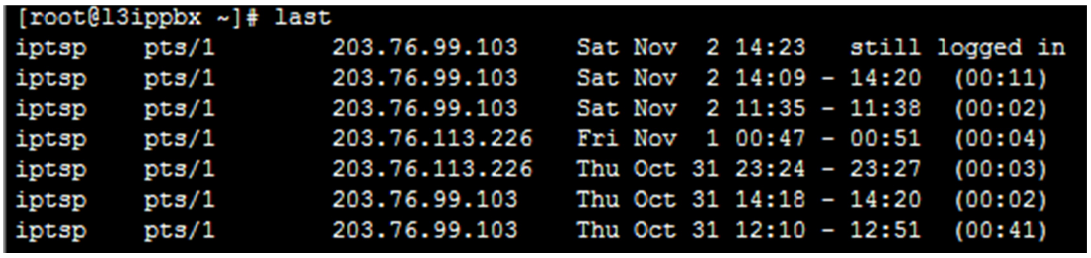
```
[root@l3ippbx ~]# tail -f /var/log/secure
Oct 12 10:34:24 localhost sshd[24809]: Failed password for root from
203.76.99.103 port 49908 ssh2
Oct 12 10:34:28 localhost sshd[24809]: Accepted password for root from
203.76.99.103 port 49908 ssh2
Oct 12 10:34:28 localhost sshd[24809]: pam_unix(sshd:session): session opened
for user root by (uid=0)
Oct 12 10:34:37 localhost sudo:     root : TTY=unknown ; PWD=/root ; USER=root ;
COMMAND=/usr/sbin/asterisk -vvvvvrx core show channels
Oct 12 13:31:21 localhost sshd[27343]: Disconnecting: Too many authentication
failures for root
Oct 12 13:31:21 localhost sshd[27340]: PAM 3 more authentication failures;
logname= uid=0 euid=0 tty=ssh ruser= rhost=67.198.172.202  user=root
Oct 12 13:31:21 localhost sshd[27340]: PAM service(sshd) ignoring max retries;
4 > 3
Oct 12 13:31:23 localhost sshd[27348]: reverse mapping checking getaddrinfo for
67.198.172.202.customer.krypt.com failed - POSSIBLE BREAK-IN ATTEMPT!
Oct 12 13:31:23 localhost sshd[27348]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=67.198.172.202  user=root
Oct 12 13:31:25 localhost sshd[27348]: Failed password for root from
67.198.172.202 port 3778 ssh2
Oct 12 13:31:32 localhost last message repeated 3 times
Oct 12 10:34:47 localhost last message repeated 9 times
[root@l3ippbx~]#tail -f /var/log/elastix/audit.log
[Oct 10 19:58:36] NAVIGATION admin: User admin visited "PBX >> PBX Configuration"
from 203.76.99.103.
```

```
[Oct 10 20:00:03] NAVIGATION admin: User admin visited "PBX >> Tools >>
Asterisk-Cli" from 203.76.99.103.
[Oct 10 20:00:09] NAVIGATION admin: User admin visited "PBX >> Tools >>
Asterisk-Cli" from 203.76.99.103.
```

Directories to see the Linux kernel and full asterisk logs:

```
[root@l3ippbx~]#tail -f /var/log/messages
[root@l3ippbx~]#tail -f /var/log/asterisk/full
```

The Linux kernel log also gives the all successful remote SSH login session with IP address or hostname, login time-date-month, username and time of the active sessions. To view this administrator can run a single command that given bellow Figure 3.

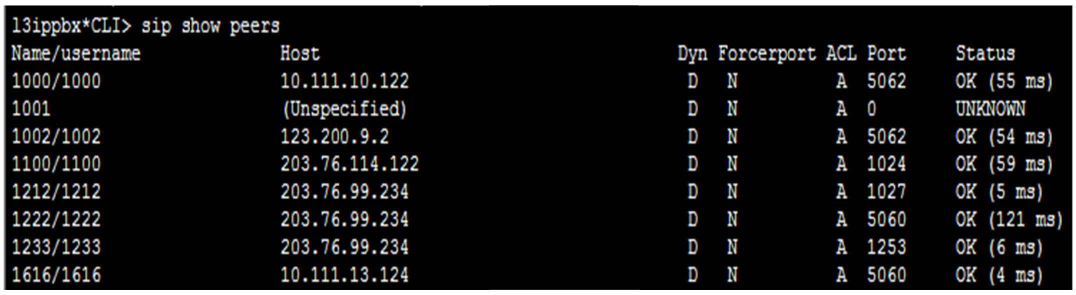


```
[root@l3ippbx ~]# last
iptsp    pts/1        203.76.99.103    Sat Nov  2 14:23   still logged in
iptsp    pts/1        203.76.99.103    Sat Nov  2 14:09 - 14:20  (00:11)
iptsp    pts/1        203.76.99.103    Sat Nov  2 11:35 - 11:38  (00:02)
iptsp    pts/1        203.76.113.226   Fri Nov  1 00:47 - 00:51  (00:04)
iptsp    pts/1        203.76.113.226   Thu Oct 31 23:24 - 23:27  (00:03)
iptsp    pts/1        203.76.99.103    Thu Oct 31 14:18 - 14:20  (00:02)
iptsp    pts/1        203.76.99.103    Thu Oct 31 12:10 - 12:51  (00:41)
```

Figure 3. Host IP addresses list of SSH login to the SIP server

Some useful Asterisk CLI commands help the administrators to see registered extensions with corresponding IP addresses, the active number of calls and channels through bellow Figures 4-6 commands

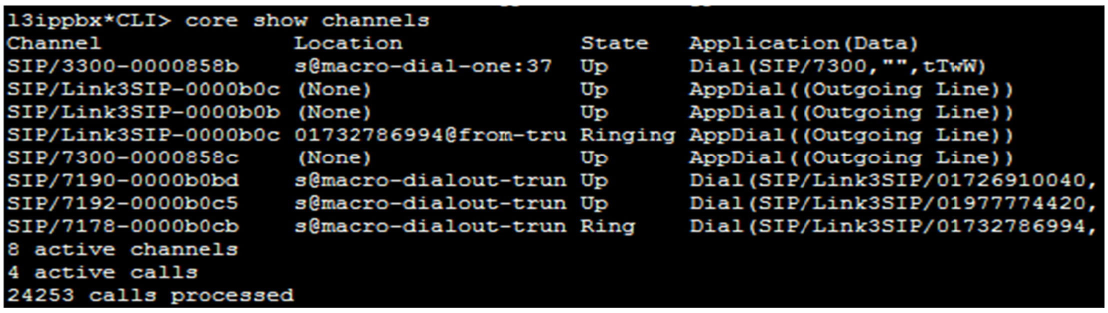
```
[root@l3ippx ~]# asterisk  -r
l3ippx*CLI>
```



```
l3ippbx*CLI> sip show peers
Name/username          Host                       Dyn Forcerport ACL Port     Status
1000/1000              10.111.10.122               D   N          A  5062     OK (55 ms)
1001                   (Unspecified)               D   N          A  0        UNKNOWN
1002/1002              123.200.9.2                 D   N          A  5062     OK (54 ms)
1100/1100              203.76.114.122              D   N          A  1024     OK (59 ms)
1212/1212              203.76.99.234               D   N          A  1027     OK (5 ms)
1222/1222              203.76.99.234               D   N          A  5060     OK (121 ms)
1233/1233              203.76.99.234               D   N          A  1253     OK (6 ms)
1616/1616              10.111.13.124               D   N          A  5060     OK (4 ms)
```

Figure 4. Registered SIP IP addresses list

To see the active calls or channels just run the bellow command at asterisk CLI showed in Figure 5.



