



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Sistema de monitorización de seguridad de una red doméstica

Grado en Ingeniería Mecánica

ALUMNO: Borja Moriano Rivera

DIRECTOR: Norberto Fernández García

CURSO ACADÉMICO: 2016-2017

UniversidadeVigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Sistema de monitorización de seguridad de una red doméstica

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

Universidade de Vigo

RESUMEN

Este proyecto abordará la creación de un sistema de monitorización de seguridad de una red doméstica. Para ello se usará una placa Raspberry Pi que actuará como elemento intermediario entre la pasarela de acceso del hogar y los dispositivos de los usuarios. Dicha Raspberry Pi se configurará como punto de acceso inalámbrico en el que se instalará el siguiente software especializado: Squid, con el propósito de interceptar, filtrar y redirigir el tráfico web tanto entrante como saliente a nuestra red de área local, SquidGuard como complemento a Squid que generará una base de datos de URLs a filtrar en base a nombres de dominio y palabras clave y Snort, que nos permitirá recibir avisos de intrusión desde el exterior a nuestra red y así como avisos de accesos no autorizados desde nuestra red a dominios o contenidos concretos. Tras la instalación de las herramientas y su configuración se han llevado a cabo una serie de pruebas de funcionamiento para validar la utilidad del sistema.

PALABRAS CLAVE

Red, Raspberry Pi, ciberdefensa, cortafuegos, proxy.

AGRADECIMIENTOS

Quiero agradecer este trabajo a mi tutor D. Norberto Fernández García por haberme guiado a lo largo de todo el proyecto, así como las pautas marcadas y las dudas resueltas. Al Centro Universitario de la Defensa por haberme proporcionado los conocimientos necesarios a lo largo de estos 5 años para desempeñar mi carrera con total profesionalidad. A Dña. Carmen García Jurado por haberme proporcionado la asesoría legal de dicho proyecto. A Marta Muriel García por apoyarme especialmente durante estos últimos meses animándome a lo largo del proyecto. A mis padres por haber sido una guía fundamental a lo largo de estos años pese a estar a mucha distancia.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	5
1 Introducción y objetivos	7
1.1 Internet. Un medio en constante crecimiento.....	7
1.2 Internet ¿Un entorno seguro?	9
1.3 Motivación y objetivos del TFG	11
1.4 Estructura de la memoria	11
2 Estado del arte	13
2.1 Trabajos previos	13
2.1.1 Captura y análisis de tráfico de red con Raspberry Pi y CapAnalysis.....	13
2.1.2 Creación de una suite de seguridad con Raspberry Pi.....	14
2.1.3 Sistemas Operativos preparados para realizar ciberataques	15
2.2 Hardware	16
2.2.1 Raspberry Pi.....	16
2.2.2 Encaminador (Router)	20
2.3 Software	21
2.3.1 Raspbian	21
2.3.2 Snort.....	21
2.3.3 Squid.....	23
2.3.4 Diladele Web Filter.....	24
2.3.5 SquidGuard	26
3 Desarrollo del TFG.....	27
3.1 Arquitectura y funcionamiento básico del sistema	27
3.2 Instalación y configuración del Sistema Operativo	28
3.2.1 Instalación del Sistema Operativo Raspbian	28
3.2.2 Configuración de la conexión remota a la Raspberry Pi	30
3.2.3 Configurar la Raspberry Pi como punto de acceso.....	31
3.3 Instalación de software de control de tráfico de red	35
3.3.1 Instalación de Snort	35
3.3.2 Instalación de Squid y SquidGuard	38
3.3.3 Reglas de iptables	47
3.3.4 Creación de scripts de aviso	48

4 Pruebas	50
4.1 Filtrado de URL's	50
4.2 Filtrado por palabras/repositorio Black List	51
4.3 Aviso via mail de intrusión no deseada	52
4.4 Pruebas de carga y tiempos de respuesta	54
4.5 Prueba de número de equipos conectados simultáneamente	55
5 Conclusiones y líneas futuras	56
5.1 Conclusiones	56
5.2 Líneas Futuras	57
6 Bibliografía.....	58
Anexo I: Contenido HTML	61
Anexo II: Scripts	62

ÍNDICE DE FIGURAS

Figura 1-1 Evolución del porcentaje de viviendas (2006-2016) que disponen de acceso a Internet. Eje X años y eje Y porcentaje. (Fuente: www.ine.es)	7
Figura 1-2 Gráfico de Población que ha utilizado Internet en los últimos 3 meses. Enero de 2017. (Fuente www.ine.es)	8
Figura 2-1 CapAnalysis leyendo un archivo .pcap (Fuente www.securityartwork.es)	13
Figura 2-2 CapAnalysis filtrando por países de destinos los paquetes (Fuente www.securityartwork.es)	14
Figura 2-3 Sistema Operativo PwnPi 3.0 (Fuente: pwnpi.sourceforge.net)	15
Figura 2-4 Sistema Operativo Kali Linux (Fuente: www.kali.org)	16
Figura 2-5 Sistema Operativo Poison Tap (Fuente: www.samy.pl)	16
Figura 2-6 Primeras placas de circuitos basadas en microcontroladores (Fuente MagPi ed. nº1) ...	17
Figura 2-7 Primer prototipo de Raspberry Pi del tamaño de un stick USB (Fuente MagPi ed. nº1)	17
Figura 2-8 Primeras placas "Alfa" sostenida por el co-desarrollador Eben Upton (Fuente MagPi ed. nº1)	18
Figura 2-9 Raspberry Pi 2 Model B (Fuente MagPi nº31)	19
Figura 2-10 Raspberry Pi 3 (Fuente: http://tech.scargill.net/raspberry-pi-3-grand-opening/)	20
Figura 2-11 Router D-Link DIR 615 (Fuente: www.dlink.com)	20
Figura 2-12 OS Raspbian (Fuente: www.raspbian.org)	21
Figura 2-13 Sistema de detección de intrusiones Snort (Fuente: www.snort.org)	21
Figura 2-14 Esquema de servidor proxy (Fuente: autoría propia)	23
Figura 2-15 Filtrado de anuncios por idiomas en Web Filter (Fuente: autoría propia)	24
Figura 2-16 Configuración del bloqueo de anuncios en Web Filter (Fuente: autoría propia)	25
Figura 2-17 Boqueo de URL por categorías de Web Filter (Fuente: autoría propia)	25
Figura 2-18 Configuración de Squid en modo transparente en Web Filter (Fuente: autoría propia)	26
Figura 2-19 Logotipo de SquidGuard (Fuente: www.squidguard.org)	26
Figura 3-1 Arquitectura del sistem propuesto (Fuente: autoría propia)	27
Figura 3-2 SD Formatter (Fuente: Autoría propia)	28
Figura 3-3 Comando diskutil (Fuente: Autoría propia)	29
Figura 3-4 Escritorio Raspbian (Fuente: Autoría propia)	29
Figura 3-5 Imágen de <i>Raspberry Configuration Tool</i> (Fuente: Autoría propia)	30
Figura 3-6 Submenú de interfaz de Raspberry Configuration Tool (Fuente: Autoría propia)	30
Figura 3-7 Fichero interfaces (Fuente: Autoría propia)	32
Figura 3-8 Redes inalámbricas disponibles (Fuente: Autoría propia)	33

Figura 3-9 Archivo <i>snort.debian.conf</i> (Fuente: Autoría propia)	36
Figura 3-10 Archivo de configuración de Snort (Fuente: Autoría propia).....	36
Figura 3-11 Archivo de configuración de Snort (Fuente: Autoría propia).....	37
Figura 3-12 Archivo <i>local.rules</i> (Fuente: Autoría propia)	37
Figura 3-13 Archivo <i>sources.list</i> (Fuente: Autoría propia)	38
Figura 3-14 Fichero <i>debian.rules</i> (Fuente: Autoría propia)	40
Figura 3-15 Imágen de cómo quedaría el directorio <i>"/home/pi"</i> (Fuente: Autoría propia).....	41
Figura 3-16 fichero <i>testdomains</i> creado para SquidGuard (Fuente: Autoría propia).....	42
Figura 3-17 Fichero <i>squid.conf</i> (Fuente: Autoría propia)	44
Figura 3-18 Carpeta <i>gamble</i> de la base de datos de SquidGuard (Fuente: Autoría propia).....	47
Figura 3-19 Diagrama de flujo del script de aviso (Fuente: Autoría propia)	49
Figura 4-1 Comprobación de filtrados de URL con Squid (Fuente: Autoría propia)	50
Figura 4-2 Búsqueda de "futbol" en el buscador de Google (Fuente: Autoría propia)	51
Figura 4-3 Squid bloqueando la búsqueda de "futbol" (Fuente: Autoría Propia)	52
Figura 4-4 Correo electrónico de aviso (Fuente: Autoría propia)	52
Figura 4-5 Capturas de paquetes mediante Wireshark (Fuente: autoría propia)	53
Figura 4-6 Fichero de alertas de Snort (Fuente: Autoría propia)	54
Figura 4-7 Ejecución del script "load_time" (Fuente: Autoría propia)	55
Figura 5-1 E2guardian (Fuente: e2guardian.org)	57
Figura 5-2 BroIDS (Fuente: bro.org)	57

ÍNDICE DE TABLAS

Tabla 1-1 Datos del último estudio del INE (Fuente www.ine.es)	8
Tabla 1-2 Procedimientos incoados en 2015. Resumen Fiscalía General del Estado año 2016 (Fuente: www.fiscal.es)	9
Tabla 4-1 Comprobación de Benchmarks I (Fuente: Autoría propia).....	55
Tabla 4-2 Comprobación de Benchmarks II (Fuente: Autoría propia)	55

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Internet. Un medio en constante crecimiento.

Vivimos en una sociedad en la cual ha ido creciendo exponencialmente el uso de Internet con diversos fines adaptando las nuevas tecnologías a tareas tan cotidianas como podrían ser leer el periódico, escuchar música o mantener conversaciones en tiempo real a través de videoconferencias o mensajería instantánea. Según el último estudio del Instituto Nacional de Estadística (INE) con fecha de Octubre de 2016, más de 3 millones de hogares españoles con 4 miembros disponen de acceso a Internet [1]. En la Figura 1-1 del INE vemos como ha sido esa evolución a lo largo de los años:

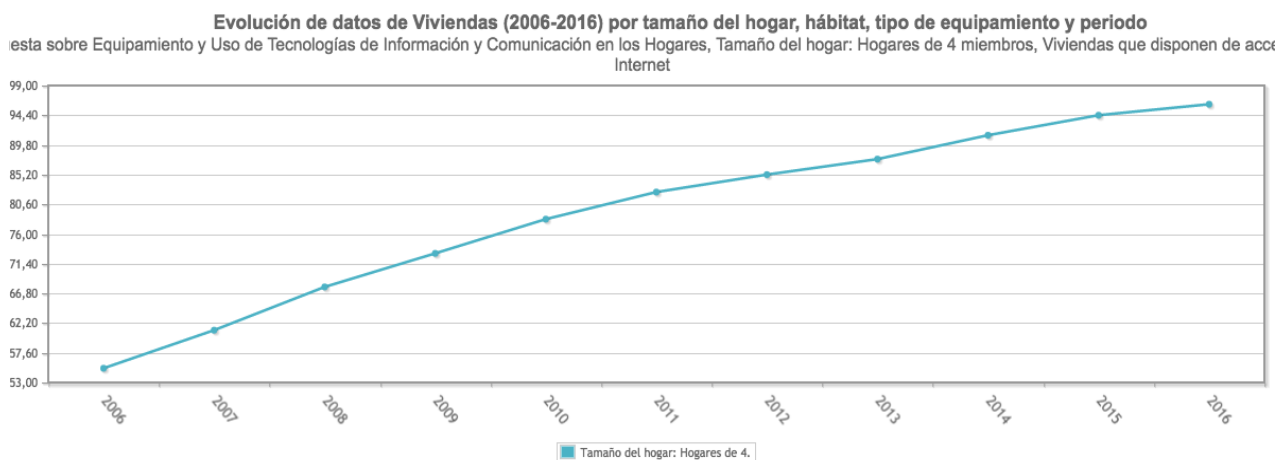


Figura 1-1 Evolución del porcentaje de viviendas (2006-2016) que disponen de acceso a Internet. Eje X años y eje Y porcentaje. (Fuente: www.ine.es)

Así mismo si nos fijamos en los últimos resultados de las encuestas del INE, vemos resultados esclarecedores de que la población española está apostando por esta tecnología [2]:

	Valor en %	Variación respecto al año anterior
Hogares con conexión a Internet	81,9	3,2
Hogares con conexión de banda ancha	81,2	3,4
Personas que han usado Internet	80,6	1,9
Usuarios frecuentes de Internet (al menos 1 vez/semana en los últimos 3 meses)	76,5	1,8
Usuarios que han realizado compras a través de Internet (últimos 3 meses)	34,9	2,8

Tabla 1-1 Datos del último estudio del INE (Fuente www.ine.es)

Podemos apreciar que los hogares españoles apuestan por el uso de Internet. Es más, si nos fijamos en la Figura 1-2 podremos observar que el 98,4% de los jóvenes entre 16 y 24 años ha hecho uso de Internet en los últimos 3 meses [3]:

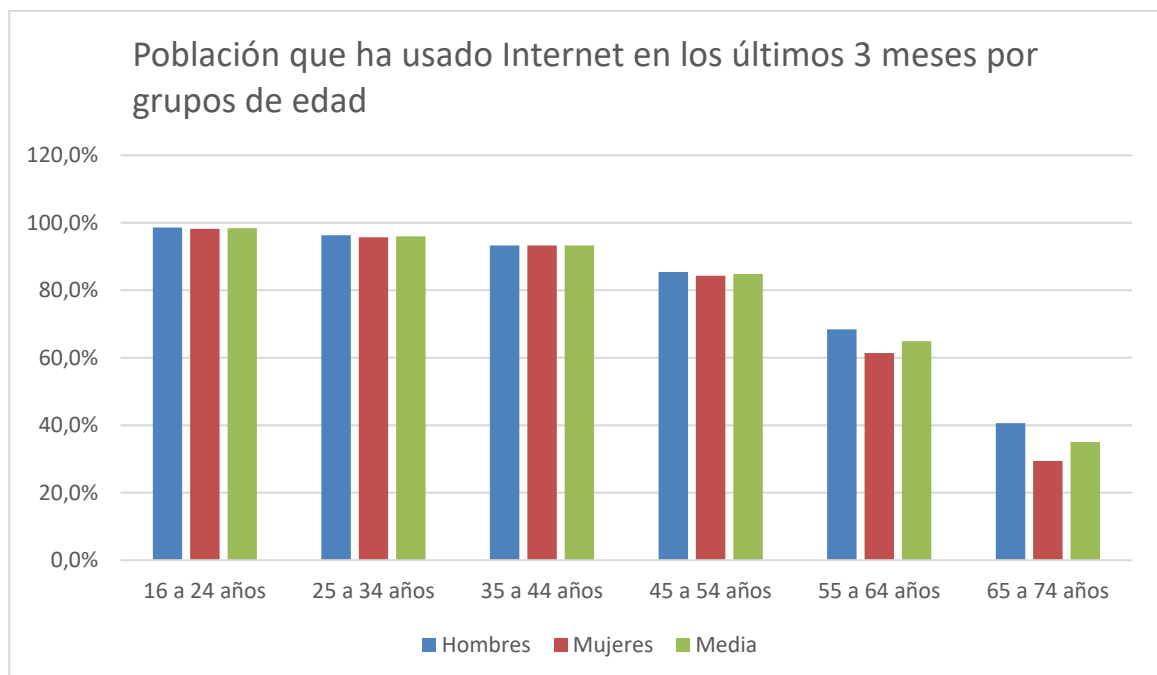


Figura 1-2 Gráfico de Población que ha utilizado Internet en los últimos 3 meses. Enero de 2017. (Fuente www.ine.es)

1.2 Internet ¿Un entorno seguro?

Como mencionamos en el apartado anterior un gran número de usuarios españoles hace uso de esta “reciente” tecnología, pero también hemos de ser cautelosos a la hora de introducirnos en un mundo completamente desconocido a nivel usuario sin conocimientos básicos de redes.

Y es aquí donde entran los términos de ciberataque y ciberseguridad. Si buscamos en el diccionario de la RAE (Real Academia Española de la lengua) no encontraremos dichas palabras, sino que encontraremos palabras compuestas por el prefijo “ciber-” que *“Indica relación con redes informáticas”* [4] y por las palabras “ataque” que significa en este contexto *“Acción de atacar, acometer o emprender una ofensiva”* [5] y “seguridad” cuya acepción aquí es *“Cualidad de seguro, libre o exento de riesgo”* [6]

Para confirmar que Internet es un entorno inseguro basta con ir al resumen de la Fiscalía General del estado del año 2016, concretamente al capítulo tercero sección 8, en la cual se resumen los procedimientos incoados como se ve en la Tabla 1-2:

Delitos informáticos		Procedimientos judiciales	%
Contra la libertad	Amenazas/coacciones a través de TICs (arts. 169 y ss. y 172 y ss.).	1.009	4,47
	Acoso a través de TICs (art. 172 ter).	96	0,43
Contra la integridad moral	Trato degradante a través de TICs (art. 173).	226	1,00
Contra la libertad sexual	Pornografía infantil/discapaces a través de TICs (art. 189).	767	3,40
	Acoso menores a través de TICs (art. 183 ter).	98	0,43
	Otros delitos c/libertad sexual a través TIC.	77	0,34
Contra la intimidad	Ataques/interceptación sistemas y datos (art. 197 bis y ter).	220	0,97
	Descubrimiento/revelación secretos a través TIC (art. 197).	597	2,64
Contra el honor	Calumnias/injurias autoridades a través TIC (art. 215).	254	1,13
Contra el patrimonio y el orden socio-económico	Estafa cometida a través de las TICs (art. 248 y 249).	18.201	80,62
	Descubrimiento secretos empresa a través TIC (arts. 278 y ss.).	276	1,22
	Delitos c/ servicios de radiodifusión/ interactivos (art. 286).	27	0,12
	Delitos de daños informáticos (arts. 264, 264 bis y 264 ter).	295	1,31
	Delitos c/ propiedad intelectual a través TIC (art. 270 y ss.).	70	0,31
De falsedad	Falsificación a través de la TICs.	193	0,85
Contra Constitución	Discriminatoria a través TIC (art. 510).	40	0,18
Otros.		129	0,57
Total		22.575	100,00

Tabla 1-2 Procedimientos incoados en 2015. Resumen Fiscalía General del Estado año 2016 (Fuente: www.fiscal.es)

El Instituto Nacional de Ciberseguridad perteneciente al Ministerio de Energía, Turismo y Agenda Digital ha publicado en Febrero de 2017 el ranking de los diez principales incidentes de ciberseguridad producidos en 2016:

“El ranking de los principales “sucesos” de ciberseguridad, que vuelve a poner de manifiesto que ningún sector es inmune a los ataques y que es importante seguir los procedimientos de seguridad, lo encabeza el robo de 81 millones de dólares al Banco Central de Bangladés perpetrado por piratas informáticos que lograron acceder a los sistemas informáticos del Banco y transferir esa cantidad de dinero a varios casinos de Filipinas. Un error ortográfico en el nombre de uno de los destinatarios levantó las alarmas evitando así el mayor robo de la historia, puesto que en realidad se habían realizado 35 peticiones para obtener casi 1.000 millones de dólares.

