VoIP Server Firewalling and Hardening

by Amit Blum

September 2016

1. Introduction

0.1

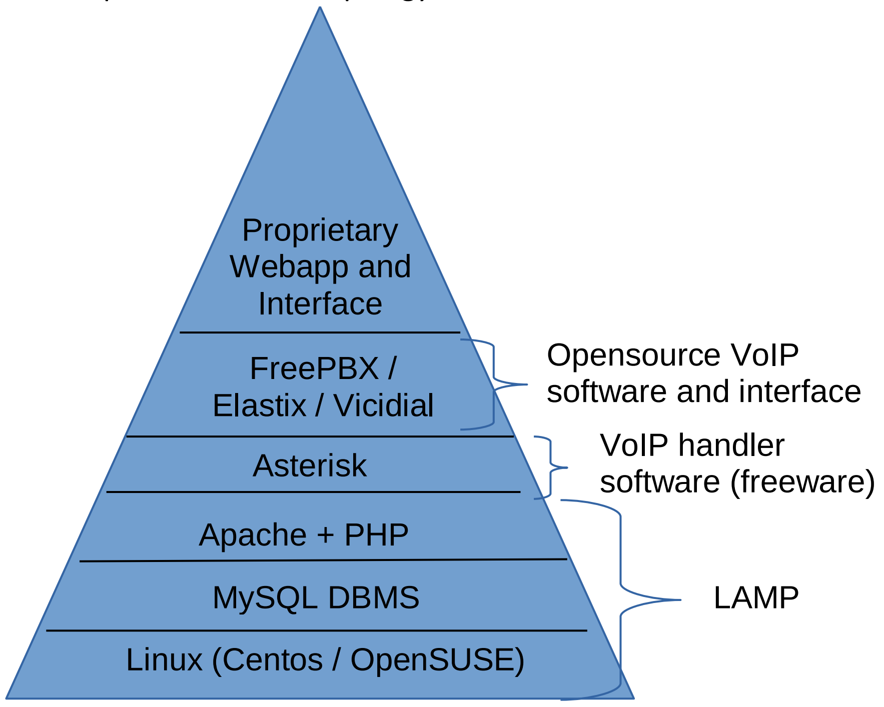
What is VoIP?

Voice over Internet Protocol is a methodology and technologies aimed at a delivery of voice communication and other multimedia sessions over the internet. Servers providing VoIP services are of a wide range – from standard PBX (telephony switching) through automatic dialers, and large scale VoIP switches and authentication servers. Some handle large volumes of traffic and eed to be distributed into clusters and use load balancers.
There are many VoIP protocols – In this document I will stick to a specific type of server configuration and the SIP protocol.

0.2

Server design and topology:

The servers I will attempt to harden are typically the standalone servers that the end user directly connects to, such as PBX and Dialers.
These specific servers's topology is as follows:


Proprietary Webapp and Interface
FreePBX / Elastix / Vicidial
Asterisk
Apache + PHP
MySQL DBMS
Linux (Centos / OpenSUSE)
Opensource VoIP software and interface
VoIP handler software (freeware)
LAMP

The project is to harden this type of server, including such servers that are clustered into specific tasks (SQL server, Web server and Asterisk server, for example).
Such servers can either be local network based (easier to harden) or Cloud based. Cloud based are the "interesting" ones and are becoming rapidly more common and thus that is the type that I will attempt to harden.