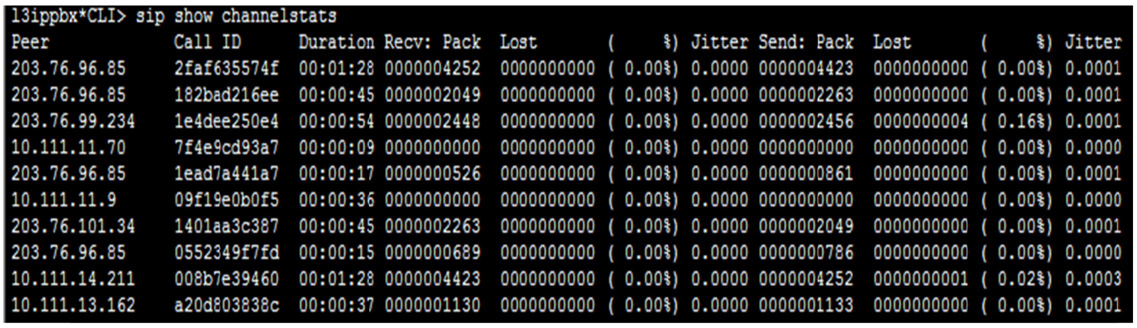
```
l3ippbx*CLI> core show channels
Channel              Location             State   Application(Data)
SIP/3300-0000858b    s@macro-dial-one:37  Up      Dial(SIP/7300,"",tTwW)
SIP/Link3SIP-0000b0c (None)               Up      AppDial((Outgoing Line))
SIP/Link3SIP-0000b0b (None)               Up      AppDial((Outgoing Line))
SIP/Link3SIP-0000b0c 01732786994@from-tru Ringing AppDial((Outgoing Line))
SIP/7300-0000858c    (None)               Up      AppDial((Outgoing Line))
SIP/7190-0000b0bd    s@macro-dialout-trun Up      Dial(SIP/Link3SIP/01726910040,
SIP/7192-0000b0c5    s@macro-dialout-trun Up      Dial(SIP/Link3SIP/01977774420,
SIP/7178-0000b0cb    s@macro-dialout-trun Ring    Dial(SIP/Link3SIP/01732786994,
8 active channels
4 active calls
24253 calls processed
```

Figure 5. Current active calls list at asterisk CLI

Here administrator can hang-up above active call of the extension SIP/7190-0000b0bd by using command form asterisk CLI as given bellow:

```
l3ippx*CLI> channel request hangup SIP/7190-0000b0bd
```

Command to see the active peers with send and receive packet jitter and also the time duration of calls at asterisk CLI



```
l3ippbx*CLI> sip show channelstats
Peer            Call ID      Duration Recv: Pack  Lost       (     %) Jitter Send: Pack  Lost       (     %) Jitter
203.76.96.85    2faf635574f  00:01:28 0000004252  0000000000 ( 0.00%) 0.0000 0000004423  0000000000 ( 0.00%) 0.0001
203.76.96.85    182bad216ee  00:00:45 0000002049  0000000000 ( 0.00%) 0.0000 0000002263  0000000000 ( 0.00%) 0.0001
203.76.99.234   1e4dee250e4  00:00:54 0000002448  0000000000 ( 0.00%) 0.0000 0000002456  0000000004 ( 0.16%) 0.0001
10.111.11.70    7f4e9cd93a7  00:00:09 0000000000  0000000000 ( 0.00%) 0.0000 0000000000  0000000000 ( 0.00%) 0.0000
203.76.96.85    1ead7a441a7  00:00:17 0000000526  0000000000 ( 0.00%) 0.0000 0000000861  0000000000 ( 0.00%) 0.0001
10.111.11.9     09f19e0b0f5  00:00:36 0000000000  0000000000 ( 0.00%) 0.0000 0000000000  0000000000 ( 0.00%) 0.0000
203.76.101.34   1401aa3c387  00:00:45 0000002263  0000000000 ( 0.00%) 0.0000 0000002049  0000000000 ( 0.00%) 0.0001
203.76.96.85    0552349f7fd  00:00:15 0000000689  0000000000 ( 0.00%) 0.0000 0000000786  0000000000 ( 0.00%) 0.0000
10.111.14.211   008b7e39460  00:01:28 0000004423  0000000000 ( 0.00%) 0.0000 0000004252  0000000001 ( 0.02%) 0.0003
10.111.13.162   a20d803838c  00:00:37 0000001130  0000000000 ( 0.00%) 0.0000 0000001133  0000000000 ( 0.00%) 0.0001
```

Figure 6. Active peers with send and receive jitter and also the time duration of calls at asterisk CLI

4.3 Apply Extreme Level Security for SIP Server by Linux IP Tables

IPtables is firewall systems in Linux kernel which provides the users to change and construct an individual freedom. Net filter is the fundamental concept of IPtables. Actually it's a supply filtering firewall of Linux machine that filters packets by the fields in IP, TCP, UDP, and ICMP packet headers etc. By Applying rules at Linux iptables administrator can allow or disallow a certain IP address or range of IP addresses with on the foundation of port number for a certain asterisk server. To make certain more security for the asterisk server administrator can apply rules at Linux IPtables for remote SSH login, Web Browsing (https, http). SIP is a software layer protocol in TCP IP which can be widely using for VOIP to ascertain a period utilizing the default port 5060 which uses UDP and RTP for voice/data transmission by allowing a certain IP or IP addresses range. Network mapping "nmap" in Linux is a critical command to see the open protocols with port number as bellow command output:

```
[root@l3ippbx ~]# nmap 123.200.0.30
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2013-10-12 11:50 BDT
Interesting ports on 123.200.0.30:
Not shown: 1670 closed ports
PORT     STATE SERVICE
10/tcp   open  ssh
80/tcp   open  http
110/tcp  open  pop3
111/tcp  open  rpcbind
143/tcp  open  imap
443/tcp  open  https
993/tcp  open  imaps
995/tcp  open  pop3s
1002/tcp open  windows-icfw
3306/tcp open  mysql
nmap finished: 1 IP address (1 host up) scanned in 0.083 seconds
```

Iptables rules for remote SSH login to the asterisk server:

```
[root@l3ippx ~]# /sbin/iptables -A INPUT -p tcp -s 203.76.0.0/16 --dport 10 -j
ACCEPT
```

```
[root@l3ippx ~]# /sbin/iptables -A INPUT -p tcp --dport 10 -j DROP
[root@l3ippx ~]# service iptables save
[root@l3ippx ~]# iptables -L
Chain INPUT (policy ACCEPT)
target          prot opt source                  destination
ACCEPT     tcp  --  203.76.0.0/16         anywhere          tcp dpt:ssh
DROP       tcp  --  anywhere              anywhere          tcp dpt:ssh
```

IPtables rules for allowing and disallowing IP addresses for Session Initiation Protocol (SIP) Register request through the default port 5060. Session Initiation Protocol (SIP), an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with a number of applicants. This session integrated internet telephone calls, multimedia conferences, instant messaging, voice, video, interactive games, and virtual reality etc. A SIP message includes a header by having an elective body. These messages are either requests or responses. SIP entity received request and it performs the corresponding action and sends back an answer to the originator of the request. Responses are three digit status codes.

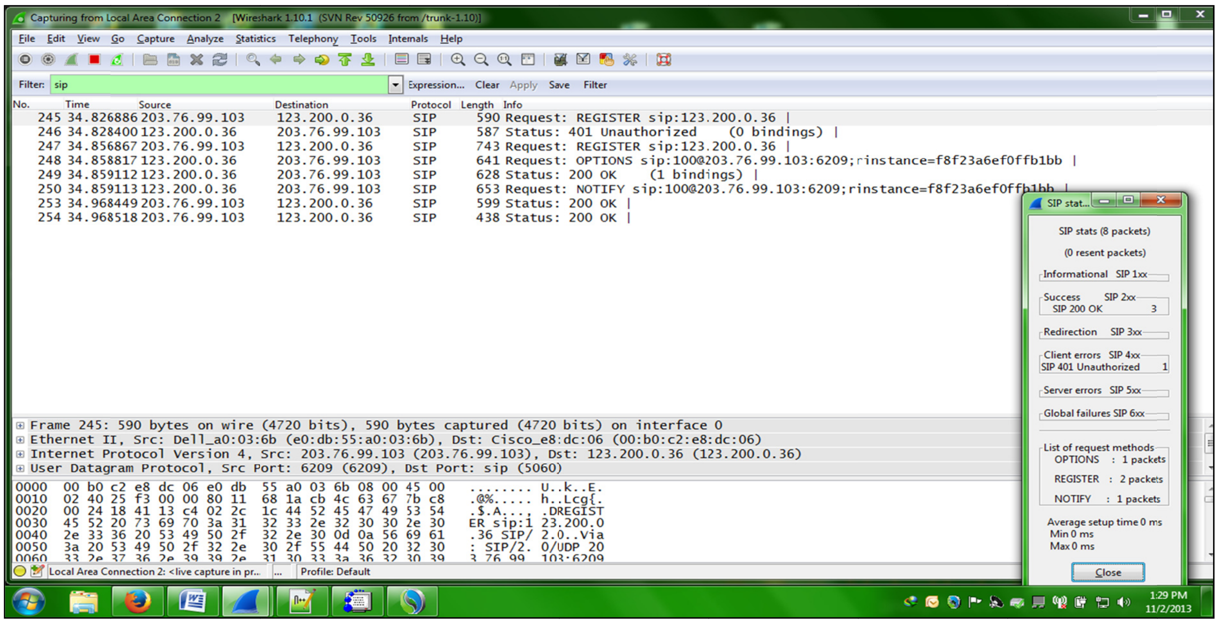
Table 1. SIP Requests

SIP Request	Response
INVITE	Initiates a call by inviting user to participate in session
ACK	Confirms that the client has received a final response to an INVITE request
BYE	Indicates termination of the call
OPTIONS	Used to query the capabilities of a server
CANCEL	Cancels a pending request
REGISTER	Registers the user agent