El segundo puesto lo ocupa el robo de unos 64 millones de dólares en bitcoins a la plataforma de intercambio Bitfinex de Hong Kong, el mayor operador mundial de intercambio de bitcoin basado en dólares, lo que provocó una caída de la cotización del bitcoin superior al 23 por ciento.

La tercera posición del ranking es para la publicación de datos de 154 millones de votantes de Estados Unidos. Los datos incluían información personal como dirección, correo electrónico, número de teléfono o enlaces a redes sociales.

En cuarto lugar, encontramos la publicación de datos personales de 93 millones de ciudadanos de México, debido a la defectuosa configuración de la base de datos MongoDB utilizada por el Instituto Nacional Electoral de México. Este hecho, en el que quedaron expuestos detalles como la dirección postal y la fecha de nacimiento, levantó un gran revuelo en un país donde los secuestros son un crimen habitual.

En la quinta posición aparece el robo de 1.000 millones de cuentas a Yahoo!. Además de fechas de nacimiento, direcciones de correo electrónico, números de teléfono, contraseñas en MD5, la información robada también contenía preguntas y respuestas de seguridad sin cifrar.

Otro suceso relacionado con la misma compañía ocupa el puesto sexto. Concretamente el robo de 500 millones de cuentas en un ataque que de acuerdo a la información publicada podría estar respaldado por un estado, ocurrido en a finales de 2014. Entre los datos robados, de nuevo se encontraban preguntas y respuestas de seguridad sin cifrar.

Continuando con este ranking, en séptima posición aparece el robo de 400 millones de cuentas de Friend Finder Network Inc., compañía que gestiona diferentes páginas de citas. La información robada incluía datos personales, como correo electrónico, patrones de navegación, patrones de compra y orientación sexual de los usuarios.

El octavo puesto es para el mayor ataque de DDoS producido hasta la fecha con dispositivos de Internet de las cosas (IoT). Producido por la botnet Mirai, compuesta por cientos de miles de cámaras IP junto a otros dispositivos IoT, dejó fuera de juego a múltiples servicios de Internet, llegando a afectar a PlayStation Network y Twitter, entre otros. Además se sospecha que este ciberataque habría sido también una prueba de concepto para afectar al funcionamiento de los sistemas de voto electrónico de EE.UU. previamente a las elecciones presidenciales del 8 de noviembre.

El noveno lugar de esta clasificación lo ocupa el fallo en la implementación de la pila TCP en sistemas Linux posteriores a la versión de Kermel 3.6. Este fallo, que permitía la infección de malware de forma remota o incluso el secuestro del tráfico de los usuarios, afectaba a todos los dispositivos móviles con sistema operativo Android y a las televisiones inteligentes, además de degradar las conexiones mediante el protocolo Https y redirigir el tráfico.

Cierra este ranking el fallo en los procesadores Qualcomm que permitía acceder a la información cifrada sin que se activaran los mecanismos de borrado. Esta vulnerabilidad en la generación de las claves de cifrado afectó aproximadamente al 60% de los móviles Android del mercado.” [7]

Otra motivación sería la alerta que hay que mantener respecto a nuestros dispositivos móviles, dado que recientemente se han descubierto casos de *malware* instalado por intermediarios ejecutándose en segundo plano dando múltiple información como veremos en el apartado 2.1.1.

Esto pone de manifiesto el peligro que supone navegar por Internet sin unas nociones básicas de ciberseguridad. Para más información se puede consultar la siguiente dirección [8] para obtener un informe actual de los ciberataques acontecidos en el día.

1.3 Motivación y objetivos del TFG

Dado que según el apartado 1.2 Internet no es un medio seguro, necesitamos una serie de herramientas de seguridad para disminuir el impacto de los ciberataques en nuestros equipos.

Así pues el objetivo del presente trabajo consiste en estudiar la viabilidad de emplear herramientas de código libre (*open source*) gratuitas y un dispositivo hardware de bajo coste (una placa Raspberry Pi) como sistema que permita monitorizar el tráfico en una red local. Con este sistema de monitorización se pretende mejorar la seguridad de la red local, restringiendo el acceso a sitios web potencialmente peligrosos de los usuarios domésticos y detectando/evitando intrusiones externas.

1.4 Estructura de la memoria

En dicha Memoria se va a documentar todo el proceso de creación de firewall que monitorizará el tráfico de red de nuestro hogar. Para ello se seguirá el siguiente índice:

Capítulo 1: Introducción

En este capítulo se comenta el uso de Internet en hogares y el problema que supone su seguridad

Capítulo 2: Estado del arte: Hardware y Software a usar

En dicho capítulo se hablará de los dispositivos y programas a utilizar en nuestro proyecto para monitorizar el tráfico de red.

Capítulo 3: Desarrollo e instalación de software

Aquí se detallará la instalación y configuración de los programas

Capítulo 4: Pruebas

En este capítulo realizaremos las pruebas pertinentes para comprobar que los programas instalados funcionan correctamente

Capítulo 5: Conclusiones

Aquí se hablará de los resultados obtenidos en el proyecto, así como de las líneas futuras a seguir.

2 ESTADO DEL ARTE

2.1 Trabajos previos

Antes de abordar la creación de un dispositivo capaz de satisfacer los objetivos del presente proyecto, es necesario conocer los trabajos que hay en este campo hasta ahora.

2.1.1 Captura y análisis de tráfico de red con Raspberry Pi y CapAnalysis

Yolanda Olmedo (grupo de auditorias de seguridad “s2”) en la web www.securityartwork.es creó un artículo sobre cómo usó su Raspberry Pi 3 Model B para monitorear sus dispositivos de origen chino, concretamente dos móviles y una tableta, debido a que tenía sospechas de que estos enviaban información sensible al exterior a través de aplicaciones trabajando en segundo plano como “Analyticscore.apk” [9]. Mediante el programa CapAnalysis (cuya interfaz gráfica se muestra en la Figura 2-1) consigue clasificar y filtrar los paquetes enviados:

Date ↑	Time	Source IP	Destination IP	Destination Name	Source Port	Destination Port	L4	Protocol	Country	Bytes
2016-09-16	18:01:00	31.13.83.4	31.13.83.4	scontent-mad1-1.xx.fbcdn.net	50178	443	TCP	SSLFacebook	IE	1.1 K
2016-09-16	18:01:00	31.13.93.3	31.13.93.3	graph.facebook.com	55552	443	TCP	SSLFacebook	IE	2.3 K
2016-09-16	18:01:00	8.8.8.8	8.8.8.8		15179	53	ESP	DNS	US	46
2016-09-16	18:01:00	8.8.8.8	8.8.8.8		29779	53	ESP	DNS	US	36
2016-09-16	18:00:05	31.13.83.4	31.13.83.4	scontent-mad1-1.xx.fbcdn.net	50176	443	TCP	SSLFacebook	IE	1.1 K
2016-09-16	18:00:05	8.8.8.8	8.8.8.8		42170	53	ESP	DNS	US	46
2016-09-16	18:00:04	179.60.192.3	179.60.192.3	graph.facebook.com	55332	443	TCP	SSLFacebook	US	1.7 K
2016-09-16	18:00:04	8.8.8.8	8.8.8.8		16015	53	ESP	DNS	US	36
2016-09-16	17:59:54	216.58.201.142	216.58.201.142	app-measurement.com	53667	80	TCP	HTTPGoogle	US	176
2016-09-16	17:55:06	216.58.201.142	216.58.201.142	app-measurement.com	56439	80	TCP	HTTPGoogle	US	352
2016-09-16	17:55:06	8.8.8.8	8.8.8.8		21677	53	ESP	DNS	US	37
2016-09-16	17:52:11	52.43.240.174	52.43.240.174	shavar.services.mozilla.com	58713	443	TCP	SSL	US	685
2016-09-16	17:52:11	8.8.8.8	8.8.8.8		3062	53	ESP	DNS	US	46

Figura 2-1 CapAnalysis leyendo un archivo .pcap (Fuente www.securityartwork.es)

CapAnalysis es una herramienta de código abierto que nos permite analizar grandes cantidades de paquetes. Unas de las interfaces que ofrece dicho programa es la representación gráfica de los países a los que se envían los paquetes. En la Figura 2-2 se puede ver un ejemplo de uso de esta interfaz, siendo los países destino los siguientes: EEUU, Reino Unido, Irlanda, Alemania, Japón y China.

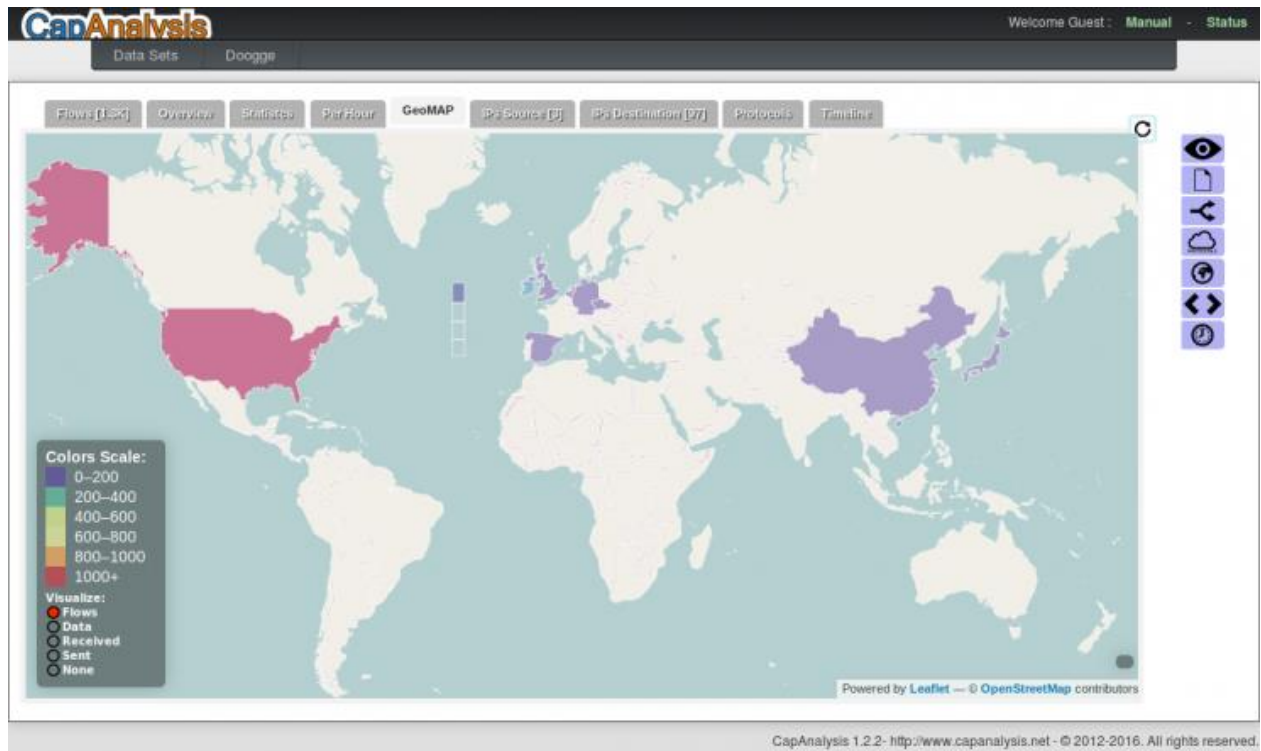


Figura 2-2 CapAnalysis filtrando por países de destinos los paquetes (Fuente www.securityartwork.es)

Centrándose en los paquetes con destino China y usando la aplicación Wireshark descubre que se envían a una empresa dedicada al alojamiento web (*hosting*), concretamente www.adsunflower.com los siguientes datos: Marca, Modelo, Número de serie, “*International Mobile station Equipment Identity*” (IMEI), dirección “*Media Access Control*” (MAC) e Identidad Internacional del Abonado a un Móvil o *International Mobile Subscriber Identity* (IMSI). [10]

2.1.2 Creación de una suite de seguridad con Raspberry Pi

En este caso Travis Smith de la web www.tripwire.com aprovecha este dispositivo para analizar el tráfico malicioso que pudiese llegar a sus dispositivos que cuentan con IoT (Internet de las cosas). Para ello usa diferentes programas:

- Bro IDS: Creado por Vern Paxton en 1995, Bro es un detector de intrusiones basado en eventos,. Se caracteriza por su bajo consumo de recursos, pero su nivel de complejidad a la hora del manejo a nivel usuario es alto. [11]
- Critical Stack: software libre encargado de mantener una base de datos de amenazas actualizadas. Básicamente su funcionamiento consta de dos partes: el usuario escoge el tipo de amenaza que quiere detectar con Bro y Critical Stack se encarga de transcribirla al lenguaje de Bro y este la carga mediante guiones (*scrpits*) automáticamente. [12]

- Logstash: Software utilizado para alertarnos vía correo electrónico de alguna anomalía en el registro de Bro cuando este detecta alguna amenaza. [13]
- Elasticsearch: Permite almacenar los registros o “logs” de Logstash. [14]
- Kibana: Es una herramienta gráfica que permite al usuario visualizar los eventos de los registros, obteniendo una idea general de las amenazas a un simple golpe de vista. Además podrá depurar las detecciones de Bro con posibles falsos positivos, así como analizar los paquetes recogidos con software como CapAnalysis. [15]

2.1.3 Sistemas Operativos preparados para realizar ciberataques

Mientras que en el apartado anterior se describía la posibilidad de crear un sistema de defensa frente a ciberataques, la Raspberry Pi también puede ser utilizada como herramienta para efectuar ataques. Hay una distribución dedicada exclusivamente a esto que se llama *PwnPi* (Figura 2-3). Esta distribución basada en Linux permite realizar auditorías de seguridad con más de 200 herramientas de *pentesting* que abarcan desde la obtención de claves de redes Wi-Fi privadas hasta la realización de ataques *Man-In-The-Middle* para ver los contenidos cifrados por *Secure Socket Layer*. [16]

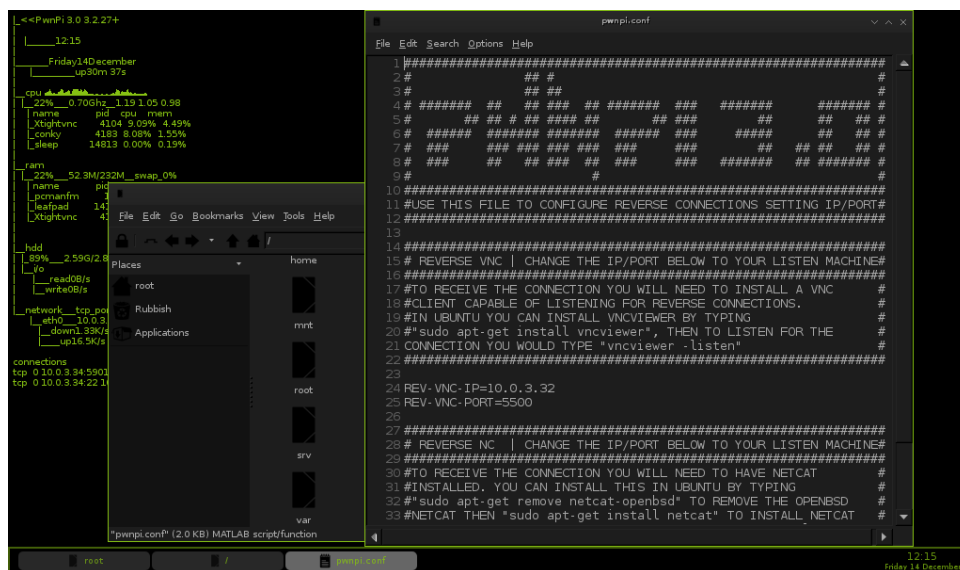


Figura 2-3 Sistema Operativo PwnPi 3.0 (Fuente: pwnpi.sourceforge.net)

Por otro lado tenemos otra distribución de código abierto y completamente gratis basada en Debian llamada Kali Linux [17], versión actualizada de la distribución basada en Linux BackTrack. Kali Linux (Figura 2-4) es una de las distribuciones más usadas para auditorías de seguridad por su gran abanico de herramientas, más de 300, que abarcan desde una simple sustracción de contraseñas al análisis forense. Además cuenta con una gran cantidad de adaptaciones, para utilizarla en plataformas que no son comunes por ejemplo en las Raspberry Pi.

