Common Vulnerabilities in VoIP Networks

A demonstration of threats caused by default network configuration and the knowledge required for remediation measures

Jane Kocan-Payne

CMP508: Information and Network Security

MSc Cyber Security and Ethical Hacking

2023/24

Note that Information contained in this document is for educational purposes.

.

Contents

.

1 INTRODUCTION

Voice over IP (VoIP) networks can be found in corporations of all sizes across the globe (Keromytis, 2010), and their popularity continues to grow, as can be demonstrated in Figure 1:

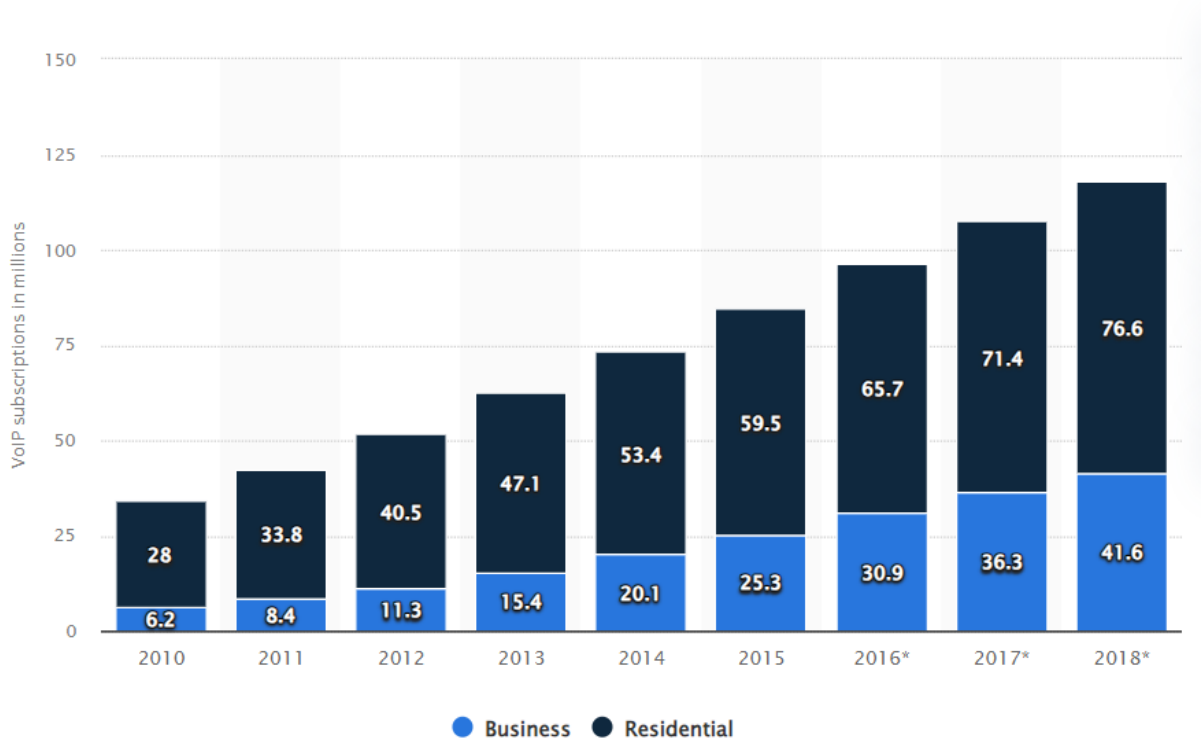

Figure 1: VoIP subscriptions in the United States, 2010-18 (Statista, 2015).

Despite this popularity, VoIP networks are well-known for being vulnerable to a large and varied number of attacks (McClure, et al., 1999), with the highest number of vulnerabilities creating denial-of-service opportunities (Keromytis, 2010). The majority of vulnerabilities present in VoIP systems are caused by misconfigurations during the implementation of the network (Keromytis, 2010), as shown in Figure 2:

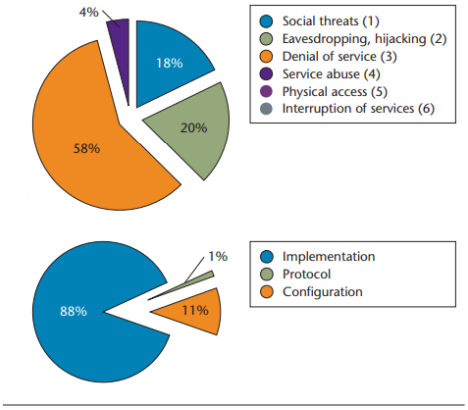

Figure 2: Vulnerabilities in VoIP by type and cause (Keromytis, 2010).

Furthermore, the diversity of use cases for VoIP (Vitel Global, 2022) necessitates that PBX vendors release their products in a default state with little-to-no security customisations applied (Nuno, et al., 2020), requiring comprehensive knowledge of both networking and cybersecurity (Androulidakis, 2016) to ensure their appropriate configuration.

The aim of this report is to explore the threats to VoIP networks and provide recommendations for securing network machines against those threats using secure design practices. An out-of-the-box implementation of a PBX platform and two client machines will be used to investigate and demonstrate the vulnerabilities present and develop suggestions for their remediation.

2 METHODOLOGY

2.1 OVERVIEW OF PROCEDURE

Corporate VoIP networks can scale into very complex architectures with multiple components, as can be seen in Figure 3:

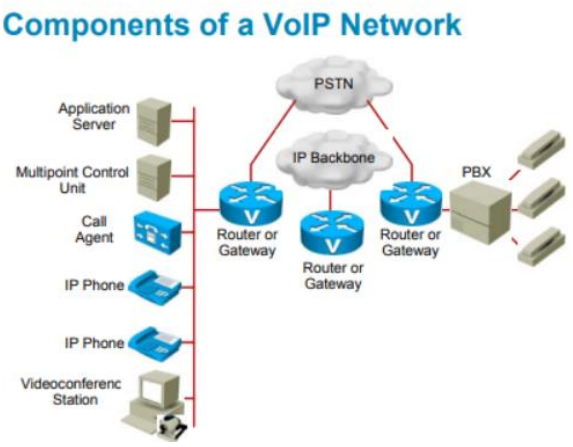

Figure 3: Typical components of a VoIP network (Router-switch, 2020).

