Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Sistema de consulta y notificación de alertas de seguridad mediante VoIP en Raspberry Pi

Autor: Ismael Narváez Berenjeno

Tutor: Antonio José Estepa Alonso

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Sistema de consulta y notificación de alertas de seguridad mediante VoIP en Raspberry Pi

Autor:
Ismael Narváez Berenjeno

Tutor:
Antonio José Estepa Alonso
Profesor titular

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2015

Proyecto Fin de Carrera: Sistema de consulta y notificación de alertas de seguridad mediante VoIP en Raspberry Pi

Autor: Ismael Narváez Berenjeno

Tutor: Antonio J. Estepa Alonso

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2015

El Secretario del Tribunal

A mi familia, amigos y profesores

Agradecimientos

Mi primera mención de agradecimiento va dirigida a mi familia. Sin su apoyo nada podía haber sido posible.

También quiero agradecer a mis amigos su apoyo, por estar siempre ahí para sacarme una sonrisa.

Por último, quiero dar las gracias a mi tutor Antonio José Estepa Alonso, por sus consejos y su ayuda en todo el proceso de desarrollo.

Resumen

Este documento recoge el proceso de desarrollo y prueba de un IDS, que usa un sistema de notificación y consulta de alertas, basado en llamadas VoIP. Todo ello ha sido instalado en un sistema de bajo consumo y coste, como es la Raspberry Pi B+.

El sistema tiene dos funciones principales:

- **Función de notificación**: Analiza el tráfico que le llega, y si detecta una amenaza, **notifica** al administrador mediante una llamada telefónica.
- **Función de consulta**: El administrador puede **consultar** la existencia de alertas de una cierta prioridad, realizando una llamada al sistema.

Estas tareas son realizadas gracias al trabajo conjunto de dos módulos:

- **Snort**, es el encargado de analizar el tráfico en busca de ataques.
- **Asterisk**, se encarga de realizar (**notificación**) o cursar (**consulta**) las llamadas telefónicas.

Se ha simulado una red convencional, para probar su funcionamiento.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

"El único sistema seguro es aquél que está apagado en el interior de un bloque de hormigón protegido en una habitación sellada rodeada por guardias armados"
— Gene Spafford-

1.1 Motivación

La motivación de este proyecto reside en conseguir que cualquier usuario, ya sea especializado o no, pueda proteger su red mediante un IDS, gracias a un sistema de **notificación** y **consulta** basado en llamadas telefónicas. En la Figura 1-1 podemos observar ambas funciones.

Este tipo de notificación y consulta, facilita la tarea de administración, evitando la necesidad de consultar un registro de logs, ya sea a través de ficheros de registro o mediante interfaces gráficas.

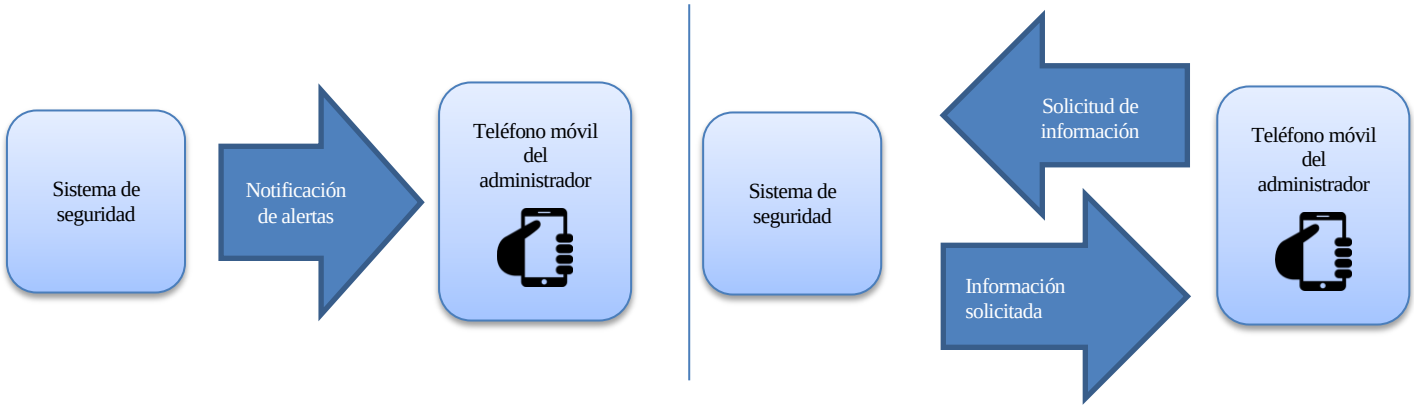


Figura 1-1: Funciones de notificación y consulta del sistema

1.2 Objetivos

Los objetivos de este proyecto son:

- **Reducir la complejidad** de proteger una red, permitiendo realizar la tarea a cualquier usuario.
- **Conseguir estar informados permanentemente,** gracias al teléfono móvil.
- **Implementar el sistema en un dispositivo de bajo coste, consumo y tamaño**, que pueda ser comprado y usado por cualquier usuario.

1.3 Estructura de la memoria

La memoria está estructurada en siete capítulos:

- En el primer capítulo, se hace una introducción del tema que se desarrollará en esta memoria, los objetivos y la estructura de la misma.
- En el segundo capítulo, se analiza las tecnologías existentes para la notificación y consulta de alertas de seguridad.
- En el tercer capítulo, se expone la planificación seguida en la realización del proyecto.
- En el cuarto capítulo, se resumen las herramientas necesarias para crear el sistema.
- En el quinto capítulo, se detalla cómo funciona el sistema en los dos modos (**notificación** y **consulta**), los distintos módulos que lo forman y cómo se coordinan.
- En el sexto capítulo, se muestran las distintas pruebas que se han realizado al sistema para comprobar su correcto funcionamiento.
- En el último capítulo, se exponen las conclusiones alcanzadas al realizar el proyecto, y los puntos en los que el sistema puede evolucionar.

Por último se adjuntan una serie de anexos, que muestran los pasos seguidos a la hora de crear el sistema.

2 ESTADO DEL ARTE

"If you reveal your secrets to the wind, you should not blame the wind for revealing them to the trees"

-Kahlil Gibran-

2.1 Sistemas para la consulta de alertas

2.1.1 BASE

BASE [1] nos ofrece una interfaz web fácil de usar, funcional y con diversos idiomas. Es la más popular de todas las herramientas de consulta, y cuenta con más de 215.000 descargas.

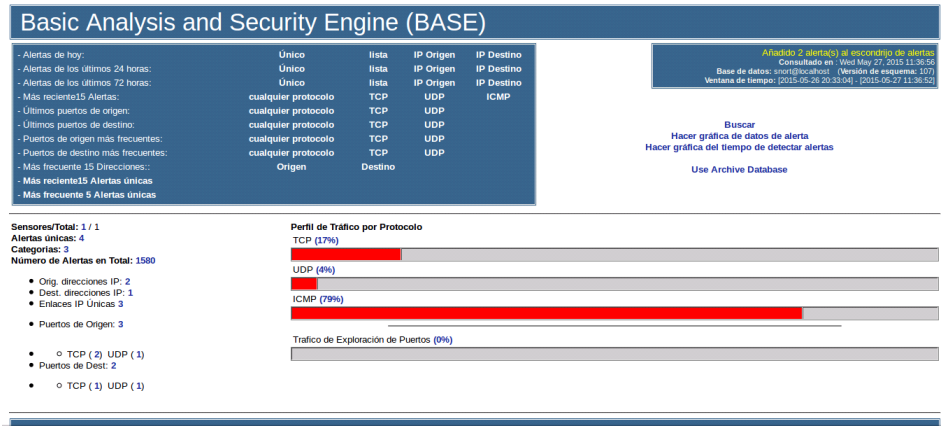


Figura 2-1: Portada BASE

Nos ofrece una portada con el tráfico ordenado por protocolos.

Las **ventajas** de usar BASE son:

- Fácil instalación.
- Permite exportar datos mediante email.
- Muchos tipos de gráficas diferentes: pastel, barras, líneas y mapamundi.
- Gráficas con clasificaciones muy diversas: por fecha, tiempo, puertos, firmas, países. etc.
- Puede eliminar alertas de la base de datos.

Los **inconvenientes**:

- No posee un desarrollo activo desde 2003.
- No posee herramientas de clasificación automáticas de alertas.

2.1.2 Snorby

Snorby [2] ofrece una portada con muchos efectos visuales, que la convierten en una herramienta agradable de usar.

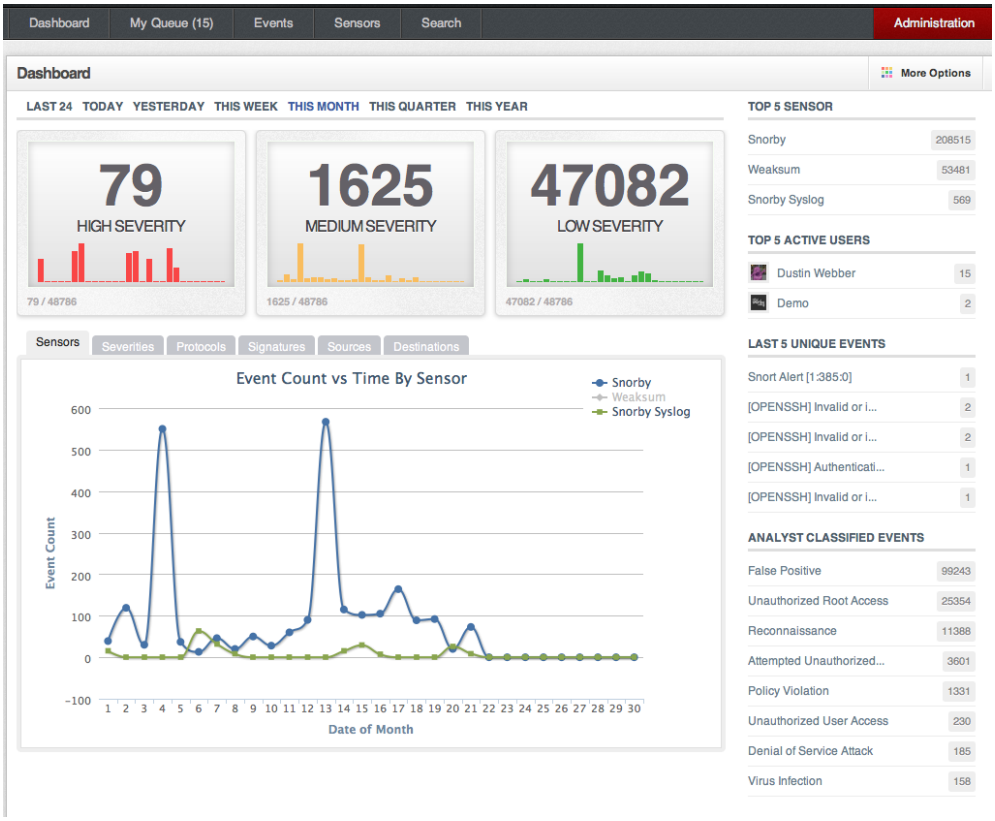


Figura 2-2: Portada de Snorby

Posee una serie de características únicas, como:

- Portada con mucha información, lo que permite ver de un simple vistazo muchos parámetros diferentes.
- Clasifica automáticamente las alertas, en severidad alta, media o baja.
- Permite exportar datos mediante email o xml, además de poder generar informes PDF.

Pero respecto a BASE, está más limitado en la realización de gráficas:

- Solo permite gráficas de tipo pastel y mediante líneas.
- No permite gráficas con datos sobre puertos o países.

2.2 Sistemas para la notificación de alertas

2.2.1 Swatch

Swatch permite detectar eventos de Snort mediante la monitorización de Syslog. Podemos configurarlo [3] para que envíe notificaciones mediante email cuando se produce una alerta. El problema radica en que los

mensajes de Syslog son de una línea, y las alertas de Snort tienen varias, por lo que algunas veces puede producir el envío de notificaciones erróneas, enviando un email por cada línea.

2.2.2 OpenNMS

OpenNMS [4] es una aplicación de monitorización y administración de redes de código abierto, que permite integrarse con Snort [5].

OpenNMS captura los traps generados por Snort, y envía notificaciones al administrador vía email [6].

2.3 Conclusiones

Estas soluciones nos aportan mucha información, pero necesitan usuarios especializados que sean capaces de sintetizar los datos, tanto de las **notificaciones** vía email, como de las **consultas** en las interfaces web.

Además, los sistemas de consulta restan movilidad al usuario, ya que deben encontrarse en un puesto fijo para hacer uso de ellos.

Estas deficiencias son las que trata de solucionar este proyecto.

3 PLANIFICACIÓN

A goal without a plan is just a wish.

- Antoine de Saint-Exupéry-

3.1 Tareas

Las tareas realizadas en el desarrollo del proyecto han sido las siguientes:

Tabla 3-1: Tareas del proyecto

ID	Nombre	Descripción	Fecha de inicio	Fecha de fin
T1[1]	Planteamiento de ideas	Recolección y selección de ideas para el proyecto	11/11/14	1/12/14
T2	Búsqueda de información	Búsqueda de fuentes de información que ayuden en el desarrollo del sistema	2/12/14	15/01/15
T3	Planificación	Planificación temporal de las distintas tareas a realizar	16/01/15	23/01/15
T4	Desarrollo de componentes	Instalación, configuración y prueba de los distintos componentes que forman el sistema	10/02/15	1/05/15
T4.1	Instalación y configuración del IDS	Instalación y configuración del sistema de detección de intrusos	10/02/15	20/02/15
T4.2	Pruebas del IDS	Comprobar que el sistema se ajuste al funcionamiento deseado	23/02/15	27/02/15
T4.3	Solución de problemas	Periodo para solucionar los fallos en la configuración del IDS detectados	2/03/15	10/03/15
H1[2]	IDS	Hito. Conseguir IDS funcional. Creación de backup	11/03/15	11/03/15

[1] Tarea número 1 en el desarrollo
[2] Hito número 1 en el desarrollo

T4.4	Instalación de interfaces gráficas	Instalación de distintas interfaces gráficas que permiten realizar una consulta tradicional de las alertas	11/03/15	1/04/15
T4.5	Instalación y configuración de Asterisk	Instalación y configuración de la centralita VoIP que realizará las llamadas telefónicas	2/04/15	16/04/15
T4.6	Pruebas centralita VoIP	Pruebas de funcionamiento de la centralita	17/04/15	24/04/15
T4.7	Solución de problemas	Periodo para solucionar los fallos en la configuración de la centralita detectados	27/04/15	1/05/15
H2	Centralita VoIP	Hito. Conseguir centralita VoIP funcionando correctamente.	4/05/15	4/05/15
T5	Integración de componentes	Fase de unión de los distintos componentes desarrollados	4/05/15	3/06/15
T5.1	Desarrollo de scripts	Creación de los scripts que procesarán las alertas	4/05/15	8/05/15
T5.2	Pruebas del conjunto	Búsqueda de fallos en el sistema	11/05/15	19/05/15
T5.3	Solución de problemas	Periodo para solucionar los problemas encontrados en la fase anterior	20/05/15	3/06/15
H3	Sistema funcional	Hito. Sistema totalmente funcional. Creación de backup.	4/06/15	4/06/15
T6	Documentación del proceso	Recolección de toda la información usada en el desarrollo	10/02/15	3/06/15
T7	Redacción	Elaboración de la memoria del proyecto	4/06/15	30/06/15

