Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Ciberataques y contramedidas en servicios críticos

Cyberattacks and countermeasures in critical services

Damián Eduardo Domínguez De Barros

La Laguna, *31* de *mayo* de 2021

D. **Pino Caballero Gil**, con N.I.F. 45.534.310-Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y Sistemas de la Universidad de La Laguna, como tutora

D. **Carlos Rosa Remedios**, con N.I.F. 43.786.084-H responsable de la Unidad de Tecnologías de la Información y la Comunicación de Gestión de Servicios para la Salud y Seguridad en Canarias, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Ciberataques y contramedidas en servicios críticos”

ha sido realizada bajo su dirección por D. **Damián Eduardo Domínguez De Barros**,

con N.I.F. 43.385.303-N.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 12 de febrero de 2021.

Agradecimientos

A mi tutora, Pino Caballero Gil, y a mi cotutor, Carlos Rosa Remedios, por permitirme la oportunidad de realizar este trabajo y su ayuda durante el proceso.

A mis amigos, especialmente los compañeros de la carrera, por los buenos momentos en tiempos de estudio.

A mi familia, por todo el apoyo.

Licencia

Resumen

El objetivo de este trabajo ha sido el análisis de una centralita telefónica IP y el estudio de las posibles vulnerabilidades y ciberataques que puedan afectar a las comunicaciones, comprobando así su nivel de seguridad y mejorarlo en el caso de que esto fuera posible.

Para ello se ha utilizado el programa de software libre Asterisk, que aporta todas las funcionalidades necesarias para un correcto funcionamiento de la central telefónica. Se ha comprobado que una instalación básica de este servicio no es lo suficientemente seguro, pues permite capturar las comunicaciones sin ningún tipo de cifrado y reproducirlas posteriormente. Tras esto, se ha configurado adecuadamente para cifrar las conexiones y evitar ataques contra el servicio.

También se han estudiado distintas vulnerabilidades del tipo de cifrado utilizado, que es SSL/TLS, con el fin de evitar que se puedan realizar ataques del tipo Man in the Middle. El primer ataque investigado es el denominado Raccoon Attack, que afecta hasta la versión 1.2 de TLS y se basa en las mediciones de tiempo en el intercambio de los mensajes, de forma que cuando se dispone de suficientes pruebas se podrían conocer las claves de Diffie Hellman y descifrar los mensajes.

Palabras clave: IP-PBX, Asterisk, ciberataques, TLS, Raccoon Attack, Diffie Hellman.

Abstract

The aim of this work has been the analysis of an IP PBX and the study of possible vulnerabilities and cyber-attacks that may affect communications, thus checking its security level and improving it if possible.

For this purpose, the free software program Asterisk has been used, which provides all the necessary functionalities for the correct operation of the telephone exchange. It has been verified that a basic installation of this service is not secure enough, since it allows capturing communications without any type of encryption and reproducing them later. Subsequently, it has been properly configured to encrypt the connections and prevent attacks against the service.

Different vulnerabilities of the type of encryption used, which is SSL/TLS, have also been studied in order to prevent Man in the Middle attacks. The first attack investigated is the so-called Raccoon Attack, which affects up to TLS version 1.2 and is based on time measurements in the exchange of messages so that when sufficient evidence becomes available, the Diffie Hellman keys could be known and messages could be decrypted.

Keywords: IP-PBX, Asterisk, cyberattacks, TLS, Raccoon Attack, Diffie Hellman.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

1.1 Motivación

Desde hace aproximadamente dos décadas, impulsado por el desarrollo de las tecnologías, las comunicaciones y la definición del protocolo H.323 [1] por parte de la ITU-T, han propiciado que el mundo de la telefonía se integre dentro de Internet. En España, la cobertura de fibra óptica hasta la casa (*FTTH*) es de un 80% [2], donde las compañías han unificado sus servicios y los ofrecen a través de un cable de este tipo.

Todos estos factores han favorecido que las centrales telefónicas evolucionaran de analógicas a digitales y éstas, a su vez, a centrales telefónicas IP. En la actualidad, estas centrales son capaces de implementarse en un servidor o en un simple ordenador, con el consecuente ahorro que esto supone.

Sin embargo, hay que tener en cuenta que ningún sistema es seguro en su totalidad, y el hecho de que estos servicios estén conectados a Internet hace que se tengan que enfrentar a numerosos ataques, vulnerabilidades, fallos de seguridad, etc. Otro problema al que se enfrentan es que, por diversos motivos, se configuren de manera incorrecta, no se actualice el software o no se apliquen parches de seguridad. Es decir, no se realice un mantenimiento adecuado.

Si bien estos ataques no suelen acaparar titulares, no quiere decir que no existan. Los que sí lo suelen hacer son los de las vulnerabilidades en protocolos y herramientas que se consideran seguros y se utilizan para cifrar las comunicaciones.

1.2 Objetivos

Los objetivos de este proyecto son:

- Analizar la seguridad de una instalación básica de una centralita telefónica usando el programa Asterisk [3] y las casuísticas de ataques contra el servicio.
- Configuración del programa para evitar las posibles vulnerabilidades.
- Comprobar que las medidas adoptadas funcionan correctamente.
- Análisis de un ataque contra el protocolo SSL/TLS [4] que pudiera afectar al servicio.

1.3 Estado del arte

Solo en los Estados Unidos, entre los años 2010 y 2018, el número de líneas VoIP en empresas ha pasado de 6.2 millones a 41.6 [5]. Además, se espera que en el mundo el tamaño del mercado VoIP tenga una tasa de crecimiento anual compuesto del 3.1% entre 2021 y 2026 [6].

La lista de programas disponibles para implementar una *PBX*, tanto de software libre como propietario, es extensa. Sin embargo, hay que destacar la presencia de Asterisk, el programa elegido para el desarrollo de este proyecto. (Ver figura 1).

Fig. 1. Logo de Asterisk.

Publicado por primera vez en 1999, cuenta con numerosas funciones como buzones de voz, soporte para conferencias y un gran soporte para numerosos protocolos. Existen otros muchos productos que están basados o utilizan Asterisk de base, como Issabel [7] o Elastix antes de ser adquirida por la compañía 3CX [8]. (Ver figuras 2 y 3).

Fig. 2. Logo de Issabel.

Fig. 3. Logo de Elastix.

En la actualidad, Asterisk presume de unas dos millones de descargas anuales y estar instalado en alrededor de un millón de servidores [9]. Su última versión estable se publicó el 25 de marzo de 2021, siendo ésta la 18.3.0. Por esto es normal que, con el paso de los años, exista una lista de vulnerabilidades que afecten al servicio [10]. (Ver figura 4).

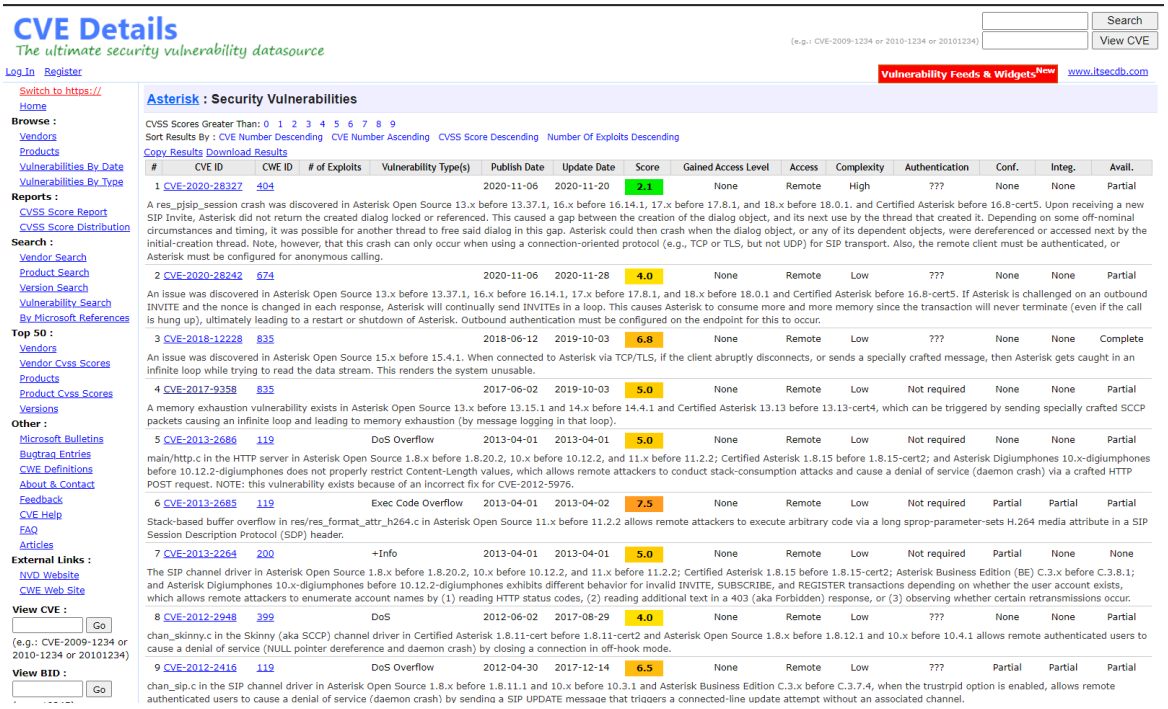

Fig. 4. Lista de vulnerabilidades de Asterisk.

1.4 Estructura de la memoria

En el capítulo dos se llevará a cabo una introducción de la aplicación explicando su uso y la seguridad de la misma así como el trabajo que se ha realizado. Además se explicarán los otros programas que se han utilizado en el desarrollo del proyecto. En los siguientes capítulos se explicará detalladamente el trabajo realizado, tratando en el tercero la instalación, primeros usos y posterior configuración de Asterisk, en el cuarto un análisis del nivel se seguridad del servicio y en el quinto otro análisis sobre una vulnerabilidad que afecta al protocolo SSL/TLS, utilizado para cifrar las comunicaciones de la aplicación.