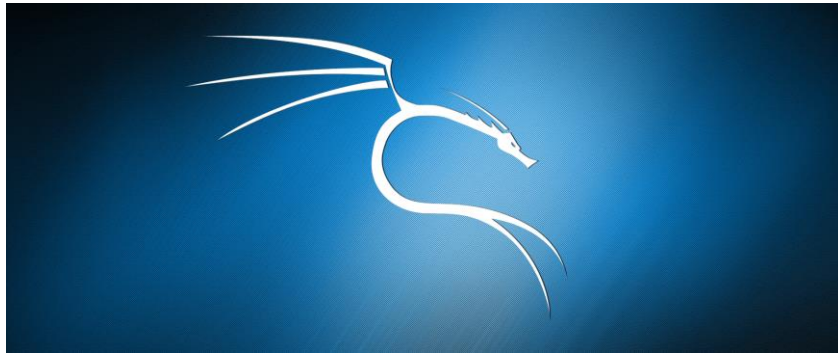


Figura 2-4 Sistema Operativo Kali Linux (Fuente: www.kali.org)

En Noviembre de 2016 el ingeniero informático Samy Kamkar crea una distribución para Raspberry Pi llamada Poison Tap (Figura 2-5) capaz de sustraer contraseñas de cualquier ordenador protegido con contraseña. Emulando una conexión nueva de *Ethernet*, crea una puerta trasera permanente en el equipo. Simplemente teniendo un navegador abierto puede empezar a sustraer las contraseñas en este y enviarla al atacante. [17]



Figura 2-5 Sistema Operativo Poison Tap (Fuente: www.samy.pl)

2.2 Hardware

Después de exponer los trabajos previos, vamos a explicar el hardware o conjunto de dispositivos físicos que vamos a necesitar en nuestro proyecto.

2.2.1 Raspberry Pi

La Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito creado por Eben Upton, David Braben, Jack Lang, Pete Lomas, Alan Mycroft y el doctor Robert Mullins.

El proyecto Raspberry Pi empezó en 2006 en la Universidad de Cambridge. Con este proyecto se intentaba recuperar el espíritu de la computación de los 80. La idea era construir un ordenador basado en microcontroladores con el lenguaje de programación Python (Figura 2-6), de ahí el nombre de “Pi” en Raspberry Pi.

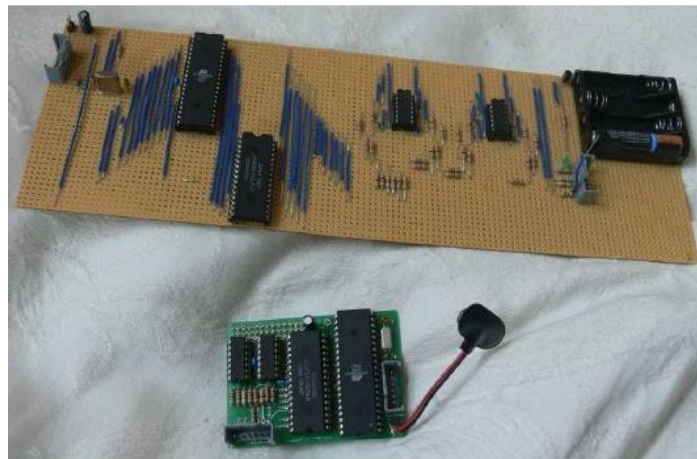


Figura 2-6 Primeras placas de circuitos basadas en microcontroladores (Fuente MagPi ed. n°1)

Debido al coste de tener que producir un sistema operativo que incluyese controladores de dispositivos, capacidad de soportar el uso de redes, etc. se optó por usar un sistema operativo de fuente abierta como Linux, de esta manera permitirían la libertad de usar diferentes lenguajes de programación así como otro tipo de software.

Del uso de microcontroladores se pasó a la idea del “Sistema en un Chip” (*System on a Chip*, SoC en inglés), del cual se escogió uno de la compañía Broadcom. La idea de la Fundación Raspberry Pi fue desarrollar una placa SoC con el procesador ARM11 de Broadcom, de bajo coste, para así potenciar el que nuevos ingenieros informáticos se fuesen sumando al proyecto.

El primer prototipo (Figura 2-7) fue del tamaño de una memoria USB, con un puerto USB en un extremo a un lado y una salida HDMI al otro. Este prototipo ya usaba una tarjeta MicroSD para almacenar el sistema operativo. En este punto los desarrolladores se dieron cuenta de que era demasiado pequeño para añadir el puerto de red, las salidas de audio y video y el conjunto de pines de uso general (General Purpose In/Out, GPIO en inglés).



Figura 2-7 Primer prototipo de Raspberry Pi del tamaño de un stick USB (Fuente MagPi ed. n°1)

En este punto se decide adoptar el tamaño de una tarjeta de crédito y es cuando empiezan a colaborar con la comunidad de desarrolladores de Linux para adaptarse al procesador ARM1176JZF-S a 700 MHz.



Figura 2-8 Primeras placas "Alfa" sostenida por el co-desarrollador Eben Upton (Fuente MagPi ed. n°1)

Las placas fueron dotadas de un adaptador de corriente en forma de micro USB, de cara a que en un futuro cualquier persona con un cargador de móvil pudiese usarlas. Su lanzamiento fue el 29 de Febrero de 2012. [18]

Con el paso del tiempo nace la Raspberry Pi Model 2 (Figura 2-9) que supone un gran avance en cuanto a hardware, ya que podemos desde ver películas usando esta como una Smart TV hasta crear nuestros propios videojuegos. Esto se debe a la mejora del procesador, incluyendo ahora un ARM Cortex-A7 de cuatro núcleos a 900MHz y 1GB de memoria RAM. En cuanto a su precio sigue siendo muy barata, utilizando como estrategia de mercado la frase *“Con el dinero que gastas en el Café Starbucks en una semana podrías tener una”* [20]. Su fecha de lanzamiento fue el 2 de Febrero de 2015.

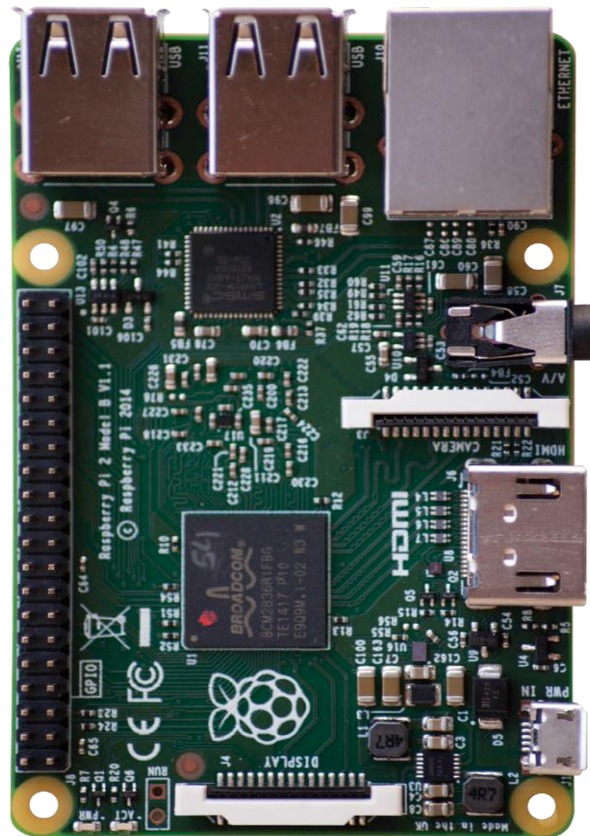


Figura 2-9 Raspberry Pi 2 Model B (Fuente MagPi n°31)

En Marzo de 2016 llega Raspberry Pi 3 B (Figura 2-10) con las novedades siguientes:

- Tarjeta de red para Wireless y Bluetooth de la marca Broadcom: este elemento solo se puede apreciar en la placa con un microscopio o unas lentes de aumento. La tarjeta de red modelo BCM43438 proporciona conectividad a la red de manera inalámbrica y Bluetooth
- Sistema en chip (SOC): Hecho especialmente para Raspberry Pi 3, la compañía Broadcom ha desarrollado dicho chip con un procesador ARM con cuatro núcleos modelo A53 Cortex a 1,2 GHz con 32 kB de memoria caché de nivel 1 y 512 kB de nivel 2, un procesador gráfico Videocore IV y una memoria RAM de 1GB LPDDR2.

En cuanto al resto de componentes, conserva los mismos modelos que su predecesora. [21]

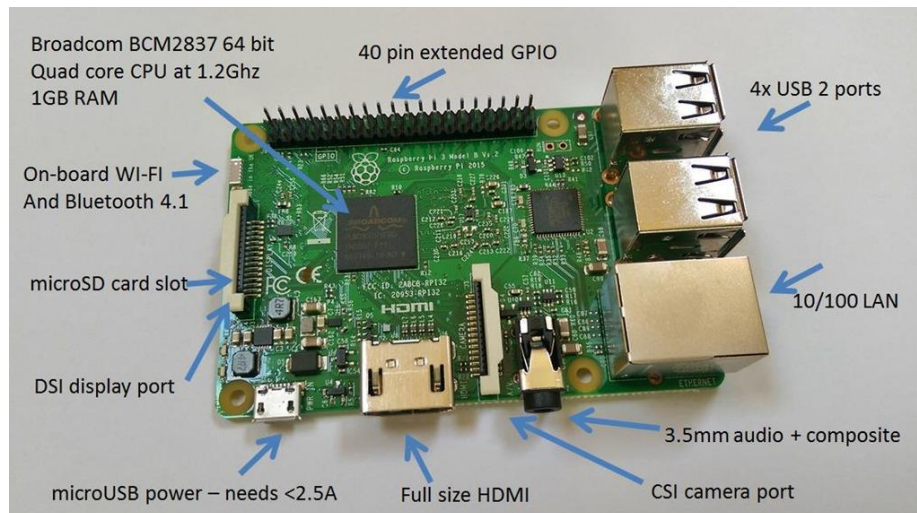


Figura 2-10 Raspberry Pi 3 (Fuente: <http://tech.scargill.net/raspberry-pi-3-grand-opening/>)

Tras ver la evolución de la Raspberry Pi a lo largo de los años, se opta para el proyecto por el modelo 3 B debido a la inclusión de la tarjeta de red inalámbrica, no teniendo que comprar ninguna tarjeta de red externa a la placa, así como un procesador más potente que el modelo 2.

2.2.2 Encaminador (Router)

Dado que la mayoría de los hogares disponen de un encaminador, a la hora de llevar a cabo nuestro proyecto asumiremos que se dispone de uno. Normalmente disponemos de uno cuando contratamos un servicio de Internet con alguna compañía (ISP o *Internet Service Provider*, encargado de proporcionar el “enlace” a Internet desde tu casa). En nuestra prueba contamos con el *router* de la marca D-Link modelo DIR-615 (Figura 2-11) con las siguientes características: [21]

- Soporte de protocolos estándar 802.11n/g , 802.3 y 802.3u
- Trabajo en banda de 2,4 GHz
- Soporte a WPA/WPA2
- Acceso vía WPS (PIN)
- Filtrado de MAC's /URL



Figura 2-11 Router D-Link DIR 615 (Fuente: www.dlink.com)

2.3 Software

Una vez se ha descrito los dispositivos a emplear en nuestro proyecto, vamos a explicar los programas o *software* a emplear.

2.3.1 Raspbian



Figura 2-12 OS Raspbian (Fuente: www.raspbian.org)

Raspbian (Figura 2-12) es el sistema operativo (*Operative System* o por sus siglas OS) que vamos a utilizar en nuestra Raspberry Pi 3. Dicho OS nace en 2012 de la mano de los desarrolladores Mike Thompson y Peter Green y proviene de Debian Jessie 8.0 en su adaptación “armhf” (para procesadores ARM). Está pensada para utilizar Debian en sistemas pequeños y obsoletos, así como los más actuales de arquitectura de 32 bits, como es el caso de nuestra Raspberry. Instalando Raspbian se consigue que se suavice el consumo de recursos de procesos que antes requerían de muchos, como Squid, del que hablaremos más adelante. Cabe decir que es una adaptación no oficial de Debian. [22]

2.3.2 Snort



Figura 2-13 Sistema de detección de intrusiones Snort (Fuente: www.snort.org)

Lo primero será definir lo que es una intrusión para introducirnos a este tipo de programas:

“Llamaremos intrusión a un conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso” [23]

Con esta definición podríamos describir la base del Sistema de Detección de Intrusiones (IDS). Nos podríamos llegar a preguntar si haría realmente falta dicho sistema teniendo normalmente por defecto los ordenadores un cortafuegos. La respuesta es sí, porque en caso de que el cortafuegos fallara tendríamos un detector externo a nuestro equipo que nos podría avisar de la intrusión y conseguir algo más de seguridad. Por lo menos a la hora de detección y aviso.

Estos sistemas comenzaron a desarrollarse en 1980. Las cuatro características principales que deben regir un IDS son:

- Independencia, es decir, que el IDS se ejecute de manera continua sin necesidad de la supervisión de un usuario humano, independientemente de los protocolos de seguridad a aplicar previamente introducidos por una persona. También este ha de ser transparente al resto de procesos y que el usuario y atacante no se dé cuenta de que se están ejecutando.
- Aceptabilidad, o capacidad del sistema de soportar la carga de trabajo producida por el IDS en cuestión, así como la capacidad de minimizar los falsos positivos generados por este, ya que si no confiamos en las alertas generadas, de nada nos sirve.
- Adaptabilidad, o capacidad de aceptar los cambios que se producen mediante las actualizaciones del sistema operativo, teniendo en cuenta hasta los más mínimos cambios en el núcleo o *kernel* de nuestro sistema (Linux).
- Tolerancia a fallos, teniendo en cuenta que nuestro equipo estará sometido a cambios inesperados en el funcionamiento del mismo como por ejemplo un reinicio inesperado o un intento de engaño por parte del atacante, para burlar la seguridad que nos proporciona el IDS.

Contamos con dos grandes familias de IDS: por un lado tenemos los *Host IDS* (HIDS) que buscan las amenazas en nuestra propia máquina y por otro lado los *Network-based IDS* (NIDS) que buscan fuera de nuestra propia máquina en la red.

Hay tres tipos de IDS basados en host o máquina:

- Verificadores de contenido mediante huellas hash y la creación de una base de datos de las originales de cara a comparar cada cierto tiempo, asegurando su integridad almacenándolas en medios portátiles como USBs o CDs. Un ejemplo claro de implementación sería aplicarlo a los ficheros de contraseñas. Esto daría a simple vista una gran sensación de seguridad, pero en el momento que cambiemos la contraseña va a saltar nuestra alarma
- *Honeypots*: Estos IDS se encargan de crear un sistema “trampa” para el atacante, haciéndole creer que está entrando a donde él quiere con total impunidad y pasando desapercibido, cuando realmente habremos conseguido “encerrar” el ejecutable del atacante, alertarnos vía correo electrónico e incluso generar una respuesta automática bloqueando la dirección del atacante.
- Analizadores de registro, encargados de ver minuciosamente los registros o *logs* de cada una de las actividades generadas en la máquina pudiendo llegar a diferenciar cuales de ellas pueden llegar a ser potencialmente amenazas para la seguridad, llamadas vulnerabilidades, por donde puede pasar nuestro atacante.

Nuestro IDS, Snort, se trata de un sistema NIDS basado en comprobación de reglas, empleando el rastreo de paquetes o “*packet sniffer*” para detectar intrusiones en la red. Básicamente la máquina donde estamos ejecutando Snort actuará en modo promiscuo, viendo todo el tráfico que pasa por la red y posteriormente analizándolo. Para ello utiliza unos campos a modo de “filtros” para identificar posibles patrones de ataque e informarlos. A continuación se nombrarán los más comunes con una regla sencilla:

<pre>alert tcp 192.168.1.0/24 any -> 151.80.138.20 80 (sid: 1000001; msg: "Acceso a servidor no autorizado";)</pre>
--

- Tipo de paquete, en este caso TCP.
- Dirección/Rango IP origen/destino: 19.168.1.0/24 origen y 151.80.138.20 destino.
- Puerto origen y destino: Aquí el primero es cualquiera (*any*) y destino 80.
- Campos de datos. Aquí aparece el identificador de la regla (*sid*) y el mensaje de alerta que se genera cuando la regla se dispara, es decir, cuando desde una IP de origen se intenta acceder a IP destino por el puerto 80.

Para explicar el concepto de comprobación de reglas nos basaremos en lo siguiente: imaginemos que tenemos un cortafuegos o firewall instalado en nuestro equipo, y que sólo tenemos habilitado el puerto número 80 que es el que se encarga de gestionar por defecto el tráfico HTTP (HyperText Transfer Protocol) sobre el que van la mayoría de páginas web. Nuestro encaminador va a ver que todo el tráfico pasa por ese puerto pero no va a poder diferenciar el contenido HTML (HyperText Markup Language) de un paquete que va a la página <http://www.marca.com> que nos va a cargar la imagen de nuestro deportista favorito, de un atacante ajeno a nuestra red que está intentando penetrar en nuestros equipos y sustraer información usando HTTP.

A pesar de que existen distintos IDS en el mercado, se escoge Snort debido a su coste nulo, bajo consumo de recursos y su fácil configuración.

2.3.3 Squid

Squid se trata de un servidor web con funciones de proxy y cache para HTTP y File Transfer Protocol (FTP) entre otros. Además permite hacer un filtrado transparente al usuario cogiendo paquetes de dominios o Domain Name Server (DNS) y Secure Socket Layer (SSL). Para entender un poco mejor este concepto vamos a definir a continuación una serie de conceptos:

Un servidor proxy se trata de un servidor intermediario entre nuestro equipo y el *router* que enlaza a Internet. Cuando intentamos navegar por una página web, nuestro equipo solicitará al servidor proxy la página, y como intermediario, este la solicitará al servidor a través del *router*. Una vez que se acepte la solicitud el servidor externo la enviará a nuestro servidor proxy y este a nuestro equipo, como podemos ver en la Figura 2-14.