Table 2. SIP Response Codes

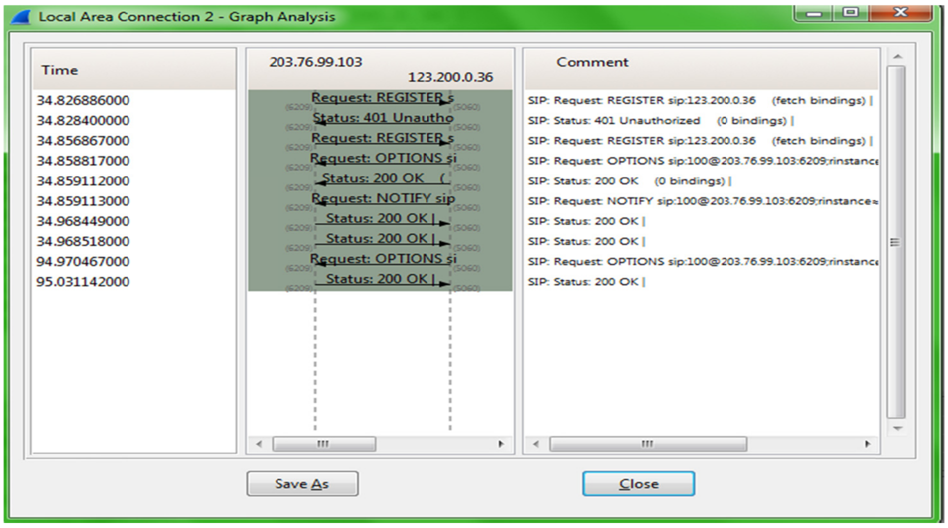
SIP Response codes	Meanings
1XX	Informational Responses
2XX	Successful Responses
3XX	Redirection Responses
4XX	Request Failure Responses
5XX	Server Failure Responses
6XX	Global Failures Responses

Without applying the rules at Linux iptables for SIP default port 5060, it is seen that a user agent with IP address 203.76.99.103 provides 2 register packet requests to the SIP server 123.200.0.36 according to the Figures 4 and 5 at wireshark then the SIP server 123.200.0.36 provides the OPTIONS and successful response code 200 OK as an acknowledgement message to the user agent 203.76.99.103 as given as bellow Figures 7, 8 at wireshark.



```
Capturing from Local Area Connection 2   [Wireshark 1.10.1 (SVN Rev 50926 from /trunk-1.10)]
Filter: sip
No.   Time      Source        Destination    Protocol Length Info
  245 34.826886 203.76.99.103 123.200.0.36   SIP      590 Request: REGISTER sip:123.200.0.36 |
  246 34.828400 123.200.0.36  203.76.99.103  SIP      587 Status: 401 Unauthorized    (0 bindings) |
  247 34.856867 203.76.99.103 123.200.0.36   SIP      743 Request: REGISTER sip:123.200.0.36 |
  248 34.858817 123.200.0.36  203.76.99.103  SIP      641 Request: OPTIONS sip:100@203.76.99.103:6209;rinstance=f8f23a6ef0ffb1bb |
  249 34.859112 123.200.0.36  203.76.99.103  SIP      628 Status: 200 OK    (1 bindings) |
  250 34.859113 123.200.0.36  203.76.99.103  SIP      653 Request: NOTIFY sip:100@203.76.99.103:6209;rinstance=f8f23a6ef0ffb1bb |
  253 34.968449 203.76.99.103 123.200.0.36   SIP      599 Status: 200 OK |
  254 34.968518 203.76.99.103 123.200.0.36   SIP      438 Status: 200 OK |

⊞ Frame 245: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits) on interface 0
⊞ Ethernet II, Src: Dell_a0:03:6b (e0:db:55:a0:03:6b), Dst: Cisco_e8:dc:06 (00:b0:c2:e8:dc:06)
⊞ Internet Protocol Version 4, Src: 203.76.99.103 (203.76.99.103), Dst: 123.200.0.36 (123.200.0.36)
⊞ User Datagram Protocol, Src Port: 6209 (6209), Dst Port: sip (5060)

0000  00 b0 c2 e8 dc 06 e0 db  55 a0 03 6b 08 00 45 00   ........ U..k..E.
0010  02 40 25 f3 00 00 80 11  68 1a cb 4c 63 67 7b c8   .@%..... h..Lcg{.
0020  00 24 18 41 13 c4 02 2c  1c 44 52 45 47 49 53 54   .$.A..., .DREGIST
0030  45 52 20 73 69 70 3a 31  32 33 2e 32 30 30 2e 30   ER sip:1 23.200.0
0040  2e 33 36 20 53 49 50 2f  32 2e 30 0d 0a 56 69 61   .36 SIP/ 2.0..Via
0050  3a 20 53 49 50 2f 32 2e  30 2f 55 44 50 20 32 30   : SIP/2. 0/UDP 20
0060  33 2e 37 36 2e 39 39 2e  31 30 33 3a 36 32 30 39   3.76.99. 103:6209
Local Area Connection 2: <live capture in pr...   Profile: Default

SIP stats (8 packets)
(0 resent packets)
Informational   SIP 1xx
Success         SIP 2xx
  SIP 200 OK                 3
Redirection     SIP 3xx
Client errors   SIP 4xx
SIP 401 Unauthorized         1
Server errors   SIP 5xx
Global failures SIP 6xx
List of request methods
  OPTIONS   :  1 packets
  REGISTER  :  2 packets
  NOTIFY    :  1 packets
Average setup time 0 ms
 Min 0 ms
 Max 0 ms
```

Figure 7. SIP register request packets flow form IP 203.76.99.103 to SIP server 123.200.0.36



```
Local Area Connection 2 - Graph Analysis
Time          203.76.99.103  123.200.0.36   Comment
34.826886000  Request: REGISTER s           SIP: Request: REGISTER sip:123.200.0.36    (fetch bindings) |
34.828400000  Status: 401 Unautho           SIP: Status: 401 Unauthorized    (0 bindings) |
34.856867000  Request: REGISTER s           SIP: Request: REGISTER sip:123.200.0.36    (fetch bindings) |
34.858817000  Request: OPTIONS si           SIP: Request: OPTIONS sip:100@203.76.99.103:6209;rinstance
34.859112000  Status: 200 OK   (            SIP: Status: 200 OK    (0 bindings) |
34.859113000  Request: NOTIFY sip           SIP: Request: NOTIFY sip:100@203.76.99.103:6209;rinstance=
34.968449000  Status: 200 OK |              SIP: Status: 200 OK |
34.968518000  Status: 200 OK |              SIP: Status: 200 OK |
94.970467000  Request: OPTIONS si           SIP: Request: OPTIONS sip:100@203.76.99.103:6209;rinstance
95.031142000  Status: 200 OK |              SIP: Status: 200 OK |
```

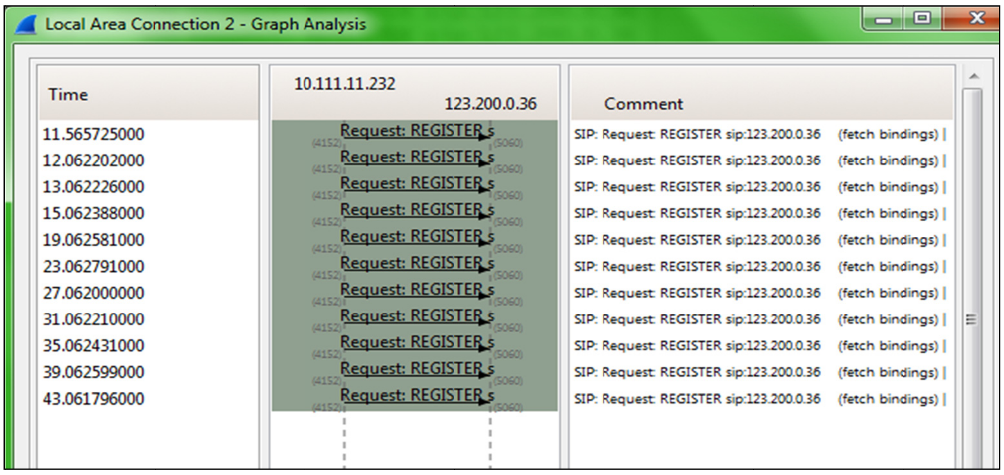
Figure 8. Flow graph of SIP register request form the host 203.76.99.103 to SIP server 123.200.0.36

Now apply the rules at Linux kernel iptables to allow the IP block 203.76.0.0./16 and disallow rest of the IP addresses of the world for SIP Register request through the default port 5060 for the SIP server with IP address 123.200.0.36 by using bellow commands.

```
[root@l3ippx ~]# /sbin/iptables -A INPUT -p udp -s 203.76.0.0/16  --dport 5060
-j ACCEPT
[root@l3ippx ~]#  /sbin/iptables -A INPUT -p udp --dport 5060 -j DROP
[root@l3ippx ~]# service iptables save
Saving firewall rules to /etc/sysconfig/iptables:            [  OK  ]
[root@l3ippx ~]# iptables -L
Chain INPUT (policy ACCEPT)
target          port    opt source                destination
```

```
ACCEPT    udp --  203.76.0.0/16       anywhere             udp dpt:sip
DROP      udp  --  anywhere           anywhere             udp dpt:sip
```

Now send register request form the user agent with IP address 10.111.11.232 which is not at allowed ip block therefore Linux kernel dropped the SIP register request though the agent 10.111.11.232 send 11 packets of register request to the SIP server 123.200.0.36 according to the wireshark flow graph in Figure 9.