For the purposes of this report, a VoIP network was implemented as per the diagram in Figure 4:

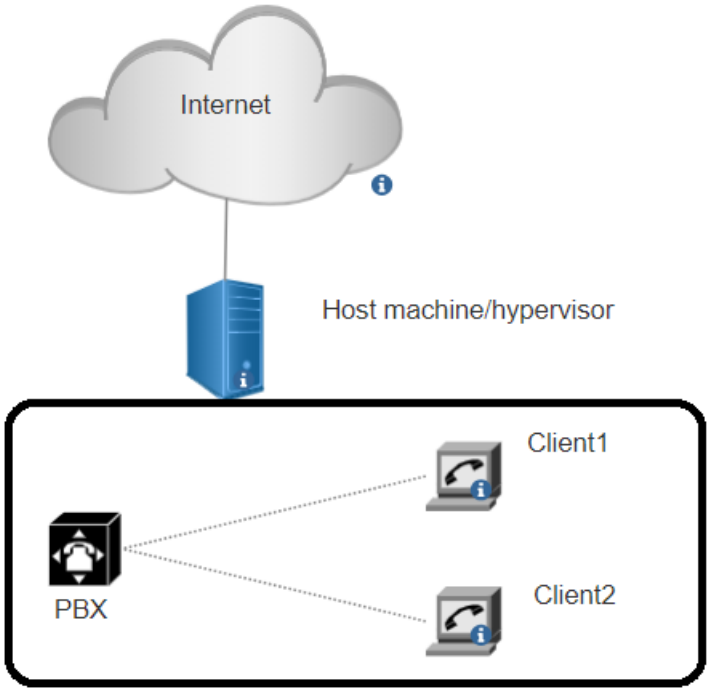

Figure 4: Network diagram of the exemplar VoIP network.

Several platforms were utilised for the implementation of the VoIP network. These platforms and their purposes can be seen in Appendix A.

In instances where the author used a command line containing a variable value, these are indicated with the use of chevrons (<>) within the command line, for example:

```
tar -xf <fileName>
```

2.2 PRE-REQUISITES

The procedure followed for the installation of any pre-requisites can be found in Appendix B.

2.3 ASTERISK INSTALLATION

Upon the completion of the operating system installation, the author downloaded the most recent Asterisk release from the following URL:

https://www.asterisk.org/downloads/

The downloaded file was extracted with the command below. The `-x` and `f` options instruct `tar` to extract the file contents and passes a file name to the tool, respectively.

```
tar -xf <downloadedFileName>
```

Once the file content had been extracted, a sequence of installation commands was executed. These can be found, along with their condensed terminal outputs, in Appendix C.

Finally, the Asterisk instance was started with the following command:

```
sudo asterisk
```

2.4 ASTERISK CONFIGURATION

Once the installation of Asterisk had completed, the author implemented some basic configuration to enable the two client machines to make calls to each other. Two configuration files that had been created during the basic PBX set-up were backed up using the following commands, executed from within the directory that held the Asterisk configuration files (/etc/asterisk):

```
sudo mv extensions.conf extensions.conf.bak

sudo mv pjsip.conf pjsip.conf.bak
```

To complete the Asterisk installation, the author created a new extensions.conf file with the following configuration items:

- Definition of a dial plan.
- Specification of two extension numbers for allocation to the client machines.
- Action to take when a call was placed to the extension numbers.

The contents of the newly created extensions.conf file can be found in Figure 5:

```
[from-internal]
exten=>6001,1,Dial(PJSIP/6001,20)
exten=>6002,1,Dial(PJSIP/6002,20)
```

Figure 5: Contents of the new extensions.conf file.

Finally, the author created a new pjsip.conf file containing the following configuration items:

- Definition of transport protocol.
- Call handling rules for both of the client extension numbers.
- Authentication information for both of the client extension user accounts.

The contents of the newly created pjsip.conf file can be found in Appendix D.

2.5 Client SIP Phone Client Installation

The SIP client was installed on Client1 with the following command:

```
sudo apt update && sudo apt install linphone
```

Once installed, the Linphone client was opened. Registration to the Asterisk instance was done by using the "Use a SIP account" button in the client's welcome page:

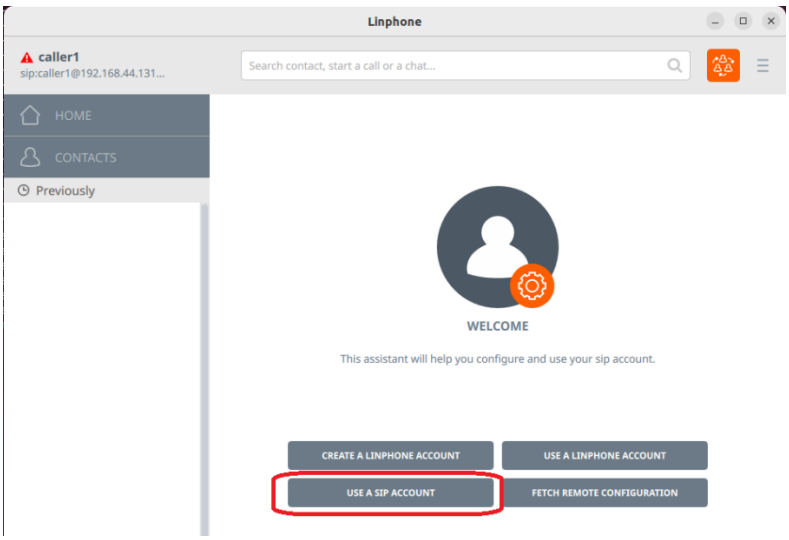

Figure 6: Signing into Linphone for the first time.

The username and password (as stipulated in the pjsip.conf file) were entered, and the IP address of the PBX machine used in the “SIP Domain” field as shown in Figure 7:

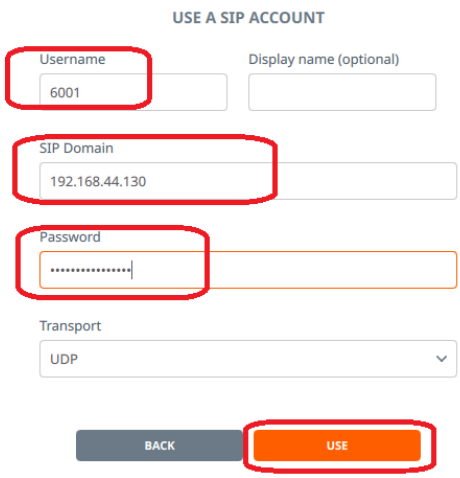

Figure 7: Setting up the SIP account in the Linphone application.

This process was repeated for Client2 and a test call made between the two nodes to test connectivity, which was successful.