3.2 Diagrama de Gantt

Con el objetivo de ofrecer una visión más clara de la distribución del tiempo, se usará un diagrama de Gantt [7]:

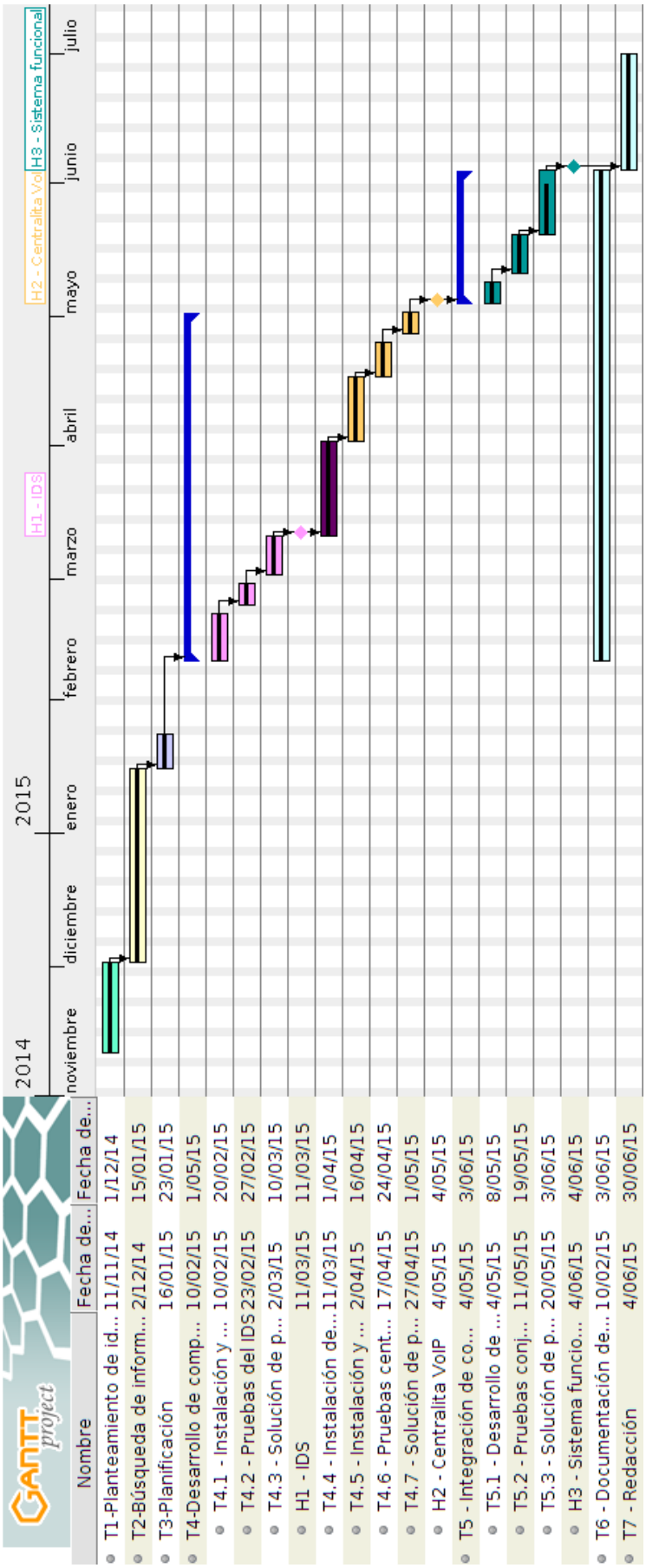


Figura 3-1: Diagrama de Gantt

4 HERRAMIENTAS UTILIZADAS

Simple is good

-Jim Henson-

Para conseguir que el sistema desempeñe sus funciones, se necesitan una serie de herramientas trabajando conjuntamente. En este apartado se explicarán cuales son, y el motivo de su elección.

- **Snort** [8]**:** Sistema detección de intrusos en red. Analiza los paquetes que llegan al sistema, en busca de amenazas. Posee una gran base de datos con patrones de ataques, que se actualiza continuamente. Además, ofrece la posibilidad de que la comunidad contribuya en la búsqueda de nuevos ataques. Estas características lo convierten en un IDS actualizado y robusto.
- **AWK** [9]**:** Lenguaje para el procesado de datos en formato texto. En este caso lo usaremos para procesar las alertas generadas por Snort, y extraer los campos de interés.
- **Text2wave** [10]: Transforma los campos extraidos mediante AWK, en formato audio, para que puedan ser enviados a través de una llamada.
- **SoX** [11]: Editor de audio digital que adecua el audio generado por Text2wave, a Asterisk.
- **Asterisk** [12]**:** Programa de software libre que ofrece funciones de central telefónica. Realiza las llamadas con las **notificaciones** y cursa las peticiones de **consulta**. En nuestro caso necesitamos un control total, y Asterisk nos ofrece esto frente a otras soluciones como Elastix [13] o FreePBX [14].
- **MySQL** [15]**:** Sistema de gestión de bases de datos relacional, multiusuario y multihilo, donde almacenar las alertas generadas por Snort. Recurriremos a esta herramienta en el caso de que se quiera hacer uso de un sistema tradicional de consulta.
- **Apache** [16]: Servidor HTTP multiplataforma con una arquitectura modular. Al igual que con MYSQL, se deberá usar únicamente en el caso de que se opte por incorporar un sistema tradicional de consulta.
- **Telegram** [17]: Servicio de mensajería por Internet. Su seguridad, y la gran integración con diferentes plataformas, han sido los factores determinantes para su elección frente a sus competidores en el mercado.

5 FUNCIONAMIENTO DEL SISTEMA

When you want to know how things really work, study them when they're coming apart

-William Gibson, Zero History-

Como hemos expuesto anteriormente, el sistema posee dos funciones principales, **notificación** y **consulta** de alertas. En este apartado explicaremos como funciona en ambos casos.

5.1 Notificaciones de alertas

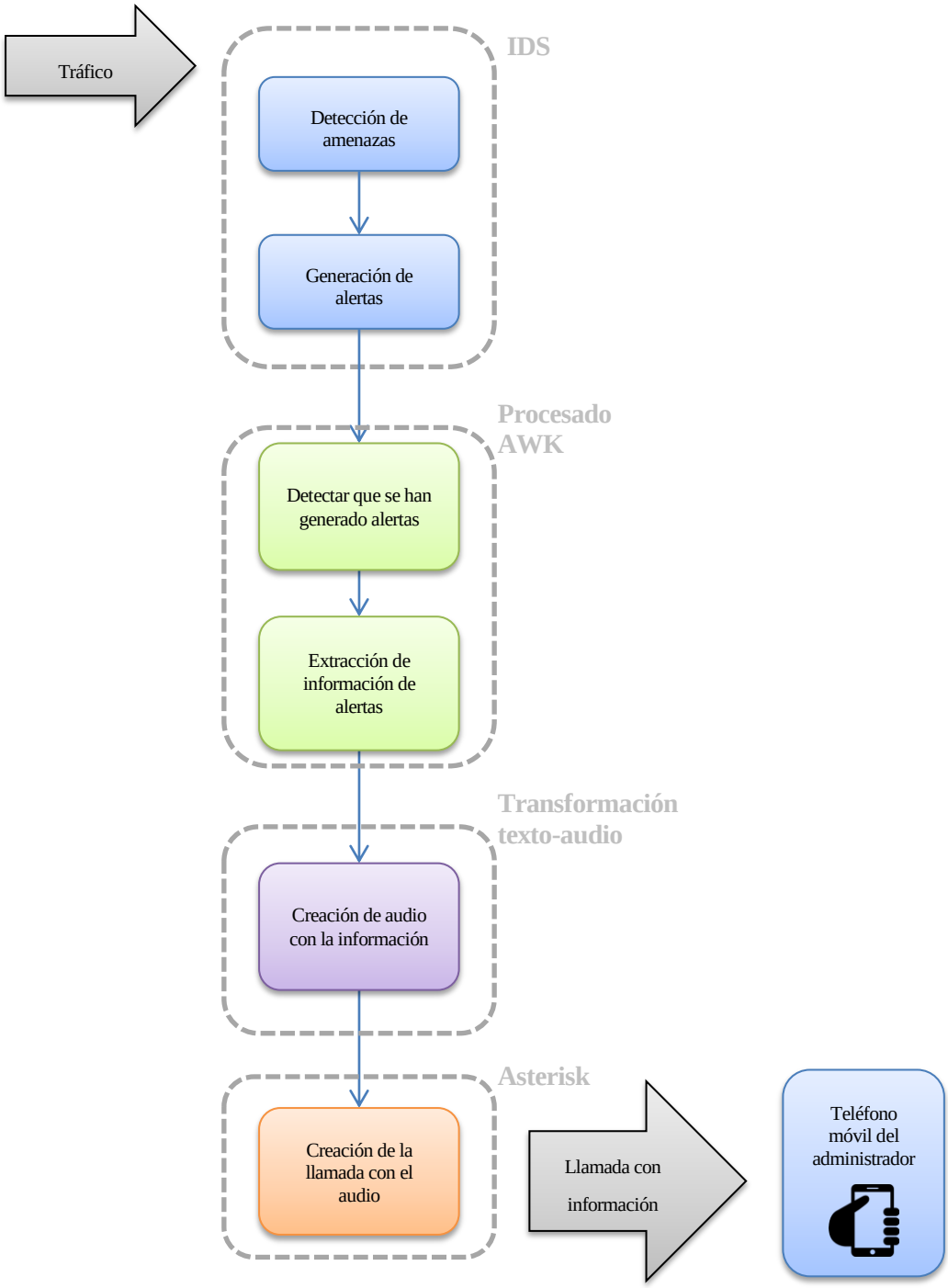


Figura 5-1: Funcionamiento de notificación del sistema

En la Figura 5-1 se pueden distinguir cuatro tareas o módulos principales:

1. El tráfico proveniente de la red es analizado por el IDS. Si se encuentra alguna amenaza, generará una alerta, que será almacenada en un fichero de registro.
2. Detectamos si se han generado alertas. En caso afirmativo, extremos los campos que contienen la información. En este caso solo nos interesa el campo mensajes[3], que se corresponde con una descripción breve de la amenaza.

[3] Para saber como es una alerta y el mensaje que contiene, véase el Anexo III.

3. Transformamos la información obtenida de los mensajes, en un archivo de audio.
4. Generamos una llamada con Asterisk, con el audio creado.

Estos cuatro módulos se encuentran implementados en un mismo sistema, una Raspberry Pi B+ [18].

En los siguientes apartados se profundizará más en cada uno de los módulos.

5.1.1 IDS

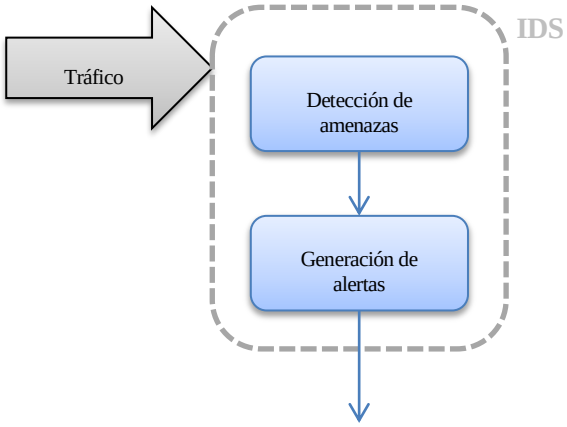


Figura 5-2: Módulo IDS del sistema

La arquitectura interna del IDS usado, Snort, es la siguiente [19]:

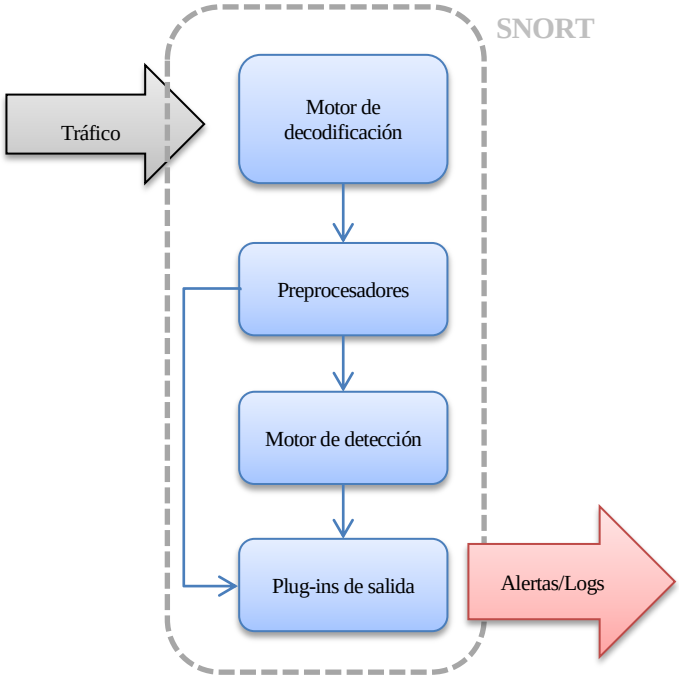


Figura 5-3: Arquitectura interna de Snort

- **Motor de decodificación**: Extrae información importante de los paquetes que le llegan, y la guarda en estructuras de datos, facilitando su posterior procesamiento.

- **Preprocesadores:** Permiten añadir distintos módulos (plugins), aumentando las funcionalidades. Pueden lanzar alertas, clasificar, descartar o modificar un paquete, antes de enviarlo al motor de detección, que cuenta con un alto coste computacional.
- **Motor de detección**: Analiza el contenido del paquete utilizando información de los módulos anteriores, y la compara con los patrones de las reglas. Es el corazón de Snort.
- **Plugins de salida**: Si se detecta un paquete sospechoso, ya sea por los preprocesadores o por el motor de detección, imprimen una alerta en el formato especificado en el archivo de configuración de Snort.