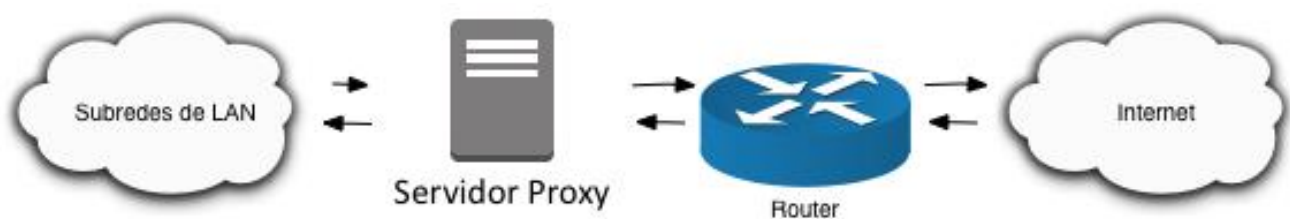


Figura 2-14 Esquema de servidor proxy (Fuente: autoría propia)

Este servidor nos va a proporcionar bastantes ventajas:

- No permitirle el acceso directo al *router* a otros usuarios que no sean el administrador, permitiéndonos controlar todo el flujo de tráfico, tanto entrante como saliente.
- Aceleramos la carga de páginas mediante la memoria caché del servidor proxy, que guardará la información de una página que se haya cargado por primera vez, extrayendo la próxima vez que accedamos a esta la información de la memoria y no del servidor, agilizando el proceso.
- Permitirnos un filtrado de URLs no deseadas, como veremos más adelante.
- Gestionar el uso de determinadas subredes de nuestro entorno.

- Ver los registros de páginas no deseadas a las que acceden nuestros usuarios, para incluirlas en nuestro listado de URLs prohibidas.
- Protege nuestros equipos de ataques externos, ya que se usa a modo de cortafuegos.

Pero no todo son ventajas en el uso de este servidor, como podemos ver a continuación:

- Para que nuestras aplicaciones funcionen correctamente y accedan a Internet sin problemas, debido a la configuración en nuestro trabajo (se verá en el apartado 3.3.2), deberemos de configurarlas una a una: navegadores, servicios de correos, etc.
- Al redirigir todo nuestro tráfico a través de un simple nodo, estaremos también expuestos a que en caso de que falle ese nodo, será imposible acceder a Internet, por lo que se recomienda tener uno secundario a modo de repuesto.
- Se requiere un mantenimiento constante de dicho proxy para evitar incompatibilidades con nuestros equipos. [24]

2.3.4 Diladele Web Filter

Hay una alternativa de pago a Squid ya creada por la empresa Diladele que nos permite filtrar todo tipo de contenido web con una interfaz de usuario muy intuitiva mediante el servidor Apache y un framework de Python llamado Django. Utiliza una base de datos propia que se actualiza de manera semanal automáticamente en la que va incluyendo sitios potencialmente peligrosos dentro de categorías como pornografía, juegos, deportes, etc. En dicha aplicación se nos facilita una versión de prueba de 15 días en la cual tenemos control absoluto en el filtrado como vemos a continuación. [26]

Bloqueo por el idioma utilizado en los anuncios (Figura 2-15):

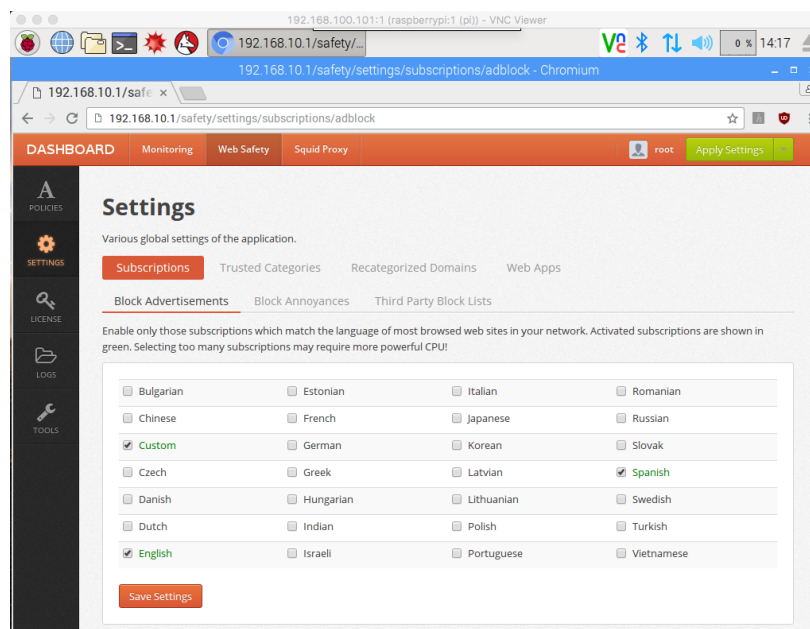


Figura 2-15 Filtrado de anuncios por idiomas en Web Filter (Fuente: autoría propia)

Bloqueo de anuncios incluyendo los anuncios originados por grandes plataformas como *Youtube* antes de cargar los vídeos de su página (Figura 2-16).

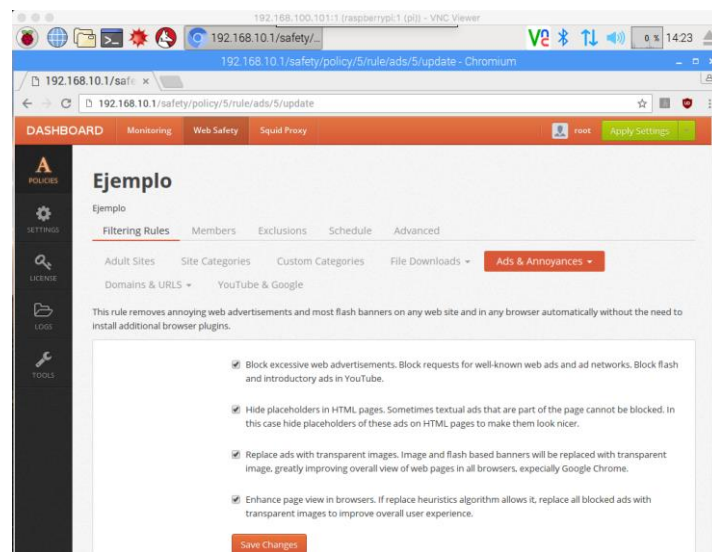


Figura 2-16 Configuración del bloqueo de anuncios en Web Filter (Fuente: autoría propia)

Bloqueo por categorías de la base de datos propia (Figura 2-17):

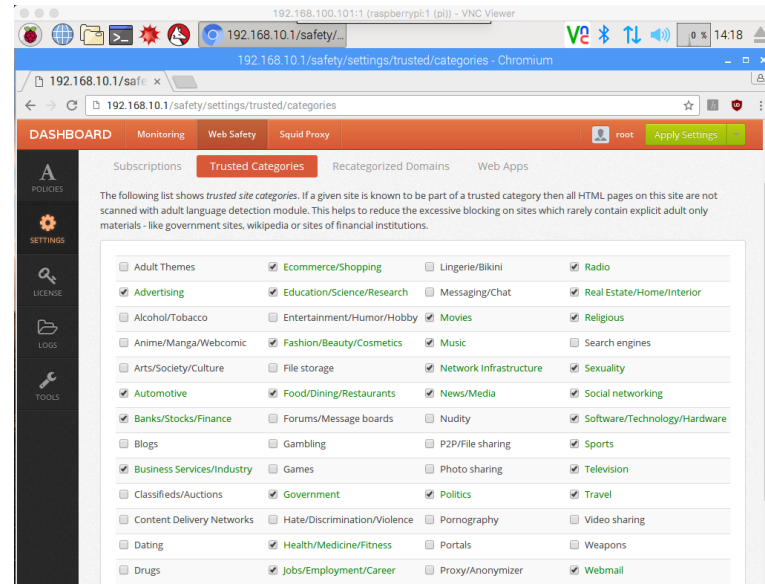


Figura 2-17 Bloqueo de URL por categorías de Web Filter (Fuente: autoría propia)

Así podemos ver que Squid dispone de un modo para interceptar HTTPS transparente al usuario (Figura 2-18):

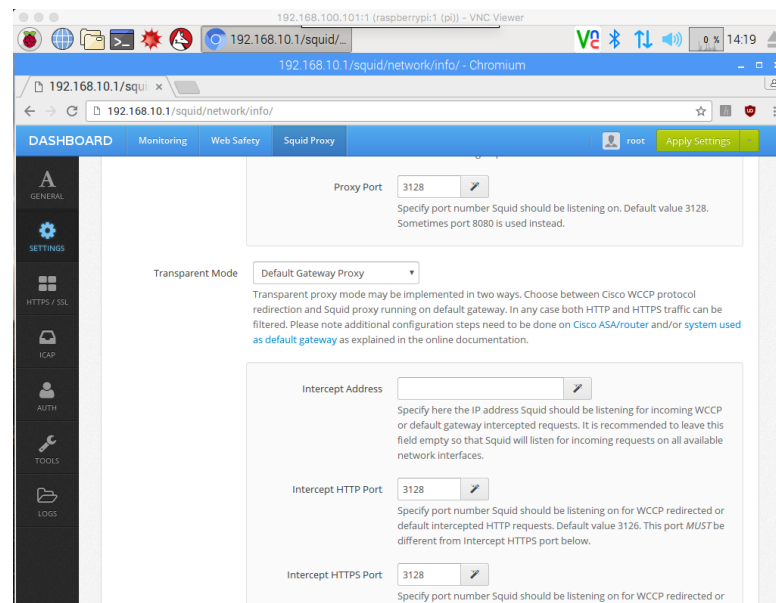


Figura 2-18 Configuración de Squid en modo transparente en Web Filter (Fuente: autoría propia)

Para configurar dicha aplicación usaríamos un servidor VNC o *Virtual Network Computing*, para acceder de manera remota a nuestra Raspberry. [26]

Esta opción se contempló a lo largo de la elaboración del proyecto implementándose desde un principio, pero debido a que es un software de pago y solamente ofrecía una prueba de 15 días se descartó.

2.3.5 SquidGuard



Figura 2-19 Logotipo de SquidGuard (Fuente: www.squidguard.org)

SquidGuard (Figura 2-19) es un complemento gratuito y de código libre de Squid encargado de actuar como filtro, redirigir a una página determinada y controlador de acceso a páginas web. Puede ser usado para:

- Limitar el acceso a ciertas páginas web a determinados usuarios.
- Bloquear páginas web de una lista negra creada previamente.
- Bloqueo por palabras no deseadas contenidas en un listado.
- Forzar a usar nombres de los dominios en vez de las IP de los servidores.
- Redirigir las URL bloqueadas a una página web determinada.
- Hacer bloqueos temporales, estableciendo horas de trabajo, días de la semana, etc.
- Establecer diferentes reglas para diferentes grupos de trabajo.

Por el contrario, SquidGuard no puede filtrar:

- Contenidos ubicados dentro de documentos de texto (se filtra sólo en base a URLs).
- Contenidos implícitos en scripts de lenguajes de programación como Javascript.

3 DESARROLLO DEL TFG

3.1 Arquitectura y funcionamiento básico del sistema

En este primer apartado del desarrollo vamos a ver el esquema básico de nuestro sistema. Siguiendo la Figura 3-1, vamos a colocar nuestra Raspberry Pi modelo 3 como nexo entre el encaminador y nuestros equipos de la red local. En dicha Raspberry se van a instalar los programas Squid, SquidGuard y Snort con los siguientes propósitos:

- Squid: filtrar por dominios y palabras clave las URLs a las que podrán acceder los usuarios.
- SquidGuard: proporcionará una base de datos de dominios y palabras clave robusta a Squid de la que dispondrá permanentemente y que actualizaremos mediante un *script*.
- Snort: Con este programa avisaremos al usuario vía correo electrónico tanto de un intento de intrusión externo como de un intento de acceso desde nuestros equipos a URL no deseadas.

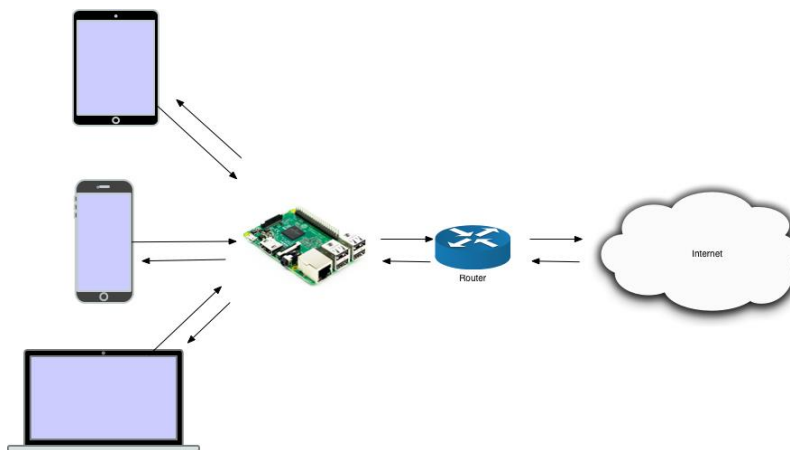


Figura 3-1 Arquitectura del sistema propuesto (Fuente: autoría propia)

3.2 Instalación y configuración del Sistema Operativo

3.2.1 Instalación del Sistema Operativo Raspbian

Lo primero de todo será instalar el Sistema Operativo en nuestra Raspberry. Para ello deberemos entrar en la página oficial de Raspberry www.raspberrypi.org y descargaremos el archivo “Raspbian Jessie with Pixel” con la versión del núcleo o *kernel* 4.4 de fecha del 11 de Enero de 2017. Cuando lo tengamos listo procederemos a instalarlo en la tarjeta microSD que previamente tendremos formateada. Usaremos un software recomendado por la página oficial de Raspberry Pi llamado *SD Formatter* [27] (Figura 3-2), el cual nos permite hacer un borrado completo de todo el contenido de la tarjeta de memoria.

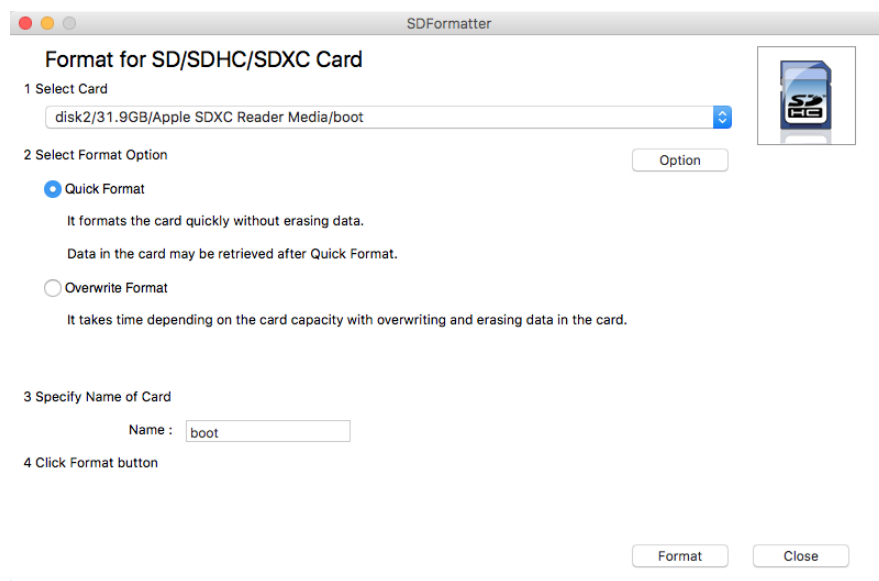


Figura 3-2 SD Formatter (Fuente: Autoría propia)

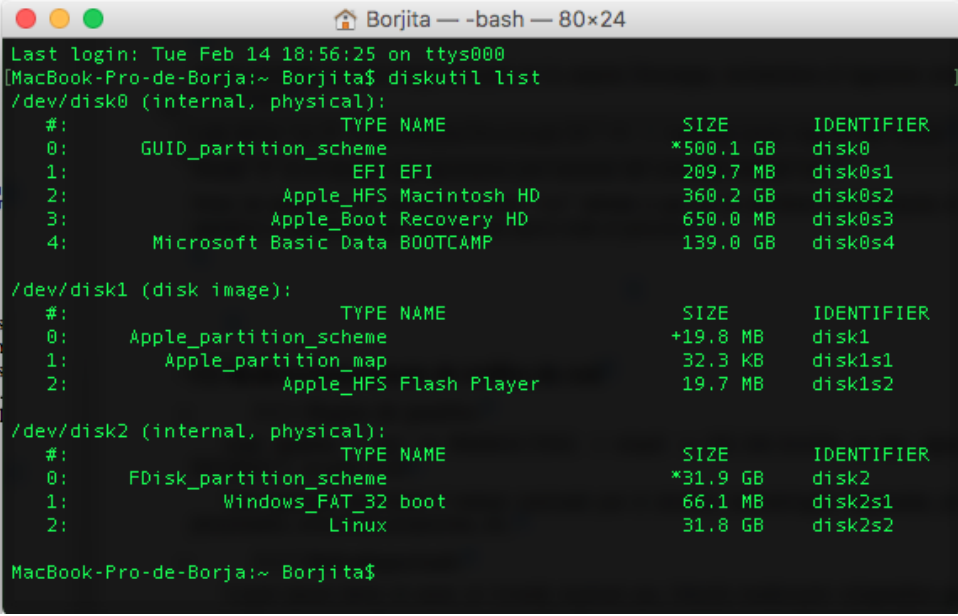
Debemos prestar atención antes de formatear la tarjeta puesto que debe estar desmontada de nuestro sistema. Después de formatear nuestra microSD toca introducir nuestra imagen descargada en ella. Para esto primero deberemos saber dónde está nuestra unidad de disco utilizando el comando:

```
diskutil list
```

El cual nos mostrará en nuestro terminal las unidades de almacenamiento de nuestro ordenador. Aquí apuntaremos la ruta de nuestra microSD con la forma:

```
diskn
```

Donde “n” es un número que asigna nuestro ordenador a dicha tarjeta (Figura 3-3)



```

Last login: Tue Feb 14 18:56:25 on ttys000
[MacBook-Pro-de-Borja:~ Borjita$ diskutil list
/dev/disk0 (internal, physical):
#    TYPE NAME             SIZE       IDENTIFIER
0:    GUID_partition_scheme      *500.1 GB   disk0
1:     EFI EFI                209.7 MB    disk0s1
2:     Apple_HFS Macintosh HD    360.2 GB    disk0s2
3:     Apple_Boot Recovery HD    650.0 MB    disk0s3
4:     Microsoft Basic Data BOOTCAMP 139.0 GB    disk0s4

/dev/disk1 (disk image):
#    TYPE NAME             SIZE       IDENTIFIER
0:    Apple_partition_scheme     +19.8 MB    disk1
1:    Apple_partition_map        32.3 KB     disk1s1
2:    Apple_HFS Flash Player     19.7 MB     disk1s2

/dev/disk2 (internal, physical):
#    TYPE NAME             SIZE       IDENTIFIER
0:    FDisk_partition_scheme     *31.9 GB    disk2
1:    Windows_FAT_32 boot        66.1 MB     disk2s1
2:    Linux                      31.8 GB     disk2s2

MacBook-Pro-de-Borja:~ Borjita$

```

Figura 3-3 Comando diskutil (Fuente: Autoría propia)

Ahora teniendo nuestra imagen Raspbian alojada en la carpeta Descargas, primero desmontaremos la tarjeta de nuestro sistema y teclearemos el siguiente comando en el terminal:

```
sudo dd bs=1m if=/Users/Borjita/Downloads/2017-01-11-raspbian-jessie.img of=/dev/rdiskn
```

Donde “n” es el número que apuntamos previamente de ejecutar el comando *diskutil*. Conviene aclarar que no se debe poner en “n” secuencias como “1s1” debido a que no se escribirá correctamente el sistema operativo y tendremos que empezar de nuevo todo el proceso. [28]

Como resultado de este proceso, deberíamos tener instalado nuestro sistema operativo Raspbian en nuestra Raspberry Pi. Si la conectamos a un monitor mediante la conexión HDMI se debería ver, posteriormente a la configuración de idioma, teclado, etc, una imagen como la que muestra la Figura 3-4.