Local Area Connection 2 - Graph Analysis

```
Time            10.111.11.232           123.200.0.36    Comment
11.565725000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
12.062202000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
13.062226000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
15.062388000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
19.062581000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
23.062791000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
27.062000000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
31.062210000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
35.062431000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
39.062599000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
43.061796000    (4152) Request: REGISTER s (5060)       SIP: Request: REGISTER sip:123.200.0.36   (fetch bindings) |
```

Figure 9. Wireshark flow graph for SIP register request from user agent 10.111.11.232 to SIP server 123.200.36

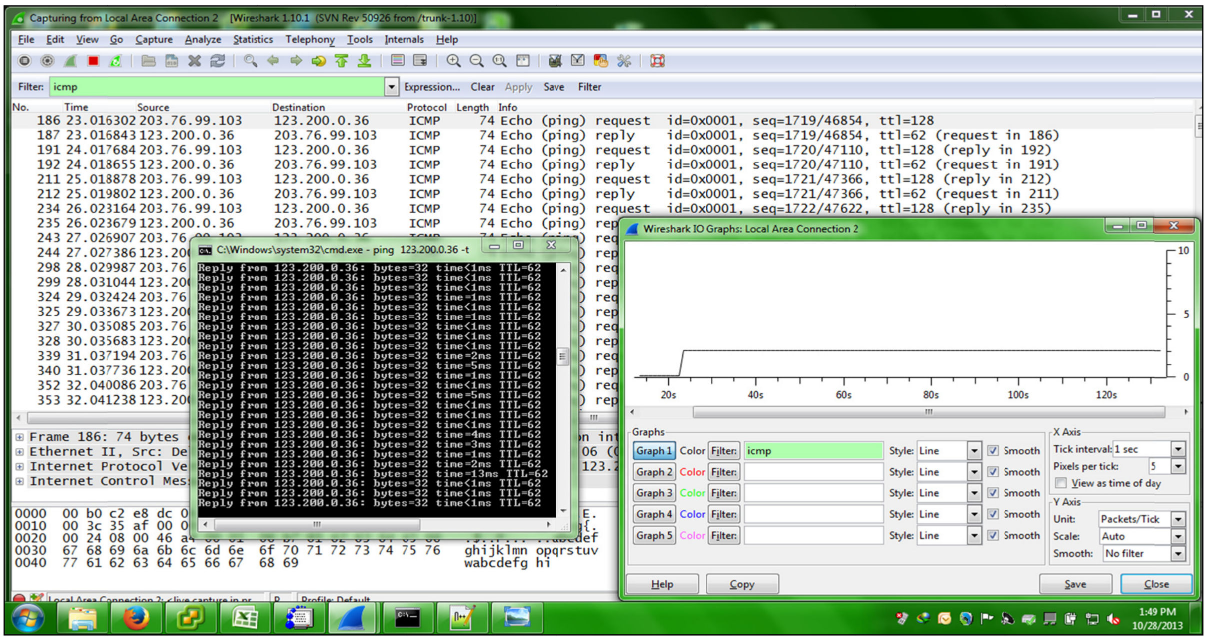
4.4 Prevent Web Based Attacks

As some asterisk based PBX software provides the graphical user interface, so administrator can allow a particular IP or IP address ranges for graphical usually admin login to the server by allowing or blocking the HTTPS protocol port number as given bellow:

```
[root@l3ippx ~]# /sbin/iptables -A INPUT -p tcp -s 203.76.99.0/24 --dport 443
-j ACCEPT
[root@l3ippx ~]# /sbin/iptables -A INPUT -p tcp --dport 443 -j DROP
[root@l3ippx ~]#  service iptables save
[root@l3ippx ~]# iptables -L
Chain INPUT (policy ACCEPT)
target         prot opt source                 destination
ACCEPT   tcp   --  203.76.99.0/24      anywhere            tcp dpt:https
DROP     tcp   --  anywhere            anywhere            tcp dpt:https
```

4.5 Prevent Internet Control Message Protocol (ICMP) Flood Attack

To confuse the hacker or unauthorized users administrator can allow ICMP ping, trace route request for particular IP address or IP ranges and disallow rest of the IP address of the world for a particular SIP server.


Capturing from Local Area Connection 2 [Wireshark 1.10.1 (SVN Rev 50926 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save Filter
Wireshark IO Graphs: Local Area Connection 2
10
5
0
20s
40s
60s
80s
100s
120s
Help Copy Save Close
1:49 PM
10/28/2013

Filter: icmp

```
No.  Time      Source        Destination   Protocol Length Info
 186 23.016302 203.76.99.103 123.200.0.36  ICMP     74 Echo (ping) request  id=0x0001, seq=1719/46854, ttl=128
 187 23.016843 123.200.0.36  203.76.99.103 ICMP     74 Echo (ping) reply    id=0x0001, seq=1719/46854, ttl=62 (request in 186)
 191 24.017684 203.76.99.103 123.200.0.36  ICMP     74 Echo (ping) request  id=0x0001, seq=1720/47110, ttl=128 (reply in 192)
 192 24.018655 123.200.0.36  203.76.99.103 ICMP     74 Echo (ping) reply    id=0x0001, seq=1720/47110, ttl=62 (request in 191)
 211 25.018878 203.76.99.103 123.200.0.36  ICMP     74 Echo (ping) request  id=0x0001, seq=1721/47366, ttl=128 (reply in 212)
 212 25.019802 123.200.0.36  203.76.99.103 ICMP     74 Echo (ping) reply    id=0x0001, seq=1721/47366, ttl=62 (request in 211)
 234 26.023164 203.76.99.103 123.200.0.36  ICMP     74 Echo (ping) request  id=0x0001, seq=1722/47622, ttl=128 (reply in 235)
 235 26.023679 123.200.0.36  203.76.99.103 ICMP     74 Echo (ping) rep
 243 27.026907 203.76                                         ) req
 244 27.027386 123.20                                         ) rep
 298 28.029987 203.76                                         ) req
 299 28.031044 123.20                                         ) rep
 324 29.032424 203.76                                         ) req
 325 29.033673 123.20                                         ) rep
 327 30.035085 203.76                                         ) req
 328 30.035683 123.20                                         ) rep
 339 31.037194 203.76                                         ) req
 340 31.037736 123.20                                         ) rep
 352 32.040086 203.76                                         ) req
 353 32.041238 123.20                                         ) rep

⊞ Frame 186: 74 bytes                                         n int
⊞ Ethernet II, Src: De                                        06 (
⊞ Internet Protocol Ve                                        123.
⊞ Internet Control Mes

0000  00 b0 c2 e8 dc 0                                         E.
0010  00 3c 35 af 00 0                                         {.
0020  00 24 08 00 46 a                                         def
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi

Local Area Connection 2: <live capture in pr   P   Profile: Default
```

C:\Windows\system32\cmd.exe - ping 123.200.0.36 -t

```
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time=1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time=1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time=2ms TTL=62
Reply from 123.200.0.36: bytes=32 time=5ms TTL=62
Reply from 123.200.0.36: bytes=32 time=1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time=5ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time=4ms TTL=62
Reply from 123.200.0.36: bytes=32 time=3ms TTL=62
Reply from 123.200.0.36: bytes=32 time=1ms TTL=62
Reply from 123.200.0.36: bytes=32 time=2ms TTL=62
Reply from 123.200.0.36: bytes=32 time=13ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
Reply from 123.200.0.36: bytes=32 time<1ms TTL=62
```