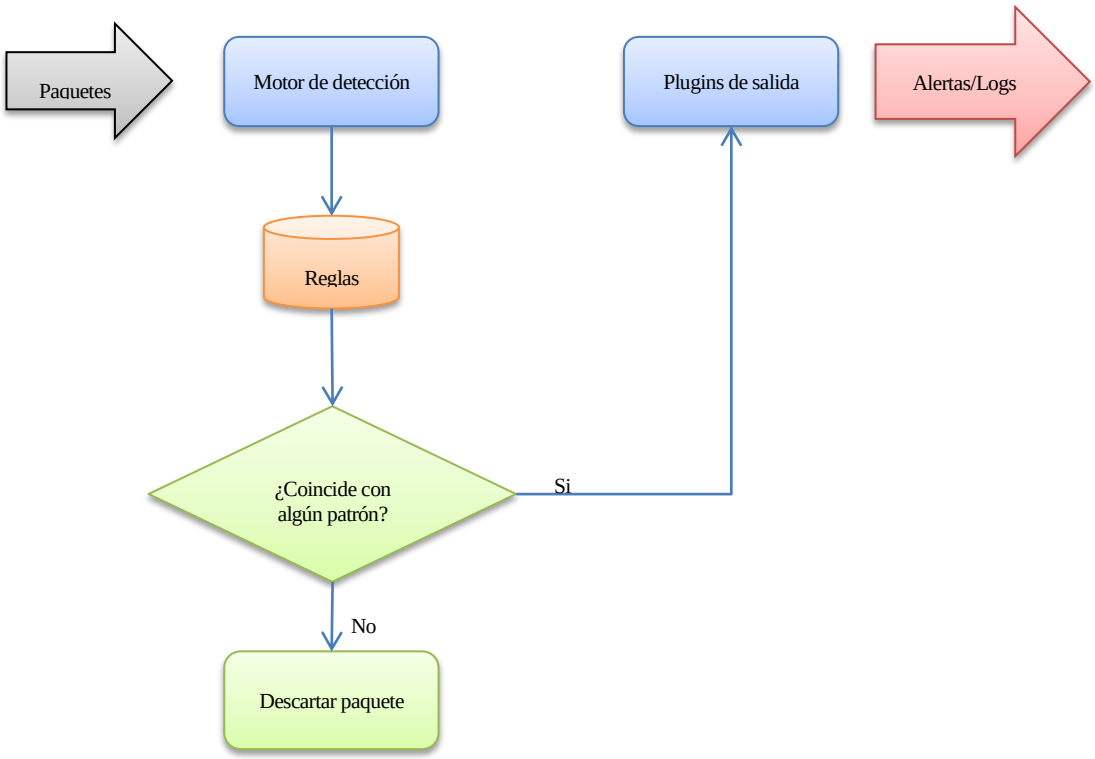


Figura 5-4: Funcionamiento del motor de detección y los plugins de salida

Las alertas generadas por los plugins de salida, se almacenan en un fichero de registro, especificado en el archivo de configuración de Snort. En nuestro caso */var/log/snort/alertas*. Este fichero es de gran importancia, ya que será la fuente de información para el sistema, de donde extraerá las alertas, en los dos modos de funcionamiento.

5.1.2 Procesador AWK y transformación a audio

Figura 5-5: Módulos para el procesado de alertas y transformación a audio

Las dos tareas principales que muestra la Figura 5-5 son realizadas por el script alertas.sh. Su funcionamiento se explica mediante el diagrama de flujo de la Figura 5-6.

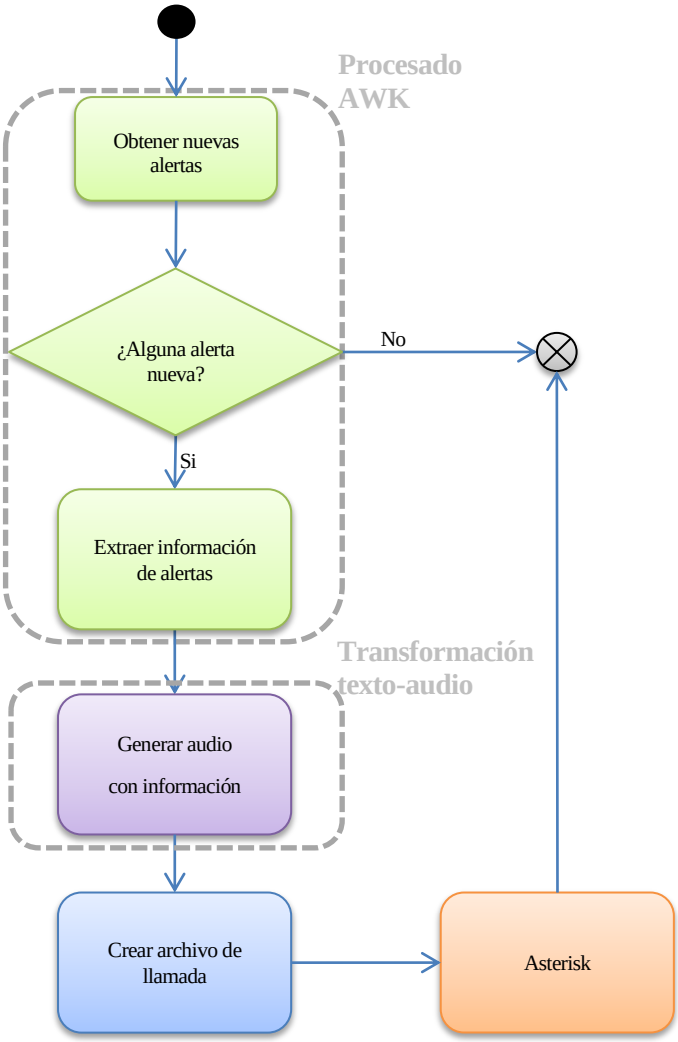


Figura 5-6: Diagrama de procesado AWK y transformación de audio

1. Obtenemos las nuevas alertas de prioridad 0[4] generadas por Snort desde la última ejecución.
2. Si no existe ninguna alerta con esa prioridad, salimos de la ejecución. Por el contrario, si existe alguna, significa que hemos recibido un ataque, por lo que debemos notificarlo:
   - **Visualmente**, apagando la luz de funcionamiento correcto, mediante el script apagaverde.sh, y encendiendo una luz de alerta, ejecutando el script encienderojo.sh.
   - **Con una llamada al administrador,** gracias a los pasos siguientes.
3. Extraemos los mensajes de las alertas y eliminamos los repetidos. Estos mensajes contienen la información de la amenaza.
4. Transformamos la información extraida en un fichero de audio y lo movemos a la biblioteca de sonidos de Asterisk (*/var/lib/asterisk/sounds/en*), para que sea accesible.
5. Creamos y movemos un fichero de llamada[5] al directorio */var/spool/asterisk/outgoing/*, y automáticamente se pasará el control al siguiente módulo, Asterisk.

Como el script comprueba si existen nuevas alertas desde la última ejecución, ejecutarlo continuamente no consume recursos. De esta forma, nos aseguramos que las alertas se detectan inmediatamente. Esta tarea la realiza el script ciclo.sh.

[4] El motivo de usar la prioridad 0 queda reflejado en el Anexo III.
[5] En el apartado c del Anexo IV se muestra el formato del fichero de llamada.

5.1.3 Asterisk

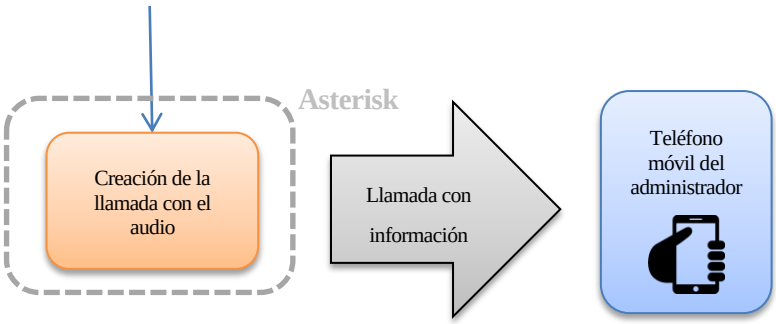


Figura 5-7: Módulo Asterisk

Asterisk es el encargado de generar la llamada, con la información extraída de los pasos anteriores. A la llamada se le debe asignar un contexto de entrada, que no es más que una sección con conjunto de acciones a realizar, como por ejemplo realizar llamada, colgar, etc [20].

En concreto, el contexto utilizado es *[prioridad0]* del fichero /etc/asterisk/extensions.conf.

Para facilitar la comprensión de las acciones usadas en este contexto, se utilizará la Figura 5-8:

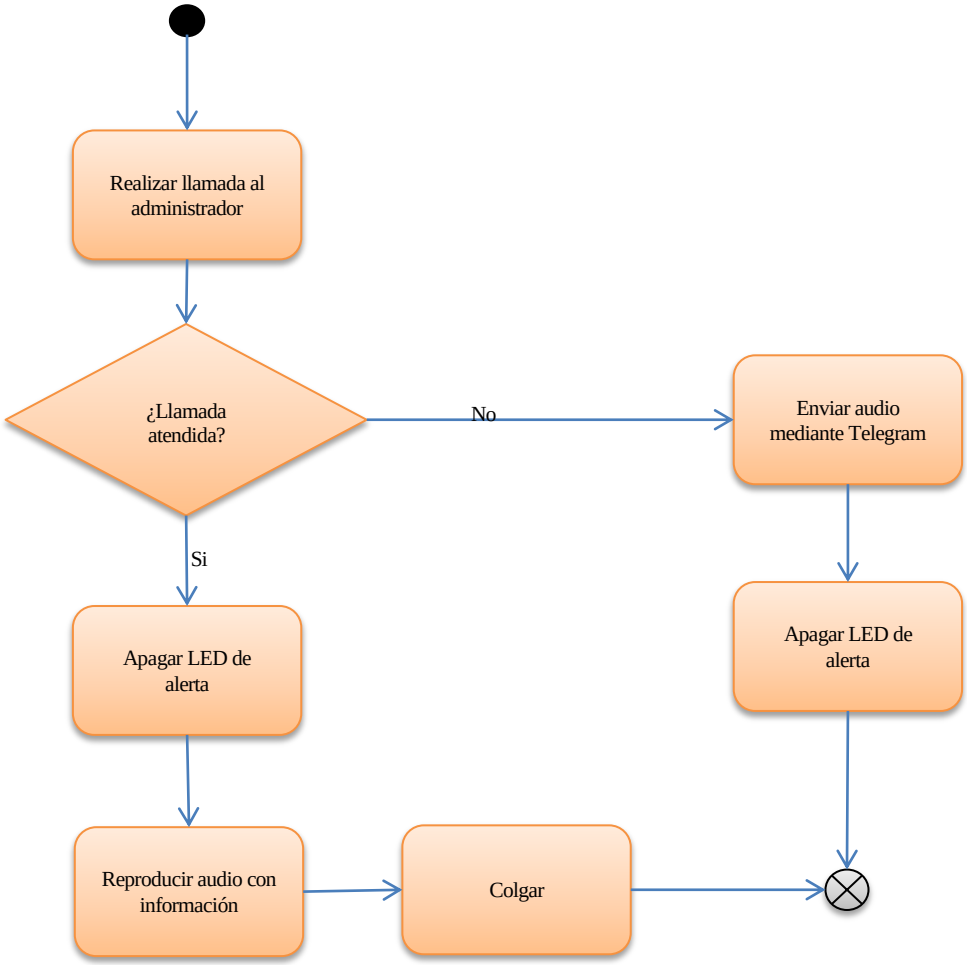


Figura 5-8: Diagrama de funcionamiento de Asterisk

Para manejar la excepción producida cuando falla la llamada, Asterisk incorpora una extensión especial [21].

Si se produce cualquier tipo de fallo, se ejecutará el código que se encuentra en ella[6], enviando por Telegram el archivo de audio que contiene la información, y apagando la alerta visual.

5.2 Acciones desde el teléfono móvil para la consulta de alertas

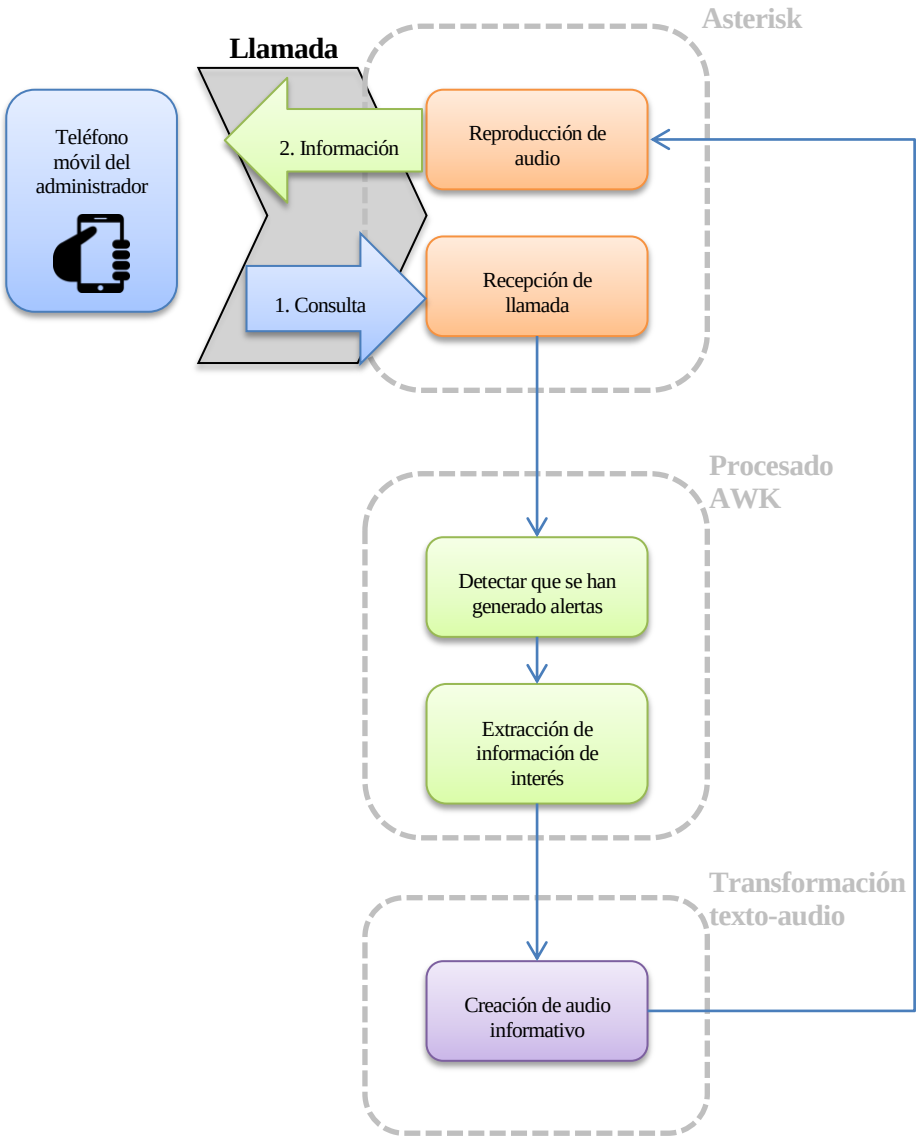


Figura 5-9: Funcionamiento de consulta del sistema

Como muestra la Figura 5-9, el sistema tiene la capacidad de aceptar acciones para la **consulta** de alertas, mediante la realización de llamadas desde el teléfono. Estas llamadas son analizadas por Asterisk, y dependiendo la extensión que posean, se ejecutará una acción u otra. Se han incluido dos:

- Acción 1: Busca la existencia de alertas de prioridad 1 en el fichero */var/log/snort/alertas* y, en caso de que se encuentren, comunica al administrador el número de alertas diferentes que se han producido.

[6] La extensión failed o de fallo, se encuentra en el fichero /etc/asterisk/extensions.conf en el apartado b del Anexo IV.

Para ejecutarla, debemos llamar a la extensión 1@DIRECCIÓN_IP_SISTEMASEGURIDAD.

- Acción 2: Misma acción que el caso anterior, pero con alertas de prioridad 2. Para ejecutarla, debemos llamar a 2@DIRECCIÓN_IP_SISTEMASEGURIDAD.

En ambos casos, si el sistema no encuentra ninguna alerta, el audio informará de que no existe ninguna con esa prioridad.

En la Figura 5-10, se muestra mediante un diagrama de flujo, el funcionamiento con más nivel de detalle.