Capítulo 2 Introducción al sistema

2.1 Máquinas virtuales

Para el desarrollo del trabajo se han utilizado una serie de máquinas virtuales utilizando el programa de virtualización Oracle VM VirtualBox [11] (ver figura 5). Esta elección en lugar de la plataforma de la que dispone la Universidad de La Laguna, el IaaS [12] (ver figura 6), se debe a la necesidad de disponer de una entrada y salida de audio.

Fig. 5. Logo de Oracle VM VirtualBox.

En total se crearon cuatro máquinas virtuales basadas en Linux [13]. Tres de ellas, la centralita telefónica y dos clientes, cuentan con Ubuntu 20.04 LTS [14]. La cuarta, que simula a un atacante y sirve para realizar algunas pruebas de seguridad, dispone de Kali Linux 2021.1 [15]. Esta elección se debe a que ambas disponen de muchas herramientas preinstaladas, facilitando el trabajo al no tener que llevar a cabo un extenso proceso de configuración.

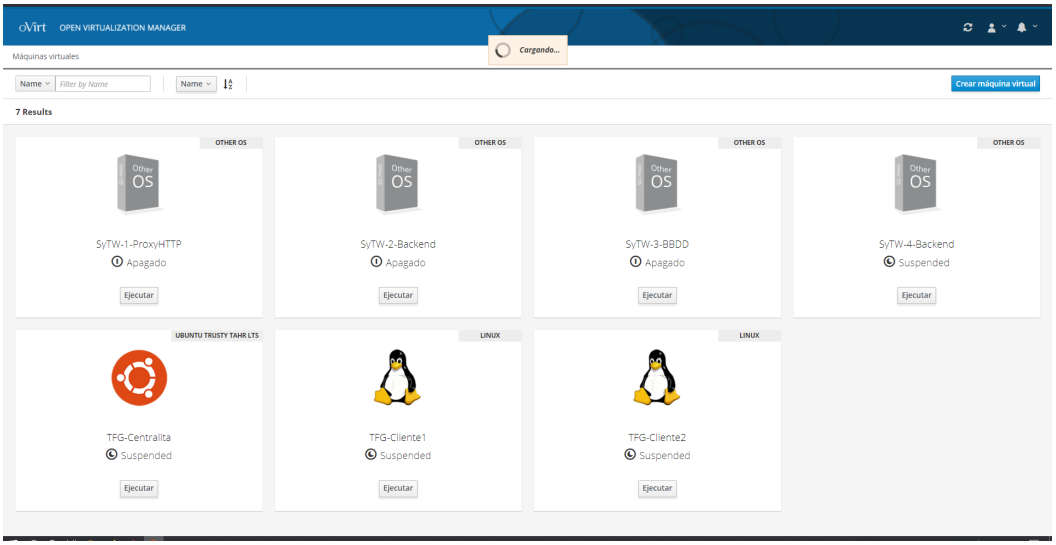

Fig. 6. IaaS de la ULL.

2.2 Definición del problema

El principal problema al que nos enfrentamos es que, tras instalar Asterisk en la supuesta centralita y los programas o *softphones* para realizar las llamadas en las máquinas que harán de clientes, comprobamos que no están cifradas y se pueden capturar fácilmente con cualquier capturador de paquetes de red o *sniffer*.

Además, la configuración del programa cuenta por defecto con la posibilidad de que usuarios anónimos sean capaces de realizar llamadas, por lo que si no se tiene en cuenta es una vía que cualquier atacante puede aprovechar para llevar a cabo estafas telefónicas.

2.3 Herramientas y programas utilizados

TABLA I

Software utilizado para la realización de este trabajo

Nombre	Uso
Oracle VM VirtualBox	Programa de virtualización
Asterisk	Software para la implementación de la *PBX*

Blink [16]	Software para los softphones
Wireshark [17]	Capturador de paquetes
Nmap [18]	Herramienta para escanear los puertos abiertos
SIPVicious svmap [19]	Conjunto de herramientas para escanear los puertos de la centralita, extensiones y realizar ataques de fuerza bruta.
Fail2Ban [20]	Programa para evitar ataques de fuerza bruta
Iptables [21]	*Firewall*

Nota: Se indican los nombres de los programas y su uso para el análisis de la seguridad de la central telefónica.

2.4 Conceptualización de la propuesta

El planteamiento llevado a cabo, como es para el estudio de posibles vulnerabilidades y no para una implementación en una empresa o institución, no tiene en cuenta ningún requisito ni configuración extra que pudiera necesitar éstas como pudieran ser llamadas telefónicas a números de teléfono, buzones de voz, múltiples niveles, grabación de llamadas, etc.

Por estos motivos, la propuesta es la de configurar Asterisk para cifrar las comunicaciones y que no sea posible que un capturador de paquetes pueda tener acceso a las llamadas. Además, no se permitirán usuarios sin autenticación, lo que ayudará a dificultar ataques contra los usuarios y las extensiones disponibles así como evitar en lo posible el fraude telefónico.

Capítulo 3 Asterisk

3.1 Instalación

En este capítulo se explica como se ha llevado a cabo la instalación de Asterisk y los programas para que los supuestos usuarios puedan realizar llamadas, siguiendo el esquema de la figura 7, así como una prueba para comprobar que las comunicaciones funcionan correctamente y, aprovechando esta prueba, confirmar que las llamadas no están cifradas, teniendo que configurar Asterisk para ello.

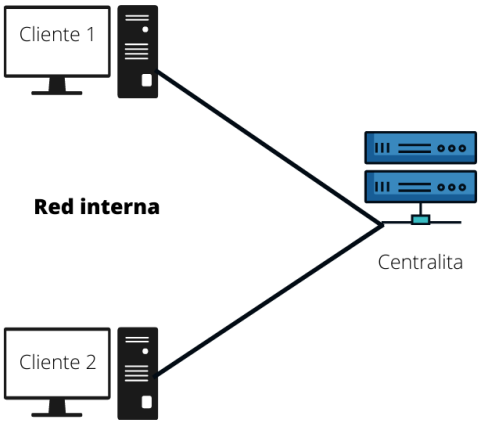

Fig. 7. Esquema de los dispositivos.

Para instalar Asterisk en la máquina que hará de centralita telefónica usaremos la línea de comandos y el proceso no difiere de cualquier otra instalación por este medio en Linux. Primero se actualizan los repositorios y después se instalan los paquetes.

```
sudo apt-get update
sudo apt-get install asterisk
```

Por defecto, el idioma predeterminado para los sonidos de Asterisk es el inglés aunque hay disponibles paquetes para otros idiomas. En nuestro caso instalaremos los que existen en español de la siguiente manera.

```
sudo apt-cache search asterisk //Lista con los paquetes disponibles

sudo apt-get install asterisk-prompt-es asterisk-core-sounds-es
asterisk-core-sounds-es-g722 asterisk-core-sounds-es-gsm
asterisk-core-sounds-es-wav

dpkg -l asterisk* //Lista de paquetes instalados

sudo service asterisk start  // Arrancamos el servicio
sudo service asterisk status // Comprobamos su estado
```

Al comprobar el estado del servicio se observa que existe un error en las rutas de dos ficheros de configuración:

- `cel.conf`.
- `cdr.conf`.

Ambos archivos hacen referencia a un directorio llamado `/radiusclient-ng` pero no existe. Dicho directorio se encuentra pero con otro nombre, `/radcli`, por lo que se tienen que actualizar las rutas, que se encuentran al final de ambos ficheros como se puede observar en las figuras 8 y 9.

```
;radiuscfg => /usr/local/etc/radiusclient-ng/radiusclient.conf
;

radiuscfg => /etc/radcli/radiusclient.conf
root@centralitaTFG:~# cat /etc/asterisk/cel.conf
```

Fig. 8. Captura del archivo `cel.conf`.

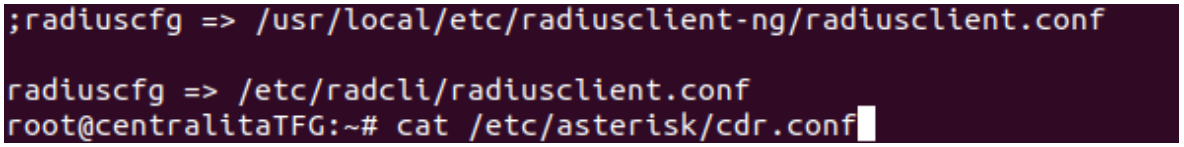

Fig. 9. Captura del archivo `cdr.conf`.

Si volvemos a comprobar el estado del servicio, veremos como este funciona correctamente, por lo que se podrá continuar con la configuración y añadir a los dos usuarios de prueba para que se puedan comunicar entre ellos.

3.2 Configuración inicial

Para añadir a nuestros usuarios y las extensiones de marcación, trabajaremos en los dos siguientes ficheros:

- `sip.conf`: En él se encuentra la configuración del programa: codecs permitidos, protocolos de la capa de transporte, plantillas de usuario, etc.
- `extensions.conf`: Contiene el plan de llamadas o dialplan, que indica el comportamiento de nuestra centralita.