3 RESULTS

According to an AI and community-powered article for LinkedIn (n.d.), the most common mistakes made upon implementing a VoIP system fall into four technical categories:

1. Lack of encryption.
2. Poor password management.
3. Inadequate firewall configuration.
4. Insufficient network monitoring.

3.1 LACK OF ENCRYPTION

When a call is initiated between two clients in a VoIP network, the session is opened and closed with a series of signaling messages between the two clients, via the PBX controller. These signaling messages are transported using the SIP protocol, shown as steps 1-4 and 6-7 in Figure 8. The voice call itself travels through the network using the RTP protocol. The flow of communications between two SIP clients can be seen in Figure 8:

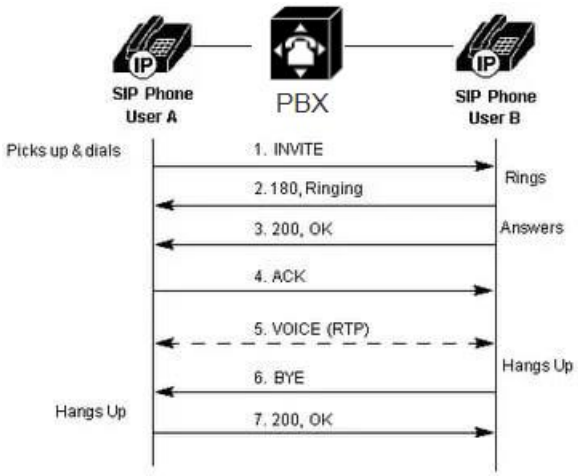

Figure 8: Flow of communications between SIP clients (Joy, 2020).

In a default implementation of Asterisk PBX, neither the SIP nor RTP protocols are encrypted, rendering any communications vulnerable to man-in-the-middle attacks (Androulidakis, 2016). As the author did not take any steps during the creation of the exemplar VoIP network to enable encryption, the test calls made following the installation of the network were not encrypted.

According to Pisarčík (2012), implementation for encryption of traffic is possible by following three basic steps:

1. Obtain/generate a private SSL key in .pem format.
2. Set the SSL/TLS options within the configuration file. In the case of the exemplar network this will be the pjsip.conf file.
3. Configure the clients to use SSL/TLS for their communications.

The specific process followed by the author for implementing encryption using a self-signed certificate for the pjsip channel in Asterisk can be found in Appendix E, and includes the steps taken to enable TLS encryption for both SIP and RTP (utilising SRTP) communications. The impact upon the communications is evidenced in the Wireshark packet captures in Figures 9 and 10, showing an unencrypted and encrypted call session initiation, respectively.

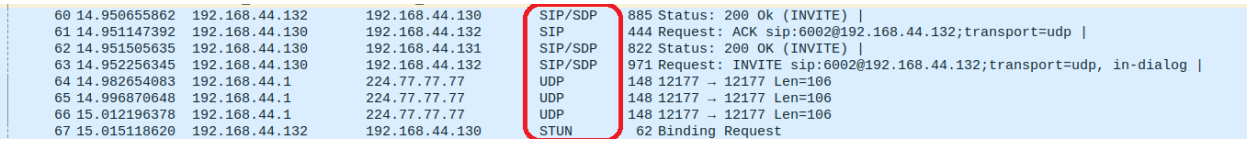

Figure 9: Call initiation using SIP (unencrypted).

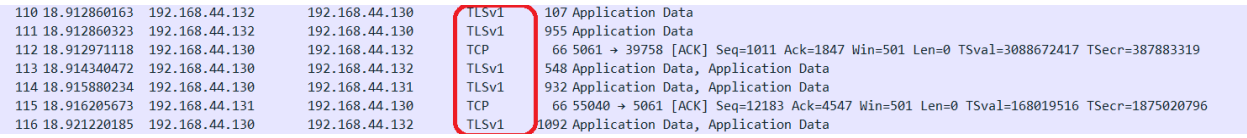

Figure 10: Call initiation using TLS (encrypted).

3.2 Poor Password Management

When using the Asterisk set-up option to create a basic PBX configuration, passwords created for endpoints are stored in plaintext in the pjsip.conf file. Furthermore, the passwords applied in the set-up process are very weak, as can be seen in Figure 11:

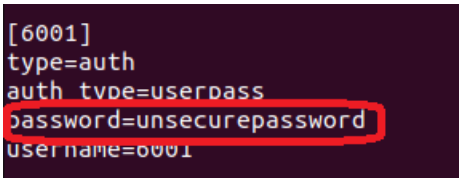

Figure 11: Plaintext password in pjsip.conf file.

It is recommended that passwords of at least sixteen randomly generated characters are used, which contain a mix of letters, digits and special characters (Pisarcik, 2012). It is further recommended that the passwords are stored in the pjsip.conf file in a pre-hashed format. The specific process followed by the author for implementing password hashing for endpoint users is detailed in Appendix F. The impact upon the discoverability of the password is evidenced by the interrogation of the contents of the pjsip.conf file after the completion of this process, as shown in Figure 12:

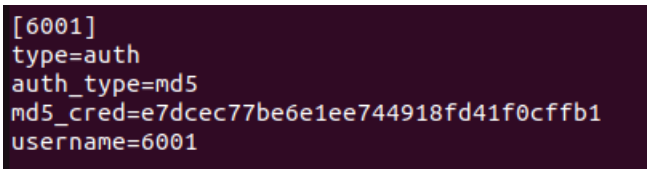

Figure 12: Hashed password in pjsip.conf file.

3.3 Inadequate Firewall Configuration

As with any networking system, ports and the services running on them within a VoIP network should be secured so that only authorised persons can gain access to them (Whitman & Mattord, 2019). This can be achieved through the use of a firewall and its associated access control lists (ACLs), provided that the firewall is configured correctly.

Asterisk operates independently of a firewall, and as such no firewall configuration took place during the network installation, which was evidenced when querying the PBX controller machine for the status of the default firewall: Uncomplicated Firewall (UFW):

```
pbxadmin@PBX:~$ sudo ufw status
Status: inactive
```

Figure 13: Checking the status of the Uncomplicated Firewall on the PBX controller.

Furthermore, there was a listening port on the PBX controller operating on behalf of the Asterisk service, as shown in Figure 14:

```
pbxadmin@PBX:~$ sudo netstat -tulpn | grep LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      574/systemd-resolve
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      901/cupsd
tcp        0      0 0.0.0.0:5061            0.0.0.0:*               LISTEN      2343/asterisk
tcp6       0      0 ::1:631                 :::*                    LISTEN      901/cupsd
```

Figure 14: Checking the status of listening ports on the PBX controller.