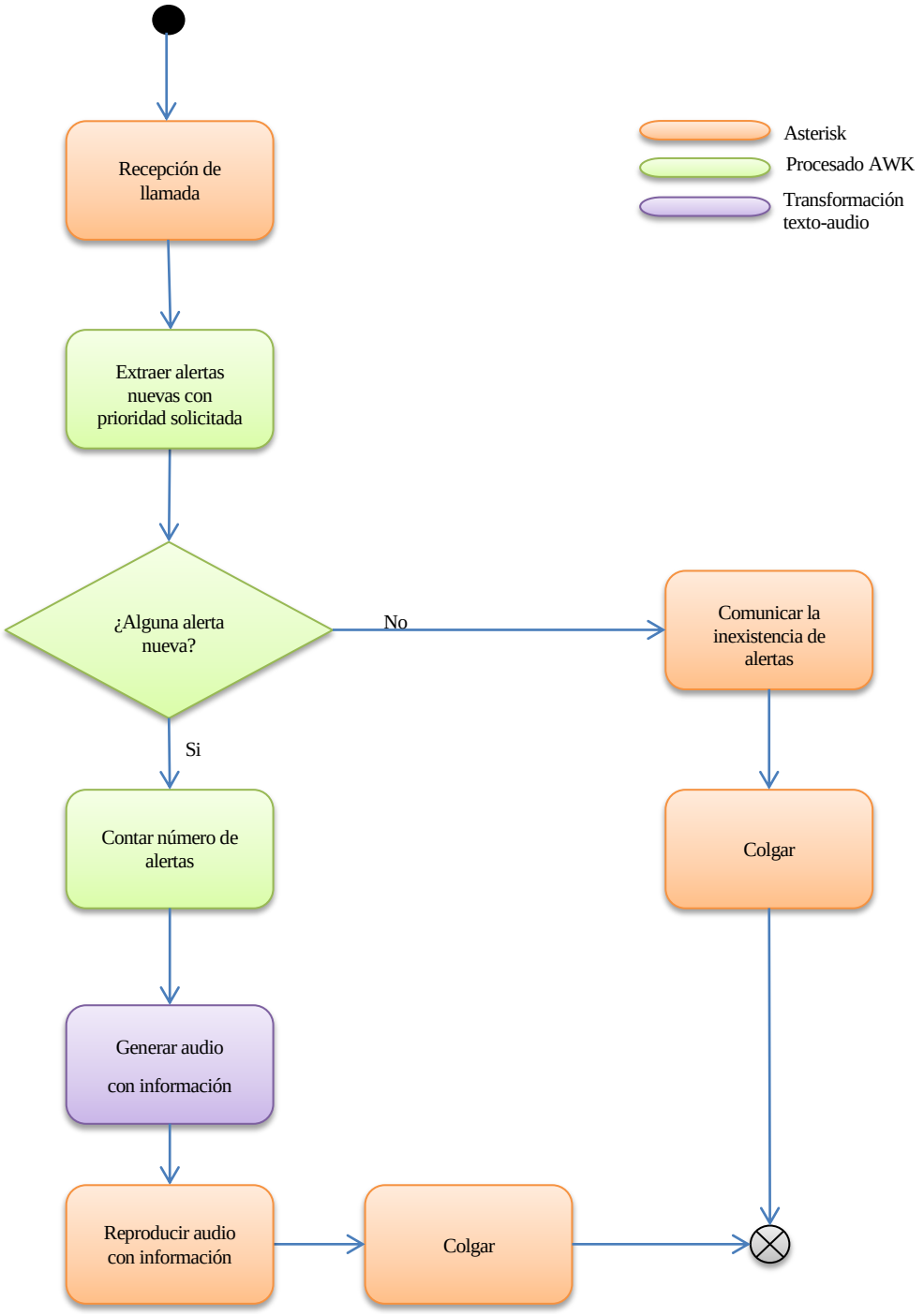


Figura 5-10: Funcionamiento detallado consulta de alertas

El sistema ha sido diseñado para que estas acciones de consulta solo la puedan realizar usuarios autorizados. Están declarados en el fichero /etc/asterisk/sip.conf.

6 PRUEBAS

I have not failed. I've just found 10,000 ways that won't work.

- Thomas A. Edison-

En esta sección se mostrarán las pruebas realizadas al sistema. Para estas pruebas se ha diseñado un escenario, que intenta ser lo más fiel posible a la realidad.

6.1 Escenario de pruebas

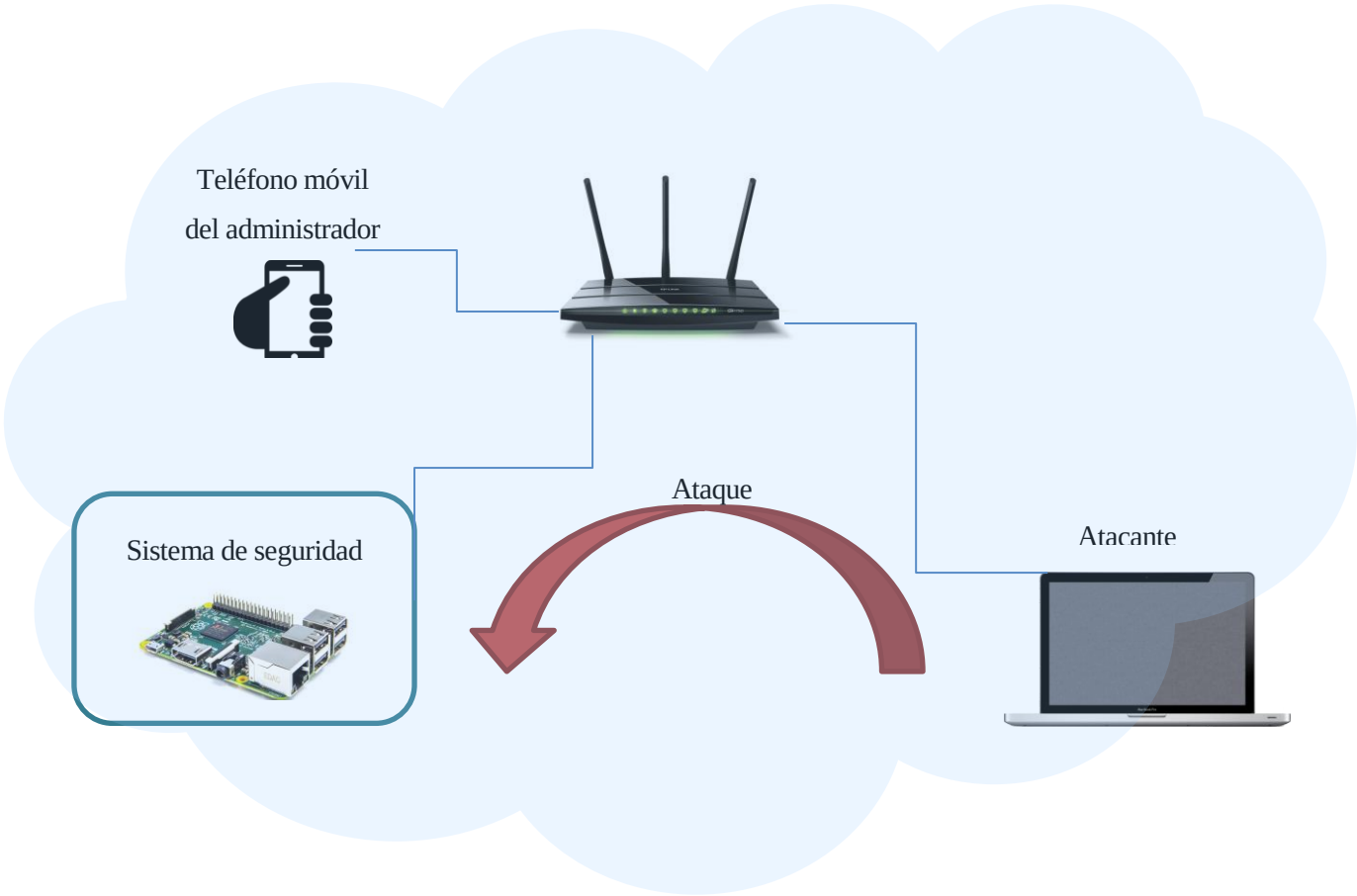


Figura 6-1: Escenario de pruebas

6.1.1 Sistema de seguridad

Estará implementado en una Raspberry Pi B+.Su bajo precio, bajo consumo, pequeño tamaño, y la posibilidad

de usar distribuciones Linux y LEDs (mediante sus GPIOs [22]), la hacen ideal para cumplir los requisitos del sistema.

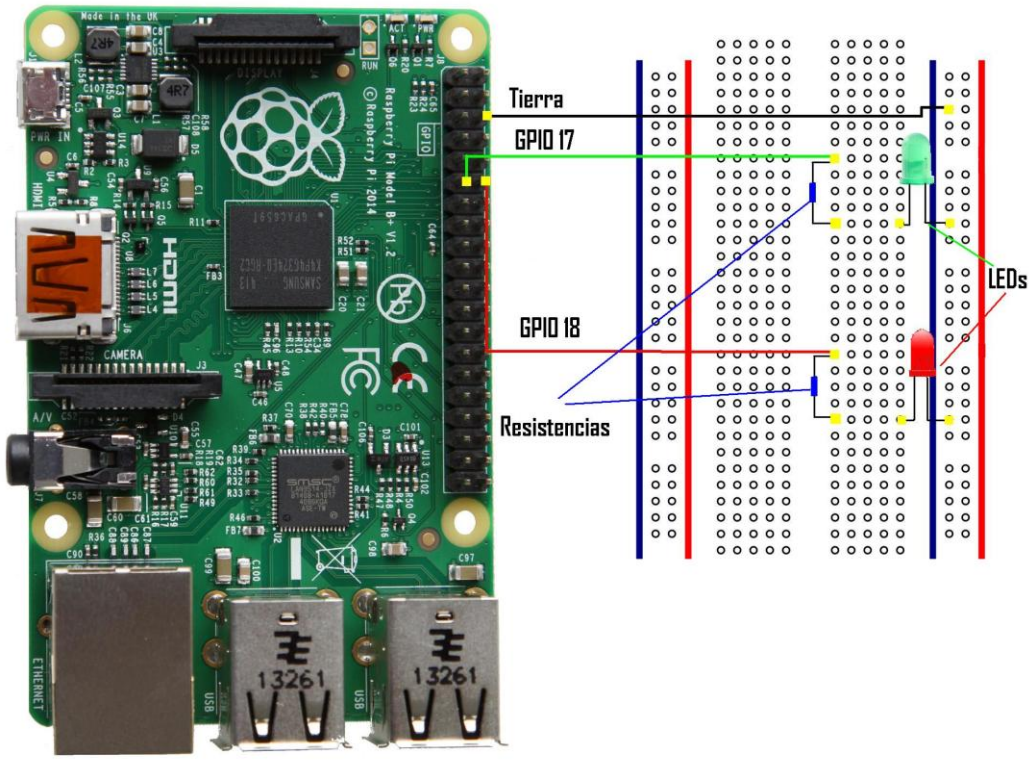


Figura 6-2: Raspberry Pi B+

Usa una distribución basada en Debian Wheezy [23], llamada Raspbian [24].

6.1.2 Teléfono móvil

El administrador está informado continuamente gracias a su teléfono móvil. En este caso, el dispositivo utiliza Android 4.4 [25].

No existe la necesidad de instalar ninguna aplicación en especial para recibir las llamadas VoIP, ya que Android ofrece la posibilidad de realizar llamadas a través de Internet[7].

Debe tener instalado la aplicación Telegram, para recibir las notificaciones en caso de que falle una llamada.

6.1.3 Atacante

Equipo que genera paquetes sospechosos, con el objetivo de que el sistema los detecte y pueda notificar al administrador.

6.2 Pruebas de la función de notificación

Para probar la función de notificación del sistema se realizan dos pruebas diferentes:

1. Prueba sin fallo en la llamada
2. Prueba con fallo en la llamada

[7] Para configurar un dispositivo, véase el apartado d del Anexo IV.

6.2.1 Prueba sin fallo en la llamada

El primer paso para probar el sistema, es enviar un paquete que genere una alerta.

Para ello, podemos crear una regla de Snort [26], que genere una alerta cuando se realiza un ping al sistema:

alert icmp any any -> $HOME_NET any (msg:"Ping a la red";sid: 10000001;rev:1;)

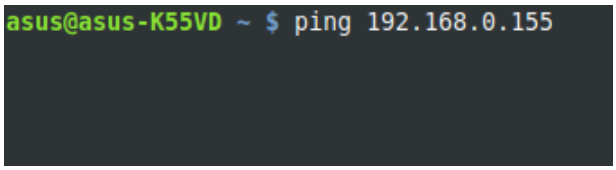


Figura 6-3: Simulación de ataque

Como podemos observar la regla está formada por dos partes:

- Cabecera

[acción] [protocolo] [IPs origen] [Puertos origen] <>/-> [IPs destino] [Puertos destino]

- Opciones

(Opción1:<parámetro>;)

Una vez generada la alerta, el sistema la analizará, extraerá el mensaje *"Ping a la red"* y generara una llamada para informar al administrador.

La demostración de la prueba se puede ver en el siguiente video:

https://www.youtube.com/watch?v=97FAbN4nmLY

6.2.2 Prueba con fallo en la llamada

Misma prueba que la anterior, pero provocando que la llamada falle. El sistema deberá tratar el fallo, y enviar la alerta por otro medio de comunicación diferente:

http://youtu.be/a6AGs81yX0Y

6.3 Pruebas de acciones de consulta desde el teléfono móvil

Se realizarán tres llamadas para probar el funcionamiento del sistema:

1. Llamada a la extensión que analiza las alertas de prioridad 1 existentes.
2. Llamada a la extensión que analiza las alertas de prioridad 2 existentes.
3. Repetición de la última llamada para comprobar como cambia el mensaje cuando no hay alertas.

Estas pruebas están reflejadas en el siguiente vídeo:

http://youtu.be/flnXeZhlXkY

7 CONCLUSIONES

The important thing is not to stop questioning.

- Albert Einstein-

La posibilidad de recibir notificaciones en tiempo real en los teléfonos móviles, reduce el tiempo para llevar a cabo acciones correctoras sobre los ataques. Por otro lado, permite que el administrador disponga de cierta movilidad, y no tenga que estar en un puesto fijo.

Además, como hemos podido observar en las pruebas, las notificaciones recibidas son muy sencillas y claras, por lo que cualquier usuario puede proteger su red.

7.1 Líneas de avance

Algunas de las mejoras del sistema están relacionadas con el lugar donde se va a implementar. Por ejemplo, se podrían ofrecer distintos paquetes de reglas personalizados, destinados a distintas actividades empresariales.

Otra posible mejora, podría consistir en configurar la centralita VoIP para que realice llamadas a dispositivos que no estén en su subred, pero esto lleva un riesgo implícito, ya que el sistema de seguridad estaría expuesto a muchos ataques provenientes de Internet. Este riesgo ha supuesto que no se decida implementar en este proyecto, y se incorpore el uso de Telegram para satisfacer este requisito.

ANEXOS

A. Anexo I. Preparación del sistema

Raspbian es el sistema operativo usado en este proyecto. Los pasos a seguir para su instalación [27] son los siguientes:

A. Descargar la imagen: http://downloads.raspberrypi.org/raspbian_latest
B. Extraer la imagen del .zip.
C. Escribir la imagen en la tarjeta microSD: $ sudo dd if=<rutaimagen> of=<rutatarjeta>

Una vez realizada la instalación, para conectarse hay que usar ssh.

```
$ ssh pi@DIRECCIONIPASIGNADA
```

La contraseña por defecto es "raspberry".