Por defecto, estos ficheros vienen con muchos ejemplos a modo de documentación y su extensión es muy grande. Por este motivo y para facilitar la modificación del fichero se ha decidido realizar una copia y borrar todos los espacios en blanco y líneas comentadas. De esta forma podemos observar los parámetros que están indicados, como que el protocolo utilizado es UDP [22], no están permitidas la conexiones con TCP [23] y que se sobreescriben ciertos parámetros por defecto.

```
[general]
context=public   ; Default context for incoming calls. Defaults to 'default'
allowoverlap=no ; Disable overlap dialing support. (Default is yes)
udpbindaddr=0.0.0.0 ; IP address to bind UDP listen socket to (0.0.0.0 binds to
all)
tcpenable=no    ; Enable server for incoming TCP connections (default is no)
```

```
tcpbindaddr=0.0.0.0   ; IP address for TCP server to bind to (0.0.0.0 binds to all interfaces)
transport=udp   ; Set the default transports.  The order determines the primary default transport.
srvlookup=yes   ; Enable DNS SRV lookups on outbound calls
```

A continuación añadiremos la configuración para nuestros usuarios. Para ello creamos una plantilla llamada *usuario* en la que se indica que aquellos que la tengan asignada serán del tipo *friend*, es decir, podrán realizar y recibir llamadas y que usarán el contexto llamado *supuestoTFG*. También indicamos que su *host* será dinámico, por lo que se podrán conectar desde cualquier dirección IP siempre que inicien sesión con sus credenciales. Por último, solo permitiremos el codec *alaw*, que viene a ser el estándar G.711 de ITU-T para la codificación de audio [24]. Esta versión es usada en Europa y en el resto del mundo salvo en los Estados Unidos y Japón, que usan la versión *ulaw*.

Lo siguiente que haremos será añadir a nuestros dos usuarios, indicando que usarán la plantilla que hemos creado e indicaremos su nombre de usuario y contraseña.

```
[usuario](!)
type=friend
host=dynamic
disallow=all
allow=alaw
context=supuestoTFG

;Cliente 1
[cliente1](usuario)
username=cliente1
secret=********

;Cliente 2
[cliente2](usuario)
username=cliente2
secret=**********
```

Llegados a este punto, nuestros dos clientes podrán registrarse en la centralita pero no podrán realizar ninguna llamada pues no hemos especificado ningún plan de llamadas.

Así pues, añadimos al fichero `extensions.conf` el contexto que usarán nuestros usuarios así como la extensión o el número de marcación, la prioridad y la aplicación con la tecnología usada y el usuario.

```
[general]

[supuestoTFG]
exten => 101,1,Dial(SIP/cliente1)
exten => 102,1,Dial(SIP/cliente2)

exten => 001,1,Playback(demo-congrats)
exten => 001,2,Dial(SIP/cliente1)
exten => 001,3,Hangup()
```

La prioridad sirve para realizar acciones antes o después de una llamada. Por ejemplo, si el cliente dos marca el número 001, primero escuchará un audio de prueba, después llamará al cliente uno y al final, se colgará la llamada.

3.3 Instalación de softphones en los clientes

Dado que nuestros dos clientes ya pueden registrarse y llamarse entre ellos, el siguiente paso es instalar el software necesario para puedan realizar dichas llamadas. Para ello usarán el *softphone* Blink, que está disponible para Linux, Windows [25] y MacOS [26]. (Ver figura 10).

Fig. 10. Logo de Blink.

Para instalarlo es necesario obtener la firma del código para comprobar que no ha sido alterado y modificar el fichero ubicado en `/etc/apt/sources.list`.

1. Obtenemos la firma.

```
sudo curl -o /etc/apt/trusted.gpg.d/agp-debian-key.gpg
http://download.ag-projects.com/agp-debian-key.gpg
```

2. Modificamos el fichero /etc/apt/sources.list.

```
deb     http://ag-projects.com/ubuntu focal main
deb-src http://ag-projects.com/ubuntu focal main
```

3. Actualizamos los repositorios e instalamos el paquete.

```
sudo apt-get update
sudo apt-get install blink
```

Al entrar en la aplicación, en el proceso de añadir una nueva cuenta, marcaremos que ya disponemos de una cuenta SIP [27]. En el campo de la dirección SIP se tendrá que indicar el nombre que está especificado en el fichero `sip.conf` y en nuestro caso la IP de la centralita, que en caso de disponerlo se podría sustituir por un dominio. (Ver figura 11).

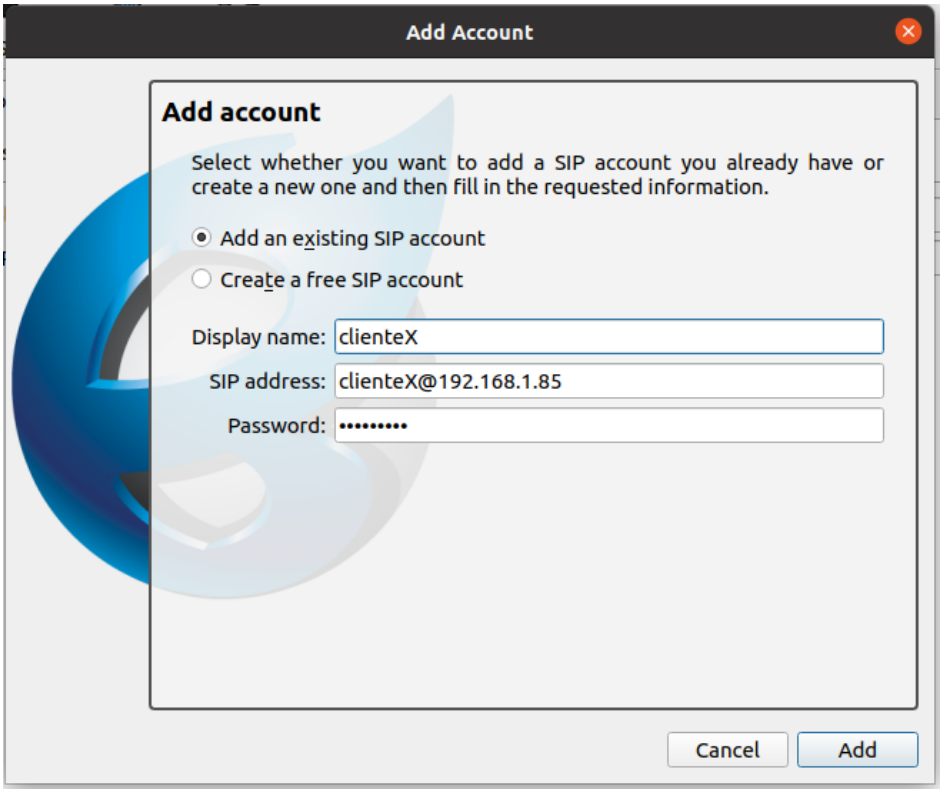

Fig. 11. Añadiendo cuenta de los clientes.

3.4 Prueba del servicio

Para comprobar que el servicio funciona correctamente realizaremos una llamada entre ambos usuarios. Antes de que eso ocurra, instalaremos el *sniffer* Wireshark (ver figura 12) de forma similar a las anteriores aplicaciones.

Fig. 12. Logo de Wireshark.

```
sudo apt update
sudo apt install wireshark
```

Tras la instalación, iniciamos la aplicación, grabamos los paquetes de red y realizamos la llamada. Cuando ésta termine, detenemos la grabación y accedemos al menú de telefonía del programa. En él podremos observar y reproducir las comunicaciones realizadas mientras capturamos los paquetes de red como se observa en la figura 13.

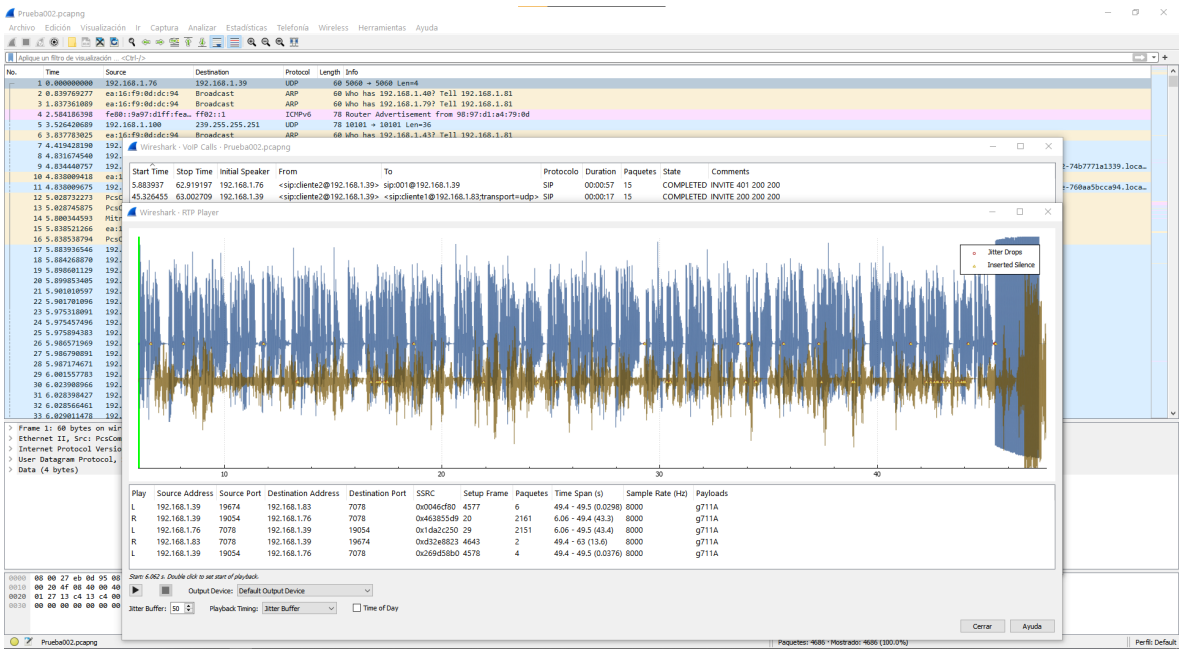

Fig. 13. Llamada capturada con Wireshark.