The process followed for enabling the firewall and implementing a basic configuration is detailed in Appendix G. The author applied a single rule to the firewall, explicitly allowing TCP traffic from within the internal network (source IP address beginning with 192.168.44) to port 5601. As firewalls also include an implicit "DENY" rule at the end of their rulesets, this has the effect of dropping all other incoming traffic to the PBX controller. The new firewall rule set is shown in Figure 15:

```
pbxadmin@PBX:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
5601/tcp                   ALLOW       192.168.44.0/24
```

Figure 15: Checking the newly configured firewall status.

3.4 Insufficient Network Monitoring

According to Keromytis (2010), over half of the vulnerabilities seen in VoIP networks provide opportunities for attackers to succeed in denial of service (DoS) attacks. These can be prevented by the use of network monitoring using an Intrusion Detection System (IDS) or Intrusion Prevention System (IPS). This course of action is recommended by Pisarčík (2012), who further suggests that the open-source Snort tool can be used to fulfill this purpose.

As with the firewall implementation investigated, Asterisk operates independently of network monitoring tools, and as such no IDS or IPS configuration took place during the VoIP network installation.

Snort can be implemented as either an IDS or an IPS solution. However, the complexity of its installation and configuration stipulate that this task falls outside the scope of this report. A default installation of Snort incorporates sample rule files, which includes a rule set for SIP traffic. Though now deprecated, the file provides background for the customisation of appropriate rules for an implementation with a VoIP network. The contents of this rule set can be found in Appendix H.

4 CONCLUSION

This report set out to explore common vulnerabilities found in VoIP networks and the threats they pose using a VoIP network installed with a default configuration. Whilst the author has been able to demonstrate the vast scope of both networking and security knowledge required to sufficiently secure a VoIP network, the investigation has been limited by the scope of the exemplar network used. With the understanding that the complexity of a network increases exponentially with its scale (Mcquerry, 2010), the author would propose further work be conducted to demonstrate the inherent vulnerabilities that a default configuration encompasses:

- Further investigation into the implementation of an IDS/IPS solution.
- Use of allow/block lists or configuration of the fail2ban tool to restrict access to the PBX controller.
- Network segmentation using VLANs.
- Hardening of VoIP infrastructure on a wider scale using penetration testing and vulnerability assessment tools.

5 REFERENCES

Androulidakis, I., 2016. *VoIP and PBX Security and Forensics A Practical Approach.* 2 ed. s.l.:Springer.

Joy, M., 2020. *SIP Call Flow Explained.* [Online]
Available at: https://www.onsip.com/voip-resources/voip-solutions/sip-call-flow-explained
[Accessed 16 December 2023].

Keromytis, A., 2010. Voice-over-IP Security: Research and Practice. *IEEE Security & Privacy,* Volume 8, pp. 76-78.

LinkedIn, n.d. *What are the most common mistakes to avoid when securing a cloud-based VOIP system?.* [Online]
Available at: https://www.linkedin.com/advice/3/what-most-common-mistakes-avoid-when-securing-kdprf
[Accessed 16 December 2023].

McClure, S., Scambray, J. & Kurtz, G., 1999. *Hacking Exposed 7: Network Security Secrets and Solutions.* 7th ed. s.l.:McGraw-Hill.

Mcquerry, S., 2010. *Interconnecting Cisco network devices. Part 2 (ICND2), Authorized self-study guide.* 6 ed. Indianapolis: Cisco Press.

Nuno, P. et al., 2020. A Diagnosis and Hardening Platform for an Asterisk VoIP PBX. *Security and Communication Networks,* Volume 2020.

Pisarcik, B., 2012. *OSSLab.* [Online]
Available at: http://www.osslab.tw:8880/@api/deki/files/3354/=asterisk-security-hardening-1.0.pdf
[Accessed 16 December 2023].

Router-switch, 2020. *3 Solutions for CISCO VoIP System.* [Online]
Available at: https://blog.router-switch.com/2020/11/3-solutions-for-cisco-voip-system/
[Accessed 14 December 2023].

Statista, 2015. *VoIP residential and business telephone lines in the United States from 2010 to 2018 (in millions).* [Online]
Available at: https://www.statista.com/statistics/615387/voip-telephone-lines-in-the-us/
[Accessed 14 December 2023].

Vitel Global, 2022. *Use Cases of Cloud VoIP System in Various Industries.* [Online]
Available at: https://www.vitelglobal.com/blog/use-cases-of-cloud-voip-system-in-various-industries/
[Accessed 16 December 2023].

Whitman, M. E. & Mattord, H. J., 2019. *Management of Information Security.* 6 ed. Boston: Cengage.

6 BIBLIOGRAPHY

Asterisk, n.d. *Installing Asterisk from Source.* [Online]
Available at: https://docs.asterisk.org/Getting-Started/Installing-Asterisk/Installing-Asterisk-From-Source/
[Accessed 18 November 2023].

ClusterPBX, 2019. *How-to Configure Linphone for Encrypted Communication.* [Online]
Available at: https://docs.clusterpbx.com/clusterpbx/how-to-articles/how-to-configure-linphone-for-encrypted-communication/
[Accessed 16 December 2023].

HotKey, n.d. *Asterisk security – hash your endpoint's password with md5 (md5secret).* [Online]
Available at: https://hotkey404.com/asterisk-security-hash-your-endpoints-password-with-md5-md5secret/
[Accessed 16 December 2023].

HotKey, n.d. *Asterisk security: using self-signed SSL Certificate for TLS registration.* [Online]
Available at: https://hotkey404.com/asterisk-security-using-self-signed-ssl-certificate-for-tls-registration/
[Accessed 16 December 2023].

Ubuntu, 2023. *Security - Firewall.* [Online]
Available at: https://ubuntu.com/server/docs/security-firewall
[Accessed 16 December 2023].