El sistema operativo trae una herramienta de configuración que se debe ejecutar al encender el equipo por primera vez:

```
$sudo raspi-config
```

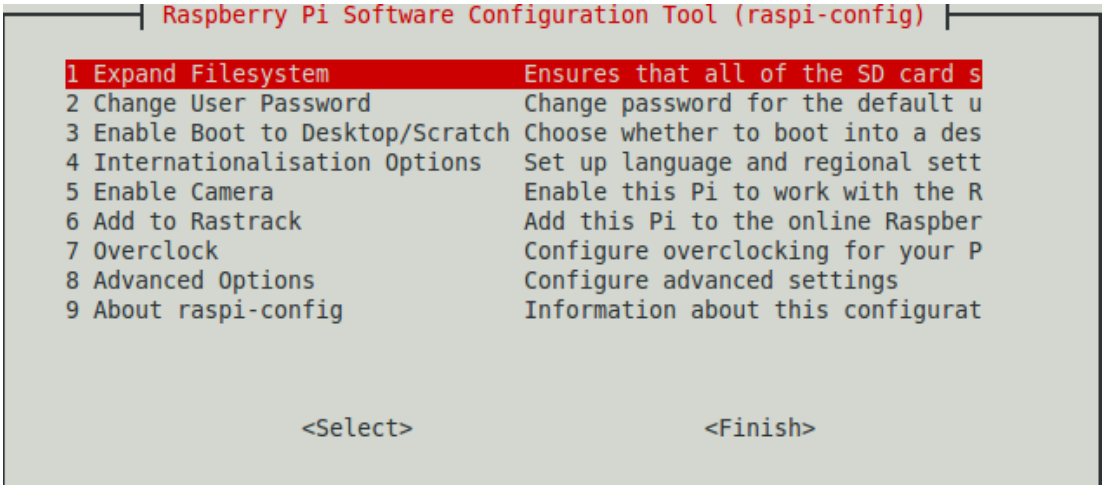


Figura 0-1: Herramienta de configuración de Raspbian

En nuestro caso, solo tenemos que expandir el sistema de archivos para que ocupe toda la capacidad de la microSD. El sistema se reiniciará tras llevar a cabo esta tarea.

Con el objetivo de usar paquetes nuevos y evitar problemas de seguridad, se debe actualizar el sistema.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Como el sistema hará uso de herramientas web, se necesita un servidor web, que permita alojarlas. Se instalará Apache2 .

```
$ sudo su
# apt-get install apache2 apache2-utils apache2.2-bin apache2.2-common libapache2-mod-php5 php5
```

Mucha información obtenida se guardará en distintas base de datos. Para ello se necesita un servidor que permita crearlas y gestionarlas. En este caso se usará MYSQL.

```
# apt-get install mysql-client mysql-common mysql-server
```

Siguiendo las recomendaciones de Snort [28], se debe desactivar LRO [29] y GRO [30]:

```
# apt-get install ethtool
# ethtool -K eth0 gro off
# ethtool -K eth0 lro off
```

Con estas características activadas, la tarjeta de red realiza el reensamblaje de paquetes antes de que sean procesados por el núcleo. Por defecto, Snort trunca paquetes mayores de 1518 bytes. Además LRO y GRO pueden causar problemas con el reensamblado en el preprocesador Stream5.

Se necesitan varias dependencias adicionales para que el sistema funcione correctamente:

```
# apt-get install g++ make autoconf automake libtool flex bison gcc libnet1 libnet1-dev libcrypt-ssleay-perl libpcre3 libpcre3-dev libmysqlclient-dev libphp-adodb libssl-dev libtool libwww-perl libmysqlclient-dev client ntp php5-cli php5-gd php5-mysql php-pear syslog-ng sox
```

Para mayor comodidad, se fijará la dirección IP del sistema:

/etc/network/interfaces

```
auto lo

iface lo inet loopback
auto eth0
iface eth0 inet static
#Mi direccion IP
address 192.168.0.155
#Puerta de enlace
gateway 192.168.0.1
netmask 255.255.255.0
#Red local
network 192.168.0.0
broadcast 192.168.0.255

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

En caso de que el sistema no pueda notificar mediante una llamada VoIP, necesitamos un método alternativo de notificación. En este caso utilizaremos el envío de mensajes a un smartphone mediante Telegram. Los pasos para su instalación los encontramos en [31].

B. Anexo II. Instalación y configuración del IDS

a. Snort

El IDS utilizado será Snort. Se instalará desde el código fuente [32]. Para realizar esta tarea se necesitan varias dependencias, que será lo primero a instalar.

```
# cd /usr/src
# wget http://www.tcpdump.org/release/libpcap-1.7.3.tar.gz
# tar -zxf libpcap-1.7.3.tar.gz
# cd libpcap-1.7.3
# ./configure --prefix=/usr --enable-shared
# make
# make install
#  cd ..
#  wget http://libdnet.googlecode.com/files/libdnet-1.12.tgz
#  tar -zxf libdnet-1.12.tgz
#  cd libdnet-1.12
#  ./configure --prefix=/usr --enable-shared
#  make
#  make install
# cd ..
# wget https://snort.org/downloads/snort/daq-2.0.4.tar.gz
# tar xvfz daq-2.0.4.tar.gz
# cd daq-2.0.4
# ./configure; make; sudo make install
```

Para actualizar la ruta de la librería:

```
# echo >> /etc/ld.so.conf /usr/lib
# echo >> /etc/ld.so.conf /usr/local/lib && ldconfig
```

Para instalar Snort:

```
# cd /usr/src
# wget https://snort.org/downloads/snort/snort-2.9.7.2.tar.gz
# tar xvfz snort-2.9.7.2.tar.gz
# cd snort-2.9.7.2/
# ./configure --enable-sourcefire; make; sudo make install
# mkdir /etc/snort /etc/snort/rules /var/log/snort /var/log/barnyard2 /usr/local/lib/snort_dynamicrules
# touch /etc/snort/rules/white_list.rules /etc/snort/rules/black_list.rules
# groupadd snort && useradd -g snort snort
# chown snort:snort /var/log/snort /var/log/barnyard2
# cp /usr/src/snort-2.9.7.2/etc/*.conf* /etc/snort/
# cp /usr/src/snort-2.9.7.2/etc/*.map /etc/snort
# /usr/local/bin/snort –version
```

```
# snort  -V


  ,,_     -*> Snort! <*-
 o"  )~   Version 2.9.7.2 GRE (Build 177)
  ""    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
        Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
        Copyright (C) 1998-2013 Sourcefire, Inc., et al.
        Using libpcap version 1.7.3
        Using PCRE version: 8.31 2012-07-06
        Using ZLIB version: 1.2.7
```

Tras concluir la instalación, se necesita modificar el archivo de configuración de Snort:

/etc/snort/snort.conf

Línea 45:

```
# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.0.155

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET
```

Línea 104:

```
var RULE_PATH rules
var SO_RULE_PATH so_rules
var PREPROC_RULE_PATH preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snor$
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH rules
var BLACK_LIST_PATH rules
```

Se debe modificar los plugins de salida, con el objetivo de obtener ficheros en formato binario, para que puedan ser utilizados por el módulo encargado de rellenar la base de datos.

Línea 536:

```
output unified2: filename snort.log, limit 128
```

Línea 544:

```
include $RULE_PATH/local.rules
include $RULE_PATH/snort.rules
```

Se eliminan o comentan el resto de reglas.

b. PulledPork

Para poner en funcionamiento Snort, se necesitan una serie de reglas, que se pueden obtener desde su página.

Para obtener estas reglas, el primer paso es registrarse en la página de Snort [33].

Una vez que completado el registro, se podrá obtener un oinkcode:

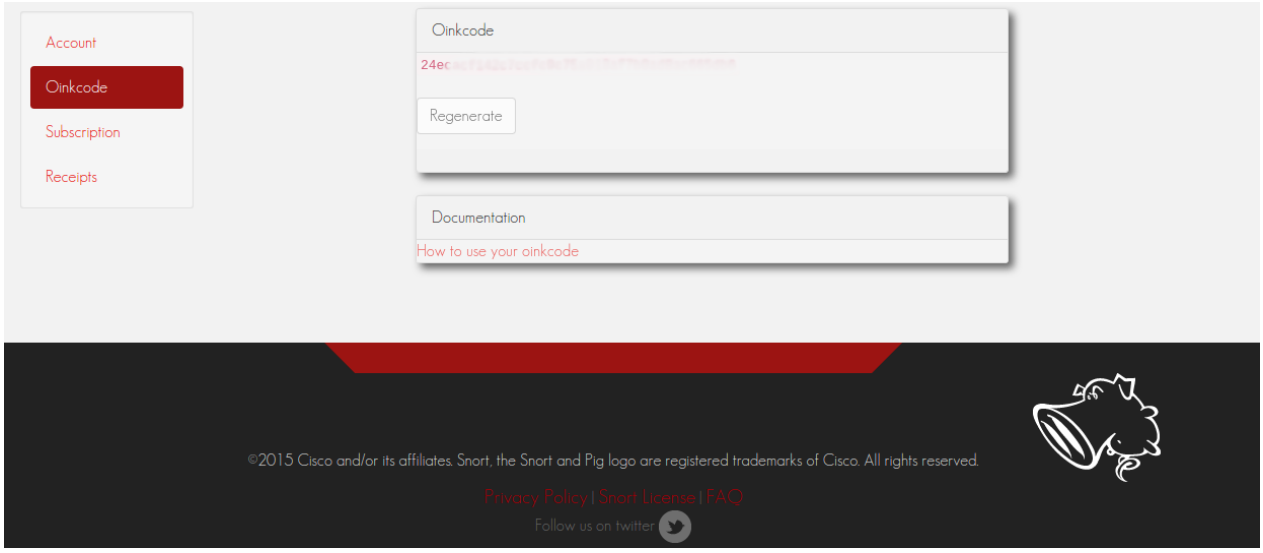


Figura 0-2: Oinkcode

Un oinkcode es una clave asociada a nuestra cuenta que permite descargar reglas.

La descarga de reglas se puede realizar manualmente, pero existe una herramienta llamada PulledPork [34], que permite automatizar dicha tarea.

```
# cd /usr/src/
# wget https://pulledpork.googlecode.com/files/pulledpork-0.7.0.tar.gz
# tar -zxf pulledpork-0.7.0.tar.gz
# cd pulledpork-0.7.0
# cp pulledpork.pl /usr/local/bin && cp etc/*.conf /etc/snort
# pulledpork.pl -V
PulledPork v0.7.0 - Swine Flu!
```

Se crean algunos ficheros y directorios que necesita PulledPork:

```
# mkdir /etc/snort/rules/iplists
# touch /etc/snort/rules/iplists/default.blacklist
```

Hay que editar el fichero de configuración:

/etc/snort/pulledpork.conf

```
rule_url=https://www.snort.org/reg-rules/|snortrules-snapshot.tar.gz|<oinkcode>
rule_url=https://s3.amazonaws.com/snort-org/www/rules/community/|community-rules.tar.gz|Community
rule_url=http://labs.snort.org/feeds/ip-filter.blf|IPBLACKLIST|open
```

```
ignore=deleted.rules,experimental.rules,local.rules
temp_path=/tmp
rule_path=/etc/snort/rules/snort.rules
local_rules=/etc/snort/rules/local.rules
sid_msg=/etc/snort/sid-msg.map
sid_msg_version=1
sid_changelog=/var/log/sid_changes.log

sorule_path=/usr/local/lib/snort_dynamicrules/
snort_path=/usr/local/bin/snort
config_path=/etc/snort/snort.conf
distro=Debian-6-0
black_list=/etc/snort/rules/iplists/default.blacklist
IPRVersion=/etc/snort/rules/iplists
snort_control=/usr/local/bin/snort_control
enablesid=/etc/snort/enablesid.conf
dropsid=/usr/local/etc/snort/dropsid.conf
disablesid=/etc/snort/disablesid.conf
modifysid=/etc/snort/modifysid.conf

version=0.7.0
```

Para desbloquear las reglas bloqueadas:

```
# echo pcre:fwsam >> /etc/snort/disablesid.conf
```

Ya se puede poner en funcionamiento PulledPork:

```
# chmod +x /usr/local/bin/pulledpork.pl
/usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf  -l

   http://code.google.com/p/pulledpork/
     _____ ____
    `----,\    )
     `--==\\  /    PulledPork v0.7.0 - Swine Flu!
      `--==\\/
    .-~~~~-.Y|\\_  Copyright (C) 2009-2013 JJ Cummings
  @_/       /  66\_  cummingsj@gmail.com
   |    \  \  _(")
    \   /-| ||'--'  Rules give me wings!
     \_\ \_\\
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Checking latest MD5 for snortrules-snapshot-2972.tar.gz....
They Match
Done!
Checking latest MD5 for community-rules.tar.gz....
They Match
Done!
IP Blacklist download of http://labs.snort.org/feeds/ip-filter.blf....
Reading IP List...
Checking latest MD5 for opensource.gz....
They Match
Done!
Checking latest MD5 for emerging.rules.tar.gz....
They Match
Done!
Fly Piggy Fly!
```

Las opciones usadas son:

-c /etc/snort/pulledpork.conf: Ruta del fichero de configuración.

-l: Escribir logs en /var/log

Por último, para automatizar por completo el proceso de actualización de las reglas, se puede programar una tarea para que se ejecute periódicamente. Para programar una tarea, Linux ofrece un demonio que permite ejecutar procesos en intervalos de tiempo regulares. Este demonio se llama *cron.*

Para añadir una tarea se utiliza el comando *crontab -e*. El formato es el siguiente:

Minuto(0-59) Hora(0-23) Día(1-31) Mes(1-12) Día de la semana(0=domingo, 1=lunes, ...) comando

Si no se quiere usar un campo, a la hora de especificar la fecha, se utiliza "*".

En nuestro caso:

```
0 5 * * * /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

Gracias a esto, las reglas se actualizarán diariamente.

PulledPork guardará todas las reglas en el fichero /etc/snort/rules/snort.rules. Antes de poner en funcionamiento a Snort, se debe crear el fichero que almacenará nuestras reglas (etc/snort/rules/local.rules).

Ejecutamos Snort en modo prueba para comprobar que esté correctamente configurado:

```
# snort -c /etc/snort/snort.conf -T
```

Al final se debe obtener este resultado si está correctamente configurado:

```
      --== Initialization Complete ==--


 ,,_     -*> Snort! <*-
o"  )~   Version 2.9.7.2 GRE (Build 177)
```

```
"" By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
   Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
   Copyright (C) 1998-2013 Sourcefire, Inc., et al.
   Using libpcap version 1.7.3
   Using PCRE version: 8.31 2012-07-06
   Using ZLIB version: 1.2.7

   Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.4  <Build 1>
   Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
   Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
   Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
   Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
   Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
   Preprocessor Object: SF_POP  Version 1.0  <Build 1>
   Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
   Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
   Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
   Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
   Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
   Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
   Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
   Preprocessor Object: SF_SSH  Version 1.1  <Build 3>

Snort successfully validated the configuration!
Snort exiting
```

c. Barnyard2

Los logs se almacenarán en una base de datos. Barnyard2 será el encargado en enviar información a la base de datos.

```
#cd /usr/src
# wget https://github.com/firnsy/barnyard2/archive/v2-1.13.tar.gz
# tar -zxf v2-1.13.tar.gz
# cd barnyard2-2-1.13/
# autoreconf -fvi -I ./m4 && ./configure --with-mysql && make && make install
# mv /usr/local/etc/barnyard2.conf /etc/snort
# cp schemas/create_mysql /usr/src
```

Pero Barnyard2 necesita que las alertas estén en formato binario para poder leerlas. Se utilizarán los ficheros snort.log.XXXXX generados en el directorio /var/log/snort.