3.5 Configurando el cifrado de las comunicaciones

Para evitar que esto sea posible, hay que cambiar la configuración del fichero `sip.conf` así como ejecutar un script para crear los certificados necesarios para la centralita y los clientes y de esta forma poder usar el protocolo TLS.

El script no se encuentra instalado por defecto, por lo que hay que obtenerlo del repositorio de Asterisk en GitHub [28]. Tras clonarlo en nuestro ordenador, lo ejecutaremos tres veces: una para los certificados de la centralita y dos para los de los clientes.

1. Generamos los certificados de la *PBX*.

```
./ast_tls_cert -C 192.168.1.85 -O "SupuestoTFG" -d /etc/asterisk/keys
```

2. Volvemos a ejecutar para obtener los certificados de los usuarios.

```
./ast_tls_cert -m client -c /etc/asterisk/keys/ca.crt -k
/etc/asterisk/keys/ca.key -C 192.168.1.83 -O "SupuestoTFG" -d
/etc/asterisk/keys -o cliente1

./ast_tls_cert -m client -c /etc/asterisk/keys/ca.crt -k
/etc/asterisk/keys/ca.key -C 192.168.1.76 -O "SupuestoTFG" -d
/etc/asterisk/keys -o cliente2
```

A continuación, deberemos modificar el contenido del `sip.conf`, indicando que usaremos el protocolo TLS, la ruta a los certificados y en la plantilla de los usuarios añadiremos la opción para que las comunicaciones sean cifradas. También deshabilitaremos la opción de que usuarios no autenticados puedan realizar llamadas.

```
[general]
transport=tls      ; Set the default transports.  The order determines the primary
default transport.
srvlookup=yes      ; Enable DNS SRV lookups on outbound calls
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/keys/asterisk.pem
tlscafile=/etc/asterisk/keys/ca.crt
tlscipher=ALL
tlsclientmethod=tls1
allowguest=no
```

```
[usuario](!)
type=friend
host=dynamic
disallow=all
allow=alaw
context=supuestoTFG
encryption=yes

;Cliente 1
[cliente1](usuario)
username=cliente1
secret=********


;Cliente 2
[cliente2](usuario)
username=cliente2
secret==========
```

Por parte de los usuarios, estos deberán disponer de dos ficheros:

- `ca.crt`
- `clienteX.pem`

Ambos ficheros se encuentran en el directorio `/etc/asterisk/keys`, tal y como se especificó al ejecutar el *script*. También se deberá cambiar la configuración del *softphone* por la siguiente:

- Marcar el campo *encryption: SDES mandatory* [29] en las opciones RTP [30]. (Ver figura 14).
- Usar el puerto 5061 e indicar en el campo de transporte el protocolo TLS. (Ver figura 15).
- Indicar las rutas a los certificados tanto para la cuenta del cliente como para la centralita. (Ver las figura 16 y 17).

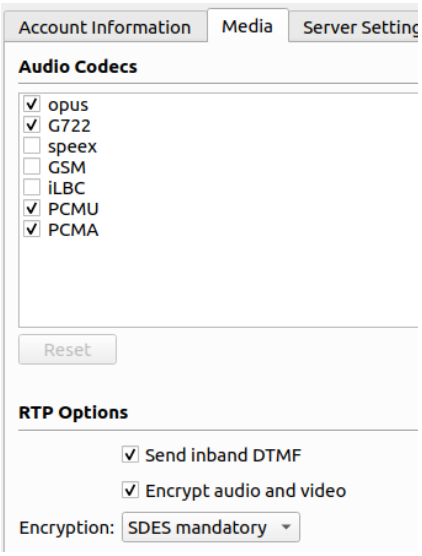

Fig. 14. Captura campo de cifrado.

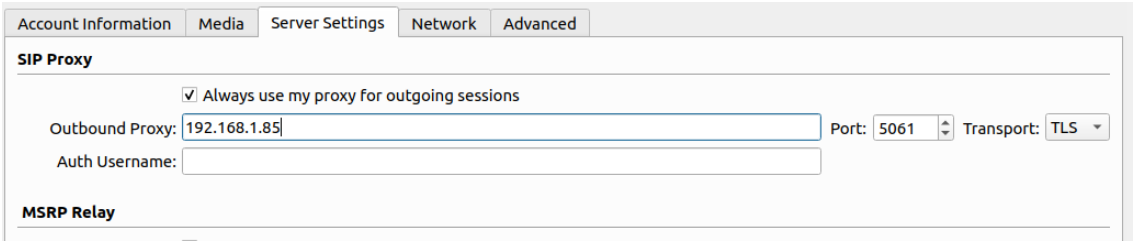

Fig. 15. Captura puerto y transporte.

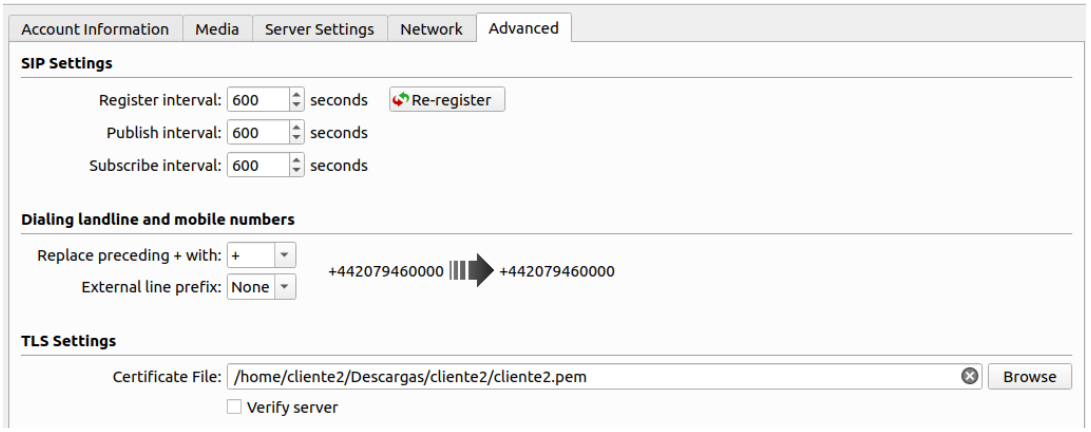

Fig. 16. Ruta del certificado para la cuenta del cliente.

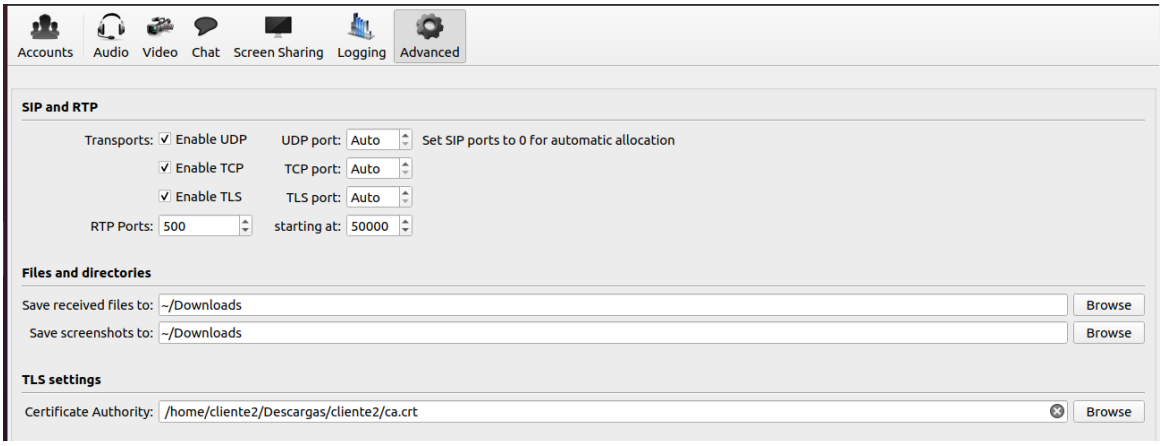

Fig. 17. Ruta para el cifrado de la centralita.

Si repetimos el proceso que realizamos anteriormente y hacemos una nueva llamada capturando los paquetes con el Wireshark, se observa, en la figura 18, que en esta ocasión no existe ningún rastro de la llamada.

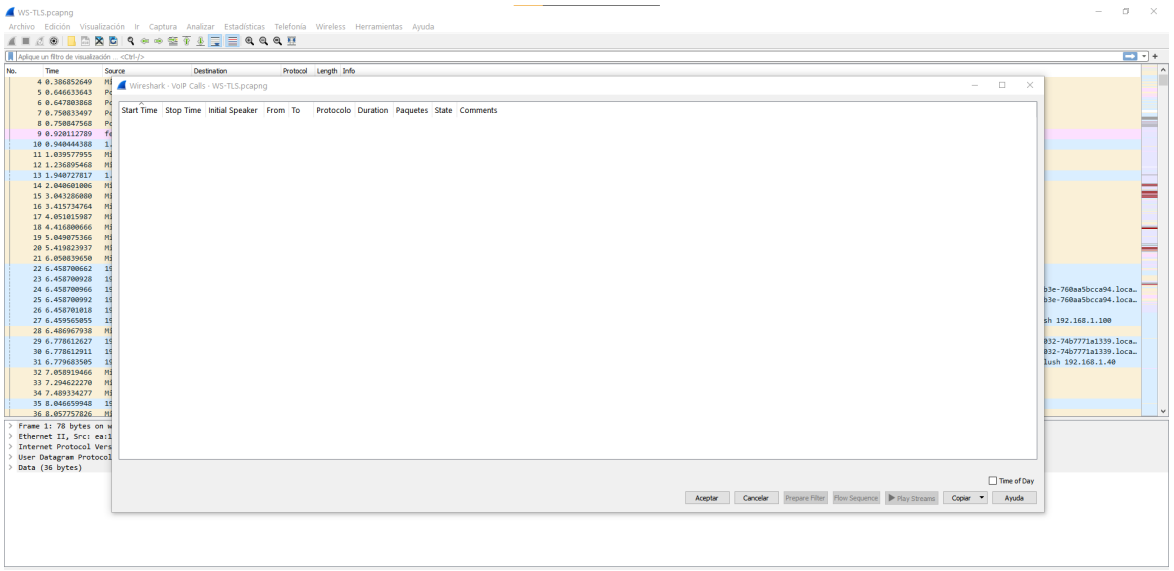

Fig. 18. Captura sin rastro de llamadas.