Graphs

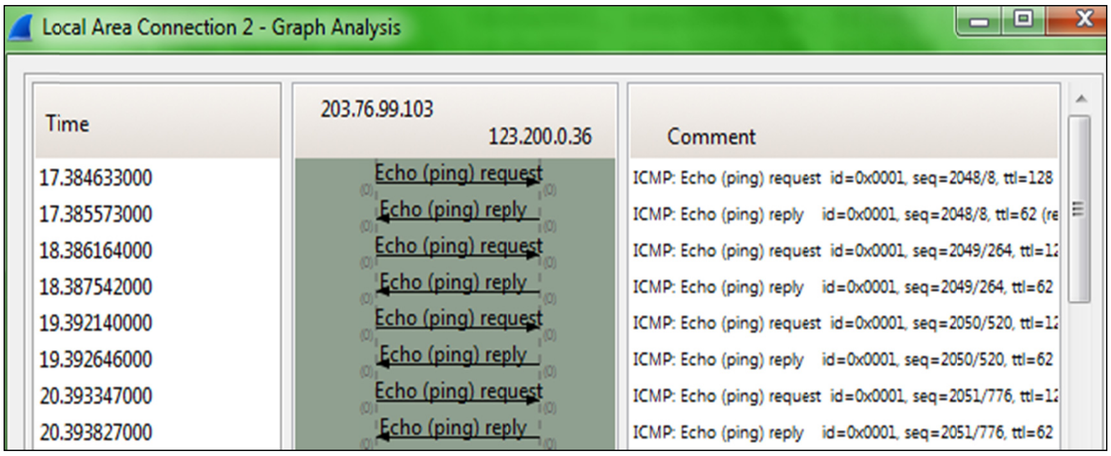
Graph 1	Color	Filter: icmp	Style: Line	✓	Smooth
Graph 2	Color	Filter:	Style: Line	✓	Smooth
Graph 3	Color	Filter:	Style: Line	✓	Smooth
Graph 4	Color	Filter:	Style: Line	✓	Smooth
Graph 5	Color	Filter:	Style: Line	✓	Smooth

X Axis
Tick interval: 1 sec
Pixels per tick: 5
☐ View as time of day

Y Axis
Unit: Packets/Tick
Scale: Auto
Smooth: No filter

Figure 10. Wireshark view for ping request form 203.76.99.103 to 123.200.0.36 and packet/tick graph at wireshark

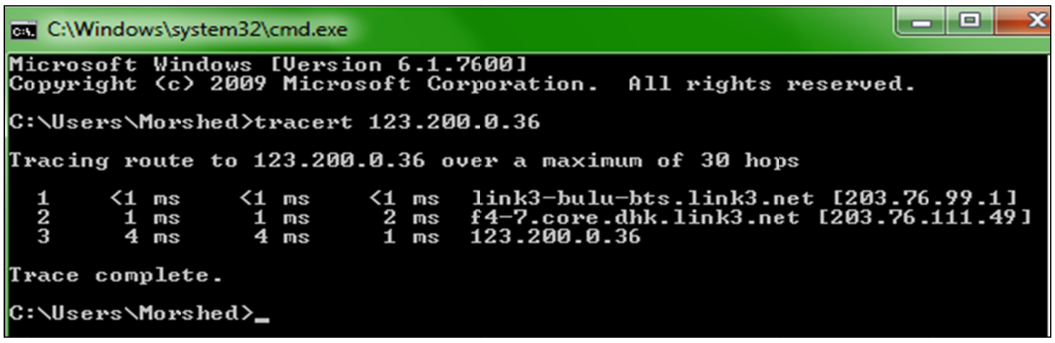
ICMP works at the network layer and ICMP flood attack which enables users to send an echo packet to a remote host to check whether it's alive or not. Without applying rules at Linux iptables for ICMP ping request at the asterisk server, it allows input ping request packet from different host and then transmit a echo or reply packet to that host as an acknowledgement as like as Figure 8. Here a host 203.76.99.103 which OS is windows-7 sends ping request to the asterisk server which OS is Red Hat Linux (centos-6.4) with ip address 123.200.0.36. Since still Linux sever do not apply iptables rules for ICMP, host 203.76.99.103 gets continuous TTL response as wireshark shows the ping request from 203.76.99.103 and reply request from 123.200.0.36. Therefore Packet/Tick graph at wireshark gives the continuous line as like as bellow Figure 11.



Time	203.76.99.103 / 123.200.0.36	Comment
17.384633000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=2048/8, ttl=128
17.385573000	Echo (ping) reply	ICMP: Echo (ping) reply id=0x0001, seq=2048/8, ttl=62 (re
18.386164000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=2049/264, ttl=1
18.387542000	Echo (ping) reply	ICMP: Echo (ping) reply id=0x0001, seq=2049/264, ttl=62
19.392140000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=2050/520, ttl=1
19.392646000	Echo (ping) reply	ICMP: Echo (ping) reply id=0x0001, seq=2050/520, ttl=62
20.393347000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=2051/776, ttl=1
20.393827000	Echo (ping) reply	ICMP: Echo (ping) reply id=0x0001, seq=2051/776, ttl=62

Figure 11. Wireshark flow graph for ping request form 203.76.99.103 to 123.200.0.36

Without ICMP rules at Linux firewall the trace route command by host 203.76.99.103 gives the complete path to reach the SIP server 123.200.0.36 as given bellow Figure 12.

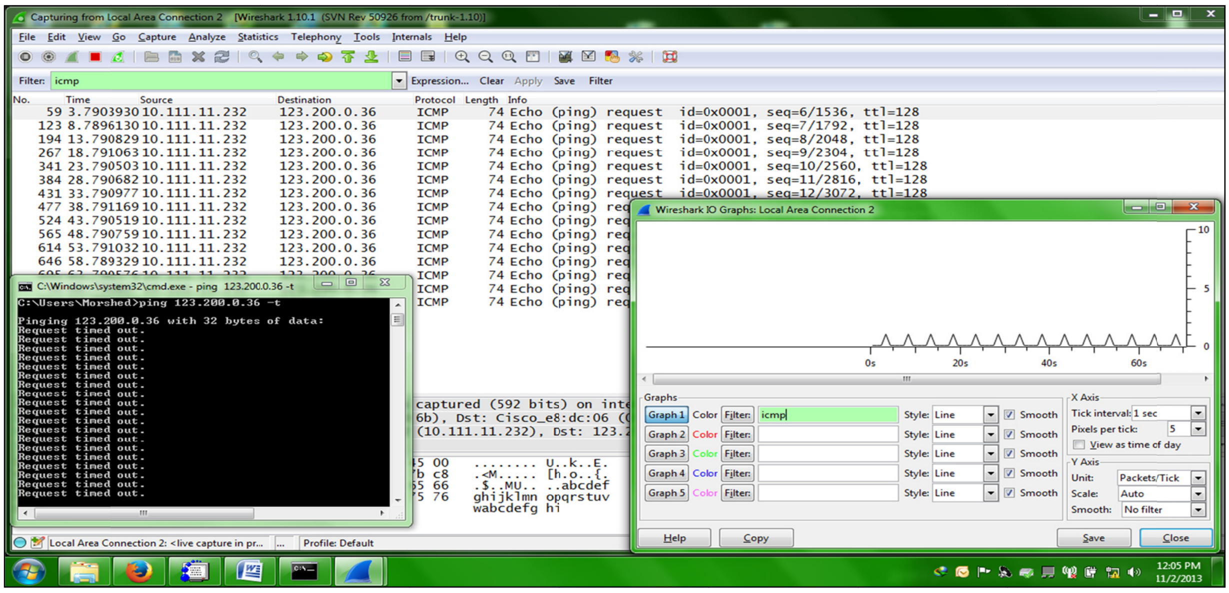


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Morshed>tracert 123.200.0.36

Tracing route to 123.200.0.36 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms  link3-bulu-bts.link3.net [203.76.99.1]
  2     1 ms     1 ms     2 ms  f4-7.core.dhk.link3.net [203.76.111.49]
  3     4 ms     4 ms     1 ms  123.200.0.36

Trace complete.

C:\Users\Morshed>_
```

Figure 12. Trace route path from the host 203.76.99.103 to SIP server 123.200.0.36

Now apply the rules for ICMP at iptables of Linux kernel by using bellow commands, here input icmp echo-request of the server with IP 123.200.0.36 is allowed only for IP block 203.76.99.0/24 and rest of the IP addresses of the world will be dropped when they send the icmp ping request to this asterisk server with IP address 123.200.0.36 by the Linux kernel.