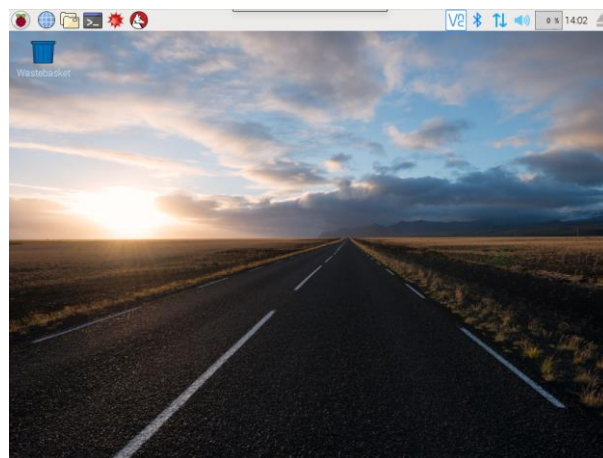


Figura 3-4 Escritorio Raspbian (Fuente: Autoría propia)

3.2.2 Configuración de la conexión remota a la Raspberry Pi

Ahora que tenemos Raspbian instalado vamos a configurar la conexión remota segura vía *Secure Shell*. Para eso accederemos a la configuración desde el terminal escribiendo:

```
sudo raspi-config
```

Buscaremos la opción que ponga “5 Interfacing options” (Figura 3-5) y presionaremos Enter.

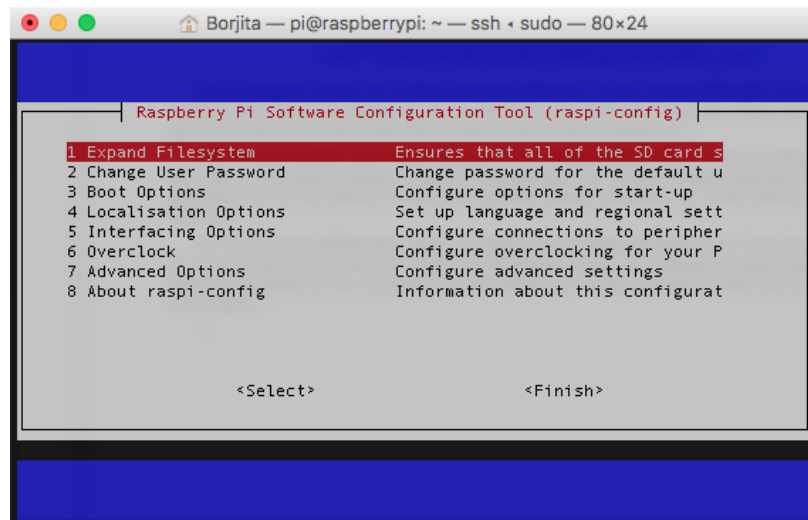


Figura 3-5 Imágen de *Raspberry Configuration Tool* (Fuente: Autoría propia)

En el siguiente menú tendremos las opciones de Activar/Desactivar SSH como vemos en la Figura 3-6. Si queremos utilizar en un futuro la aplicación de pago descrita en el apartado 2.3.4, deberemos habilitar también el *Virtual Network Computing* (VNC) para la interfaz gráfica de Web Filter. Con esto daríamos por concluida la parte de configuración básica.

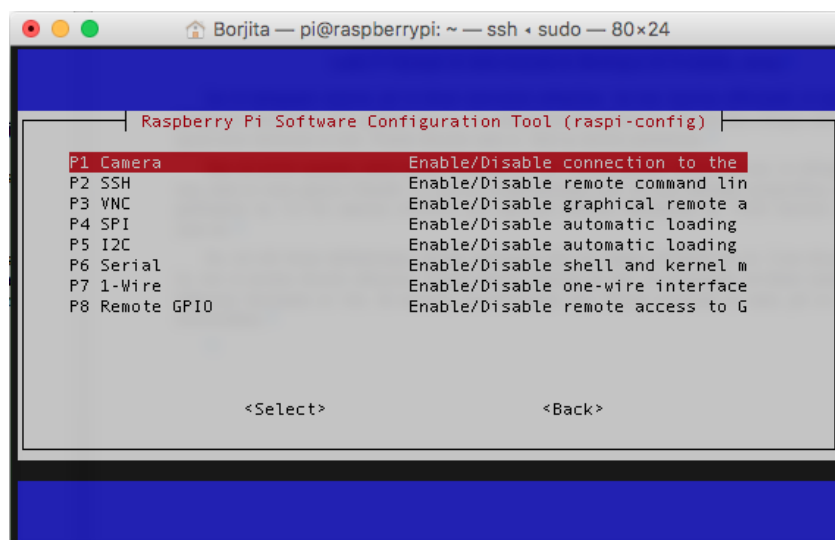


Figura 3-6 Submenú de interfaz de *Raspberry Configuration Tool* (Fuente: Autoría propia)

Al final de estos pasos seremos capaces de conectarnos a nuestra Raspberry Pi mediante SSH escribiendo en nuestra terminal:

```
sudo ssh pi@<dirección IP>
```

Donde <dirección IP> es la dirección que tendrá nuestra Raspberry obtenida desde el encaminador.

3.2.3 Configurar la Raspberry Pi como punto de acceso

Para ello vamos a empezar por actualizar el sistema operativo tecleando el siguiente comando:

```
sudo apt-get update && apt-get upgrade && apt-get dist-upgrade && reboot
```

Posterior a esto, instalaremos los paquetes de dnsmasq y hostapd:

- Con hostapd permitimos el poder usar la Raspberry como punto de acceso
- Con dnsmasq vamos a crear nuestro propio servidor DNS y DHCP que asignará las direcciones IP de manera automática a los equipos que conectemos a dicha red.

```
sudo apt-get install dnsmasq hostapd
```

Continuaremos configurando nuestra interfaz de Internet. Para ello vamos a asignar una dirección IP estática a nuestra Raspberry accediendo primero al archivo de configuración dhcpd.conf

```
sudo nano /etc/dhcpd.conf
```

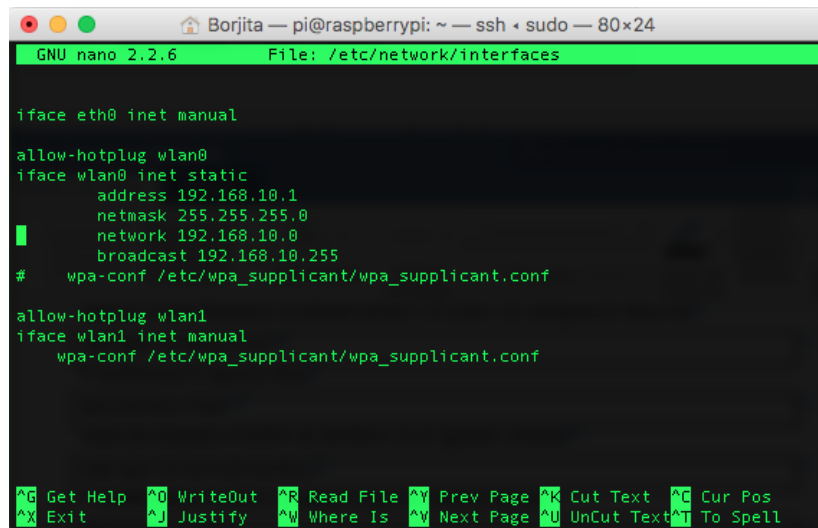
Y escribiremos la siguiente línea para que el DHCP que trabaja en segundo plano ignore la interfaz wlan0 ya que se va a configurar con una IP estática:

```
denyinterfaces wlan0
```

Ahora nos dirigimos al archivo de interfaces con el siguiente comando

```
sudo nano /etc/network/interfaces
```

Y modificaremos el archivo quedando la parte de wlan0 tal como se muestra en la Figura 3-7:



```

GNU nano 2.2.6 File: /etc/network/interfaces

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.10.1
    netmask 255.255.255.0
    network 192.168.10.0
    broadcast 192.168.10.255
#    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

```

Figura 3-7 Fichero interfaces (Fuente: Autoría propia)

Luego guardaremos el fichero y reiniciaremos dhcpd y wlan0 con los siguientes comandos:

```

sudo service dhcpd restart
sudo ifdown wlan0
sudo ifup wlan0

```

El siguiente paso sería modificar el fichero hostapd:

```

sudo nano /etc/hostapd/hostapd.conf

```

Incluyendo en dicho fichero los siguientes datos:

```

# Este es el nombre de la interfaz que hemos configurado previamente
interface=wlan0
# Usar el driver nl80211 con la tarjeta de red de la raspberry pi
driver=nl80211
# Nombre de la red
ssid=Pi_AP
# Usar la banda de 2.4GHz
hw_mode=g
# Usar el canal 6
channel=6
# Activar 802.11n
ieee80211n=1
# Activar Wi-Fi MultiMedia prioriza estos contenidos y asegura una QoS

```

```

wmm_enabled=1
# Activar canales de 40MHz con 20ns de intervalo de guarda entre ellos
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
# Aceptar todas las direcciones MAC
macaddr_acl=0
# Usar autenticación WPA
auth_algs=1
# Solicitar a los clientes que sepan el nombre de la red
ignore_broadcast_ssid=0
# Usar WPA2
wpa=2
# Usar una clave pre-compartida
wpa_key_mgmt=WPA-PSK
# Contraseña de red
wpa_passphrase=raspberry
# Usar AES en vez de TKIP
rsn_pairwise=CCMP

```

Para comprobar que esta funcionando correctamente podemos ejecutar:

```
sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf
```

En el caso de que se esté ejecutando correctamente aparecerá la red “Pi_AP” en el listado de redes inalámbricas (Figura 3-8).



Figura 3-8 Redes inalámbricas disponibles (Fuente: Autoría propia)

Ahora deberemos decirle a hostapd qué archivo de configuración debe mirar cada vez que comience el programa. Para ello tecleamos lo siguiente:

```
sudo nano /etc/default/hostapd
```

Y buscamos una línea que ponga `#DAEMON_CONF=""` y le introducimos la ruta de nuestro archivo de hostapd.conf quedando así la línea:

```
DAEMON_CONF=" /etc/hostapd/hostapd.conf"
```

Lo siguiente que haremos será configurar dnsmasq. Primero crearemos un archivo de copia de seguridad (*backup*) para conservar los originales con el siguiente comando:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
```

Y pondremos las siguientes líneas:

```
interface=wlan0      # Usar interfaz wlan0
listen-address=192.168.10.1 # Dirección donde va a escuchar
bind-interfaces      # Enlazar con la interfaz que hemos asignado previamente
server=8.8.8.8       # Solicitud al DNS de Google
domain-needed        # No enviar nombres de dominios recortados como fb.com
bogus-priv           # No enviar nunca direcciones a través de espacios no contenidos #en las IP
                     # previamente asignadas
dhcp-range=192.168.10.50,192.168.10.150,12h # Asignar direcciones IP entre
#192.168.10.50 y 192.168.10.150 con 12 horas de asignación a dichas IP
```

Ahora necesitamos habilitar el reenvío de paquetes. Para ello vamos a acceder al archivo `sysctl.conf` con:

```
sudo nano /etc/sysctl.conf
```

Y eliminar el “#” del principio de línea que pone “`net.ipv4.ip_forward=1`”. Esta configuración se activará después de reiniciar el dispositivo. Para que se haga inmediatamente escribiremos:

```
sudo sh -c echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Ahora por último redigiremos el tráfico de wlan0 a eth0 activando el uso de NAT para traducir las IP privadas en públicas. Usaremos iptables:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state -state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

El comando:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Guardará nuestra configuración de iptables en /etc/iptables.ipv4.nat. Para que se aplique esta configuración cada vez que encendamos nuestra Raspberry vamos a editar en nuestro archivo rc.local ejecutando:

```
sudo nano /etc/rc.local
```

Justo encima de exit 0 escribiremos:

```
iptables-restore < /etc/iptables.ipv4.nat
```

Ahora para terminar iniciaremos las aplicaciones que hemos instalado y configurado antes, ejecutando:

```
sudo service hostapd start  
sudo service dnsmasq start
```

Para asegurarnos de tener todo configurado correctamente reiniciaremos nuestra RPi

```
sudo reboot
```

Si todo se ha realizado correctamente, deberíamos poder acceder a Internet desde nuestro nuevo punto de acceso Pi_AP sin problemas. [29]

3.3 Instalación de software de control de tráfico de red

Una vez instalado el Sistema Operativo y configurado nuestro punto de acceso, vamos a instalar los programas que nos permitirán controlar el tráfico de red.

3.3.1 Instalación de Snort

Para instalar Snort utilizaremos el repositorio de Raspbian Jessie mediante el comando:

```
sudo apt-get install snort
```

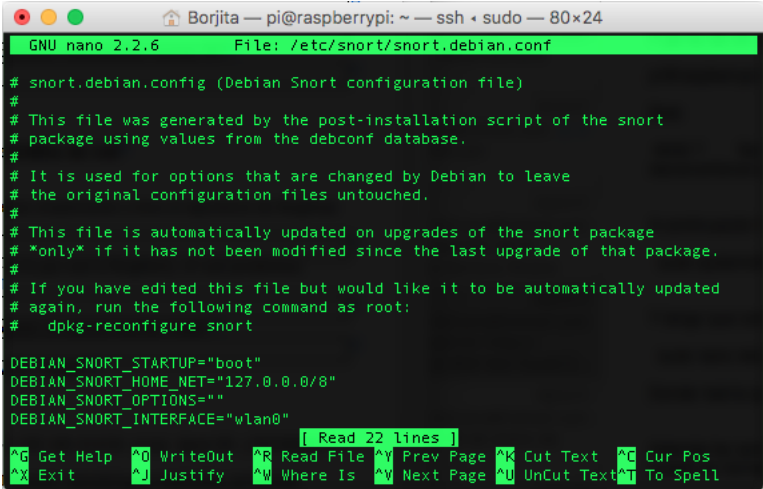
Pararemos Snort para configurarlo:

```
sudo systemctl stop snort
```

Posteriormente vamos a configurarlo para que se centre en nuestra interfaz wlan0:

```
sudo nano /etc/snort/snort.debian.conf
```

Aquí en la línea de `DEBIAN_SNORT_INTERFACE` escribiremos `DEBIAN_SNORT_INTERFACE="wlan0"` (Figura 3-9).



```

GNU nano 2.2.6 File: /etc/snort/snort.debian.conf

# snort.debian.config (Debian Snort configuration file)
#
# This file was generated by the post-installation script of the snort
# package using values from the debconf database.
#
# It is used for options that are changed by Debian to leave
# the original configuration files untouched.
#
# This file is automatically updated on upgrades of the snort package
# *only* if it has not been modified since the last upgrade of that package.
#
# If you have edited this file but would like it to be automatically updated
# again, run the following command as root:
#   dpkg-reconfigure snort

DEBIAN_SNORT_STARTUP="boot"
DEBIAN_SNORT_HOME_NET="127.0.0.0/8"
DEBIAN_SNORT_OPTIONS=""
DEBIAN_SNORT_INTERFACE="wlan0"

[ Read 22 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell

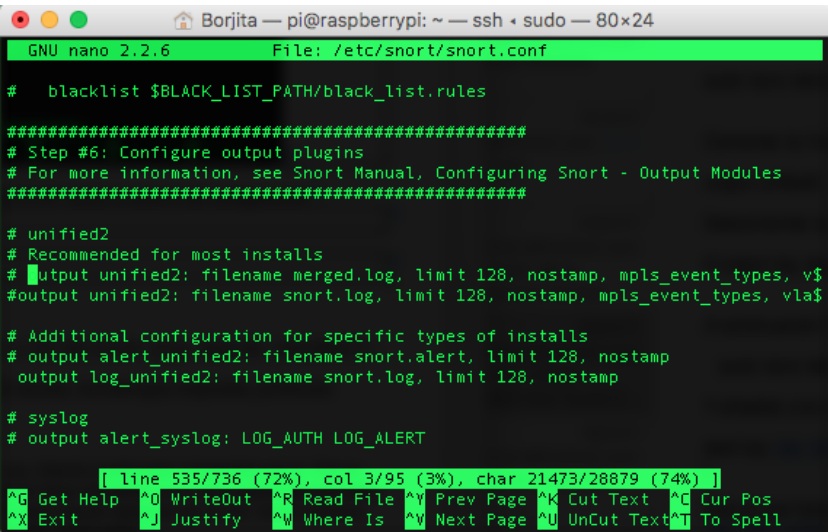
```