Barnyard2 necesita una serie de ficheros y directorios:

```
# mkdir /var/log/barnyard2/
# touch /var/log/snort/barnyard2.waldo
```

Se edita /etc/snort/barnyard2.conf para que funcione corretamente.

Lo primero es configurar el equipo y la interfaz que recibe los datos:

Linea 68:

```
config hostname:   localhost
config interface:  eth0
```

Para enviar los datos a nuestra base de datos y aun fichero, añadimos en la línea 225:

```
output alert_fast: alertas
output database: log, mysql, user=root password=raspberry dbname=snort host=localhost
```

El siguiente paso es configurar la base de datos:

```
$ mysql -u root -h localhost -p
grant usage on snort.* to snort@localhost;
grant usage on archive.* to snort@localhost;
set password for snort@localhost=PASSWORD('snort');
grant all privileges on snort.* to snort@localhost;
grant all privileges on archive.* to snort@localhost;
flush privileges;
exit;
```

La base de datos que almacene la información proveniente de Snort debe de tener una estructura concreta. Existe un fichero que permite crear las tablas de la base de datos automáticamente (/usr/src/create_mysql).

```
$ mysql -u root -h localhost -p
use snort;
source /usr/src/create_mysql;
```

Se ejecuta Barnyard2 para comprobar que funciona correctamente:

```
# /usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.log –v
```

Las opciones usadas son:

-c /etc/snort/barnyard2.conf : Archivo de configuración utilizado por Barnyard2.

-d /var/log/snort: Directorio donde se almacenan los logs.

-f snort.log: Tipo de archivos utilizados. En este caso se usarán archivos con estructura snort.log.XXXXXX

-v: Modo verbose.

El resultado debe ser similar al siguiente:

```
    --== Initialization Complete ==--
```

```
 ______   -*> Barnyard2 <*-
/ „_  \  Version 2.1.13 (Build 327)
|o"  )~|  By Ian Firns (SecurixLive): http://www.securixlive.com/
+ "" +  (C) Copyright 2008-2013 Ian Firns <firnsy@securixlive.com>

Opened spool file '/var/log/snort/snort.log.1431169171'
Closing spool file '/var/log/snort/snort.log.1431169171'. Read 8 records
Opened spool file '/var/log/snort/snort.log.1431170408'
Closing spool file '/var/log/snort/snort.log.1431170408'. Read 8 records
Opened spool file '/var/log/snort/snort.log.1431171172'
Closing spool file '/var/log/snort/snort.log.1431171172'. Read 6 records
Opened spool file '/var/log/snort/snort.log.1431172313'
Closing spool file '/var/log/snort/snort.log.1431172313'. Read 8 records
Opened spool file '/var/log/snort/snort.log.1431172355'
Closing spool file '/var/log/snort/snort.log.1431172355'. Read 16 records
Opened spool file '/var/log/snort/snort.log.1431172402'
Closing spool file '/var/log/snort/snort.log.1431172402'. Read 0 records
Opened spool file '/var/log/snort/snort.log.1431178214'
Waiting for new data
```

Para comprobar que se ha escrito en la base de datos:

```
root@raspberrypi:/home/pi# mysql -u root -p
mysql> use snort;
mysql> show tables;
+------------------+
| Tables_in_snort  |
+------------------+
| acid_ag          |
| acid_ag_alert    |
| acid_event       |
| acid_ip_cache    |
| base_roles       |
| base_users       |
| data             |
| detail           |
| encoding         |
| event            |
| icmphdr          |
```

```
| iphdr          |
| opt            |
| reference      |
| reference_system |
| schema         |
| sensor         |
| sig_class      |
| sig_reference  |
| signature      |
| tcphdr         |
| udphdr         |
+----------------+
22 rows in set (0.00 sec)


mysql> select count(*) from event;
+----------+
| count(*) |
+----------+
|    16184 |
+----------+
1 row in set (0.12 sec)
```

Si eventos > 0, significa que se han escrito eventos en la base de datos, y por lo tanto todo funciona correctamente.

El último paso en la configuración del IDS es conseguir que se ejecute con cada inicio del sistema. Para ello hay que añadir al cron las siguientes líneas:

```
@reboot /usr/local/bin/snort -c /etc/snort/snort.conf -i eth0 –D #Barnyard2
@reboot /usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.log -w /var/log/snort
/barnyard2.waldo –D
```

Para ejecutar Snort en modo demonio se necesita una memoria de intercambio [35]. Con 512 MB valdrá:

```
#nano /etc/dphys-swapfile
CONF_SWAPSIZE=512
# /etc/init.d/dphys-swapfile stop
Stopping dphys-swapfile swapfile setup ..., done.
# /etc/init.d/dphys-swapfile start
Starting dphys-swapfile swapfile setup ...
want /var/swap=512MByte, checking existing: deleting wrong size file (1048576000), generating swapfile ...
of 512MBytes
```

```
done.
```

Si todo ha ido bien, tras reiniciar el sistema se debe obtener una salida similar al ejecutar el siguiente comando:

```
# ps -aux | grep "snort"
warning: bad ps syntax, perhaps a bogus '-'?
See http://gitorious.org/procps/procps/blobs/master/Documentation/FAQ
root 3442 92.6 7.7 47668 34500 ? Rs 23:19 3:14 /usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.log -w /var/log/snort/barnyard2.waldo -D
root 3444 0.0 45.0 467784 200672 ? Ssl 23:19 0:00 /usr/local/bin/snort -c /etc/snort/snort.conf -i eth0 -D
root 3452 0.0 0.3 3548 1688 pts/0 S+ 23:22 0:00 grep snort
```

d. BASE

Para instalar BASE :

```
#nano /etc/php5/apache2filter/php.ini
```

Línea 452:

```
; Common Values:
;   E_ALL (Show all errors, warnings and notices including coding standards.)
;   E_ALL & ~E_NOTICE  (Show all errors, except for notices)
;   E_ALL & ~E_NOTICE & ~E_STRICT  (Show all errors, except for notices and cod$
;   E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR  (Show only errors)
; Default Value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED
; Development Value: E_ALL
; Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT
; http://php.net/error-reporting
error_reporting = E_ALL & ~E_NOTICE
```

```
#pear config-set preferred_state alpha && pear channel-update pear.php.net && pear install --alldeps Image_Color Image_Canvas Image_Graph
#a2enmod ssl
#service apache2 restart
#cd /usr/src && wget http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-1.4.5.tar.gz
# tar -zxf base-1.4.5.tar.gz && cp -r base-1.4.5 /var/www/base
#chown -R www-data:www-data /var/www/base
```

Para realizar graficas se necesita gd:

```
apt-get install php5-gd
service apache2 restart
pear install --force Image_Color
```

```
pear install --force Image_Canvas
pear install --force Image_Graph
service apache2 restart
```

Ya se puede acceder desde http://DireccionRaspberry/base/.

Para configurar correctamente BASE:

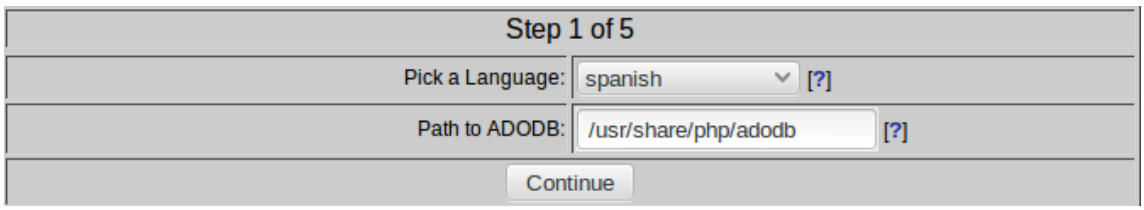


Figura 0-3: Paso 1 en la configuración de BASE

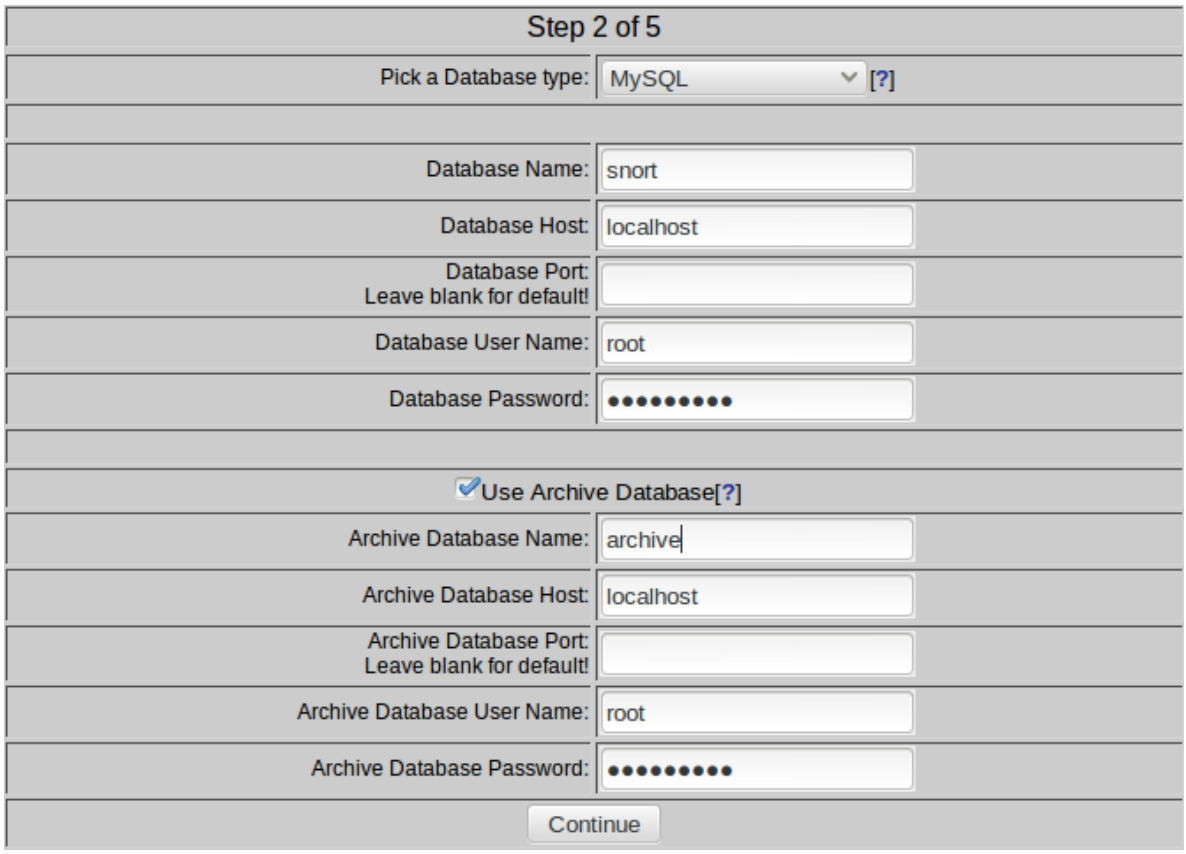


Figura 0-4: Paso 2 en la configuración de BASE

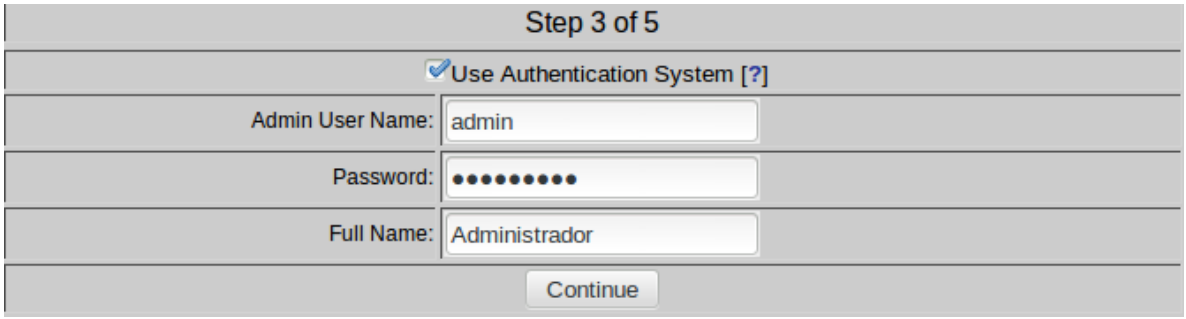


Figura 0-5 : Paso 3 en la configuración de BASE

Pulsamos el botón “Create BASE AG”:

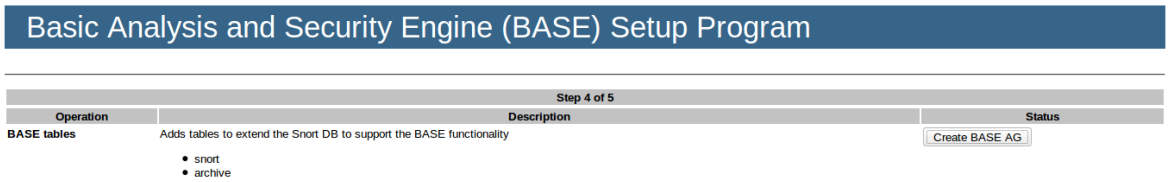


Figura 0-6: Paso 4 en la configuración de BASE

Y debe aparecer:

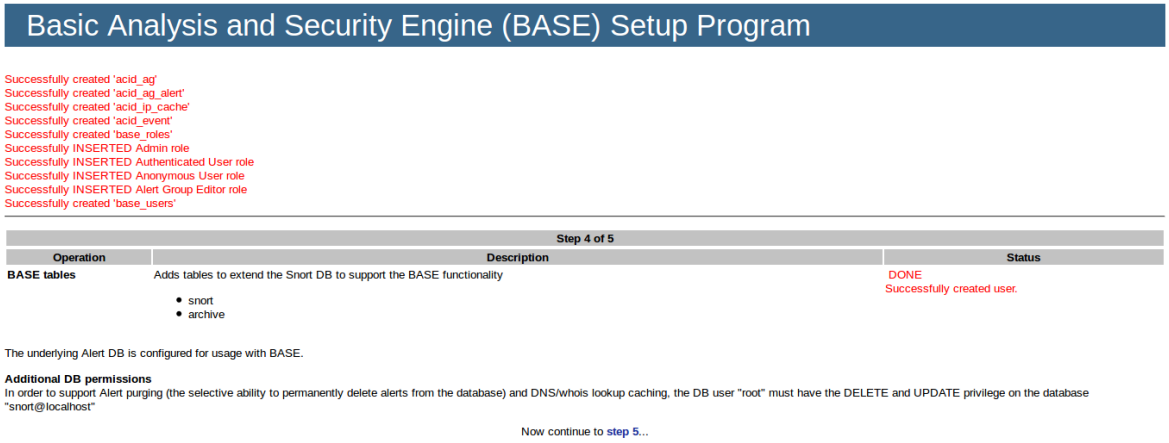
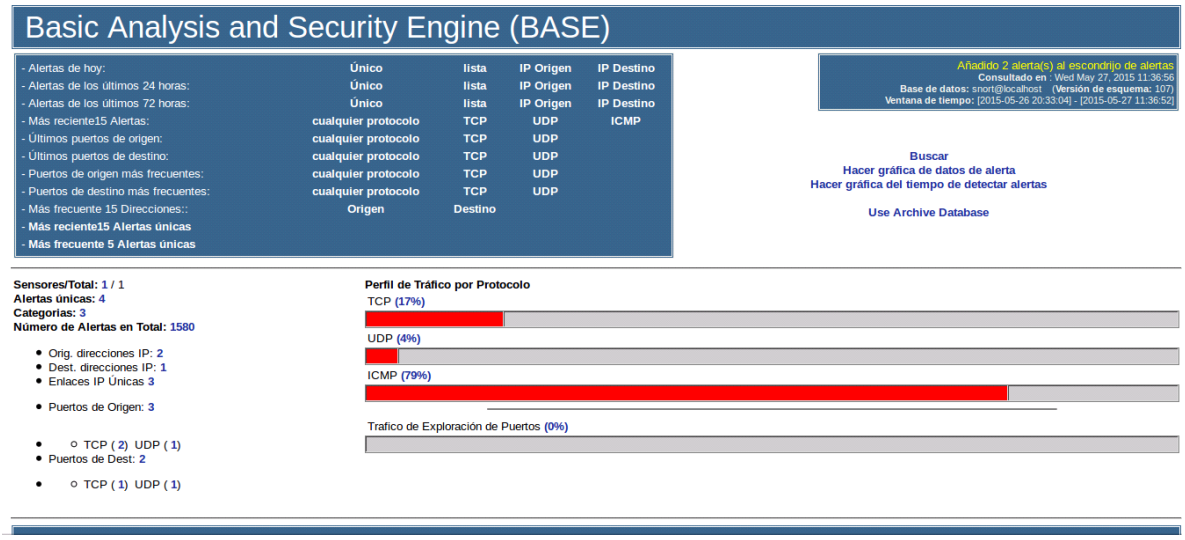


Figura 0-7: Paso 5 en la configuración de BASE

Ya se puede acceder a todas las funcionalidades de BASE, tras identificarse correctamente:

Figura 0-8: Página de identificación en BASE



C. Anexo III. Formato de las alertas de Snort

El formato de una alerta es el siguiente:

```
[**] [1:10000001:1] ping a la red [**]
[Priority: 0]
05/29-12:12:26.634959 192.168.0.156 -> 192.168.0.155
ICMP TTL:64 TOS:0x0 ID:27460 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:7084   Seq:1  ECHO
```

Figura 0-9: Ejemplo de alerta de prioridad 0

```
[**] [128:4:1] (spp_ssh) Protocol mismatch [**]
[Classification: Detection of a non-standard protocol or event] [Priority: 2]
05/29-12:12:28.602436 192.168.0.156:47059 -> 192.168.0.155:22
TCP TTL:64 TOS:0x10 ID:38235 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0x14258CA9  Ack: 0x51AE9A33  Win: 0x366D  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2424332 280217
```

Figura 0-10: Ejemplo de alerta de prioridad 2

Los mensajes los podemos localizar de la siguiente manera:

[**] [x:y:z] **mensaje** [**]

Para la generación de un archivo de audio con información sobre las amenazas, utilizaremos las alertas con prioridad 0. Esto se debe a que los mensajes de las alertas de Snort por defecto, no tienen un formato leíble estándar, por lo que es imposible su transformación a audio.

```
[**] [128:4:1] (spp_ssh) Protocol mismatch [**]
[Classification: Detection of a non-standard protocol or event] [Priority: 2]
05/29-12:12:28.602436 192.168.0.156:47059 -> 192.168.0.155:22
TCP TTL:64 TOS:0x10 ID:38235 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0x14258CA9  Ack: 0x51AE9A33  Win: 0x366D  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2424332 280217
```

Figura 0-11: Ejemplo de mensaje de alerta con prioridad 2

Las alertas de prioridad 0 son todas aquellas que creadas por nosotros, por lo que le podremos asignar unos mensajes con un formato correcto para el sistema

D. Anexo IV. Asterisk

a. Instalación

En este punto se describirán los pasos necesarios para instalar Asterisk [36].

Para descargar la última versión de Asterisk:

```
#cd /usr/src
#wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-13-current.tar.gz
#tar -zxvf asterisk-13-current.tar.gz
```

Para instalar los prerrequisitos:

```
#cd asterisk-13.3.2/contrib/scripts/
#./install_prereq install
```

Una vez instaladas las dependencias necesarias, se puede comenzar con la instalación de Asterisk:

```
#cd ../..
#./configure
#make
#make install
#make samples
#make config
#/etc/init.d/asterisk start
```

Comprobamos si tenemos instalado Festival [37]:

```
#asterisk -r
raspberrypi*CLI> module show like festival
Module                  Description                  Use Count  Status      Support Level
app_festival.so             Simple Festival Interface          0          Running          extended
1 modules loaded
```

Lo siguiente es instalarlo [38]:

```
#sudo apt-get install festival festival-dev
```

Si se quieren utilizar voces en español para Asterisk:

```
#cd /usr/src
wget http://forja.guadalinex.org/frs/download.php/154/festvox-sflpc16k_1.0-1_all.deb
```

```
dpkg -i  festvox-sflpc16k_1.0-1_all.deb
```

Se utilizará el fichero de configuración por defecto de Festival:

```
#rm /etc/festival.scm
```

Por último, hay que añadir al fichero /usr/share/festival/festival.scm las siguientes líneas:

```
(set! voice_default 'voice_JuntaDeAndalucia_es_sf_diphone)
(define (tts_textasterisk string mode)
(let ((wholeutt (utt.synth (eval (list 'Utterance 'Text string)))))
(utt.wave.resample wholeutt 8000)
(utt.wave.rescale wholeutt 5)
(utt.send.wave.client wholeutt)))
(set! server_access_list '("localhost\\.localdomain" "localhost"))
```

Para que se inicie Asterisk con cada inicio de sesión:

```
#cp  /usr/share/doc/festival/examples/festival.init /etc/init.d/festival
#chmod +x /etc/init.d/festival
```

/etc/default/festival

```
RUN_FESTIVAL=yes
```

El último paso es crear un enlace simbólico:

```
#ln -s /etc/init.d/festival /etc/rcS.d/S99festival
```

b. Configuración

/etc/asterisk/sip.conf

```
[general]
context=internal
allowguest=no
allowoverlap=no
bindport=5060
bindaddr=0.0.0.0
srvlookup=no
disallow=all
allow=ulaw
alwaysauthreject=yes
canreinvite=no
nat=yes
session-timers=refuse
localnet=192.168.0.0/255.255.255.0
```

```
[7001]
type=friend
host=dynamic
secret=123
context=internal

[7002]
type=friend
host=dynamic
secret=456
context=internal
```

/etc/asterisk/extensions.conf

```
 [internal]
exten=> 3000,1,Answer()
same => n,Wait(2)
same => n,Festival(Extension 7001)
same => n,Wait(2)
same => n,Hangup()

exten=> 7002,1,Answer()
same => n,Festival(Extension 7002)
same => n,Hangup()

exten =>1,1,system(sh /home/pi/prioridad1.sh)
same => n,Answer()
same => n,Wait(1)
same => n,Playback(finalp1)
same => n,Hangup()

exten =>2,1,system(sh /home/pi/prioridad2.sh)
same => n,Answer()
same => n,Wait(1)
same => n,Playback(finalp2)
same => n,Hangup()

[prioridad0]
exten=> 0,1,Answer()
```

```
same => n,Wait(1)
same => n,Playback(finalp0)
same => n,system(sudo nohup /home/pi/apagarojo.sh &)
same => n,system(sudo nohup /home/pi/enciendeverde.sh &)
same => n,Hangup()

exten => failed,1,system(sudo nohup /tg/telegram.sh &)
same => n,system(sudo nohup /home/pi/apagarojo.sh &)
same => n,system(sudo nohup /home/pi/enciendeverde.sh &)
```

c. Formato fichero de llamada

El fichero de llamada [39] usado tiene las siguientes características:

- Destino de llamada: Usuario 7001.
- Número máximo de reintentos: 2.
- Tiempo de espera entre llamadas: 120 segundos.
- Tiempo de espera para responder la llamada: 30 segundos.
- Contexto: prioridad0.
- Extensión: 0.

```
Channel: SIP/7001
MaxRetries: 2
RetryTime: 120
WaitTime: 30
Context: prioridad0
Extension: 0
```

d. Configuración terminal móvil

La siguiente configuración ha sido llevada a cabo en un terminal con Android 4.4.4.

En *Telefono->Ajustes* , buscamos *Ajustes de llamadas por Internet.*

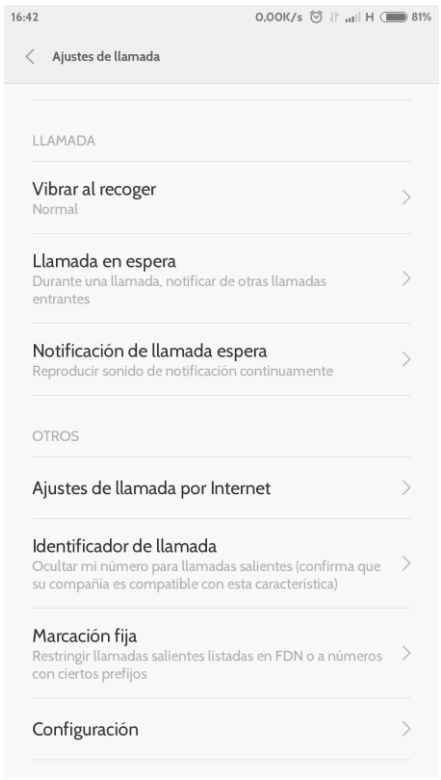


Figura 0-12: Ajustes de llamada sobre Internet

Una vez dentro, configuramos los detalles de nuestra cuenta SIP, y la guardamos.

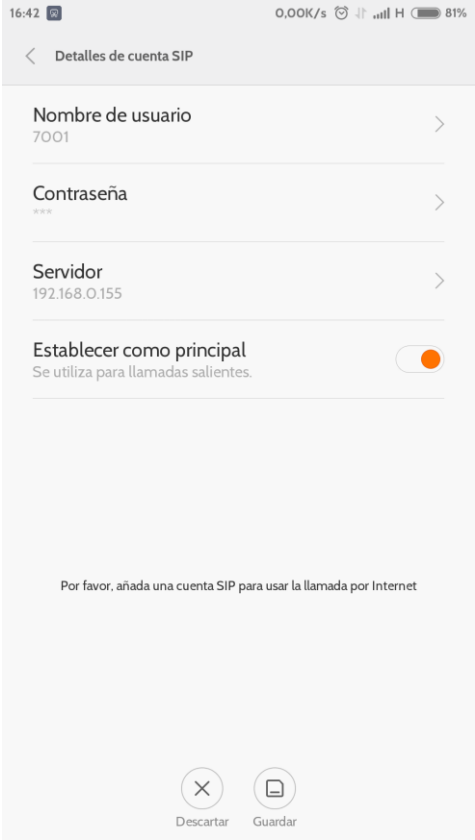


Figura 0-13: Configuración de cuenta SIP

Para conectarse, hay que estar conectado a la misma subred.

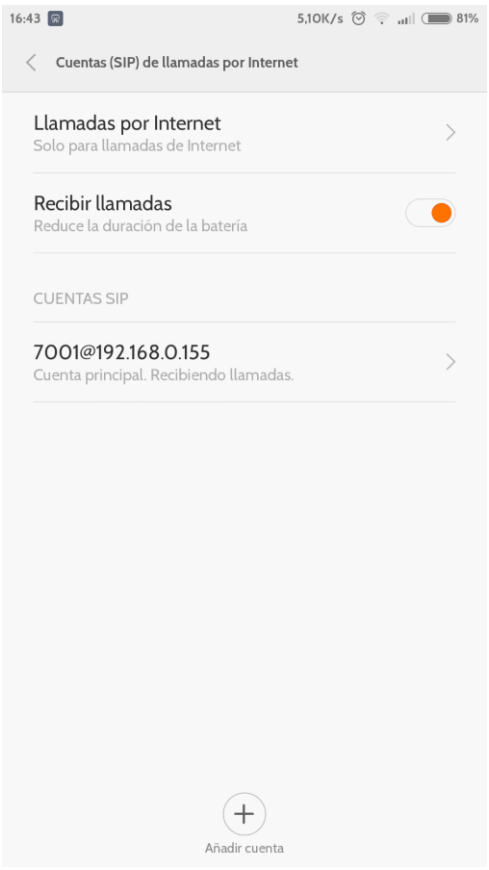


Figura 0-14: Listas de cuentas SIP

Si se quieren recibir llamadas desde el sistema, se debe activar la opción *Recibir llamadas*.