APPENDICES

APPENDIX A – PLATFORMS, SOFTWARE AND TOOLS USED

Platform name	Version number	Purpose
Windows 10 (virtual machine)	23H2 build 22631-2715	Hosting the VMware Workstation Pro virtualisation software.
VMware Workstation Pro	17.5.0 build 18363.418	Hosting the virtual machines for the VoIP network.
Ubuntu (virtual machines)	6.2.0-36-generic	One machine for hosting the PBX controller, two more to function as endpoints hosting SIP phones for calling one another.
tar	1.34	Native Linux file archiving tool. Used to extract the contents of the downloaded installation file for Asterisk.
make	4.3	Native Linux tool for compiling executable programs.
mv	8.32	Native Linux tool for moving/renaming files.
Asterisk	20.5.0	Software to host the PBX controller for call handling functions.
apt	2.4.11	Native Linux package management tool. Used to install the Linphone client software.
Linphone	Qt5.15.3	Desktop client on client endpoints for making/receiving calls.
Wireshark	3.6.2	Network packet capture for demonstrating encryption functionality.
md5sum	8.32	Native Linux tool for providing an MD5 hash of text and files.
ufw	0.36.1	Default Ubuntu firewall.
netstat	2.10-alpha	Native Linux tool for displaying system information.

APPENDIX B – PRE-REQUISITES SET-UP PROCEDURE

VMware Workstation Pro installation

VMware Workstation Pro is a paid-for software application, and as such its installation is out of scope for this report. The installer media can be downloaded from the VMware website (account creation is required):

https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

The installation was a simple process, which required the tester to select all the defaults offered by the installer. The installer was run as an administrator within the Windows machine.

Virtual machine set-up

The virtual machines used to host the VoIP network were built using the Ubuntu .iso file downloaded from the following URL:

https://ubuntu.com/download/desktop

The set-up of the operating system is out of scope for this report, which focuses on the set-up of the VoIP network components specifically. To create a new machine in VMware, the "Create a New Machine" button was selected from the home page of VMware Workstation:

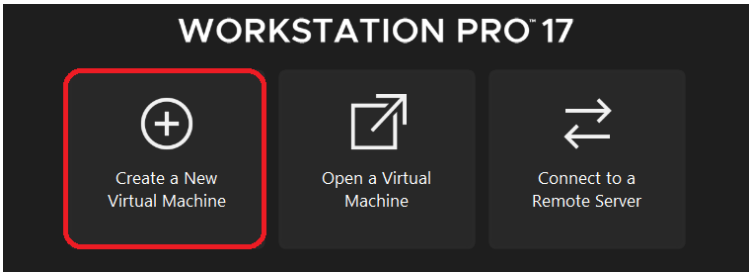

Figure 16: Creating a new virtual machine in VMware Workstation.

The installation was specified as "Typical":

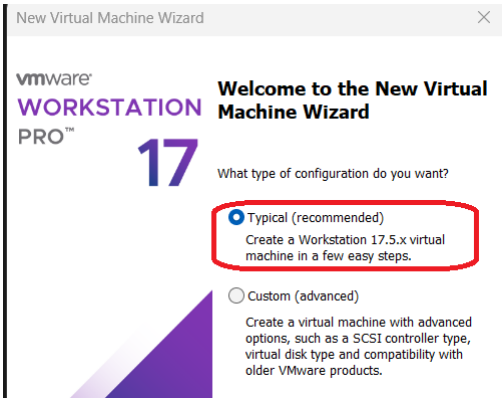

Figure 17: Specifying the installation type in VMware Workstation.

The downloaded .iso was selected as the source for the installer disc file:

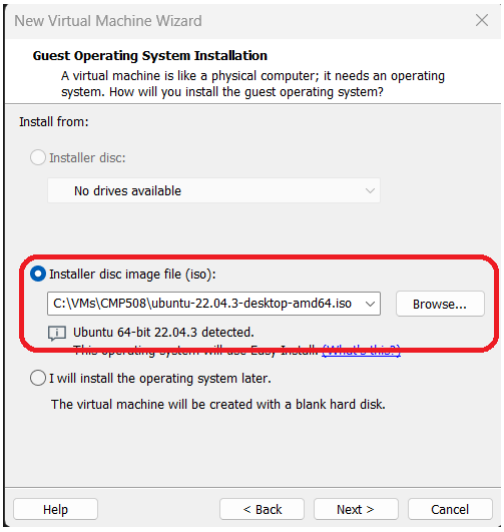

Figure 18: Selecting the .iso file for the installer disc file in VMware Workstation.

A computer name, username and password were set:

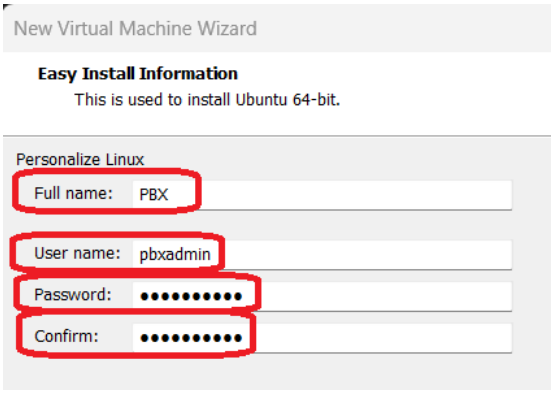

Figure 19: Setting the computer name, username and password in VMware Workstation.

A virtual machine and save location were specified:

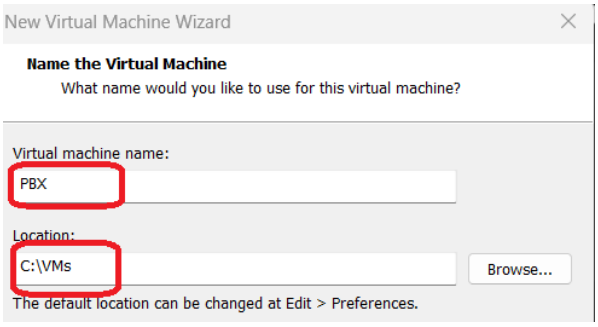

Figure 20: Specifying the virtual machine and save location in VMware Workstation.

The defaults were selected in each of the following screens in the New Virtual Machine wizard. Upon completion of the wizard, the virtual machine was powered on by VMware Workstation and the operating system was installed as per the documentation from the vendor and the prompts on the screen. This process was repeated twice more to produce a total of three virtual machines.