```
[root@l3ippx~]# iptables -A INPUT -s 203.76.99.0/24 -p icmp --icmp-type
echo-request -j ACCEPT
[root@l3ippx ~]# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
[root@l3ippx ~]# service iptables save
Saving firewall rules to /etc/sysconfig/iptables:           [  OK  ]
[root@l3ippx ~]# iptables -L
Chain INPUT (policy ACCEPT)
target             prot  opt  source                  destination
ACCEPT      icmp --  203.76.99.0/24          anywhere     icmp echo-request
DROP       icmp --  anywhere                 anywhere     icmp echo-request
```


Capturing from Local Area Connection 2 [Wireshark 1.10.1 (SVN Rev 50926 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save Filter
Wireshark IO Graphs: Local Area Connection 2
10
5
0
0s
20s
40s
60s
Help
Copy
Save
Close

```
Filter: icmp
No.     Time      Source          Destination    Protocol Length Info
    59 3.7903930 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) request  id=0x0001, seq=6/1536, ttl=128
   123 8.7896130 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) request  id=0x0001, seq=7/1792, ttl=128
   194 13.790829 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) request  id=0x0001, seq=8/2048, ttl=128
   267 18.791063 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) request  id=0x0001, seq=9/2304, ttl=128
   341 23.790503 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) request  id=0x0001, seq=10/2560, ttl=128
   384 28.790682 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) request  id=0x0001, seq=11/2816, ttl=128
   431 33.790977 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) request  id=0x0001, seq=12/3072, ttl=128
   477 38.791169 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) req
   524 43.790519 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) req
   565 48.790759 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) req
   614 53.791032 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) req
   646 58.789329 10.111.11.232   123.200.0.36   ICMP       74 Echo (ping) req
                                                ICMP       74 Echo (ping) req
                                                ICMP       74 Echo (ping) req
                                                ICMP       74 Echo (ping) req

captured (592 bits) on inte
6b), Dst: Cisco_e8:dc:06 ((
(10.111.11.232), Dst: 123.2

5 00    ........ U..k..E.
b c8    .<M..... [h.o..{.
5 66    .$..MU.. ..abcdef
5 76    ghijklmn opqrstuv
        wabcdefg hi

Local Area Connection 2: <live capture in pr...   ...   Profile: Default
```

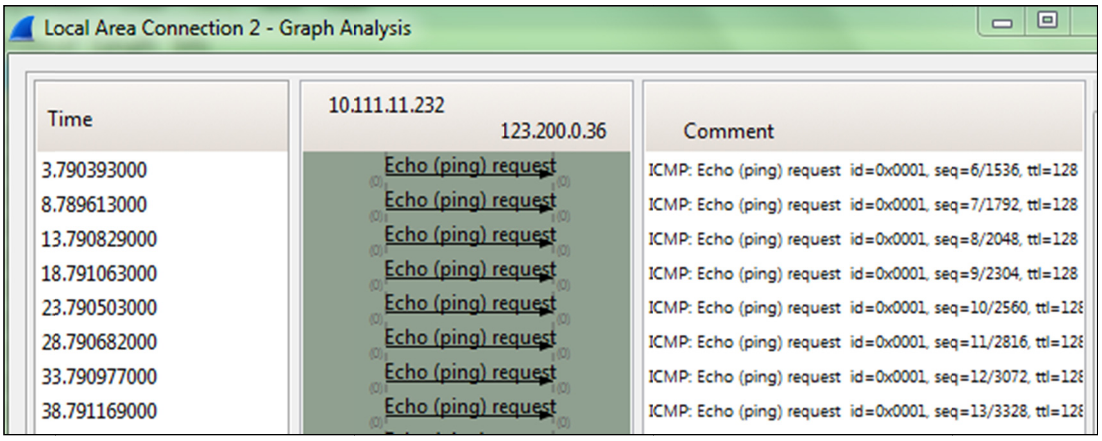
```
C:\Windows\system32\cmd.exe - ping 123.200.0.36 -t
C:\Users\Morshed>ping 123.200.0.36 -t

Pinging 123.200.0.36 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
```

```
Graphs
Graph 1  Color  Filter: icmp   Style: Line  [✓] Smooth
Graph 2  Color  Filter:        Style: Line  [✓] Smooth
Graph 3  Color  Filter:        Style: Line  [✓] Smooth
Graph 4  Color  Filter:        Style: Line  [✓] Smooth
Graph 5  Color  Filter:        Style: Line  [✓] Smooth
X Axis
Tick interval: 1 sec
Pixels per tick: 5
[ ] View as time of day
Y Axis
Unit: Packets/Tick
Scale: Auto
Smooth: No filter

12:05 PM
11/2/2013
```

Figure 13. Wireshark view for the ping request form host 10.111.11.232 to Linux server with IP address 123.200.0.36 and and packet/tick graph at wireshark

Now the ping request sender 10.111.11.232 will get request time out as an acknowledgment message from the host 123.200.0.36 because Linux kernel will dropped the ping request packets as the kernel allow only the IP block

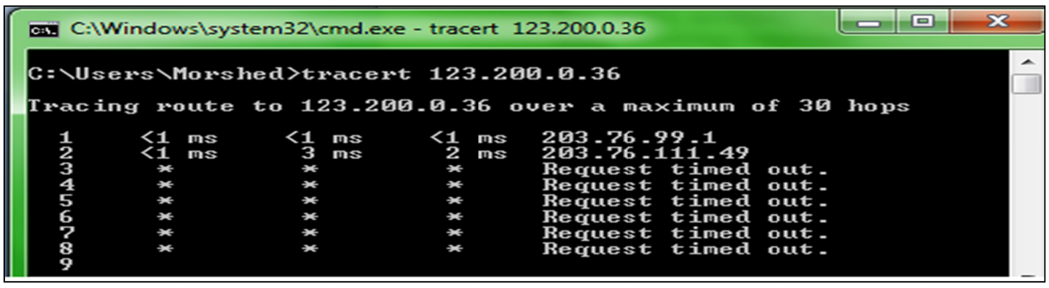
203.76.99.0/24 and there will be no reply request packet to the host 10.111.11.232 from 123.200.0.36 as wireshark flow graph shows at Figure 13. Therefore the Packet/Ticks graph Figure 14 at wireshark for icmp ping request shows the discontinuity of packets as the host 123.200.0.36 did not send reply ping request packets to host 10.111.11.232


Local Area Connection 2 - Graph Analysis

Time	10.111.11.232 123.200.0.36	Comment
3.790393000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=6/1536, ttl=128
8.789613000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=7/1792, ttl=128
13.790829000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=8/2048, ttl=128
18.791063000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=9/2304, ttl=128
23.790503000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=10/2560, ttl=12
28.790682000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=11/2816, ttl=12
33.790977000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=12/3072, ttl=12
38.791169000	Echo (ping) request	ICMP: Echo (ping) request id=0x0001, seq=13/3328, ttl=12

Figure 14. Wireshark flow graph for the ping request form host 10.111.11.232 to Linux server with IP address 123.200.0.36

Now the trace route command by host 10.111.11.232 shows the request time out response and failed to reach the SIP server 123.200.0.36 as given in Figure 14 due to applied rules for ICMP at Linux iptables.


C:\Windows\system32\cmd.exe - tracert 123.200.0.36

```
C:\Users\Morshed>tracert 123.200.0.36
Tracing route to 123.200.0.36 over a maximum of 30 hops
  1    <1 ms    <1 ms    <1 ms  203.76.99.1
  2    <1 ms     3 ms     2 ms  203.76.111.49
  3     *        *        *     Request timed out.
  4     *        *        *     Request timed out.
  5     *        *        *     Request timed out.
  6     *        *        *     Request timed out.
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9
```

Figure 15. Trace route path form the host 10.111.11.232 to SIP server 123.200.0.36 after applying ICMP firewall

Linux iptables show the accepted and dropped packets according to apply rules as like as bellow command output in Figure 16.

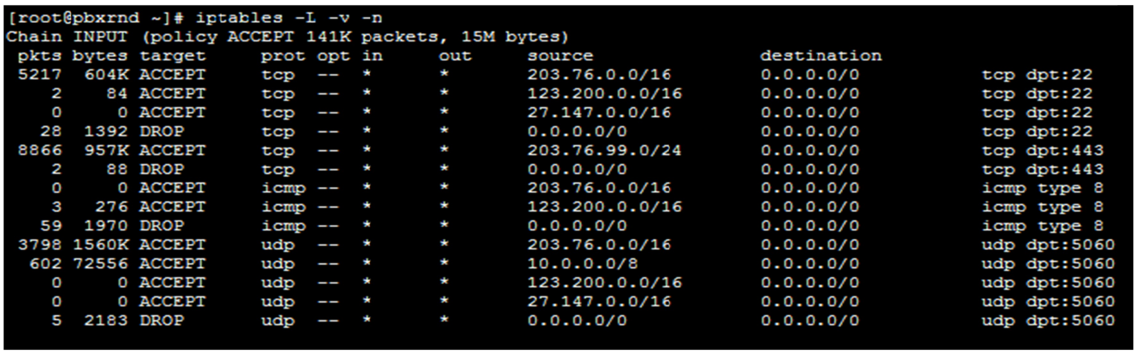

```
[root@pbxrnd ~]# iptables -L -v -n
Chain INPUT (policy ACCEPT 141K packets, 15M bytes)
 pkts bytes target     prot opt in     out     source               destination
 5217  604K ACCEPT     tcp  --  *      *       203.76.0.0/16        0.0.0.0/0           tcp dpt:22
    2    84 ACCEPT     tcp  --  *      *       123.200.0.0/16       0.0.0.0/0           tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       27.147.0.0/16        0.0.0.0/0           tcp dpt:22
   28  1392 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
 8866  957K ACCEPT     tcp  --  *      *       203.76.99.0/24       0.0.0.0/0           tcp dpt:443
    2    88 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
    0     0 ACCEPT     icmp --  *      *       203.76.0.0/16        0.0.0.0/0           icmp type 8
    3   276 ACCEPT     icmp --  *      *       123.200.0.0/16       0.0.0.0/0           icmp type 8
   59  1970 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0           icmp type 8
 3798 1560K ACCEPT     udp  --  *      *       203.76.0.0/16        0.0.0.0/0           udp dpt:5060
  602 72556 ACCEPT     udp  --  *      *       10.0.0.0/8           0.0.0.0/0           udp dpt:5060
    0     0 ACCEPT     udp  --  *      *       123.200.0.0/16       0.0.0.0/0           udp dpt:5060
    0     0 ACCEPT     udp  --  *      *       27.147.0.0/16        0.0.0.0/0           udp dpt:5060
    5  2183 DROP       udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:5060
```

Figure 16. Accepted and dropped packets list according to rules applied at Linux firewall/iptables

Administrator can delete existing rules at Iptables through following steps as given below. User has to give the command to see the iptables rules with the line number.