E. Anexo V. Scripts

a. /home/pi/ciclo.sh

```
#!/bin/sh
sleep 300
while [ 1 ]; do
 sh /home/pi/alertas.sh
 sleep 25
done
```

Hay que añadirlo al cron para que se ejecute al iniciar el sistema:

```
@reboot sh /home/pi/ciclo.sh
```

b. /home/pi/alertas.sh

```
#!/bin/sh
#Script para procesar las alertas generadas por las reglas que hemos creado en local.rules
```

```
#Este script cuenta las lineas de un fichero, las compara con el numero de la anterior comparacion, y obtiene esas lineas nuevas
#Lineasantiguas contiene el numero de lineas que tenia el fichero en la ejecución anterior. Las siguientes sentecias
#se encargan de comprobar la existencia de determinados ficheros, y si no existen los creamos.
if [ ! -f lineasantiguasp0 ]; then
echo 0 > lineasantiguasp0
fi
if [ ! -f nuevop0 ]; then
touch nuevop0
fi
if [ ! -f prioridad0 ]; then
touch prioridad0
fi
if [ ! -f prioridad0procesado ]; then
touch prioridad0procesado
fi
if [ ! -f finalp0 ]; then
touch finalp0
fi
#Obtenemos el numero de lineas del fichero
salida=$(awk 'END {print NR}' /var/log/snort/alertas)
#Rescatamos el numero de lineas en la anterior ejecución
salidaantigua=$(cat lineasantiguasp0)
#Obtenemos la lineas nuevas
resta=$(($salida-$salidaantigua))
echo "Se han añadido $(($resta)) lineas"
#Guardamos el numero de lineas para rescatarlo en la proxima ejecución
awk 'END {print NR}' /var/log/snort/alertas > lineasantiguasp0
#Guardamos las nuevas lineas añadidas
tail -n $resta /var/log/snort/alertas > nuevop0
#Y lo procesamos con awk
awk 'BEGIN{RS="\n\n"; ORS="\n\n"} /Priority: 0/' nuevop0 > prioridad0
#Solo generamos la llamada si se ha añadido algo
lineasprioridad0=$(awk 'END {print NR}' prioridad0)
if [ $lineasprioridad0 != '0' ]; then
awk 'BEGIN{RS="\n\n"; FS="\n"}
{
c=split($1,one," ");
```

```
printf "Alertas detectadas. \n";
for (x=3;x<c;x++){printf one[x] " "};
gsub("^.*] ","",$2);printf "   \n";
}' prioridad0 > prioridad0procesado
#Encendemos la alerta visual
nohup /home/pi/apagaverde.sh &
nohup /home/pi/encienderojo.sh &
#Pasamos las lineas procesadas al fichero final
awk '!array_temp[$0]++' prioridad0procesado > finalp0
#Pasamos el fichero procesado a formato audio y lo transformamos a un formato compatible con Asterisk
cat finalp0 | text2wave > finalp0.wav
sox finalp0.wav finalp0.gsm
mv finalp0.gsm /var/lib/asterisk/sounds/en
#Generamos el fichero de llamada
printf "Channel: SIP/7001\nMaxRetries: 2 \nRetryTime: 120 \nWaitTime: 30 \nContext: prioridad0\nExtension: 0 \n" > llamadap0.call
mv llamadap0.call /var/spool/asterisk/outgoing/
fi
```

Ficheros necesarios:

- lineasantiguas0: Almacena el número de líneas que tenía el fichero */var/log/snort/alertas* en la última ejecución del script. Si no existe, se crea y se almacena el valor 0.
- nuevo0: Fichero que almacena las líneas añadidas al fichero */var/log/alertas* desde la última ejecución.
- prioridad0: Contiene todas las alertas con prioridad0 del fichero nuevo0.
- prioridad0procesado: Almacena todos los mensajes de las alertas contenidas en el fichero prioridad0. Puede contener mensajes repetidos.
- finalp0: Fichero sin mensajes repetidos. Será el utilizado en la generación del fichero de audio.

c. /tg/telegram.sh

```
#!/bin/bash
tgpath=/tg
#Necesitamos estar en el directorio
cd ${tgpath}
#Enviamos el audio de la alerta
telegram-cli -W -e "send_audio Ismael_N /root/finalp0.wav"
```

d. /home/pi/prioridad1.sh

```
#!/bin/sh
#Script para procesar las alertas generadas de prioridad1
#Este script cuenta las lineas de un fichero, las compara con el numero de la anterior comparacion, y obtiene esas lineas nuevas
#Lineasantiguas contiene el numero de lineas que tenia el fichero en la ejecución anterior. Las siguientes sentecias
#se encargan de comprobar la existencia de determinados ficheros, y si no existen los creamos.
if [ ! -f lineasantiguasp1 ]; then
echo 0 > lineasantiguasp1
fi
if [ ! -f nuevop1 ]; then
touch nuevop1
fi
if [ ! -f prioridad1 ]; then
touch prioridad1
fi
if [ ! -f prioridad1procesado ]; then
touch prioridad1procesado
fi
if [ ! -f finalp1 ]; then
touch finalp1
fi
#Obtenemos el numero de lineas del fichero
salida=$(sudo awk 'END {print NR}' /var/log/snort/alertas)
#Rescatamos el numero de lineas en la anterior ejecución
salidaantigua=$(cat lineasantiguasp1)
#Obtenemos la lineas nuevas
resta=$(($salida-$salidaantigua))
echo "Se han añadido $(($resta)) lineas"
#Guardamos el numero de lineas para rescatarlo en la proxima ejecución
sudo awk 'END {print NR}' /var/log/snort/alertas > lineasantiguasp1
#Guardamos las nuevas lineas añadidas
sudo tail -n $resta /var/log/snort/alertas > nuevop1
#Y lo procesamos con awk
awk 'BEGIN{RS="\n\n"; ORS="\n\n"} /Priority: 1/' nuevop1 > prioridad1

#Solo generamos la llamada con las alertas si se ha añadido algo
lineasprioridad1=$(awk 'END {print NR}' prioridad1)
if [ $lineasprioridad1 != '0' ]; then
```

```
awk 'BEGIN{RS="\n\n"; FS="\n"}
{
c=split($1,one," ");
printf "Alertas de prioridad uno detectadas. \n";
for (x=3;x<c;x++){printf one[x] " "};
gsub("^.*] ","",$2);printf "\n";
}' prioridad1 > prioridad1procesado
#Pasamos las lineas procesadas al fichero final
awk '!array_temp[$0]++' prioridad1procesado > finalp1
#Pasamos el fichero procesado a formato audio y lo transformamos a un formato compatible con Asterisk
cat finalp1 | text2wave > finalp1.wav
sox finalp1.wav finalp1.gsm
sudo mv finalp1.gsm /var/lib/asterisk/sounds/en
#Si no hay alertas reproducimos que no hay
else
echo "No hay alertas de prioridad uno." | text2wave > finalp1.wav
sox finalp1.wav finalp1.gsm
sudo mv finalp1.gsm /var/lib/asterisk/sounds/en
fi
```

Ficheros necesarios:

- lineasantiguas1: Almacena el número de líneas que tenía el fichero */var/log/snort/alertas* en la última ejecución del script. Si no existe, se crea y se almacena el valor 0.
- nuevo1: Fichero que almacena las líneas añadidas al fichero */var/log/alertas* desde la última ejecución.
- prioridad1: Contiene todas las alertas con prioridad 1 del fichero nuevo1.
- prioridad1procesado: Almacena todos los mensajes de las alertas contenidas en el fichero prioridad1. Puede contener mensajes repetidos.
- finalp1: Fichero sin mensajes repetidos. Será el utilizado en la generación del fichero de audio.

e. /home/pi/prioridad2.sh

```
#!/bin/sh
#Script para procesar las alertas generadas de prioridad2
#Este script cuenta las lineas de un fichero, las compara con el numero de la anterior comparacion, y obtiene esas lineas nuevas
#Lineasantiguas contiene el numero de lineas que tenia el fichero en la ejecución anterior. Las siguientes
```

```
sentecias
#se encargan de comprobar la existencia de determinados ficheros, y si no existen los creamos.
if [ ! -f lineasantiguasp2 ]; then
echo 0 > lineasantiguasp2
fi
if [ ! -f nuevop2 ]; then
touch nuevop2
fi
if [ ! -f prioridad2 ]; then
touch prioridad2
fi
if [ ! -f prioridad2procesado ]; then
touch prioridad2procesado
fi
if [ ! -f finalp2 ]; then
touch finalp2
fi
#Obtenemos el numero de lineas del fichero
salida=$(sudo awk 'END {print NR}' /var/log/snort/alertas)
#Rescatamos el numero de lineas en la anterior ejecución
salidaantigua=$(cat lineasantiguasp2)
#Obtenemos la lineas nuevas
resta=$(($salida-$salidaantigua))
echo "Se han añadido $(($resta)) lineas"
#Guardamos el numero de lineas para rescatarlo en la proxima ejecución
sudo awk 'END {print NR}' /var/log/snort/alertas > lineasantiguasp2
#Guardamos las nuevas lineas añadidas
sudo tail -n $resta /var/log/snort/alertas > nuevop2
#Y lo procesamos con awk
awk 'BEGIN{RS="\n\n"; ORS="\n\n"} /Priority: 2/' nuevop2 > prioridad2

#Solo generamos la llamada si se ha añadido algo
lineasprioridad2=$(awk 'END {print NR}' prioridad2)
if [ $lineasprioridad2 != '0' ]; then
awk 'BEGIN{RS="\n\n"; FS="\n"}
{
c=split($1,one," ");
printf "Alertas de prioridad dos detectadas. \n";
```

```
for (x=3;x<c;x++){printf one[x] " "};
gsub("^.*] ","",$2);printf "\n";
}' prioridad2 > prioridad2procesado
#Pasamos las lineas procesadas al fichero final
awk '!array_temp[$0]++' prioridad2procesado > finalp2
#Pasamos el fichero procesado a formato audio y lo transformamos a un formato compatible con Asterisk
cat finalp2 | text2wave > finalp2.wav
sox finalp2.wav finalp2.gsm
sudo mv finalp2.gsm /var/lib/asterisk/sounds/en
else
echo "No hay alertas de prioridad 2" | text2wave > finalp2.wav
sox finalp2.wav finalp2.gsm
sudo mv finalp2.gsm /var/lib/asterisk/sounds/en
fi
```

Ficheros necesarios:

- lineasantiguas2: Almacena el número de líneas que tenía el fichero */var/log/snort/alertas* en la última ejecución del script. Si no existe, se crea y se almacena el valor 0.
- nuevo2: Fichero que almacena las líneas añadidas al fichero */var/log/alertas* desde la última ejecución.
- prioridad2: Contiene todas las alertas con prioridad 2 del fichero nuevo2.
- prioridad2procesado: Almacena todos los mensajes de las alertas contenidas en el fichero prioridad2. Puede contener mensajes repetidos.
- finalp2: Fichero sin mensajes repetidos. Será el utilizado en la generación del fichero de audio.

f. /home/pi/enciendeverde.sh

```
#!/bin/bash
#Exportamos el puerto GPIO 17. LED de funcionamiento
echo 17 > /sys/class/gpio/export
#Lo configuramos como salida
echo out > /sys/class/gpio/gpio17/direction
#Encendemos el LED asignandole 1 como valor lógico
echo 1 > /sys/class/gpio/gpio17/value
```

Hay que añadirlo al cron para que se ejecute al iniciar el sistema:

```
@reboot sh /home/pi/ enciendeverde.sh
```

g. /home/pi/encienderojo.sh

```
#!/bin/bash
#Exportamos el puerto GPIO 18. LED de alerta
```

```
echo 18 > /sys/class/gpio/export
#Lo configuramos como salida
echo out > /sys/class/gpio/gpio18/direction
#Encendemos el LED asignandole 1 como valor lógico
echo 1 > /sys/class/gpio/gpio18/value
```

h. /home/pi/apagaverde.sh

```
#!/bin/bash
#Apagamos el LED asignandole 0 como valor lógico
echo 0 > /sys/class/gpio/gpio17/value
#Eliminamos la entrada del puerto GPIO 17
echo 17 > /sys/class/gpio/unexport
```

a. /home/pi/apagarojo.sh

```
#!/bin/bash
#Apagamos el LED asignandole 0 como valor lógico
echo 0 > /sys/class/gpio/gpio18/value
#Eliminamos la entrada del puerto GPIO 18
echo 18 > /sys/class/gpio/unexport
```

REFERENCIAS

[1] SecureIdeas. Página web de BASE. [Online].
http://sourceforge.net/projects/secureideas/

[2] Snorby. Página de Snorby en github. [Online].
https://github.com/Snorby/snorby

[3] O'Reilly Commons. Wiki de O'Reilly Commons. [Online].
http://commons.oreilly.com/wiki/index.php/Snort_Cookbook/Logging,_Alerts,_and_Output_Plug-ins#Logging_to_Email

[4] OpenNMS. Página web de OpenNMS. [Online].
http://www.opennms.org/

[5] OpenNMS .Wiki de OpenNMS. [Online].
http://www.opennms.org/wiki/Snort_Integration

[6] OpenNMS .Wiki de OpenNMS. [Online].
http://www.opennms.org/wiki/Tutorial_Notifications#Event_Notifications

[7] Wikipedia. Wikipedia: La enciclopedia libre. [Online].
http://es.wikipedia.org/wiki/Diagrama_de_Gantt

[8] Snort. Página web de Snort. [Online].
https://www.snort.org/

[9] Jesús Alberto Vidal Cortes. *El lenguaje de programación AWK.*

[10] LinuxQuestions. Wiki de LinuxQuestions. [Online].
http://wiki.linuxquestions.org/wiki/Text2wave

[11] Elio Rojano. Blogs de Sinologic. [Online].
https://www.sinologic.net/blog/2010-08/como-convertir-audio-con-sox-compatible-con-asterisk.html

[12] Asterisk. Página web de Asterisk. [Online].
http://www.asterisk.org/

[13] Elastix. Página web de Elastix.[Online].
http://www.elastix.org/index.php/es/

[14] FreePBX. Página web de FreePBX. [Online].

http://www.freepbx.org/

[15] MySQL. Página web de MySQL. [Online].
https://www.mysql.com/

[16] Apache. Página web de Apache. [Online].
http://httpd.apache.org/

[17] Telegram. Página web de Telegram. [Online].
https://telegram.org/

[18] Raspberry Pi. Página web de Raspberry Pi. [Online].
https://www.raspberrypi.org/products/model-b-plus/

[19] Departamento de Telemática, *Apuntes Seguridad*

[20] Axelko. The Infamous TechBlog. [Online].
http://www.axelko.com/techblog/2013/09/curso-asterisk-iii-extensiones-internas/

[21] VoIp-info.Wiki de VoIP-Info. [Online].
http://www.voip-info.org/wiki/view/Asterisk+standard+extensions

[22] Raspberry Pi. Página web de Raspberry Pi. [Online].
https://www.raspberrypi.org/documentation/usage/gpio/

[23] Wikipedia. Wikipedia: La enciclopedia libre. [Online].
https://es.wikipedia.org/wiki/Debian_Wheezy

[24] Raspbian. Página web de Raspbian. [Online].
https://www.raspbian.org/

[25] Google. Página web de Android. [Online].
http://www.android.com/versions/kit-kat-4-4/

[26] Snort. Manual de Snort. [Online].
http://manual.snort.org/node27.html

[27] Raspberry Pi. Página web de Raspberry Pi. [Online].
https://www.raspberrypi.org/documentation/installation/installing-images/

[28] Snort. Manual de Snort. [Online].
http://manual.snort.org/node7.html

[29] Wikipedia. Wikipedia: La enciclopedia libre. [Online].
https://en.wikipedia.org/wiki/Large_receive_offload

[30] Jonathan Corbet. Página web LWN. [Online].
https://lwn.net/Articles/358910/

[31] Emmeshop. Página web instructables. [Online].
http://www.instructables.com/id/Telegram-on-Raspberry-Pi/?ALLSTEPS

[32] Noah Dietrich. Página web de Snort. [Online].
https://www.snort.org/documents/snort-2-9-7-x-on-ubuntu-12-lts-and-14-lts

[33] Snort. Página web de Snort. [Online].
https://snort.org/users/sign_in

[34] PulledPork. Página web de PulledPork. [Online].
https://code.google.com/p/pulledpork/

[35] Asalamon74. Página web StackExchange. [Online].
http://raspberrypi.stackexchange.com/questions/70/how-to-set-up-swap-space

[36] Digium. Manual de Asterisk. [Online].
http://www.asterisk.org/sites/asterisk/files/mce_files/documents/asterisk_quick_start_guide.pdf

[37] VoIp-info.Wiki de VoIP-Info. [Online].
http://www.voip-info.org/wiki/view/Asterisk+cmd+Festival

[38] Manuel Camargo. Página web 10000horas. [Online].
http://www.10000horas.com/2011/12/09/revisando-las-opciones-de-text-to-speech/

[39] VoIp-info.Wiki de VoIP-Info. [Online].
http://www.voip-info.org/wiki/view/Asterisk+auto-dial+out

GLOSARIO

BASE: Basic Analysis and Security Engine 3

GPIO: General-purpose input/output 24

GRO: Generic Receive Offload 30

IDS: Intrusion Detection System 2

IP: Internet Protocol 30

LED: Light-Emitting Diode 24

LRO: Large Receive Offload 30

VoIP: Voz sobre Internet 8