APPENDIX C – ASTERISK INSTALLATION COMMANDS WITH SHORTENED TERMINAL OUTPUTS

```
pbxadmin@PBX:~/asterisk-20.5.0$ cd contrib/scripts


pbxadmin@PBX:~/asterisk-20.5.0/contrib/scripts$ sudo ./install_prereq
install

[sudo] password for pbxadmin:

Hit http://gb.archive.ubuntu.com/ubuntu jammy InRelease

<shortened>

##############################################

## install completed successfully

##############################################


pbxadmin@PBX:~/asterisk-20.5.0/contrib/scripts$ cd ../..


pbxadmin@PBX:~/asterisk-20.5.0$ sudo ./configure

checking build system type... x86_64-pc-linux-gnu

checking host system type... x86_64-pc-linux-gnu

checking for gcc... gcc

checking whether the C compiler works... yes

<shortened>

configure: Package configured for:

configure: OS type  : linux-gnu

configure: Host CPU : x86_64

configure: build-cpu:vendor:os: x86_64 : pc : linux-gnu :

configure: host-cpu:vendor:os: x86_64 : pc : linux-gnu :


bxadmin@PBX:~/asterisk-20.5.0$ sudo make

CC="cc" CXX="g++" LD="" AR="" RANLIB="" CFLAGS="" LDFLAGS="" make -C
menuselect CONFIGURE_SILENT="--silent" makeopts

make[1]: Entering directory '/home/pbxadmin/asterisk-
20.5.0/menuselect'
```

```
make[1]: 'makeopts' is up to date.

make[1]: Leaving directory '/home/pbxadmin/asterisk-20.5.0/menuselect'

CC="cc" CXX="g++" LD="" AR="" RANLIB="" CFLAGS="" LDFLAGS="" make -C
menuselect CONFIGURE_SILENT="--silent" menuselect

make[1]: Entering directory '/home/pbxadmin/asterisk-
20.5.0/menuselect'

gcc -g -D_GNU_SOURCE -Wall -Wno-deprecated-declarations -DHAVE_NCURSES
-I/usr/include/libxml2   -c -o menuselect.o menuselect.c

gcc -g -D_GNU_SOURCE -Wall -Wno-deprecated-declarations -DHAVE_NCURSES
-c -o strcompat.o strcompat.c

gcc -g -D_GNU_SOURCE -Wall -Wno-deprecated-declarations -DHAVE_NCURSES
-c -o menuselect_stub.o menuselect_stub.c

gcc  -o menuselect menuselect.o strcompat.o menuselect_stub.o -lxml2

make[1]: Leaving directory '/home/pbxadmin/asterisk-20.5.0/menuselect'

Generating input for menuselect ...

<shortened>

Building Documentation For: channels pbx apps codecs formats cdr cel
bridges funcs tests main res addons

 +--------- Asterisk Build Complete ---------+

 + Asterisk has successfully been built, and +

 + can be installed by running:              +

 +                                           +

 +                make install               +

 +-------------------------------------------+


pbxadmin@PBX:~/asterisk-20.5.0$ sudo make install

CC="cc" CXX="g++" LD="" AR="" RANLIB="" CFLAGS="" LDFLAGS="" make -C
menuselect CONFIGURE_SILENT="--silent" makeopts

<shortened>

+---- Asterisk Installation Complete -------+

 +                                           +

 +    YOU MUST READ THE SECURITY DOCUMENT    +

 +                                           +
```

```
 + Asterisk has successfully been installed. +
 + If you would like to install the sample   +
 + configuration files (overwriting any      +
 + existing config files), run:              +
 +                                           +
 + For generic reference documentation:      +
 +    make samples                           +
 +                                           +
 + For a sample basic PBX:                   +
 +    make basic-pbx                         +
 +                                           +
 +                                           +
 +-----------------  or ---------------------+
 +                                           +
 + You can go ahead and install the asterisk +
 + program documentation now or later run:   +
 +                                           +
 +               make progdocs               +
 +                                           +
 + **Note** This requires that you have      +
 + doxygen installed on your local system    +
 +-------------------------------------------+
```

```
pbxadmin@PBX:~/asterisk-20.5.0$ sudo make basic-pbx
Installing basic-pbx config files...
<shortened>
Installing file configs/basic-pbx/voicemail.conf
Updating asterisk.conf
```

APPENDIX D - CONTENTS OF THE NEW PJSIP.CONF FILE

```
[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0

[6001]
type=endpoint
context=from-internal
disallow=all
allow=ulaw
auth=6001
aors=6001

[6001]
type=auth
auth_type=userpass
password=unsecurepassword
username=6001

[6001]
type=aor
max_contacts=5

[6002]
type=endpoint
context=from-internal
disallow=all
allow=ulaw
auth=6002
aors=6002

[6002]
type=auth
auth_type=userpass
password=unsecurepassword
username=6002

[6002]
type=aor
max_contacts=5
```

APPENDIX E – IMPLEMENTING SSL/TLS USING THE PJSIP CHANNEL IN ASTERISK

The command below was executed from the contrib/scripts subdirectory of the directory containing the extracted Asterisk installation file. The description of the options is shown in Table 1.

```
contrib/scripts/ast_tls_cert -C <pbxIpAddress> -O <companyName> -d
/etc/asterisk/keys
```

Table 1: Explanation of command line switches for setting a self-signed certificate for Asterisk.

Option	Purpose
-C	Sets the host. Can be defined as the IP address or the DNS name of the PBX controller.
-O	Sets the organisation name for the certificate.
-d	Instructs the script where to save the resultant certificate.

During the creation process, a pass phrase was entered for the certificates. The terminal output from the script can be seen in figure 21:

```
No config file specified, creating '/var/lib/asterisk/keys/tmp.cfg'
You can use this config file to create additional certs without
re-entering the information for the fields in the certificate
Creating CA key /var/lib/asterisk/keys/ca.key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
Creating CA certificate /var/lib/asterisk/keys/ca.crt
Enter pass phrase for /var/lib/asterisk/keys/ca.key:
Creating certificate /var/lib/asterisk/keys/asterisk.key
Creating signing request /var/lib/asterisk/keys/asterisk.csr
Creating certificate /var/lib/asterisk/keys/asterisk.crt
Certificate request self-signature ok
subject=CN = 192.168.44.130, O = CMP508
Enter pass phrase for /var/lib/asterisk/keys/ca.key:
Combining key and crt into /var/lib/asterisk/keys/asterisk.pem
```

Figure 21: Terminal output on certificate creation.

The following lines of text were added to the pjsip.conf file:

```
[transport-tls]

type = transport

protocol = tls

bind = 0.0.0.0:5061

cert_file=/var/lib/asterisk/keys/asterisk.crt

priv_key_file=/var/lib/asterisk/keys/asterisk.key
```

The following lines were added to each of the endpoints defined in the pjsip.conf file:

```
transport=tls

encryption=yes
```

The strp module was loaded within the Asterisk console using the following command:

```
module load res_srtp.so
```

The pjsip module was reloaded with the following command:

```
module reload res_pjsip.so
```

The new transport configuration can be seen in Figure 22:

```
PBX*CLI> pjsip show transports

Transport:  <TransportId........>  <Type>  <cos>  <tos>  <BindAddress.....................>
==========================================================================================

Transport:  transport-tls             tls      0      0  0.0.0.0:5061
Transport:  transport-udp             udp      0      0  0.0.0.0:5060

Objects found: 2
```

Figure 22: Asterisk console showing TLS connection on port 5061.