0.3

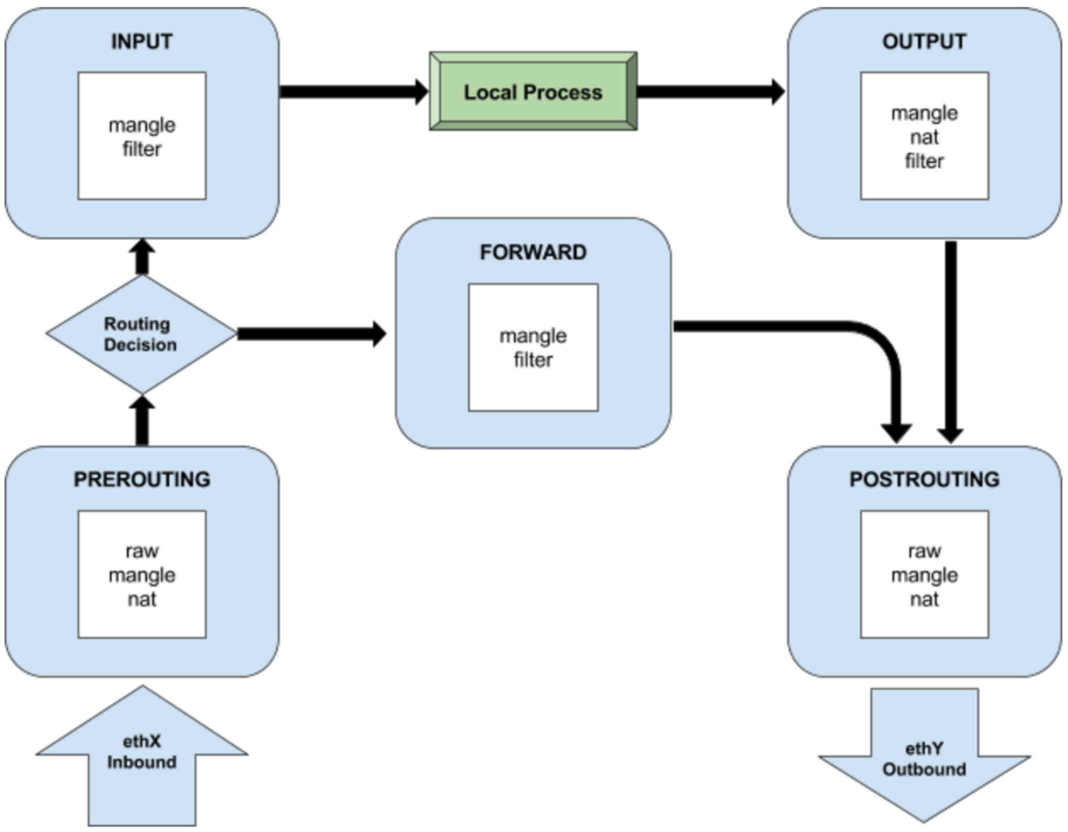
Methodology:

These are Linux stand-alone, cloud based servers. Typically their hardware is only as powerful as is required of them to handle VoIP and some web traffic. As such, it is simplest to configure a firewall on them – Iptables comes preinstalled by default.

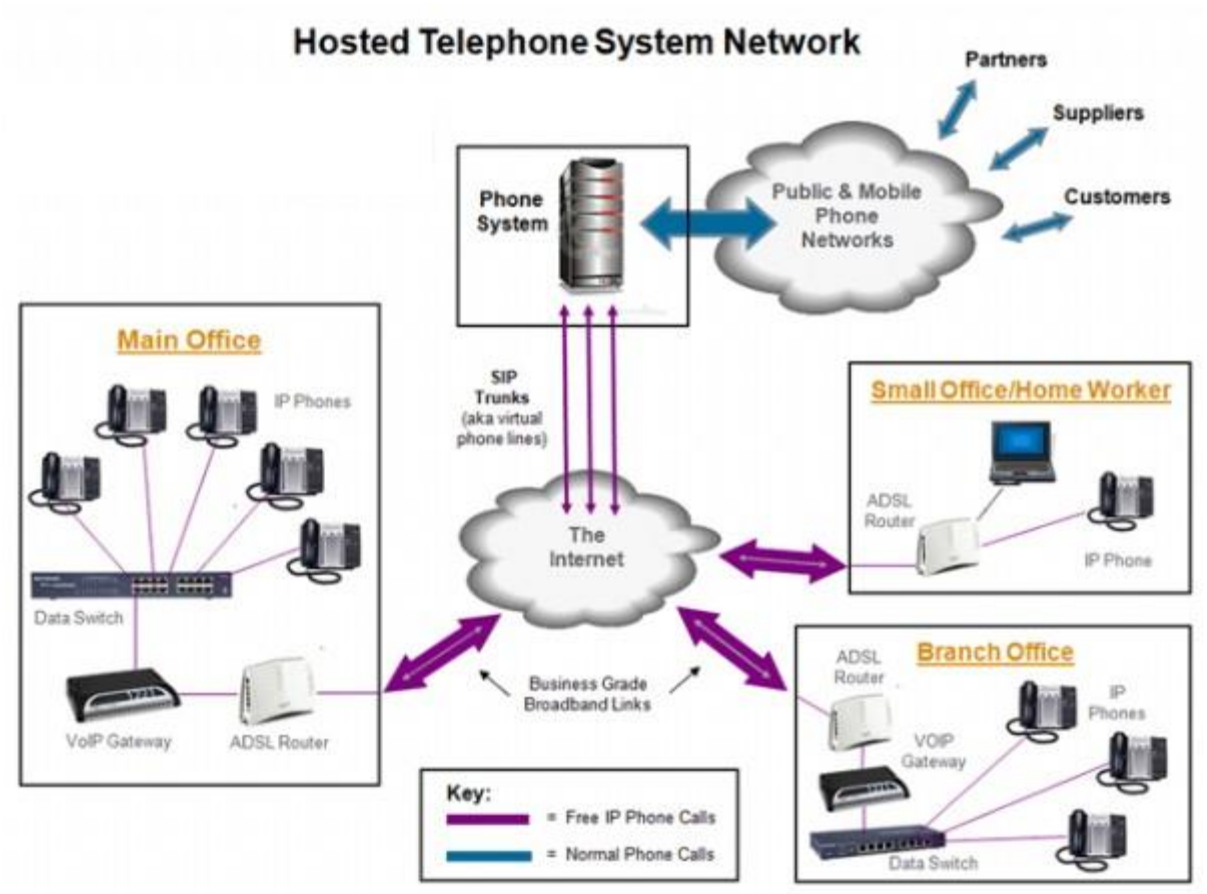
Configuring Iptables will be my main focus but I will also identify additional security weaknesses and suggest possible solutions.
The common (and in my eyes, correct) approach to configuring a firewall is - Only enable the required network traffic, or "Whitelisting".
I will configure the firewall as a whitelist on inbound traffic, but without any limitations on outbound - as is most commonly done. If a complex configuration to outbound is required due to other security breaches / contingencies it may be added later on a per scenario basis.