Si se observa en Blink la llamada en curso, aparecerán dos candados indicando que las comunicaciones están cifradas y si nos situamos encima de ellos nos indicará el protocolo utilizado. (Ver figura 19).

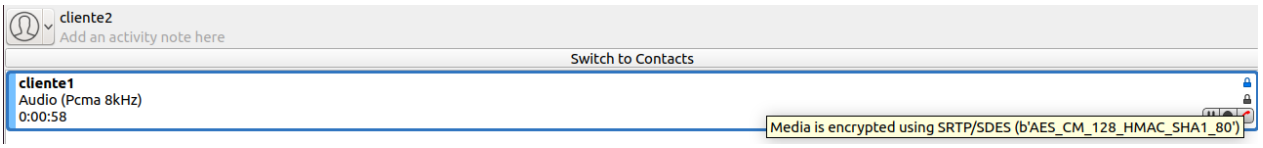

Fig. 19. Llamada cifrada con protocolo utilizado.

Capítulo 4 Seguridad del servicio

4.1 Herramientas SIPVicious

Con nuestra centralita funcionando y pudiendo nuestros usuarios comunicarse entre ellos, es momento de comprobar otros aspectos de seguridad que pueden afectar a este tipo de servicios. Concretamente nos referimos a ataques de fuerza bruta para intentar obtener las credenciales de los usuarios o las extensiones utilizadas.

Para llevar a cabo estas pruebas utilizaremos los programas que se encuentran dentro del paquete SIPVicious OSS de Enable Security. (Ver figura 20).

Fig. 20. Logo de SIPVicious.

Dentro de este paquete usaremos:

- **svmap**: Se trata de un escáner de dispositivos SIP, similar al programa nmap.
- **svwar**: También es otro escáner pero en este caso de extensiones.
- **svcrack**: Sirve para realizar ataques de fuerza bruta para obtener las contraseñas de usuarios o extensiones.

Empezaremos utilizando svmap indicando la dirección IP de nuestra centralita como se puede observar en la figura 21.

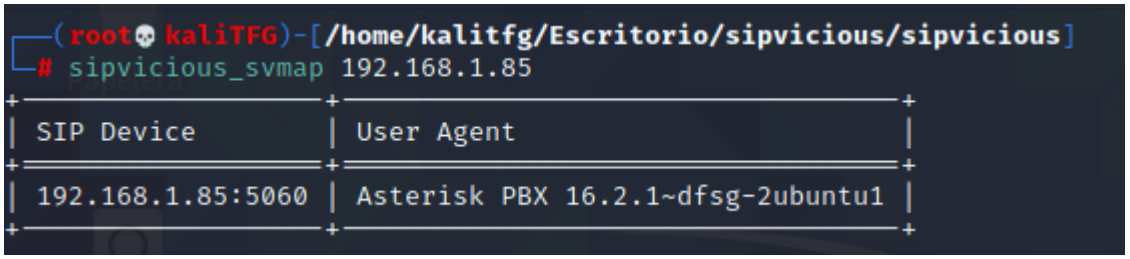

Fig. 21. Uso de svmap.

El programa nos devuelve la lista de dispositivos SIP y su *User Agent*. El principal problema que encontramos es que se indica la versión de Asterisk y esto puede ser un problema, ya que si se tratara de una versión antigua se podría explotar alguna vulnerabilidad ya conocida. Para evitar esto, se debe añadir en el `sip.conf` la siguiente entrada. (Ver figura 22).

```
useragent=Asterisk PBX
```

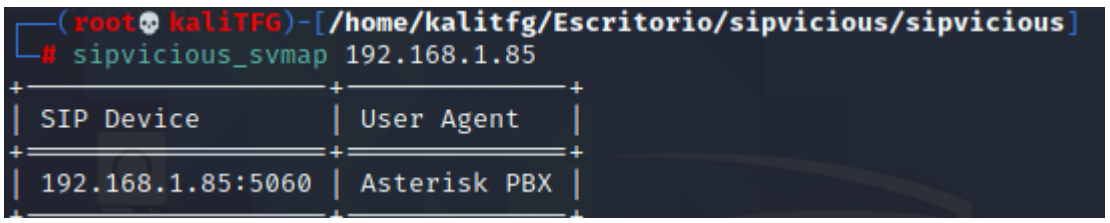

Fig. 22. Comprobación de cambio de *User Agent*.

En caso de que se quisiera conocer que puertos de la centralita están abiertos al exterior, podríamos usar nmap como hemos utilizado svmap. (Ver figura 23).

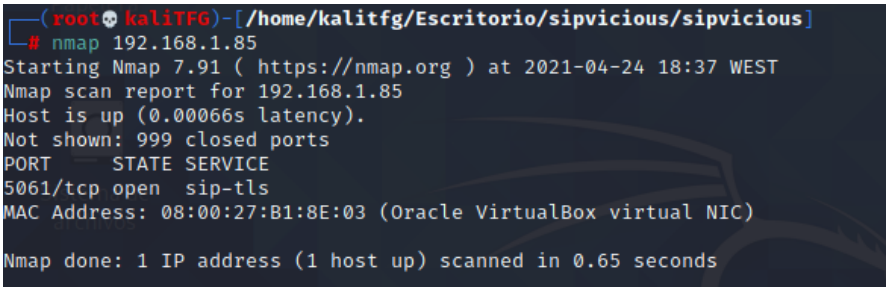

Fig. 23. Uso de Nmap.

Llegados a este punto conoceríamos los dispositivos y puertos en uso de la centralita o los del resto de la red, por lo que el siguiente paso sería conocer las extensiones o obtener la contraseña de alguno de los usuarios. Sin embargo comprobamos (Ver figuras 24 y 25) que si se intenta listar las extensiones no podremos pues necesitamos estar autenticados o en caso de acertar la contraseña no obtendremos nada como respuesta pues se necesita usar el protocolo TLS y sin los certificados esto no sería posible.

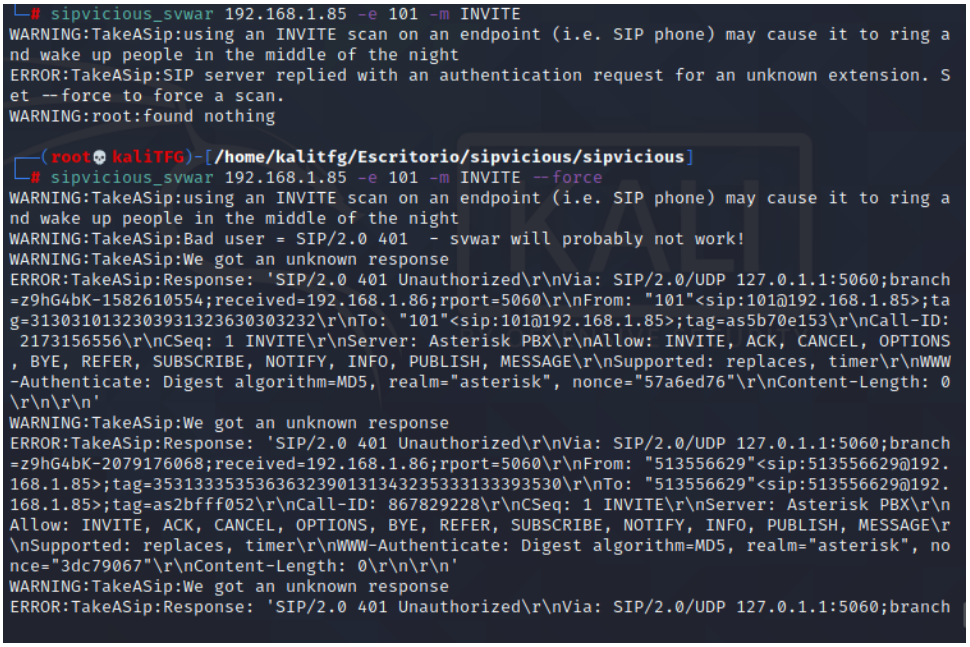

Fig. 24. Escaneo de extensiones.

Este ejemplo sería si se acertara la contraseña, ya sea porque se ha obtenido anteriormente y ya es conocida o porque una de las numerosas opciones de un ataque de fuerza bruta la contenía. Este tipo de ataques pueden utilizar un diccionario, que no es más que un fichero con miles de contraseñas filtradas en la web, palabras comunes más utilizadas o siguiendo un cierto patrón.