Several further steps were required to set up encrypted communications for the Linphone client applications. The newly created asterisk.pem certificate was copied to the /usr/share/linphone directory on both clients and renamed to rootca.pem. The certificate checking function in the Linphone client was disabled for both clients (necessary due to the self-signed nature of the certificate) by adjusting the value of the `verify_server_certs` configuration item from "`1`" to "`0`" in the linphonerc file (found in the .config/linphone directory of the user's Home directory). In order to use the TLS connection, a new SIP account was added using the Account Assistant accessed from the Home page of the Linphone application:

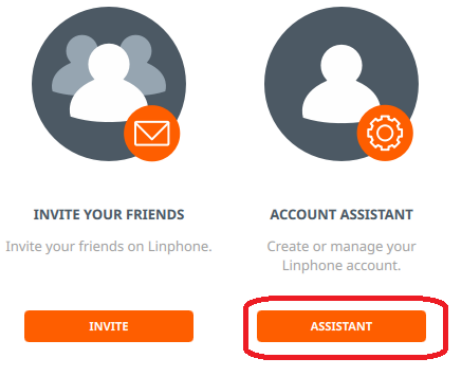

Figure 23: Adding a new account in the Linphone application, step 1.

The specific settings were input with the “Use a SIP Account” button:

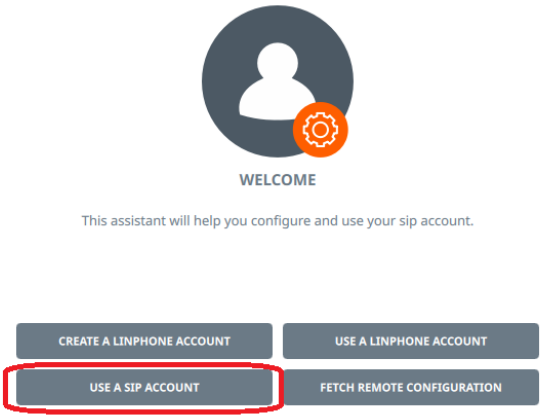

Figure 24: Adding a new account in the Linphone application, step 2.

The following options were set:

- Username.
- IP address of the PBX controller, including the port number.
- Password.
- Transport protocol set to TLS.

The settings were accepted using the “Use” button, as shown in Figure 25:

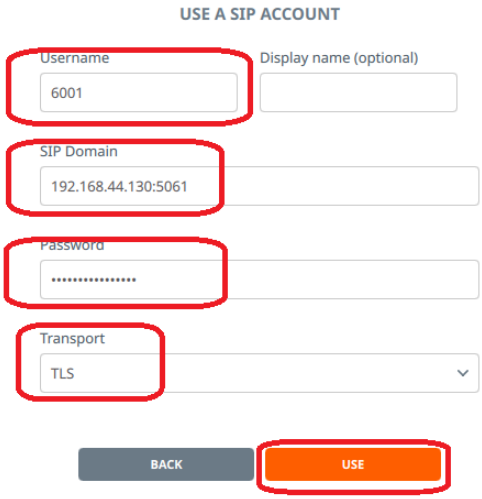

Figure 25: Adding a new account in the Linphone application, step 3.

Finally, encryption was enabled within the Linphone application by specifying the SRTP protocol and enforcing encryption in the “Calls and Chat” tab in the Preferences, accessed using the Options menu:

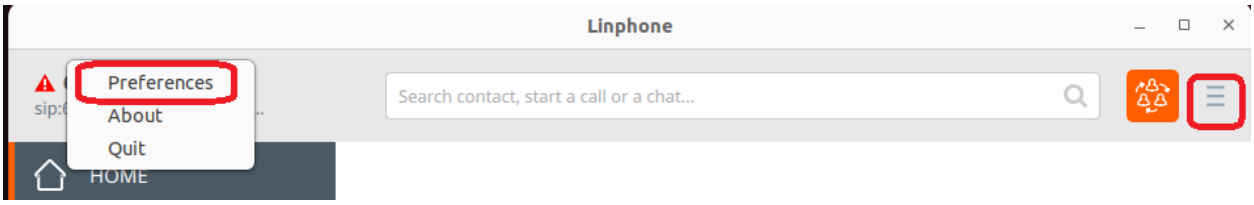

Figure 26: Enabling encryption in the Linphone application, step 1.

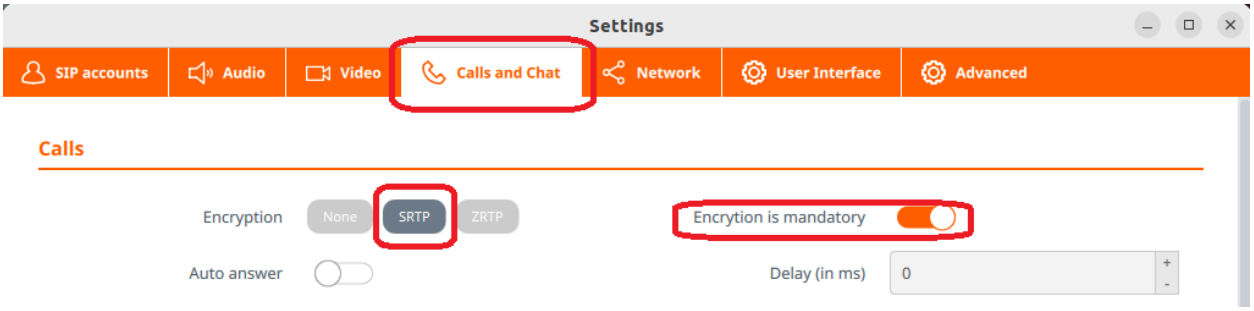

Figure 27: Enabling encryption in the Linphone application, step 2.