Figura 3-9 Archivo *snort.debian.conf* (Fuente: Autoría propia)

Para que nuestro sistema vuelque las alertas en un fichero PCAP (*Packet CAPture*) para que luego se pueda abrir con herramientas gráficas como puede ser Wireshark [30] vamos al archivo de configuración:

```
sudo nano /etc/snort/snort.conf
```

Comentaremos la línea 535 “output unified2” (Figura 3-10):



```

GNU nano 2.2.6 File: /etc/snort/snort.conf

# blacklist $BLACK_LIST_PATH/black_list.rules

#####
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort - Output Modules
#####

# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, v$
# output unified2: filename snort.log, limit 128, nostamp, mpls_event_types, v$

# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128, nostamp
# output log_unified2: filename snort.log, limit 128, nostamp

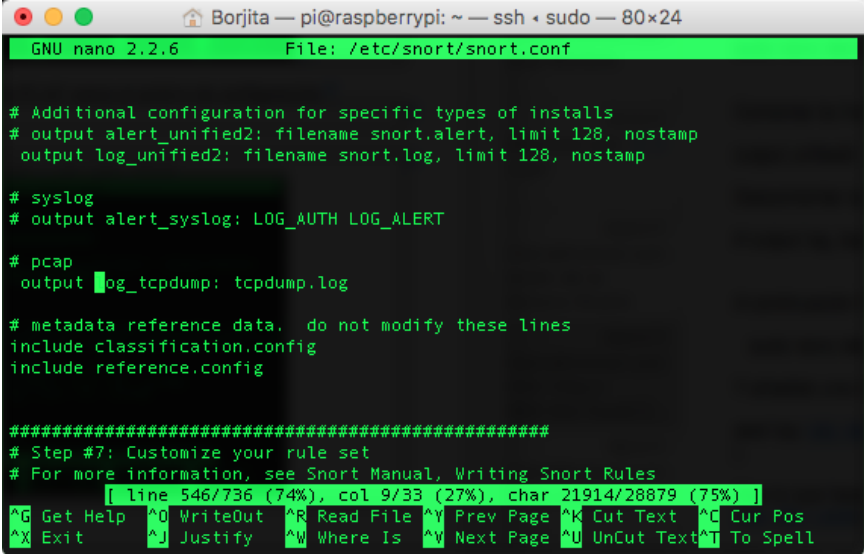
# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT

[ line 535/736 (72%), col 3/95 (3%), char 21473/28879 (74%) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell

```

Figura 3-10 Archivo de configuración de Snort (Fuente: Autoría propia)

Posteriormente vamos a activar la línea 546, eliminando el comentario (Figura 3-11):



```

GNU nano 2.2.6 File: /etc/snort/snort.conf

# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128, nostamp
output log_unified2: filename snort.log, limit 128, nostamp

# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT

# pcap
output log_tcpdump: tcpdump.log

# metadata reference data. do not modify these lines
include classification.config
include reference.config

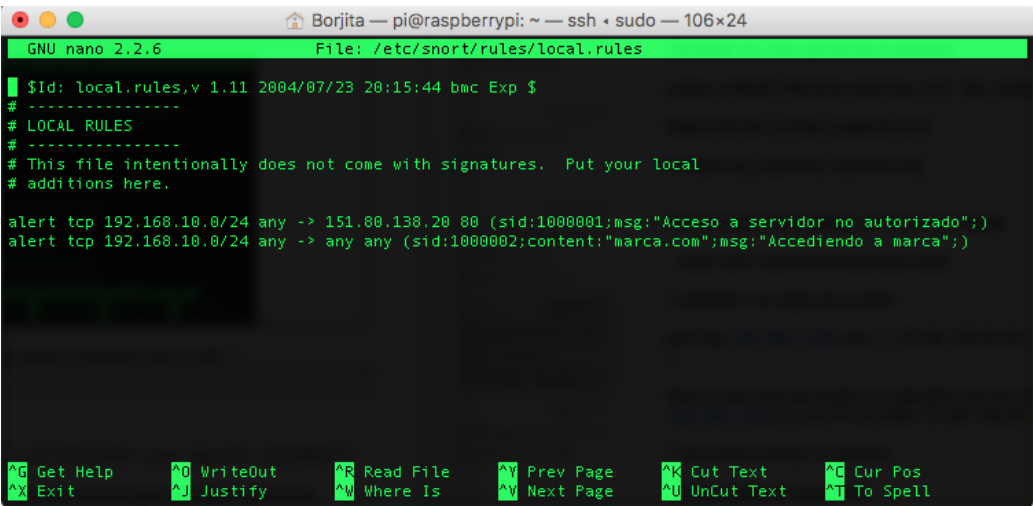
#####
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
[ line 546/736 (74%), col 9/33 (27%), char 21914/28879 (75%) ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

Figura 3-11 Archivo de configuración de Snort (Fuente: Autoría propia)

Snort por defecto carga una serie de reglas predefinidas ubicadas en ficheros disponibles en la carpeta `/etc/snort/rules` pero vamos a crear nuestro propio archivo de reglas donde volcaremos toda nuestra configuración.

```
sudo nano /etc/snort/rules/local.rules
```



```

GNU nano 2.2.6 File: /etc/snort/rules/local.rules

$Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures. Put your local
# additions here.

alert tcp 192.168.10.0/24 any -> 151.80.138.20 80 (sid:1000001;msg:"Acceso a servidor no autorizado");
alert tcp 192.168.10.0/24 any -> any any (sid:1000002;content:"marca.com";msg:"Accediendo a marca");

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

Figura 3-12 Archivo local.rules (Fuente: Autoría propia)

Añadiremos un par de reglas en ese fichero (Figura 3-12). Con esas reglas vamos a decirle a Snort que nos avise cuando se acceda a la IP 151.80.138.20 puerto 80 y que cuando se intente acceder a `www.marca.com` nos avise también.

```

alert tcp 192.168.10.0/24 any -> 151.80.138.20 80 ( sid: 1000001; msg: "Acceso a servidor no
autorizado"; )

alert tcp 192.168.10.0/24 any -> any any (sid:1000002; content:"marca.com";msg:"Accediendo a
marca";)

```

Iniciamos otra vez Snort:

```
sudo systemctl start snort
```

Podremos ver las novedades en el archivo de alertas:

```
less /var/log/snort/alert
```

Para comprobar que realmente funciona deberíamos abrir en el navegador, intentar acceder a Marca y ver que el fichero de alertas aparece un mensaje que muestre “Accediendo a marca” [31].

3.3.2 Instalación de Squid y SquidGuard

Para instalar Squid y SquidGuard seguiremos un proceso en dos partes:

Primero de todo vamos a habilitar la descarga de código fuente (*sources*) para poder compilar Squid. El motivo principal por el cual se hace este paso es para compilar Squid introduciéndole la capacidad de filtrar HTTPS que no se incluye en la distribución por defecto. Para ello editamos el fichero `/etc/apt/sources.list` como se indica en la Figura 3-13. [32]

```
sudo nano /etc/apt/sources.list
```

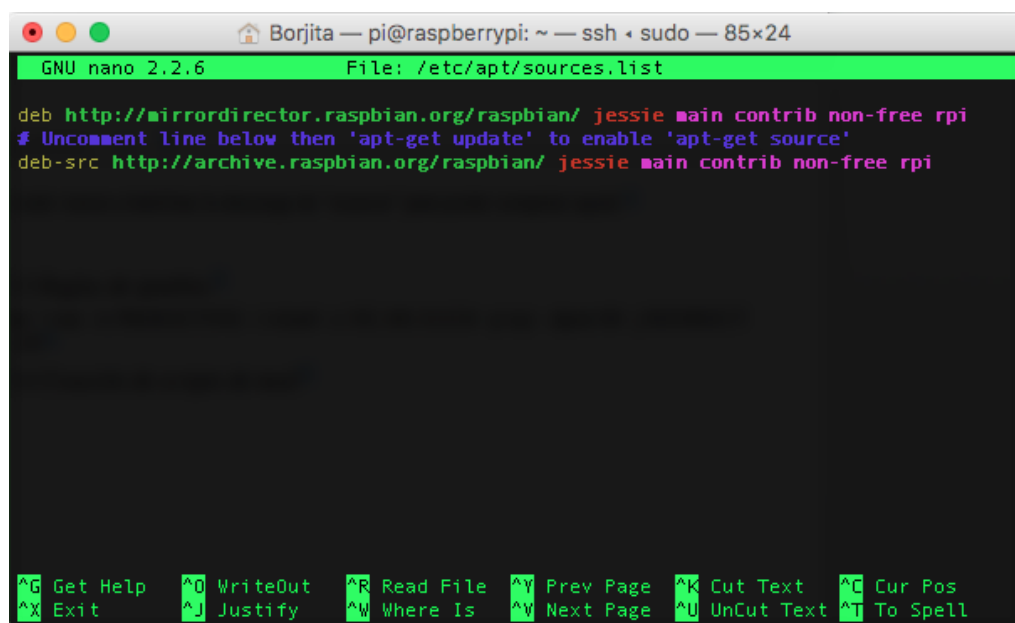


Figura 3-13 Archivo sources.list (Fuente: Autoría propia)

Actualizaremos los repositorios:

```
sudo apt-get update
```

Después de esto vamos a cargar las librerías necesarias ejecutando los siguientes comandos:

```
sudo apt-get install dpkg-dev
sudo apt-get build-dep squid3
sudo apt-get build-dep openssh
sudo apt-get install libssl-dev libcrypto++-dev
sudo apt-get build-dep openssl
sudo apt-get source squid3
sudo apt-get install devscripts build-essential fakeroot
sudo apt-get install squid-langpack apache2
```

Ahora vamos a configurar el archivo rules en la carpeta “debian” añadiendo unas líneas

```
cd squid3-3.4.8
sudo nano debian/rules
```

Añadimos las líneas siguientes debajo de “--enable-icap-client \”

```
--enable-ssl \
--enable-ssl-crt \
--with-openssl \
```

Quedando así, como se indica en la Figura 3-14

```

GNU nano 2.2.6      File: debian/rules      Modified

--enable-auth-digest="file,LDAP" \
--enable-auth-negotiate="kerberos,wrapper" \
--enable-auth-ntlm="fake,smb_lm" \
--enable-external-acl-helpers="file_userip,kerberos_ldap_group,$
--enable-url-rewrite-helpers="fake" \
--enable-eui \
--enable-esi \
--enable-icmp \
--enable-zph-qos \
--enable-ecap \
--enable-ssl \
--enable-ssl-crt \
--with-openssl \
--disable-translation \
--with-swapdir=/var/spool/squid3 \
--with-logdir=/var/log/squid3 \
--with-pidfile=/var/run/squid3.pid \
--with-filedescriptors=65536 \
--with-large-files \

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell

```

Figura 3-14 Fichero `debian.rules` (Fuente: Autoría propia)

Ahora vamos a modificar el fichero que se encuentra en la ruta `squid3-3.4.8/ssl/` llamado `gadgets.cc`

```
sudo nano /home/pi/squid3-3.4.8/ssl/gadgets.cc
```

En la línea 258 pondremos “//” delante del texto para comentarlo:

```
// NID_key_usage,
```

En las líneas 417 en adelante vamos a sustituir sha1 por sha256:

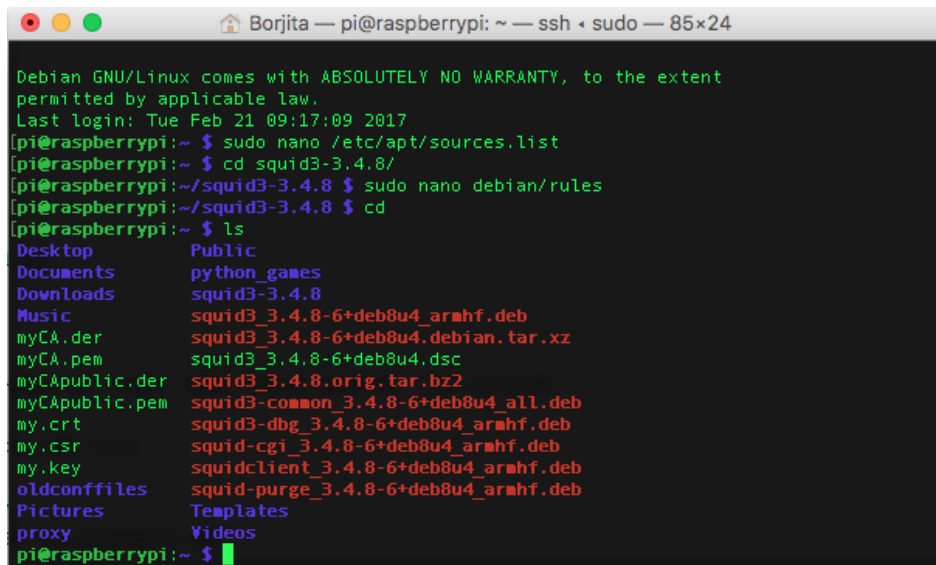
```
ret = X509_sign(cert.get(), properties.signWithPkey.get(), EVP_sha256());
ret = X509_sign(cert.get(), pkey.get(), EVP_sha256());
```

Lo siguiente será configurar y compilar, ejecutando la siguientes línea:

```
dpkg-buildpackage -rfakeroot -b
```

Después de esto, que tardará una hora y media aproximadamente, habrá que cambiar de directorio e instalar los archivos “.deb” que se han creado en el directorio de inicio (Figura 3-15):

```
cd ..
```



```

Borjita — pi@raspberrypi: ~ — ssh • sudo — 85x24

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 21 09:17:09 2017
[pi@raspberrypi:~ $ sudo nano /etc/apt/sources.list
[pi@raspberrypi:~ $ cd squid3-3.4.8/
[pi@raspberrypi:~/squid3-3.4.8 $ sudo nano debian/rules
[pi@raspberrypi:~/squid3-3.4.8 $ cd
[pi@raspberrypi:~ $ ls
Desktop          Public
Documents        python_games
Downloads        squid3-3.4.8
Music            squid3_3.4.8-6+deb8u4_armhf.deb
myCA.der         squid3_3.4.8-6+deb8u4.debian.tar.xz
myCA.pem         squid3_3.4.8-6+deb8u4.dsc
myCApublic.der   squid3_3.4.8.orig.tar.bz2
myCApublic.pem   squid3-common_3.4.8-6+deb8u4_all.deb
my.crt           squid3-dbg_3.4.8-6+deb8u4_armhf.deb
my.csr           squid-cgi_3.4.8-6+deb8u4_armhf.deb
my.key           squidclient_3.4.8-6+deb8u4_armhf.deb
oldconfigfiles  squid-purge_3.4.8-6+deb8u4_armhf.deb
Pictures         Templates
proxy            Videos
pi@raspberrypi:~ $

```

Figura 3-15 Imágen de cómo quedaría el directorio "/home/pi" (Fuente: Autoría propia)

Escribiremos las siguientes líneas para instalarlos:

```
dpkg -i *.deb
```

Ahora crearemos un directorio donde Squid va a almacenar los certificados que creemos, indicando que ese directorio va a ser propiedad del usuario "proxy" (Squid).

```
/usr/lib/squid3/ssl_crt -c -s /var/lib/ssl_db/
chown -R proxy /var/lib/ssl_db
```

Lo siguiente será generar nuestros propios certificados. Para ello crearemos un certificado con clave RSA válido para 365 días (1 año) que se llame myCA:

```
openssl req -new -newkey rsa:2048 -sha256 -days 365 -nodes -x509 -keyout myCA.pem -out myCA.pem
```

Hacemos una copia del certificado:

```
cp myCA.pem myCApublic.pem
```

Le quitamos la parte privada al certificado myCApublic.pem:

```
openssl x509 -in myCApublic.pem -outform DER -out myCApublic.der
```

Moveremos el certificado a donde Squid lo vaya a leer:

```
cd /etc/squid3
```

```
mkdir certs
cd
sudo mv myCA.pem /etc/squid3/certs/myCA.pem
```

Con esto tendremos instalado Squid, ahora faltaría instalar su apoyo, SquidGuard. Para esto vamos a introducir lo siguiente en el terminal:

```
sudo apt-get install squidGuard
```

Después de instalarlo, el siguiente paso será crear nuestra base de datos, la cual mirará Squid:

```
cd /var/lib/squidguard/db
sudo nano testdomains
```

Aquí introduciremos los dominios que queramos bloquear. Para nuestro caso bloquearemos la página `www.marca.com`, la cual se puede acceder a través de HTTP sin cifrar (Figura 3-16):

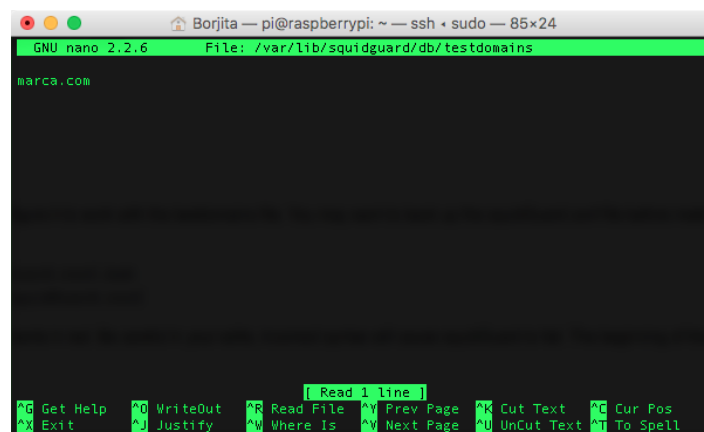


Figura 3-16 fichero `testdomains` creado para SquidGuard (Fuente: Autoría propia)