0.4

Iptables (Default) Chains and Tables Flowchart:



1. Analysis of the servers' inbound traffic and services:



VoIP is a service that in most cases is required to be open to any IP, as VoIP cloud servers are either public, or meant for a specific company's use but from many different IPs in many different locations, some dynamic (for example, by softphones on the users cellphones or from employees' home).

Web traffic should be open, but whether or not open to all will depend on the specific server usage and topology - a normal PBX may only require web access allowed to specific IPs (administrators) whereas Dialers with web interfaces require web access to all the agents and in work-from-home call centers it will include agents working from home etc.

Administrative access is given full access. Administrators are not the PBX users but rather IP's from which SSH access or any other service that will later be added may be required. This includes the actual administrators / techs that log in via ssh (the company that configured the PBX and supports it), and also the IPs of any VoIP gateway servers / switches that are in-charge of forwarding call initiations with the PBX (SIP "INVITE", also called origination), or the ones in-charge of authentication protocols with the carriers.

Services:

SSH – This service needs to be open only to specifically whitelisted addresses.

Port 5060/5744 – SIP registration and command ports (either one could be used – 5744 will be used on more “exposed” servers as to hide the VoIP functionality). Should be open to all, but monitored. Floods and brute force attempts must be blocked.

Port range 10000:20000 – for RTP media stream (VoIP sound stream up and down), should be open.

Port 80 + 443 – web should be open because these servers also provide reporting and management interfaces via WWW. According to the specific server functionality We may only open the to a specific whitelist as to avoid layer 7 attacks such as SQL injections, XSS or other advanced web hacking techniques, but either way these ports will limit the amount of connection per IP. Ammount of connections per IP will be (10 X numberOfAgnets (users) X Whitelisted IP) connections per minute. 10 connections /min per agent is a reasonable amount, and will help prevent synfloods.

ICMP requests - will be open to all, but limited to a reasonable 1 packet per second.

2. Research - how to protect SIP servers?

SIP servers are very commonly targeted by hackers, since the many of the administrators that deploy such servers have little to no knowledge of security. The more popular cloud based VoIP services become, the more hackers we'll see and with more advanced techniques.
Currently if such a server is open to the world with weak passwords to crack it is an easy and very quick source of income to even a low skilled hacker.
Such hackers typically identify the SIP server by finding port 5060 open, they then just attempt brute-force registration and if successful they can route calls to practically anywhere in the world - in the expense of the server owners. They either call for personal use of sell the registration password to a 3rd party for easy money.
According to this paper https://www.usenix.org/system/files/login/articles/login1212_ceron.pdf the steps to a successful SIP server hack are:
a. Scanning - SIP servers are typically identified by port 5060 being open, or by the web service identifying itelf as a known asterisk interface.
b. Enumerating - trying to identify configuration weaknesses and available extensions.
c. Brute Force - cracking the registration credentials.
d. Abuse - the usage of the server's service by calling to external PSTN lines.
By carefully examining these steps **firstly** I find that simply by hiding the SIP server's SIP registration port we can immediately stop a vast majority of hack attempts. Changing it from the standard 5060 to something non-standard (in this case 5744) actually works very well in real life.
Secondly - attempting to stop specific enumeration tools can further stop any hack attempt as the hacker's brute force requirements increase many times fold if he doesn't even know what extensions are available to target.
Thirdly - if the firewall is correctly configured, any brute force hack attempt should be easily thwarted.
By implementing these 3 easy counter-measures you can render your server nearly un-hackable.