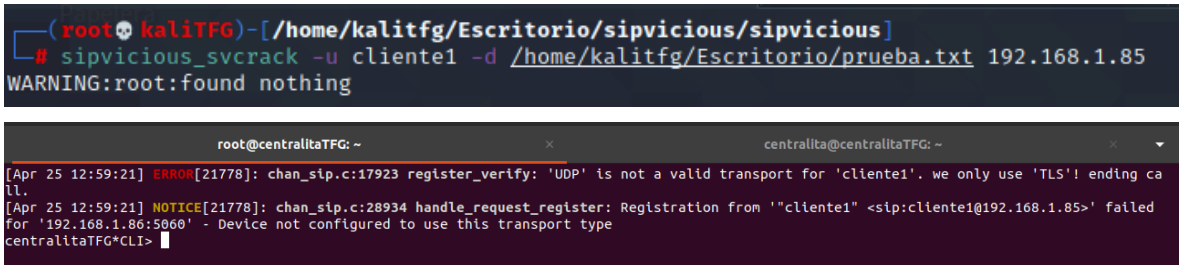

Fig. 25. Acertando contraseña.

SVCrack permite utilizar ficheros de este tipo y si utilizamos con el que cuenta Kali Linux, ubicado en `/usr/share/wordlists/` y llamado `rockyou`, contra nuestra centralita podremos observar como en la consola de Asterisk los mensajes de aviso no paran de aparecer. (Ver figura 26).

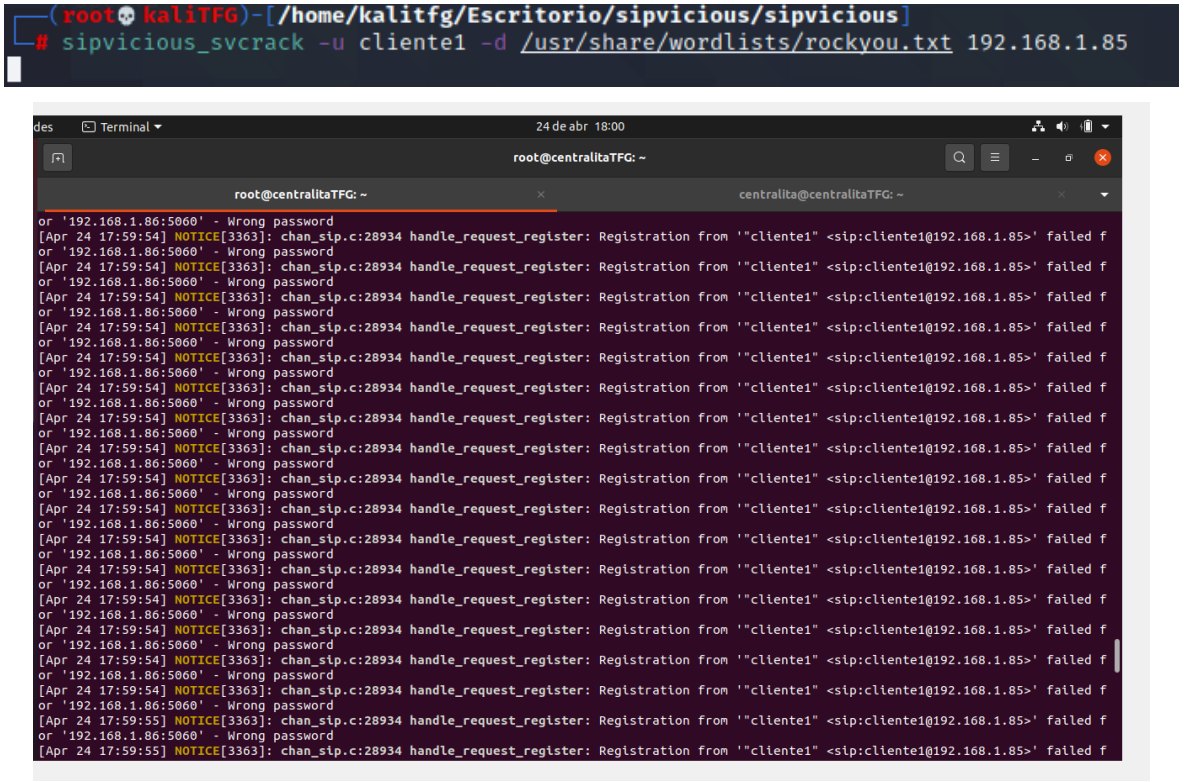

Fig. 26. Mensajes de error en la consola.

Debemos tener en cuenta que estos mensajes también se guardan en un fichero de log, por lo que si no se toman medidas contra este tipo de ataques el tamaño de este fichero aumentará considerablemente, conteniendo el mismo mensaje y en caso de ocurrir otro error y querer buscarlo será prácticamente imposible. También aumentará el ancho de banda y el uso de CPU, pudiendo dedicarse esos recursos al funcionamiento del sistema para mejorar la calidad de las comunicaciones.

4.2 Fail2Ban

Para dificultar lo máximo posible estos ataques se ha decidido instalar Fail2Ban. Este es un programa de software libre gratuito que analiza los ficheros de registros de otros programas en busca de signos de actividad sospechosa y modifica las reglas del *firewall*. De esta forma ganamos en seguridad y recuperamos el ancho de banda y los recursos que se destinan a estos ataques. (Ver figura 27).

Sin embargo, hay que tener en cuenta que Fail2Ban no es una herramienta infalible y que se debe acompañar de otras medidas, como una buena política de contraseñas, un diseño de red adecuado, un *firewall* bien configurado, etc. Además, no es la única herramienta de este tipo que existe y puede que existan otras, gratuitas o de pago, más completas.

Fig. 27. Logo de Fail2Ban.

Dado que no disponíamos de un *firewall*, instalamos iptables junto a Fail2Ban. (Ver figura 28).

Fig. 28. Iptables es un componente de Netfilter.

```
sudo apt update
sudo apt install iptables
sudo apt install Fail2ban
```

Una vez instalado, podremos empezar a configurar el servicio. Entre todos los ficheros disponibles, los que más nos interesan son:

- `etc/fail2ban/dilter.d/asterisk.conf`: Contiene una serie de expresiones regulares para examinar el fichero de *logs* de Asterisk y así poder actuar.
- `etc/fail2ban/jail.conf`: Contiene la configuración del programa y cómo se actuará.

Se recomienda que no se modifique el `jail.conf` y que se cree uno llamado `jail.local` que incluya los parámetros para las aplicaciones que se quieren vigilar. De esta forma, si al actualizar Fail2Ban se sobreescribe el fichero, el servicio seguirá funcionando al existir el `jail.local`. Así que lo creamos y añadimos lo siguiente:

```
[DEFAULT]
bantime.increment = true
bantime.rndtime   = 259200
findtime          = 43200
maxretry          = 5

[asterisk]
enabled  = true
port     = 5060,5061
action       =  %(banaction)s[name=%(__name__)s-tcp,  port="%(port)s",
protocol="tcp", chain="%(chain)s", actname=%(banaction)s-tcp]
          %(banaction)s[name=%(__name__)s-udp,        port="%(port)s",
protocol="udp", chain="%(chain)s", actname=%(banaction)s-udp]
          %(mta)s-whois[name=%(__name__)s, dest="%(destemail)s"]
logpath  = /var/log/asterisk/messages
maxretry = 5
```

La configuración que se ha indicado es la siguiente:

- Se ha establecido un tiempo de baneo aleatorio con un máximo de 72 horas. En caso de ser fijo, con un *script* se podría llegar a saber el tiempo y continuar con el

ataque una vez se pueda reanudar. De esta forma se dificulta que se pueda aprender.

- En caso de que la IP ya haya sido bloqueada anteriormente, el nuevo tiempo de baneo será mayor que el anterior.
- Solo se podrá reintentar 5 veces la contraseña en un plazo de 12 horas (43200 segundos).

En este caso no se ha especificado que se ignore alguna dirección IP. Sin embargo, sí se debería añadir este campo a la hora de implementarse en un entorno real, indicando así aquellas direcciones que sean para los administradores y cambiar o relajar los parámetros para las direcciones locales de la empresa o institución.

Tras esto, si se intentara un ataque de fuerza bruta como el del anterior ejemplo, el resultado sería el que se observa en la figura 29.

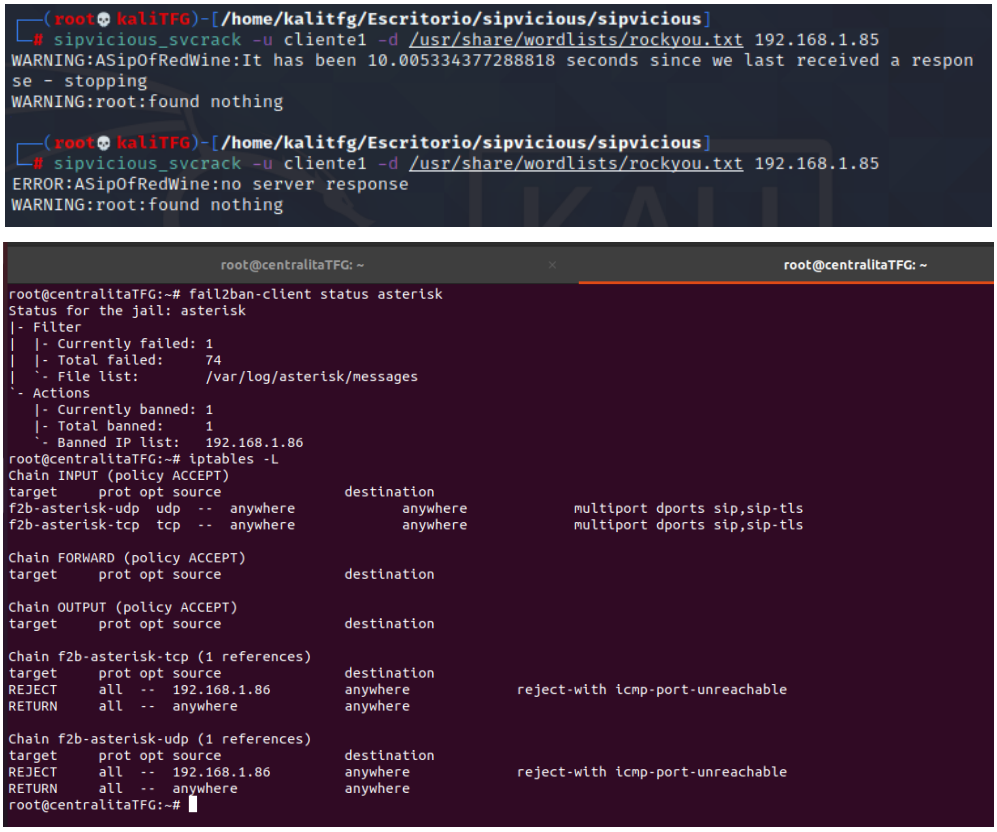

Fig. 29. Intento de ataque con Fail2Ban activado.