Continuando con la configuración de SquidGuard, debemos crear una copia de seguridad del archivo de configuración:

```
cd /etc/squidguard
sudo cp squidGuard.conf squidGuard.conf.bak
sudo nano -c /etc/squidguard/squidGuard.conf
```

Y asegurarnos de que quede así el archivo:

```
dest test {
    domainlist testdomains #Cogerá los dominios en esa lista
acl {
```

```
admin {  
    pass any  
}  
  
default {  
    pass !test any #pasarán todas excepto las de la lista #creada  
    redirect http://127.0.0.1/blocked.html #los dominios de #la lista se redirigirán a la  
web 127.0.0.1/blocked.html  
}  
}
```

Guardamos y a continuación vamos al directorio de Apache [33] donde se almacenan en local las páginas HTML creadas. Allí crearemos una llamada “blocked.html”

```
cd /var/www/html  
sudo nano blocked.html
```

Aquí meteremos la disposición de la página web que se puede ver en el Anexo I: Contenido HTML. Para que todos los cambios surtan efecto debemos de configurar de nuevo SquidGuard:

```
sudo squidGuard -C all
```

Ahora le vamos a dar permisos a Squid de cara a recoger datos de estas carpetas, cambiando su propietario:

```
sudo chown -R proxy:proxy /var/lib/squidguard/db  
sudo chown -R proxy:proxy /var/log/squidguard  
sudo chown -R proxy:proxy /usr/bin/squidGuard
```

Seguiremos configurando los dos servicios. Primero empezaremos con Squid:

```
sudo nano -c /etc/squid3/squid.conf
```

Nos iremos a la línea 1067 y tendrá que verse el conjunto de líneas así:

```

GNU nano 2.2.6 File: /etc/squid3/squid.conf

# Recommended minimum configuration:
#
# Example rule allowing access from your local networks.
# Adapt to list your (internal) IP networks from where browsing
# should be allowed
acl localnet src 10.0.0.0/8      # RFC1918 possible internal network
acl localnet src 172.16.0.0/12  # RFC1918 possible internal network
acl localnet src 192.168.0.0/24 # RFC1918 possible internal network
#acl localnet src fc00::/7       # RFC 4193 local private network range
#acl localnet src fe80::/10      # RFC 4291 link-local (directly plugged) machines

acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports

line 1073/7634 (14%), col 1/23 (4%), char 39810/277617 (14%)
^G Get Help ^O WriteOut ^R Read File ^V Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^N Next Page ^U UnCut Text ^T To Spell

```

Figura 3-17 Fichero squid.conf (Fuente: Autoría propia)

Esto permitirá escuchar a Squid en esos rangos de red.

El siguiente contenido tendrá que estar habilitado desde la línea 1217 en adelante. En caso de no existir se creará la línea:

```

http_access allow localnet
http_access allow localhost manager
http_access deny manager
http_access deny to_localhost

```

Añadiremos también una lista de acceso o *access list* (acl) debajo de las últimas:

```

acl blockeddomains dstdomain "/etc/squid3/blocked.domains.acl"
http_access deny blockeddomains
acl bad_keywords url_regex "/etc/squid3/keywords.txt"
http_access deny bad_keywords

```

Esto nos permitirá denegar el acceso a los dominios de la lista *"/etc/squid3/blocked.domains.acl"* y por palabras clave de *"/etc/squid3/keywords.txt"*, que crearemos en las rutas correspondientes. En nuestro caso la lista de dominios bloqueados contendrá a Twitter, una red social a la que hay que acceder a través de HTTPS cifrado, Marca y As, páginas web a las que se accede a través de HTTP.

```

.marca.com
.as.com
.twitter.com

```

Así mismo la lista de palabras clave contendrá palabras relacionadas con el sexo explícito, juegos de azar y deportes, prestando atención puesto que en este caso Squid es case-sensitive, es decir, tiene en cuenta las mayúsculas y minúsculas:

```
futbol
fútbol
soccer
balompie
balompié

porn
porno
juego
gamble
```

Las líneas 1642 a 1643 del fichero `/etc/squid3/squid.conf` quedarán de esta manera:

```
http_port 3129 transparent
https_port 3130 ssl-bump intercept generate-host-certificates=on dynamic_cert_mem_cache_size=4MB
cert=/etc/squid3/certs/myCA.pem
```

Esto nos permitirá ver de manera transparente al usuario el tráfico HTTP sin necesidad de configurar ningún proxy en los navegadores. Para el caso de HTTPS habrá que configurarlo en el navegador. Para el caso de safari habrá que acceder a *Acceso a llaveros > Inicio de sesión > Mis certificados y luego Archivo > Importar elementos*. Importamos nuestro certificado llamado `myCApublic.der`.

Para que pueda pinchar HTTPS introduciremos las siguientes líneas a partir de la número 2195:

```
sslproxy_options NO_SSLv2, NO_SSLv3, SINGLE_DH_USE

ssl_bump server-first all
sslproxy_cert_error allow all
sslproxy_flags DONT_VERIFY_PEER

coredump_dir /var/spool/squid3
```

Vamos a decirle a Squid que utilice las bases de datos de SquidGuard. En la línea 4145 del fichero `/etc/squid3/squid.conf` introduciremos lo siguiente:

```
url_rewrite_program /usr/bin/squidGuard
```

Ahora vamos a introducir las listas negras o *Black Lists* en SquidGuard, que son listas con contenido no deseado preestablecidas por los usuarios que han ido subiendo a repositorios. Lo primero será descargarnos el fichero más actual de una base de datos de *black lists*. Hemos escogido una de

código abierto como es Shalla Secure Service [34]. Lo siguiente será copiar los datos a nuestro directorio de base de datos de SquidGuard y le daremos permisos a Squid para que pueda manejar este:

```
cd ~/Downloads
sudo wget http://www.shallalist.de/Downloads/shallalist.tar.gz
sudo tar -xzf shallalist.tar.gz
sudo cp BL -R /var/lib/squidguard/db
cd /var/lib/squidguard/db
sudo chmod -R 755 /var/lib/squidguard/db/BL
sudo chown -R proxy:proxy /var/lib/squidguard/db/BL
```

Ahora para activar alguna de las listas sólo tenemos que configurar el archivo de configuración de SquidGuard:

```
sudo nano /etc/squidguard/squidGuard.conf
```

Por ejemplo, si queremos bloquear los sitios web de juegos y apuestas tendremos que añadir la siguiente configuración a nuestro archivo *squidGuard.conf*:

Línea 61 y sucesivas:

```
dest gamble{
    domainlist    BL/gamble/domain
    urllist       BL/gamble/urls
}
```

Línea 85 y sucesivas:

```
pass !gamble !test any
redirect http://127.0.0.1/blocked.html
```

Nos movemos a la carpeta de la base de datos y reconfiguramos SquidGuard:

```
cd /var/lib/squidguard/db
sudo squidGuard -C all
```

Y aparecerán en la base de datos, en la carpeta “gamble”, dos archivos nuevos *domains.db* y *urls.db* (ver Figura 3-18):

```

Borjita — pi@raspberrypi: /var/lib/squidguard/db/BL/gamble — ssh — sudo — 94x24
myCApublic.pem  squid3_3.4.8-6+deb8u4_armhf.deb  Videos
my.crt          squid3_3.4.8-6+deb8u4.debian.tar.xz
my.csr          squid3_3.4.8-6+deb8u4.dsc
pi@raspberrypi:~$ cd /var/lib/squidguard
pi@raspberrypi:/var/lib/squidguard$ ls
db
pi@raspberrypi:/var/lib/squidguard$ cd
pi@raspberrypi:~$ cd /etc/squidguard
pi@raspberrypi:/etc/squidguard$ ls
squidGuard.conf  squidGuard.conf.bak  squidGuard.conf.default
pi@raspberrypi:/etc/squidguard$ sudo nano squidGuard.conf
pi@raspberrypi:/etc/squidguard$ sudo squidGuard -C all
^Cpi@raspberrypi:/etc/squidguard$ cd /var/lib/squidguard/db/
^Cpi@raspberrypi:/var/lib/squidguard/db$ sudo squidGuard -C all
^Cpi@raspberrypi:/var/lib/squidguard/db$ sudo squidGuard -C all
sudo: squidguard: command not found
pi@raspberrypi:/var/lib/squidguard/db$ cd BL
pi@raspberrypi:/var/lib/squidguard/db/BL$ sudo squidGuard -C all
pi@raspberrypi:/var/lib/squidguard/db/BL$ sudo nano /etc/squidguard/squidGuard.conf
pi@raspberrypi:/var/lib/squidguard/db/BL$ sudo squidGuard -C all
pi@raspberrypi:/var/lib/squidguard/db/BL$ cd gamble
pi@raspberrypi:/var/lib/squidguard/db/BL/gamble$ ls
domains  domains.db  urls  urls.db
pi@raspberrypi:/var/lib/squidguard/db/BL/gamble$

```

Figura 3-18 Carpeta gamble de la base de datos de SquidGuard (Fuente: Autoría propia)

3.3.3 Reglas de iptables

A la hora de controlar el tráfico de red tenemos que redirigir el mismo a los puertos de escucha de Squid. Para ello accederemos al fichero *iptables.ipv4.nat* que creamos en el apartado 3.2.3:

```
sudo nano /etc/iptables.ipv4.nat
```

Y sustituiremos el contenido del archivo por lo siguiente:

```

*nat
:PREROUTING ACCEPT [661:115590]
:INPUT ACCEPT [4049:332838]
:OUTPUT ACCEPT [2700:183657]
:POSTROUTING ACCEPT [33:2313]
-A PREROUTING -i wlan0 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 3129
-A PREROUTING -i wlan0 -p tcp -m tcp --dport 443 -j REDIRECT --to-ports 3130
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
*filter
:INPUT ACCEPT [45289:14577708]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [45887:16975256]

```

```
-A FORWARD -i eth0 -o wlan0 -m state --state RELATED, ESTABLISHED -j ACCEPT
-A FORWARD -i wlan0 -o eth0 -j ACCEPT
COMMIT
```

Lo que conseguimos con esto es que primero redirija los paquetes TCP (*Transfer Control Protocol*) procedentes del puerto 80 (HTTP) al puerto 3129 y del 443 al puerto 3130 (HTTPS). En dichos puertos estará Squid escuchando.

3.3.4 Creación de scripts de aviso

Lo primero será instalar y configurar los paquetes necesarios (ssmtp y mailutils) en la Raspberry tecleando lo siguiente:

```
sudo apt-get install ssmtp mailutils
sudo cp /etc/ssmtp/ssmtp.conf /etc/ssmtp/ssmtp.conf.orig
sudo nano /etc/ssmtp/ssmtp.conf
```

Marcamos con “#” todos los comienzos de línea y a continuación vamos a definir la siguiente configuración en el fichero ssmtp.conf para utilizar la cuenta de Gmail tfgraspberry@gmail.com:

```
# Este campo define la cuenta de correo que va a recibir todos los correos
root=tfgraspberry@gmail.com
# Servidor SMTP al que reenviaremos los correos.
mailhub=smtp.gmail.com:587
# Datos de autenticación de la cuenta de correo.
AuthUser=tfgraspberry@gmail.com
AuthPass=raspberry
# Usar SSL/TLS
UseTLS=Yes
UseSTARTTLS=Yes
# Nombre del host
hostname=raspberry
```

Para que nos avise el IDS Snort vamos a crear un pequeño *script* llamado “*check*” (ver Anexo II: Scripts), cuyo diagrama de flujo aparece en la Figura 3-19, el cual nos enviará un correo electrónico cuando se detecte una novedad en el registro de alertas de Snort. Básicamente el script comparará dos ficheros de texto que extraeremos de dos fuentes: por un lado el registro de alertas de Snort y por otro lado un fichero llamado “anterior.txt”. Si hay alguna diferencia entre ambos, las diferencias se volcarán archivo llamado “diferencias.txt” que será enviado por correo al usuario.

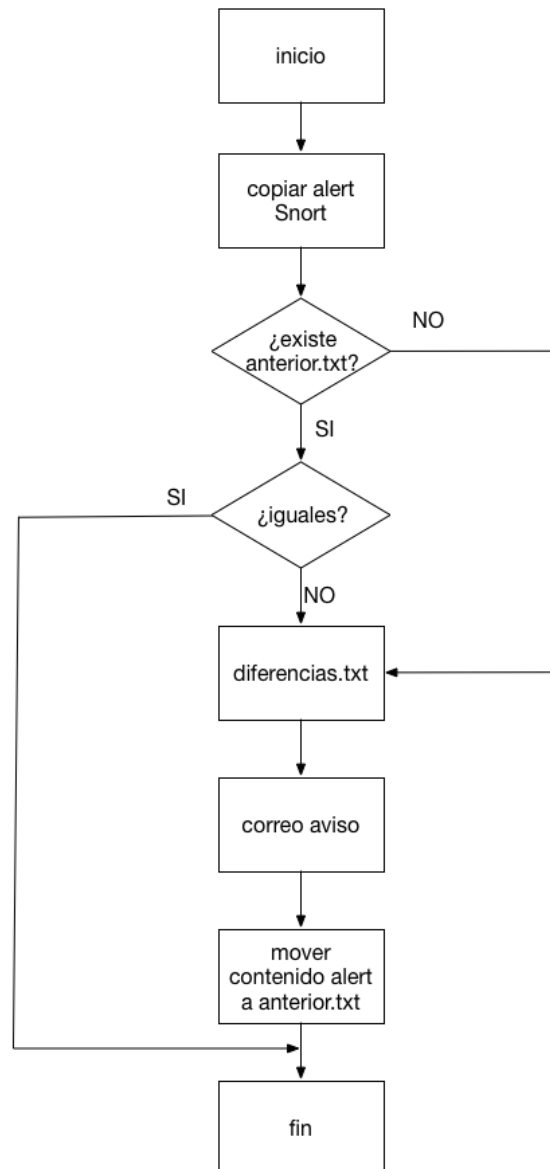


Figura 3-19 Diagrama de flujo del script de aviso (Fuente: Autoría propia)

4 PRUEBAS

4.1 Filtrado de URL's

Para comprobar el filtrado de URL's lo primero será introducir las URL a bloquear, como hemos hecho en el apartado 3.3.2 en nuestro fichero "*blocked.domains.acl*"

```
.marca.com  
.as.com  
.twitter.com
```

A continuación introduciremos en el navegador por ejemplo www.twitter.com, que utiliza HTTPS.



Figura 4-1 Comprobación de filtrados de URL con Squid (Fuente: Autoría propia)

Como podemos ver en la Figura 4-1, Squid filtra en este caso la web <https://twitter.com/> sin ningún problema.

4.2 Filtrado por palabras/repositorio Black List

A continuación, siguiendo con las comprobaciones, filtraremos por palabras clave usando el archivo que introducimos en la configuración de Squid en el apartado 3.3.2 llamado “*keywords.txt*”:

```
futbol  
fútbol  
soccer  
balompie  
balompié  
  
porn  
porno  
juego  
gamble
```

Introducimos por ejemplo la palabra “futbol” en el buscador de Google (Figura 4-2):

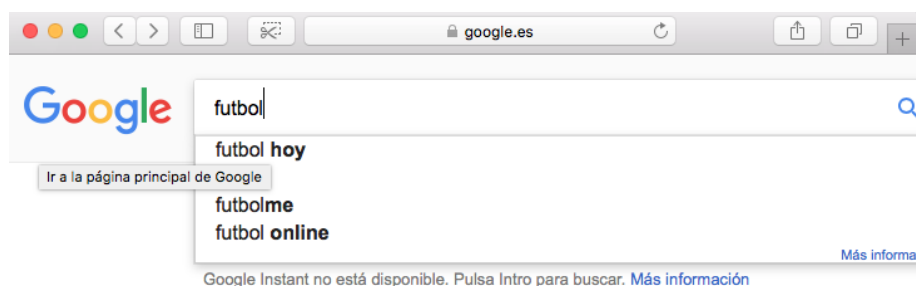


Figura 4-2 Búsqueda de "futbol" en el buscador de Google (Fuente: Autoría propia)

Al presionar Enter obtendremos la Figura 4-3:



Figura 4-3 Squid bloqueando la búsqueda de "futbol" (Fuente: Autoría Propia)

Vemos con esto que Squid filtra sin problemas por palabras clave en la URL.

4.3 Aviso via mail de intrusión no deseada

Para este apartado creamos un script de aviso en el capítulo 3.3.4, el cual nos permitirá ser avisados por correo electrónico sin tener la necesidad de controlar personalmente la Raspberry. Aprovechando el intento de conexión a www.marca.com, Snort ha aplicado la regla que le introducimos en la sección 3.3.1 y, efectivamente, al ejecutar el script “check” la Raspberry nos mandó un correo electrónico de aviso especificando la alerta y la hora a la que saltó la alarma. Para nuestro caso se accedió a la cuenta de correo a través de un Smartphone:

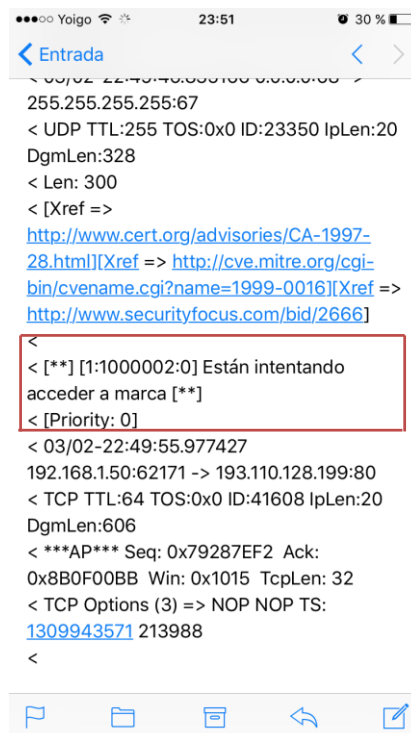


Figura 4-4 Correo electrónico de aviso (Fuente: Autoría propia)

Como podemos ver en la Figura 4-4, se intentó acceder a una URL prohibida, concretamente .marca.com, a las 22:49 de la noche.