APPENDIX F – IMPLEMENTING PASSWORD HASHING FOR ENDPOINT USERS

Firstly, the module for enabling Asterisk to handle MD5 credentials was loaded with the following command within the Asterisk console:

```
module load func_md5.so
```

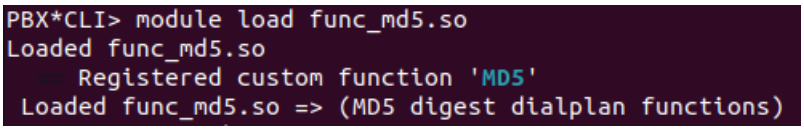

Figure 28: Output from MD5 module load success in Asterisk console.

In order to ensure that the hashed password contained the correct information, the Asterisk realm name was required. This was obtained using the following command:

```
asterisk -rx "pjsip show settings" | grep "realm"
```

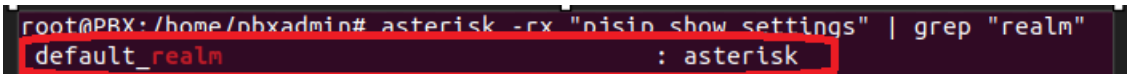

Figure 29: Obtaining the default realm name for the Asterisk instance.

The MD5 hash was then generated for both endpoints using the following command:

```
echo -n "<userId>:asterisk:<password>" | md5sum
```

The changes shown in Figure 30 were then made to the pjsip.conf file (original file content on the left):

Figure 30: Changes made to pjsip.conf file for implementation of MD5 hashed passwords.

Finally the pjsip module was reloaded with the following command:

```
reload res_pjsip.so
```

APPENDIX G – ENABLING THE FIREWALL AND IMPLEMENTING A BASIC CONFIGURATION

Firstly, the firewall was enabled with the command below and the resulting terminal output shown in Figure 31.

```
sudo ufw enable
```

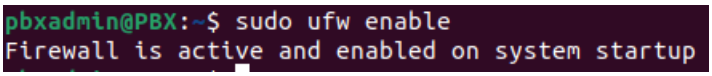

Figure 31: Terminal output after enabling the firewall.

The author determined that a basic rule would be required with the following criteria:

- The rule should apply to incoming traffic.
- Traffic from any machine on the local subnet (192.168.44.0/24) should be allowed.
- The traffic should only be allowed to port 5601.
- The protocol in use should be TCP.

Using these criteria, the rule was added to the ruleset with the following command:

```
sudo ufw allow proto tcp from 192.168.44.0/24 to any port 5601
```

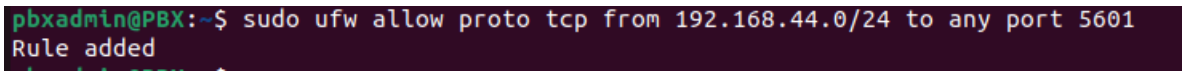

Figure 32: Adding the rule to the firewall.

APPENDIX H – CONTENTS OF COMMUNITY-SIP.RULES FILE FOR SNORT (DEPRECATED)

```
# Copyright 2005 Sourcefire, Inc. All Rights Reserved.

# These rules are licensed under the GNU General Public License.

# Please see the file LICENSE in this directory for more details.

# $Id: community-sip.rules,v 1.5 2006/06/01 15:51:28 akirk Exp $


# Note for Debian users: These rules are out of date and not

# recommended for production use. They have been commented out

# for interested users to use as a reference.

# Please use at your own risk


#Rules submitted by Jiri Markl

#Rule for alerting of INVITE flood attack:

#alert udp any any -> any 5060 (msg:"COMMUNITY SIP INVITE message
flooding"; content:"INVITE"; depth:6; threshold: type both, track
by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000158;
rev:2;)

#alert tcpp any any -> any 5060 (msg:"COMMUNITY SIP INVITE message
flooding"; content:"INVITE"; depth:6; threshold: type both, track
by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000158;
rev:2;)

#Rule for alerting of REGISTER flood attack:

#alert udp any any -> any 5060 (msg:"COMMUNITY SIP REGISTER message
flooding"; content:"REGISTER"; depth:8; threshold: type both, track
by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000159;
rev:2;)

#alert tcp any any -> any 5060 (msg:"COMMUNITY SIP REGISTER message
flooding"; content:"REGISTER"; depth:8; threshold: type both, track
by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000159;
rev:2;)

#Rule for alerting common TCP/UDP flood attack:

#alert udp any any -> any 5060 (msg:"COMMUNITY SIP TCP/IP message
flooding directed to SIP proxy"; threshold: type both, track by_src,
count 300, seconds 60; classtype:attempted-dos; sid:100000160; rev:2;)
```

```
#alert tcp any any -> any 5060 (msg:"COMMUNITY SIP TCP/IP message flooding directed to SIP proxy"; threshold: type both, track by_src, count 300, seconds 60; classtype:attempted-dos; sid:100000160; rev:2;)

#Rule for alerting attack using unresolvable DNS names:

#alert udp $DNS_SERVERS 53 -> any any (msg:"COMMUNITY SIP DNS No such name treshold - Abnormaly high count of No such name responses"; content:"|83|"; offset:3; depth:1; threshold: type both, track by_dst, count 100, seconds 60; classtype:attempted-dos; sid:100000161; rev:2;)

#Threshold rule for unauthorized responses:

#alert udp any any -> any 5060 (msg:"COMMUNITY SIP 401 Unauthorized Flood"; content:"SIP/2.0 401 Unauthorized"; depth:24; threshold: type both, track by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000162; rev:2;)

#alert tcp any any -> any 5060 (msg:"COMMUNITY SIP 401 Unauthorized Flood"; content:"SIP/2.0 401 Unauthorized"; depth:24; threshold: type both, track by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000162; rev:2;)

#alert udp any any -> any 5060 (msg:"COMMUNITY SIP 407 Proxy Authentication Required Flood"; content:"SIP/2.0 407 Proxy Authentication Required"; depth:42; threshold: type both, track by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000163; rev:2;)

#alert tcp any any -> any 5060 (msg:"COMMUNITY SIP 407 Proxy Authentication Required Flood"; content:"SIP/2.0 407 Proxy Authentication Required"; depth:42; threshold: type both, track by_src, count 100, seconds 60; classtype:attempted-dos; sid:100000163; rev:2;)

#Rule submitted by rmkml

#alert udp $EXTERNAL_NET any -> $HOME_NET 5060 (msg:"COMMUNITY EXPLOIT SIP UDP Softphone overflow attempt"; content:"|3B|branch|3D|"; content:"a|3D|"; pcre:"/^a\x3D[^\n]{1000,}/smi"; reference:bugtraq,16213; reference:cve,2006-0189; classtype:misc-attack; sid:100000223; rev:1;)
```