```
[root@pbxrnd ~]# iptables -L INPUT --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    DROP       all  --  67.198.172.202.CUSTOMER.KRYPT.COM  anywhere
2    DROP       all  --  78.16.129.52         anywhere
3    DROP        icmp --  anywhere    anywhere            icmp echo-request
```

Suppose user wants to delete the rules number 2, then follow steps given bellow:

```
[root@pbxrnd ~]# iptables -D INPUT 2
[root@pbxrnd ~]# servcie iptables save
Command to restart and Flush Linux iptabes
[root@pbxrnd ~]# service iptables restart
[root@pbxrnd ~]# iptables –F
```

4.6 SIP Configuration Security

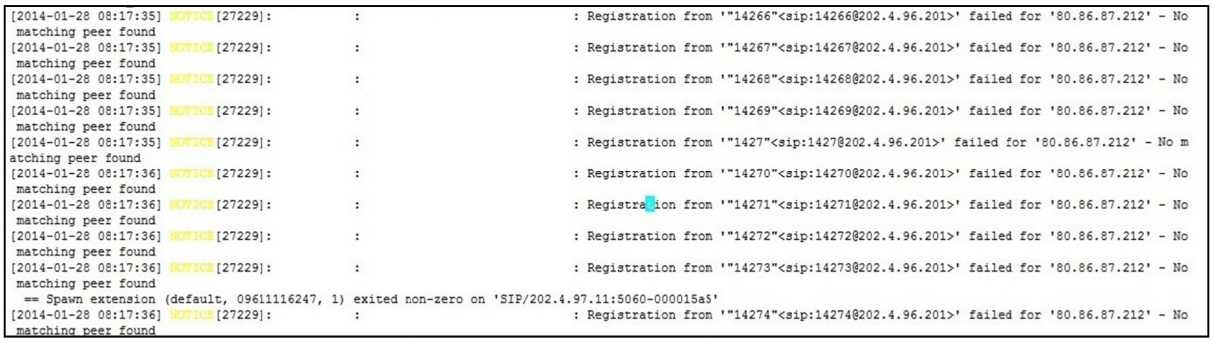
To make certain security for each static extension at `sip.conf` file of asterisk, administrator can set each extension parameter value safer like do not accept SIP authentication requests from all IP addresses. Utilize the "`permit`" and "`deny`" lines in `sip.conf` to only permit an acceptable subset of IP addresses to attain each listed extension/user. For instance parameter "`permit=10.111.0.0/255.255.0.0`" is only allowed this kind of subset of IP addresses for SIP register authentication to the asterisk server others IP or IP ranges is going to be blocked. Use strong passwords for SIP entities. To defend asterisk server from password estimating attacks, always ensure that all SIP client anecdotes have a strong password at smallest 6 characters with a blend of top and smaller case notes in addition to numeric digits and at least one non-standard feature such as for example $, !, #, *, etc. For instance secret=M0r$H3d!100 can be quite a powerful password. To ascertain the set of user anecdotes and passwords in asterisk, run the order "sip display users" at the asterisk CLI.

Always use "`type=peer`" and not ever "`type=friend`": When supplementing static SIP gazes or "SIP Trunks" in FreePBX/ Trixbox/ Elastix set the kind to "peer" and not ever set it to "friend". Utilizing "`type=friend`" will make asterisk server much more vulnerable because "`type=friend`" really determinants two objects to be conceived a SIP gaze and a SIP user. By far the poorest error that a manager could make when defining a static SIP peer would be to have both "`type=friend`" and "`insecure=invite`". In this status, a hacker could kick off calls from any isolated IP address without needing to authenticate with a password. They would only require to supposition one part of data the client name.

4.7 Extensions Security Using Fail2ban in SIP Server

Fail2ban operates by monitoring log files (e.g. /var/log/pwdfail, /var/log/auth.log, etc.) for selected entries and running scripts based on them. Most commonly this is used to block selected IP addresses that may belong to hosts that are trying to breach the system's security. It can ban any host IP that makes too many login attempts or performs any other unwanted action within a time frame defined by the administrator. Fail2ban is typically set up to unban a blocked host within a certain period, so as to not "lock out" any genuine connections that may have been temporarily misconfigured. However, an unban time of several minutes is usually enough to stop a network connection being flooded by malicious connections, as well as reducing the likelihood of a successful dictionary attack. Fail2ban can perform multiple actions whenever an abusive IP is detected: update iptables or PF firewall rules, TCP Wrapper's hosts.deny table, to reject an abuser's IP address; email notifications; or any user-defined action that can be carried out by a Python script.

It works by scanning log files and then taking action based on the entries in those logs. Fail2ban implemented with a configuration to be able to prevent SIP brute force attacks against our Asterisk PBXs. The following describes how to setup fail2ban to protect an Asterisk PBX from SIP brute force attempts and scans utilizing the iptables firewall. After install fail2band in SIP server, we need to create a configuration for Fail2Ban so that it can understand attacks against Asterisk. If having issues with system not banning properly when the "Registration from" section in your log file contains a quotation mark (") as in the Figure 17.



```
[2014-01-28 08:17:35] NOTICE[27229]:          :                         : Registration from '"14266"<sip:14266@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
[2014-01-28 08:17:35] NOTICE[27229]:          :                         : Registration from '"14267"<sip:14267@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
[2014-01-28 08:17:35] NOTICE[27229]:          :                         : Registration from '"14268"<sip:14268@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
[2014-01-28 08:17:35] NOTICE[27229]:          :                         : Registration from '"14269"<sip:14269@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
[2014-01-28 08:17:35] NOTICE[27229]:          :                         : Registration from '"1427"<sip:1427@202.4.96.201>' failed for '80.86.87.212' - No m
atching peer found
[2014-01-28 08:17:36] NOTICE[27229]:          :                         : Registration from '"14270"<sip:14270@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
[2014-01-28 08:17:36] NOTICE[27229]:          :                         : Registration from '"14271"<sip:14271@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
[2014-01-28 08:17:36] NOTICE[27229]:          :                         : Registration from '"14272"<sip:14272@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
[2014-01-28 08:17:36] NOTICE[27229]:          :                         : Registration from '"14273"<sip:14273@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
  == Spawn extension (default, 09611116247, 1) exited non-zero on 'SIP/202.4.97.11:5060-000015a5'
[2014-01-28 08:17:36] NOTICE[27229]:          :                         : Registration from '"14274"<sip:14274@202.4.96.201>' failed for '80.86.87.212' - No
 matching peer found
```

Figure 17. SIP flood attacks

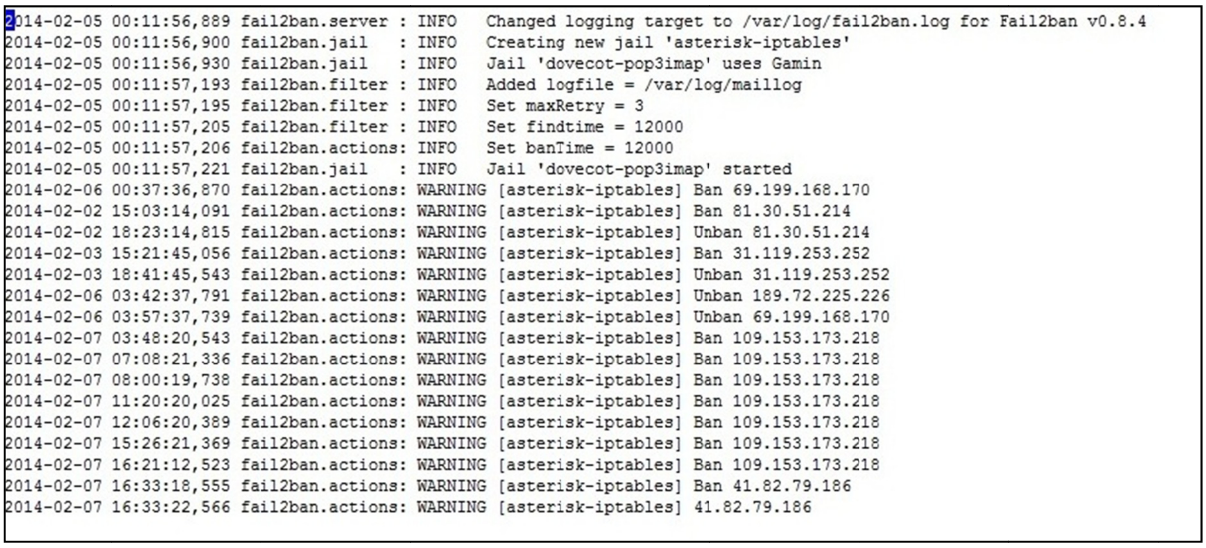
To prevent SIP flood by fail2ban need to create a new filter configuration for Asterisk:

```
touch /etc/fail2ban/filter.d/asterisk.conf
```

Need to add the following line will block these attempts. Next edit `/etc/fail2ban/jail.conf` to include the following section so that it uses the new filter. It is also recommend to turn on an iptables ban for ssh and httpd/apache if they are running on the system.