3. Firewall Implementation:

I chose to implement the Iptables in a runnable bash file, as coding with certain logic for VoIP allows for future development by translating to other coding languages and creating management interfaces.

First I created a "configurable zone", where the implementor of the firewall for the specific server can change and add parameters and lists for the firewall.
paramaters such as "port", "clientStaticIPsList", "adminStaticIPsList", "inboundGateways", "numOfAgents", "isCluster" and more are all easily configurable and subject to change from one server to another.

a. Flush all previous rules and chains:

```
iptables -t mangle -F
iptables -F
iptables -X
```

b. Allow loopback:

```
iptables -A INPUT -i lo -m comment --comment "Loopback" -j ACCEPT
iptables -A OUTPUT -o lo -m comment --comment "Loopback" -j ACCEPT
```

c. **Stop many automatic general scanning tools** and other malicious packets (using mss parameter):

```
iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
iptables -A INPUT -m conntrack --ctstate NEW -p tcp ! --syn -j REJECT --reject-with tcp-reset
iptables -t mangle -A PREROUTING -p tcp -m multiport --dport 80,443 -m conntrack --ctstate NEW -m tcpmss ! --mss 500:65535 -m
limit --limit 6/min -j LOG --log-prefix "Prevent Synflood " --log-level 6
iptables -t mangle -A PREROUTING -p tcp -m multiport --dport 80,443 -m conntrack --ctstate NEW -m tcpmss ! --mss 500:65535 -m
comment --comment "Prevent Synflood" -j DROP
```

d. Allow ICMP:

```
iptables -A INPUT -p icmp --icmp-type 8 -m limit --limit 1/s --limit-burst 3 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 8 -m limit --limit 1/s --limit-burst 3 -j ACCEPT
```

e. Prevent Spoofs ($ip is a configurable parameter for any spoof addresses):

```
iptables -A INPUT -s $ip -m comment --comment "Spoofed IP" -j ANTISPOOFING
iptables -A ANTISPOOFING -j LOG -m limit --limit 6/min --log-prefix "Spoofed IP detected: "
iptables -A ANTISPOOFING -j DROP
```

f. Allow full access to admins:

```
iptables -N FULLACCESS
iptables -A INPUT -s $ip -m comment --comment "admin Office IP" -j FULLACCESS
iptables -A FULLACCESS -j ACCEPT
```

g. Allow web access to whitelist or to all:

```
iptables -N WEBACCESS
if [ "$openPort80ForAll" = true ]; then
iptables -A INPUT -p tcp --syn -m multiport --dports 80,443 -m hashlimit --hashlimit $((10*numOfAgents))/min --hashlimit-burst
$((10*numOfAgents)) --hashlimit-mode srcip --hashlimit-name synthrottle -m comment --comment "Allow web access" -j
WEBACCESS
else
iptables -A INPUT -s $ip -p tcp --syn -m multiport --dports 80,443 -m hashlimit --$((10*numOfAgents))/min --hashlimit-burst
$((10*numOfAgents)) --hashlimit-mode srcip --hashlimit-name synthrottle -m comment --comment "Client IP web access" -j
WEBACCESS
fi
iptables -A WEBACCESS -j ACCEPT
```

h. **Stop many SIP DOS and enumeration tools**:

```
iptables -N SIPDOSCHECK
iptables -N SIPDOSDROP
iptables -A INPUT -m comment --comment "Drop SIP DoS attacks" -j SIPDOSCHECK
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "sundayddr" --algo bm --to 65535 -m comment --comment
"Deny sundayddr" -j SIPDOSDROP
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "sipsak" --algo bm --to 65535 -m comment --comment "Deny
sipsak" -j SIPDOSDROP
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "sipvicious" --algo bm --to 65535 -m comment --comment
"Deny sipvicious" -j SIPDOSDROP
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "friendly-scanner" --algo bm --to 65535 -m comment --
comment "Deny friendly-scanner" -j SIPDOSDROP
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "iWar" --algo bm --to 65535 -m comment --comment "Deny
iWar" -j SIPDOSDROP
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "sip-scan" --algo bm --to 65535 -m comment --comment
"Deny sip-scan" -j SIPDOSDROP
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "VaxSIPUserAgent" --algo bm --to 65535 -m comment --
comment "Deny VaxSIP" -j SIPDOSDROP
iptables -A SIPDOSCHECK -p udp -m udp --dport $port -m string --string "sipcli" --algo bm --to 65535 -m comment --comment "Deny
SipCLI" -j SIPDOSDROP
iptables -A SIPDOSDROP -m limit --limit 6/min -j LOG --log-prefix "firewall-sipdos: " --log-level 6
iptables -A SIPDOSDROP -j DROP
```