De esta forma estaríamos bloqueando la amenaza. En caso de seguir sufriendo este tipo de ataques y ser capaces de identificar un rango de IP, podríamos añadirlas manualmente al programa o directamente al *firewall*.

Capítulo 5 Raccoon Attack

5.1 Introducción

Ahora que se ha reforzado la seguridad de la centralita, es hora de analizar la tecnología que se utiliza para el cifrado de las comunicaciones. Para ello se analizará una nueva vulnerabilidad descubierta por los investigadores Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, y Jörg Schwenk [31].

Raccoon Attack (ver figura 30) es un nuevo ataque que explota una vulnerabilidad de tiempo y que afecta a ciertas versiones del protocolo criptográfico TLS. Se trata de un ataque más teórico que práctico debido a que se necesitan unas condiciones específicas para que tenga éxito. Esto se debe a que se necesita una medición exacta del tiempo y que se tenga habilitado la reutilización de claves de Diffie-Hellman [32], así como una versión de TLS 1.2 o inferior.

Fig. 30. Logo de Raccoon Attack.

5.2 TLS: Transport Layer Security

Este protocolo fue definido en 1999 como una actualización de SSL. A lo largo de los años se ha ido actualizando corrigiendo errores y soportando métodos más seguros según se descubrían ataques y vulnerabilidades en los métodos de cifrado. En 2018 fue definido su versión 1.3, aunque lo habitual es que se utilice la versión 1.2.

Cuando se va a realizar una conexión entre un cliente y el servidor, se realiza una serie de pasos denominados *handshake* (Ver figura 31), donde se autentica al servidor por su certificado. El primero es la fase de negociación:

- El cliente envía un mensaje *ClientHello,* donde indica la versión más alta del protocolo TLS que soporta y una lista de cifrados sugeridos
- El servidor responde con un mensaje *ServerHello*, que indica la versión elegida. Esta debe ser la más alta que soporta tanto el cliente como el servidor.
- A continuación, el servidor envía su mensaje de certificado, el mensaje *ServerKeyExchange* y el *ServerHelloDone*, indicando que terminó la negociación del *handshake*.
- El cliente responde con un mensaje *ClientKeyExchange*, donde se indica el *premaster secret* y la clave pública. El *premaster secret* es cifrado con la clave pública del certificado del servidor.
- Ambos, cliente y servidor, calculan el llamado *master secret*.
- Lo último es un intercambio de mensajes llamado *ChangeCipherSpec*, una especie de “todo lo que yo diga a continuación será autenticado”. Se envía un mensaje de *Finished*, que se deberá comprobar. En caso de que la verificación falle, la conexión se debe cortar.

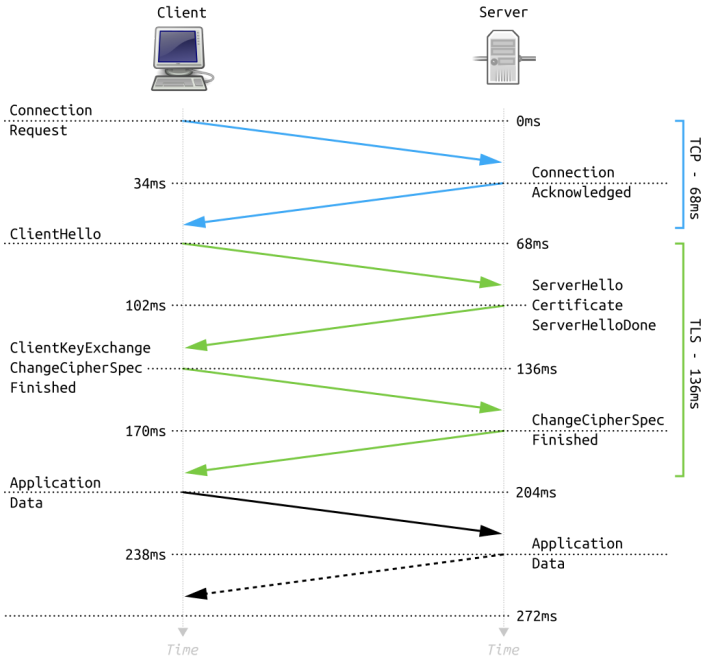

Fig. 31. Handshake TLS. [33].

El cálculo de las claves se realiza con el método de intercambio de claves Diffie-Hellman. (Ver figura 32).

5.3 Diffie-Hellman

Este protocolo hace que se pueda crear una clave compartida entre dos personas, Alice y Bob, aunque un atacante esté escuchando la conversación.

Para ello, se eligen dos números públicos compartidos, g y p, y un número privado.

A continuación, ambos locutores calculan un número que compartirán. Alice publicará A = g^a módulo p y Bob B = g^b módulo p. La clave compartida se calcula de la misma forma pero cambiando el número g por el que la otra persona acaba de compartir. K = B^a módulo p / K = A^b módulo p.

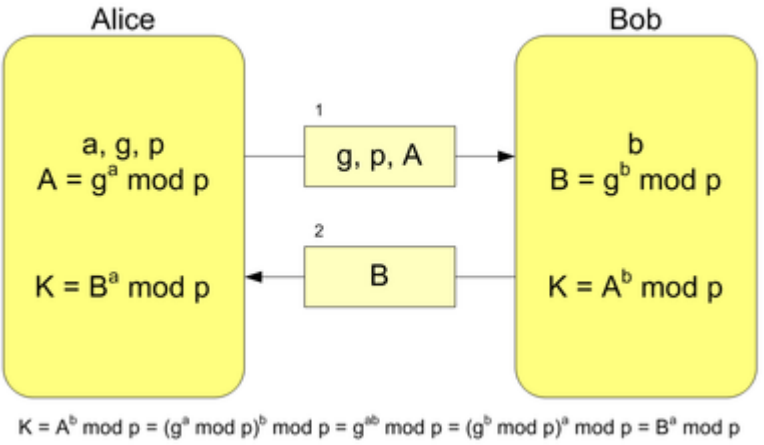

Fig. 32. Intercambio de claves usando Diffie-Hellman. [34].

La seguridad de este sistema se basa en la dificultad de calcular el logaritmo discreto, problema que se considera intratable computacionalmente siempre que se cumplan ciertas características.

5.4 Funcionamiento

Hasta la versión 1.2 del protocolo TLS se exige que se quiten los *bytes* que estén a cero al principio del *premaster secret*. Así, según el tamaño del módulo utilizado en Diffie-Hellman, algunos cálculos serán más rápidos que otros, originando que el atacante pueda adivinar los *bits* más significativos de la clave de Diffie-Hellman.

Para ello, el atacante debe capturar el *handshake* entre el cliente y el servidor. Como el servidor reutiliza las claves, inicia un *handshake* con este con una clave $g^r * g^a$, siendo r un número aleatorio. De esta forma, el *premaster secret* será $(g^r * g^a)^b = g^{rb} * g^{ab}$. El atacante puede calcular el primer término, por lo que solo queda el segundo que es la clave Diffie-Hellman. (Ver figura 33).

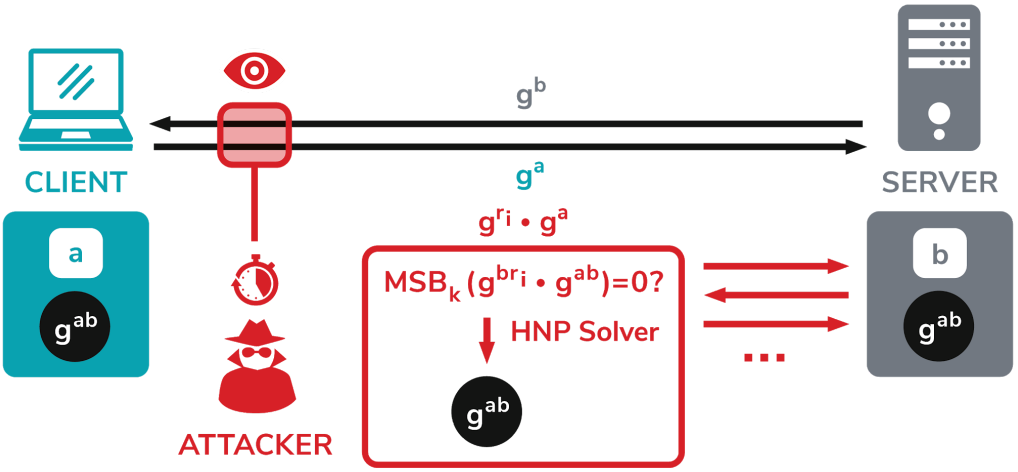

Fig. 33. Funcionamiento de Raccoon Attack. [35].

Para cada handshake se mide el tiempo, sabiendo que los secretos que empiecen por 0 obtendrán una respuesta más rápida. Cuando se tengan las mediciones suficientes, se podrá calcular g^{ab} con el problema del numero oculto y se podrá descifrar toda la información.