```
[asterisk-iptables]
enabled  = true
filter   = asterisk
action   = iptables-allports[name=ASTERISK, protocol=all]
                sendmail-whois[name=ASTERISK, dest=yeasir08@yahoo.com,
sender=fail2ban@local.local]
logpath  = /var/log/asterisk/messages
maxretry = 3
bantime = 259200
```

After using 3 times wrong authentication attacker IP address will automatically block in SIP server. The block IP address ate given in Figure 18.



```
2014-02-05 00:11:56,889 fail2ban.server : INFO   Changed logging target to /var/log/fail2ban.log for Fail2ban v0.8.4
2014-02-05 00:11:56,900 fail2ban.jail   : INFO   Creating new jail 'asterisk-iptables'
2014-02-05 00:11:56,930 fail2ban.jail   : INFO   Jail 'dovecot-pop3imap' uses Gamin
2014-02-05 00:11:57,193 fail2ban.filter : INFO   Added logfile = /var/log/maillog
2014-02-05 00:11:57,195 fail2ban.filter : INFO   Set maxRetry = 3
2014-02-05 00:11:57,205 fail2ban.filter : INFO   Set findtime = 12000
2014-02-05 00:11:57,206 fail2ban.actions: INFO   Set banTime = 12000
2014-02-05 00:11:57,221 fail2ban.jail   : INFO   Jail 'dovecot-pop3imap' started
2014-02-06 00:37:36,870 fail2ban.actions: WARNING [asterisk-iptables] Ban 69.199.168.170
2014-02-02 15:03:14,091 fail2ban.actions: WARNING [asterisk-iptables] Ban 81.30.51.214
2014-02-02 18:23:14,815 fail2ban.actions: WARNING [asterisk-iptables] Unban 81.30.51.214
2014-02-03 15:21:45,056 fail2ban.actions: WARNING [asterisk-iptables] Ban 31.119.253.252
2014-02-03 18:41:45,543 fail2ban.actions: WARNING [asterisk-iptables] Unban 31.119.253.252
2014-02-06 03:42:37,791 fail2ban.actions: WARNING [asterisk-iptables] Unban 189.72.225.226
2014-02-06 03:57:37,739 fail2ban.actions: WARNING [asterisk-iptables] Unban 69.199.168.170
2014-02-07 03:48:20,543 fail2ban.actions: WARNING [asterisk-iptables] Ban 109.153.173.218
2014-02-07 07:08:21,336 fail2ban.actions: WARNING [asterisk-iptables] Ban 109.153.173.218
2014-02-07 08:00:19,738 fail2ban.actions: WARNING [asterisk-iptables] Ban 109.153.173.218
2014-02-07 11:20:20,025 fail2ban.actions: WARNING [asterisk-iptables] Ban 109.153.173.218
2014-02-07 12:06:20,389 fail2ban.actions: WARNING [asterisk-iptables] Ban 109.153.173.218
2014-02-07 15:26:21,369 fail2ban.actions: WARNING [asterisk-iptables] Ban 109.153.173.218
2014-02-07 16:21:12,523 fail2ban.actions: WARNING [asterisk-iptables] Ban 109.153.173.218
2014-02-07 16:33:18,555 fail2ban.actions: WARNING [asterisk-iptables] Ban 41.82.79.186
2014-02-07 16:33:22,566 fail2ban.actions: WARNING [asterisk-iptables] 41.82.79.186
```

Figure 18. Attacker blocked IP address

5. Conclusion

In this paper, authors focused on capability of iptables rules is explored to protect from this attack. To verify if the network traffic is legitimate or not, associate degree iptables depends on a set of rules it has which can be predefined by way of a network or system administrator. Major concentration of the paper has been on apprehending the live traffic mistreatment the network protocol analyzer Wireshark and on the premise of study scripts mistreatment iptables are developed to allow/deny the network traffic relying upon the traffic rate of any IP address of the computer inducing the packets. Authors have analyzed how Linux iptables rule work with wireshark simulation showing the difference of filtering packets before and after applying rules. Here authors ensured security for port number and some useful protocols such as SIP, SSH, ICMP, HTTPS by filtering TCP and UDP packets with allowing a particular IP addresses with subnet mask and disallowing rest of the addresses of the world to prevent unauthorized attack to the VoIP server. Linux log files which situated at */var* directory is more useful to monitor the SIP server. This paper also shows how to choose strong password pattern that ensured more security to the SIP server.

Acknowledgements

The authors would like to thanks Department of Electrical and Electronic Engineering, School of Engineering and Computer Science, Independent University, Bangladesh.

References

Allawi, D., Rohiem, A. A., El-moghazy, A., & Ghalwash, A. (2013). New Algorithm for SIP Flooding Attack Detection.

AL-Musawi, B. Q. M. (2012). MITIGATING DoS/DDoS ATTACKS USING IPTABLES. *International Journal of Engineering & Technology, 12*(3).

Alqahtani, A. H., & Iftikhar, M. (2013). TCP/IP Attacks, Defenses and Security Tools. *International Journal of Science and Modern Engineering (IJISME), 1*(10).

Arafat, M. Y., Ahmed, F., & Sobhan, M. A. (2013). SIP Security in IP Telephony. *International conference at Elastix World* Mexico, October, 2013.

Chatterjee, K. (2013). Design and Development of a Framework to Mitigate DoS/DDoS Attacks Using IPtables Firewall. *International Journal of Computer Science and Telecommunications, 4*(3).

Chen, S., Xu, J., Kalbarczyk, Z., Iyer, R. K., & Whisnant, K. (2004). Modeling and evaluating the security threats of transient errors in firewall software. *Performance Evaluation*, *56*(1), 53-72. http://dx.doi.org/10.1016/j.peva.2003.07.013

Dantu, R., Fahmy, S., Schulzrinne, H., & Cangussu, J. (2009). Issues and challenges in securing VoIP. *Computers & Security, 28*(8), 743-753. http://dx.doi.org/10.1016/j.cose.2009.05.003

Ehlert, S., Geneiatakis, D., & Magedanz, T. (2010). Survey of network security systems to counter SIP-based denial-of-service attacks. *Computers & Security*, *29*(2), 225-243. http://dx.doi.org/10.1016/j.cose.2009.09.004

El-Moussa, F., Mudhar, P., & Jones, A. (2010). Overview of SIP attacks and countermeasures. In *Information Security and Digital Forensics* (pp. 82-91). Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-11530-1_10

Enhancing Network Security in Linux Environment. (2012). Technical Report, *IDE1202*, February 2012

Liu, L. (2010). Security Analysis of Session Initiation Protocol - A Methodology Based on Coloured Petri Nets. *1st International Cyber Resilience Conference*, Australia.

Rani, D. D., Krishna, T. S., Dayanandam, G., & Rao, T. V. (2013). TCP Syn Flood Attack Detection And Prevention. *International Journal of Computer Trends and Technology (IJCTT), 4*(10), 3412.

Sharma, B., & Bajaj, K. (2011). Packet Filtering using IP Tables in Linux. *International Journal of Computer Science Issues (IJCSI)*, *8*(4).

Taluja, S., Verma, P. K., & Dua, R. L. (2012). Network Security Using IP firewalls. *International Journal of Advanced Research in Computer Science and Software Engineering, 2*(8).

Voznak, M., & Safarik, J. (2012). Dos Attacks Targeting SIP Server and Improvements of Robustness. *International Journal of Mathematics and Computers in Simulation, 6*(1).

WM Eddy. (2007). RFC 4987 - TCP SYN Flooding Attacks and Common Mitigations.