i. Allow SIP traffic if it is not flooding:

```
iptables -N SIPTRAFFIC
iptables -A INPUT -p udp -m udp --dport $port -m comment --comment "Check legitimate SIP traffic" -j SIPTRAFFIC
iptables -A SIPTRAFFIC -p udp -m udp --dport $port -m string --string "REGISTER sip:" --algo bm -m recent --update --seconds 60 --hitcount $((6*numOfAgents)) --name VOIPREG --rsource -m limit --limit 6/min -j LOG --log-prefix "SIP Reg BForce Detected: " --log-level 6
iptables -A SIPTRAFFIC -p udp -m udp --dport $port -m string --string "REGISTER sip:" --algo bm -m recent --update --seconds 60 --hitcount $((6*numOfAgents)) --name VOIPREG --rsource -j DROP
iptables -A SIPTRAFFIC -p udp -m udp --dport $port -m string --string "REGISTER sip:" --algo bm -m recent --set --name VOIPREG --rsource
iptables -A SIPTRAFFIC -p udp -m udp --dport $port -m string --string "INVITE sip:" --algo bm -m recent --update --seconds 60 --hitcount $((6*numOfAgents)) --name VOIPINV --rsource -m limit --limit 6/min -j LOG --log-prefix "SIP Inv flood Detected: " --log-level 6
iptables -A SIPTRAFFIC -p udp -m udp --dport $port -m string --string "INVITE sip:" --algo bm -m recent --update --seconds 60 --hitcount $((6*numOfAgents)) --name VOIPINV --rsource -j DROP
iptables -A SIPTRAFFIC -p udp -m udp --dport $port -m string --string "INVITE sip:" --algo bm -m recent --set --name VOIPINV --rsource
iptables -A SIPTRAFFIC -p udp -m hashlimit --hashlimit 6/sec --hashlimit-mode srcip,dstport --hashlimit-name tunnel_limit -m udp --dport $port -j ACCEPT
iptables -A SIPTRAFFIC -p udp -m udp --dport $port -j DROP
```

j. Allow RTP stream:

```
iptables -A INPUT -p udp -m udp --dport 10000:20000 -m comment --comment "Allow RTP" -j ACCEPT
```

k. Make the firewall have stateful inspection:

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

l. Cleanup:

```
iptables -N CLEANUP
iptables -A INPUT -j CLEANUP
iptables -A CLEANUP -p tcp --dport 22 -m limit --limit 6/min -j LOG --log-prefix "Unauth SSH detected: " --log-level 6
iptables -A CLEANUP -p udp -j REJECT --reject-with icmp-port-unreachable
iptables -A CLEANUP -p tcp -j REJECT --reject-with tcp-reset
iptables -A CLEANUP -j DROP
```

4. What else can we do? (Conclusion)

iptables configuration-wise the SIP server is now very secure. However there are a few other security breaches we must take care of:

i. **ipv6 disable:** We are still vulnerable to a very devastating (and easy to reproduce) ipv6 Router Advertisements Flood attack. If we don't have any devices/servers in our networks that use ipv6 - the easiest way to protect ourselves from this attack (and any other unknown / untested attacks in this new protocol) - we just need to disable it in ***/etc/sysctl.conf***.

ii. ***Persistent iptables:*** Make sure to run the iptables's sh bash script I have created @ every reboot either by using crontab or rc.local (depending on the linux distribution). I have also made sure the script saves an iptables-restore file every time it runs for future reference or it someone chooses to use persistent iptables via sysctl etc.

iii. ***add another layer of spoof detection*** that is built in Linux. In /etc/sysctl:
net.ipv4.conf.all.rp_filter=1
net.ipv4.conf.all.log_martians=1
net.ipv4.conf.default.log_martians=1

iv. ***Protect the web service in layer 7***: make sure to update Apache, Mysql, and Linux kernel. Make sure to update the web application version to have it as secure as possible to XSS or sql injections. Install security mods such as mod_security and mod_evasive. Hide file inclusion by using -indexes option in httpd.conf. Stop slowlorris by using and configuring Apache qos_mod.

That's it !

by Amit Blum
September 2016