Las mediciones de tiempo se pueden realizar tanto del lado del servidor como del cliente. Para el servidor habría que realizar la medición cuando recibe el *ClientKeyExchange message* y devuelve el *Finished message*. Para el cliente sería prácticamente igual, calcular el tiempo que tarda en leer el *ServerKeyExchange message* y envía el *Finished message*.

5.5 Contramedidas

Como se ha dicho anteriormente, el ataque es más teórico que práctico por la dificultad de medir exactamente el tiempo entre los mensajes y encontrar la configuración del servidor correcta. Aún así, se pueden adoptar las siguientes medidas.

- Deshabilitar la opción de reutilizar claves de Diffie-Hellman.
- El ataque solo afecta cuando se utiliza el intercambio de claves Diffie-Hellman normal, pues los que utilizan *Elliptic Curve Diffie Hellman* (ECDH) no son vulnerables.
- Actualizar TLS a la versión 1.3, pues no se ve afectado.

Capítulo 6 Conclusiones

El aumento en el mercado de las líneas VoIP y la adopción de este tipo de servicios por parte de las empresas hace que se expongan cada vez más a que se utilicen las vulnerabilidades que puedan tener programas como Asterisk si se deja a un lado la seguridad y una correcta configuración o no se aplican actualizaciones periódicamente.

Hay que destacar que a pesar de que Asterisk por defecto no cifra las comunicaciones, dependiendo del programa que hará de *softphone*, es posible que la aplicación sí las cifre pero lo haga en una versión de pago. Sin embargo, lo más probable es que en una implementación de una central telefónica de este tipo se usen teléfonos IP por lo que se dependerá de la configuración de estos.

Así, la intención de este trabajo es incentivar a la configuración, mantenimiento y actualización de este o cualquier otro tipo de servicio informático para evitar en la medida de lo posible cualquier ataque que se pueda sufrir, ya sea filtraciones de datos, alguna estafa o cualquier otro ataque.

En el futuro se pretende continuar trabajando y analizando aspectos que no ha sido posible implementar debido a la situación sanitaria, como pudiera ser el uso de teléfonos IP o una conexión con una red telefónica que permita realizar o recibir llamadas del exterior. También probar otras soluciones existentes para comprobar las diferencias con el programa elegido.

Capítulo 7 Conclusions

The increase in the market for VoIP lines and the adoption of this type of service by companies means that they are increasingly exposed to the vulnerabilities that programmes such as Asterisk may have if security and proper configuration are left to one side or updates are not applied periodically.

It should be noted that although Asterisk does not encrypt communications by default, depending on the softphone software, it is possible that the application does encrypt communications but does so in a paid version. However, it is most likely that in a PBX implementation of this type, IP phones will be used, so it will depend on the configuration of these phones.

Thus, the intention of this work is to encourage the configuration, maintenance and updating of this or any other type of IT service in order to avoid, as far as possible, any attack that may be suffered, whether it be data leaks, fraud or any other type of attack.

In the future, we intend to continue working and analysing aspects that have not been possible to implement due to the sanitary situation, such as the use of IP telephones or a connection to a telephone network that allows making or receiving calls from the outside. We also intend to test other existing solutions to check the differences with the chosen programme.

Capítulo 8 Presupuesto

8.1 Coste hardware

TABLA II

Hardware necesario.

Nombre	Precio/unidad
Ordenador	450€
Teléfono IP	35€
Servidor	1000€
Router	500€

Nota: Se indica el precio estimado para un PC de oficina, teléfono, servidor y router sencillo. Para mejores prestaciones el precio puede aumentar considerablemente.

8.2 Coste software

TABLA III

Software necesario

Nombre	Precio/unidad
Softphone	0€
Asterisk	0€
Fail2Ban	0€
SIPVicious	0€
Oracle VM VirtualBox	0€

Nota: Software utilizado en este trabajo.

8.3 Coste recursos humanos

TABLA IV

Coste humano

Tarea	Jornadas 8h.
Análisis de vulnerabilidades	25
Instalación y configuración centralita telefónica	20
Configuración firewall	20
Análisis de vulnerabilidades TLS	20
Realización del informe	15

Nota: se estima un sueldo de 50€/día.

8.4 Coste total

TABLA V

Coste total

Tipos	Total
Coste del hardware	2470€
Coste del software	0€
Coste humano	5000€
Total	7470€

Nota: Precio estimado del trabajo realizado.

Capítulo 9 Apéndice 1: Código e instalación

Para poder realizar pruebas en un entorno en local es necesario crear las máquinas virtuales en el programa Oracle VM VirtualBox con los sistemas operativos indicados en el capítulo 3. En dicho capítulo también se encuentran los comandos para instalar Asterisk y el softphone Blink.

Los comandos para poder instalar Fail2Ban se especifican en el capítulo 4.2.

En el repositorio de GitHub titulado Junior808/TFG_CyCSC [36] se encuentran los ficheros de configuración de Asterisk y de Fail2Ban modificados durante el desarrollo del trabajo, siendo estos:

- `cdr.conf`
- `cel.conf`
- `sip.conf`
- `extensions.conf`
- `jail.local`

Además, se incluye una guía en el fichero `README.md` con aspectos a tener en cuenta.

Bibliografía

[1] "H.323: Sistemas de comunicación multimedia basados en paquetes", Itu.int, 2021. [Online]. Disponible: https://www.itu.int/rec/T-REC-H.323/es [Visitado: 25- Abril- 2021].

[2] "Ministerio de Asuntos Económicos y Transformación Digital - Información de cobertura", avancedigital.mineco.gob.es, 2021. [Online]. Disponible: https://avancedigital.mineco.gob.es/banda-ancha/cobertura/Paginas/informacion-cobertura.aspx [Visitado: 26- Abril- 2021].

[3] M. Spencer, Asterisk. Alabama: Digium, 1999.

[4] "RFC8446 - TLS", RFC8446, 2021. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc8446 [Visitado: 26- Abril- 2021].

[5] "VoIP telephone lines in the U.S. 2010-2018 | Statista", Statista, 2021. [Online]. Disponible: https://www.statista.com/statistics/615387/voip-telephone-lines-in-the-us/ [Visitado: 27- Abril- 2021].

[6] "VoIP Industry Statistics for 2020 | Ideacom® NC", Ideacomnc.com, 2021. [Online]. Disponible: https://ideacomnc.com/voip-industry-statistics-for-2020-infographics/ [Visitado: 27- Abril- 2021].

[7] Issabel. Community, 2017.

[8] Elastix. 3CX, 2006.

[9] "Home ⋆ Asterisk", Asterisk, 2021. [Online]. Disponible: https://www.asterisk.org/ [Visitado: 23- Febrero- 2021].

[10] "Asterisk : Security vulnerabilities", Cvedetails.com, 2021. [Online]. Disponible: https://www.cvedetails.com/vulnerability-list/vendor_id-6284/Asterisk.html [Visitado: 27- Abril- 2021].

[11] Oracle VM VirtualBox. Alemania: Innotek, 2007.

[12] "IaaS", IaaS ULL, 2021. [Online]. Disponible: https://iaas.ull.es/ [Visitado: 23- Febrero- 2021].

[13] Linux. Linux Org, 1991.

[14] Ubuntu. Canonical, 2004.

[15] Kali Linux. Offensive Security, 2013.

[16] Blink. AG Projects, 2009.

[17] G. Combs, Wireshark. Wireshark, 1998.

[18] G. Lyon, Nmap. 1997.

[19] SIPVicious. Enable Security, 2015.

[20] C. Jaquier, Fail2Ban. 2004.

[21] R. Russell, Iptables. Rusty Russell, 1998.

[22] "RFC768 UDP", RFC768, 2021. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc768 [Visitado: 15- Mayo- 2021].

[23] "RFC793 TCP", RFC793, 2021. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc793 [Visitado: 15- Mayo- 2021].

[24] "G.711: Modulación por impulsos codificados (MIC) de frecuencias vocales", Itu.int, 2021. [Online]. Disponible: https://www.itu.int/rec/T-REC-G.711/es [Visitado: 15- Mayo- 2021].

[25] Windows. Redmond, Washington, Estados Unidos: Microsoft Corporation, 1985.

[26] Mac OS. Cupertino, California, EE.UU.: Apple Inc., 2001.

[27] "RFC3261 SIP", RFC3261, 2021. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc3261 [Visitado: 15- Mayo- 2021].

[28] "asterisk/asterisk", GitHub, 2021. [Online]. Disponible: https://github.com/asterisk/asterisk/blob/master/contrib/scripts/ast_tls_cert [Visitado: 10- Marzo- 2021]

[29] "FIPS 46-2 - (DES), Data Encryption Standard", Web.archive.org, 2021. [Online]. Disponible: https://web.archive.org/web/20040410171758/http://www.itl.nist.gov/fipspubs/fip46-2.htm [Visitado: 15- Mayo- 2021].

[30] "RFC1889 RTP", RFC1889, 2021. [Online]. Disponible: https://datatracker.ietf.org/doc/html/rfc1889 [Visitado: 15- Mayo- 2021].

[31] R. Merget, M. Brinkmann, N. Aviram, J. Somorovsky, J. Mittmann and J. Schwenk, "Raccoon Attack", 2020.

[32] W. Diffie and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, 1976. Disponible: 10.1109/tit.1976.1055638.

[33] The Tango! Desktop Project,TLS Handshake. 2015.

[34] B. Stern, Diffie Hellman. 2006.

[35] R. Merget, M. Brinkmann, N. Aviram, J. Somorovsky, J. Mittmann and J. Schwenk, 2020.

[36] "Junior808/TFG_CyCSC", GitHub, 2021. [Online]. Disponible: https://github.com/Junior808/TFG_CyCSC [Visitado: 06- Junio- 2021].