En el caso de Twitter, como HTTPS irá cifrado, Snort no podrá verlo con la misma sencillez que la página de Marca, por lo tanto deberemos utilizar otra alternativa y decirle a Snort que busque en las peticiones *Domain Name Server* (DNS) con UDP a través del puerto 53 empleando la siguiente regla:

```
alert udp 192.168.1.0/24 any -> any 53 (sid: 1000009; msg:"Accediendo a twitter"; content:"|07 74 77 69 74 74 65 72 03 63 6f 6d 00|";)
```

Esta regla de Snort busca en el contenido de los paquetes (apartado “content”) el patrón hexadecimal “07 74 77 69 77 69 74 74 65 72 03 63 6f 6d 00”. Ese es el patrón que aparece en el paquete DNS cuando se trata de resolver el nombre “.twitter.com”, como se puede ver con la ayuda de Wireshark (Figura 4-5).

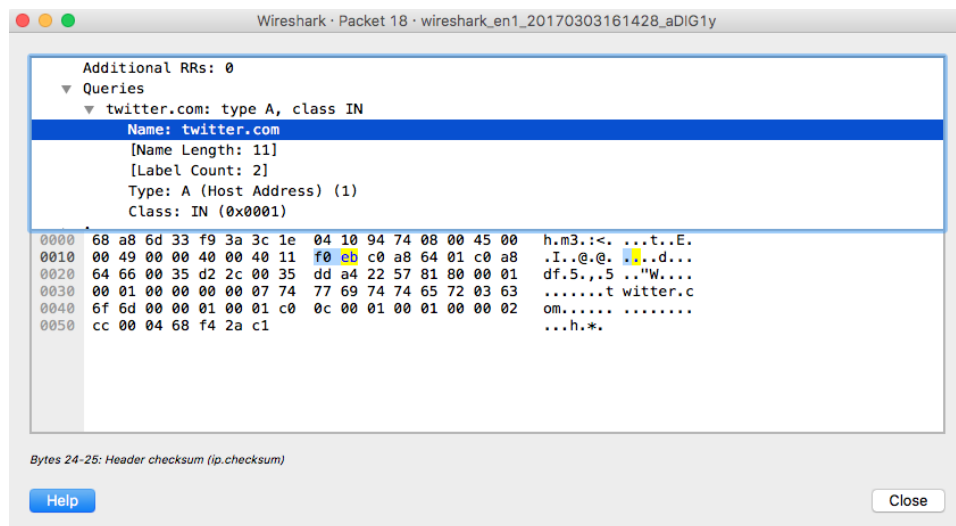


Figura 4-5 Capturas de paquetes mediante Wireshark (Fuente: autoría propia)

Al cargar dicha regla en Snort, si miramos el registro de alertas obtenemos lo siguiente al intentar acceder a la página (Figura 4-6).

```

/wwww.securityfocus.com/bid/2666]

[**] [1:1000009:0] Accediendo a twitter [**]
[Priority: 0]
03/03-15:36:09.967341 192.168.1.50:49890 -> 192.168.1.1:53
UDP TTL:255 TOS:0x0 ID:12302 IpLen:20 DgmLen:57
Len: 29

[**] [1:527:0] BAD-TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
03/03-15:36:40.241772 0.0.0.0:68 -> 255.255.255.255:67
UDP TTL:255 TOS:0x0 ID:23402 IpLen:20 DgmLen:328
Len: 300
[Xref => http://www.cert.org/advisories/CA-1997-28.html][Xref => http://cve
.mitre.org/cgi-bin/cvename.cgi?name=1999-0016][Xref => http://www.securityf
ocus.com/bid/2666]

[**] [1:1000002:0] Están intentando acceder a marca [**]
[Priority: 0]
03/03-15:36:41.219179 192.168.1.50:64762 -> 193.110.128.199:80
TCP TTL:64 TOS:0x0 ID:12969 IpLen:20 DgmLen:606
***AP*** Seq: 0x7410C775 Ack: 0xA9DDE39B Win: 0x1015 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1321283346 249747
:

```

Figura 4-6 Fichero de alertas de Snort (Fuente: Autoría propia)

4.4 Pruebas de carga y tiempos de respuesta

Para medir el impacto en el rendimiento al usar Squid y Snort, vamos a utilizar el comando *ab* de Apache para cargar simultáneamente un grupo determinado de páginas con Snort y Squid funcionando:

```
ab -n 100 -c 10 https://www.google.com/
```

En este ejemplo hemos tomado 100 solicitudes a la dirección *https://www.google.com/* con 10 conexiones simultáneas, despreciando el *caching* habiendo reiniciado Squid y Snort.

Como podemos comprobar en la Tabla 4-1 y la Tabla 4-2, el sistema no se satura, pero cabe decir que sufre un retraso de casi un 300% debido al filtrado de Snort y Squid:

	Sin Squid/Snort	Con Squid/Snort
Server Software:	-	
Server Hostname:	www.google.com	
Server Port:	443	
SSL/TLS Protocol:	TLSv1. 2, ECDHE-RSA-AES128-GCM-SHA256, 2048, 128	
Document Path:	/	
Document Length:	259 bytes	
Concurrency Level:	10	
Time taken for tests:	2.972 seconds	5.973
Complete requests:	100	

Tabla 4-1 Comprobación de Benchmarks I (Fuente: Autoría propia)

	Mínimo		Varianza		Media		Máximo	
	Sin Squid	Con Squid	Sin Squid	Con Squid	Sin Squid	Con Squid	Sin Squid	Con Squid
Connect	82	135	229	417	285	392	517	842
Processing	22	26	62	146	59	148	116	291
Waiting	21	24	60	143	57	145	115	291
Total	112	383	291	563	269	518	555	933

Tabla 4-2 Comprobación de Benchmarks II (Fuente: Autoría propia)

4.5 Prueba de número de equipos conectados simultáneamente

El objetivo de esta prueba es saber si el punto de acceso constituido por el Raspberry Pi tolera sin problemas conectar un número razonable de dispositivos teniendo en cuenta que nuestra solución va orientada al entorno doméstico. Para dicha prueba se han conectado 5 equipos de los cuales constan: un iPad 4, una tableta Samsung Tab4.0 y tres ordenadores portátiles. Se ejecutó el *script* “load_time” (ver Anexo II: Scripts) con Squid y Snort funcionando y de nuevo sin funcionar. Cargó la página www.diariodecadiz.com tardando con Squid y Snort un tiempo total de 0,373 segundos, mientras que sin estar funcionando éstos tarda un tiempo total de 0,361 segundos como se puede ver en la Figura 4-7

```

[MacBook-Pro-de-Borja:bin Borjita$ load_time www.diariodecadiz.com
-----
Lookup time:    0,263 s
Connect time:   0,268 s
Pretransfer time: 0,268 s
Starttransfer time: 0,373 s
Size download:  154 bytes
Speed download: 412,000 bytes/s

Total time:     0,373 s
[MacBook-Pro-de-Borja:bin Borjita$ load_time www.diariodecadiz.com
-----
Lookup time:    0,268 s
Connect time:   0,297 s
Pretransfer time: 0,297 s
Starttransfer time: 0,361 s
Size download:  154 bytes
Speed download: 426,000 bytes/s

Total time:     0,361 s

```

Figura 4-7 Ejecución del script "load_time" (Fuente: Autoría propia)

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Como final a nuestro trabajo de investigación podemos decir que resultaría bastante fácil y asequible realizar la monitorización de nuestra red doméstica mediante un proxy server para filtrar contenido HTTP, pero para conseguir el filtrado de mejor calidad como lo es el poder filtrar paquetes HTTPS necesitamos indagar en la configuración de Squid. En cuanto a la detección de intrusiones, el uso de Snort tampoco ha supuesto ninguna dificultad.

Respecto a la legalidad vigente a la hora de interceptar el tráfico de terceras personas, el actual Código Penal con reforma el 30 de Marzo de 2015, en el cual se incluyen los delitos informáticos, cuenta con el artículo 197 bis que dice lo siguiente:

“1. El que por cualquier medio o procedimiento, vulnerando las medidas de seguridad establecidas para impedirlo, y sin estar debidamente autorizado, acceda o facilite a otro el acceso al conjunto o una parte de un sistema de información o se mantenga en él en contra de la voluntad de quien tenga el legítimo derecho a excluirlo, será castigado con pena de prisión de seis meses a dos años.

2. El que mediante la utilización de artificios o instrumentos técnicos, y sin estar debidamente autorizado, intercepte transmisiones no públicas de datos informáticos que se produzcan desde, hacia o dentro de un sistema de información, incluidas las emisiones electromagnéticas de los mismos, será castigado con una pena de prisión de tres meses a dos años o multa de tres a doce meses.” [35]

Esto conlleva que, para poder implantar nuestro sistema, como se refleja en el apartado 2 del citado artículo, es necesario el consentimiento de los usuarios, como podrían ser los invitados a nuestro hogar. Por otro lado hay que aclarar que para filtrar contenido, tal y como se plantea en el proyecto, se requiere el consentimiento del usuario dado que este debe de instalar el certificado generado. Esto no llevaría a tanta controversia puesto que en algunas universidades como la de Edimburgo ya se solicita a los usuarios la instalación de certificados siendo la universidad la autoridad certificadora o *Certification Authority* (CA). [36]

5.2 Líneas Futuras

De cara a futuros desarrollos de nuevos Trabajos de Fin de Grado sería una buena continuación crear una interfaz de usuario similar a la de la aplicación *Diladele Web Filter* mediante Apache [33] y Django [37], de manera que quedase una ventana en la cual el usuario pudiese configurar las distintas herramientas en un único punto de una manera amigable URLs bloqueadas, palabras clave, logs, y que posteriormente el servidor interactuase con Squid y Snort para crear unas reglas de bloqueo/aviso mediante correo electrónico. Otra posible ampliación sería implementar BroIDS [11] en lugar de Snort dado que funciona basándose en eventos y no en reglas, y E2guardian [38] como filtro en lugar de Squid.



Figura 5-1 E2guardian (Fuente: e2guardian.org)



Figura 5-2 BroIDS (Fuente: bro.org)

Otras líneas futuras sería implementar Squid en su versión más actualizada, dado que la versión utilizada, que es la que proporciona el repositorio de Raspbian, no es compatible con todos los navegadores debido a un problema de obsolescencia en el algoritmo de cifrado de los certificados. Por lo tanto se insta a usar la versión 3.5.17 (implementada en *Diladele Web Filter*) o posteriores, que permite solventar dichos problemas de certificados.

6 BIBLIOGRAFÍA

- [1] Instituto Nacional de Estadística, «Instituto Nacional de Estadística,» [En línea]. Available: http://www.ine.es/jaxi/Datos.htm?path=/t25/p450/base_2011/a2016/10/&file=03005.px. [Último acceso: 1 Marzo 2017].
- [2] Instituto Nacional de Estadística, «Instituto Nacional de Estadística,» [En línea]. Available: http://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176741&menu=ultiDatos&idp=1254735976608. [Último acceso: 1 Marzo 2017].
- [3] Instituto Nacional de Estadística, «Instituto Nacional de Estadística,» [En línea]. Available: http://www.ine.es/ss/Satellite?L=es_ES&c=INESeccion_C&cid=1259925528782&p=1254735110672&pagename=ProductosYServicios%2FPYSLayout. [Último acceso: 1 Marzo 2017].
- [4] Real Academia de la Lengua Española, «Web de la RAE definición de ciber -,>» [En línea]. Available: <http://dle.rae.es/?id=98ULSyc>. [Último acceso: 4 Marzo 2017].
- [5] Real Academia de la Lengua Española, «Web de la RAE definición de ataque,>» [En línea]. Available: <http://dle.rae.es/?id=4AR9Sjs>. [Último acceso: 4 Marzo 2017].
- [6] Real Academia de la Lengua Española, «Web de la RAE definición de seguro,>» [En línea]. Available: [5] <http://dle.rae.es/?id=XTrgHXd>. [Último acceso: 4 Marzo 2017].
- [7] «Web del Instituto Nacional de Ciberseguridad,>» [En línea]. Available: <https://www.incibe.es/sala-prensa/notas-prensa/incibe-publica-el-ranking-los-10-principales-incidentes-ciberseguridad>. [Último acceso: 4 Marzo 2017].
- [8] «Web del Instituto Nacional de Ciberseguridad,>» [En línea]. Available: <https://www.certs.es/alerta-temprana/bitacora-ciberseguridad>. [Último acceso: 4 Marzo 2017].
- [9] T. Broenink, «Web de Thijs Broenink,>» [En línea]. Available: <https://www.thijsbroenink.com/2016/09/xiaomis-analytics-app-reverse-engineered/>. [Último acceso: 4 Marzo 2017].
- [10] Y. Olmedo, «Web de Security Artwork,>» [En línea]. Available: <https://www.securityartwork.es/2016/11/29/analisis-trafico-red-dispositivos-moviles/>. [Último acceso: 4 Marzo 2017].
- [11] «Web de BroIDS,>» [En línea]. Available: <https://www.bro.org/>. [Último acceso: 4 Marzo 2017].

2017].

- [12] «Web de Critical Stack,» [En línea]. Available: <https://criticalstack.com/>. [Último acceso: 5 Marzo 2017].
- [13] «Web de Elastic,» [En línea]. Available: <https://www.elastic.co/products/logstash>. [Último acceso: 4 Marzo 2017].
- [14] «Web de Elastic,» [En línea]. Available: <https://www.elastic.co/products/elasticsearch>. [Último acceso: 4 Marzo 2017].
- [15] T. Smith, «Web de Tripwire,» [En línea]. Available: <https://www.tripwire.com/state-of-security/security-data-protection/sweet-security-part-2-creating-a-defensible-raspberry-pi/>. [Último acceso: 4 Marzo 2017].
- [16] «Web de Pwnpi,» [En línea]. Available: <http://pwnpi.sourceforge.net/index.html>. [Último acceso: 4 Marzo 2017].
- [17] «Web de Kali Linux,» [En línea]. Available: <https://www.kali.org/>. [Último acceso: 4 Marzo 2017].
- [18] S. Kamkar, «Web de Samy Kamkar,» [En línea]. Available: <https://samy.pl/poisonmap/>. [Último acceso: 4 Marzo 2017].
- [19] The Raspberry Pi Foundation, «MagPi #1,» *MagPi*, nº 1, pp. 6-8, 1 Mayo 2012.
- [20] The Raspberry Pi Foundation, «MagPi #31,» *MagPi*, nº 31, pp. 6-7, 31 Marzo 2015.
- [21] The Raspberry Pi Foundation, «MagPi #43,» *MagPi*, nº 43, pp. 6-12, 20 Marzo 2016.
- [22] «Web D-link,» [En línea]. Available: <http://www.dlink.com/es/es/support/product/dir-615-wireless-n-300-router>. [Último acceso: 4 Marzo 2017].
- [23] «Web de Raspbian,» [En línea]. Available: www.raspbian.org. [Último acceso: 4 Marzo 2017].
- [24] «Web de Red española para la Interconexión de los Recursos Informáticos,» [En línea]. Available: <https://www.rediris.es/cert/doc/unixsec/node26.html#SECTION07682000000000000000>. [Último acceso: 4 Marzo 2017].
- [25] «Web del Instituto Nacional de Tecnologías Educativas y Formación del profesorado,» [En línea]. Available: http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m6/proxy_squid.html. [Último acceso: 4 Marzo 2017].
- [26] «Web de Diladele,» [En línea]. Available: https://docs.diladele.com/administrator_guide_4_9/index.html. [Último acceso: 4 Marzo 2017].
- [27] «Web de SD Formatter,» [En línea]. Available: https://www.sdcard.org/downloads/formatter_4/. [Último acceso: 4 Marzo 2017].
- [28] «Web de Raspberry Pi,» [En línea]. Available: <https://www.raspberrypi.org/documentation/installation/installing-images/mac.md>. [Último acceso: 4 Marzo 2017].
- [29] «Web de Frillip,» [En línea]. Available: <https://frillip.com/using-your-raspberry-pi-3-as-a>

wifi-access-point-with-hostapd/. [Último acceso: 4 Marzo 2017].

- [30] «Web de Wireshark,» [En línea]. Available: <https://www.wireshark.org/>. [Último acceso: 4 Marzo 2017].
- [31] «Web de Snort,» [En línea]. Available: <https://www.snort.org/>. [Último acceso: 4 Marzo 2017].
- [32] «Web de Github,» [En línea]. Available: <https://github.com/jpelias/squid3-ssl-bump>. [Último acceso: 4 Marzo 2017].
- [33] «Web de Apache,» [En línea]. Available: <https://httpd.apache.org/>. [Último acceso: 4 Marzo 2017].
- [34] «Web de Shalla Secure Service,» [En línea]. Available: www.shallalist.de. [Último acceso: 4 Marzo 2017].
- [35] «Web del Boletín Oficial del Estado,» [En línea]. Available: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2015-3439. [Último acceso: 4 Marzo 2017].
- [36] «Web de la Universidad de Edimburgo,» [En línea]. Available: <http://www.ed.ac.uk/information-services/computing/computing-infrastructure/network/certificates/install/installmacos>. [Último acceso: 4 Marzo 2017].
- [37] «Web de Django,» [En línea]. Available: <https://www.djangoproject.com/>. [Último acceso: 4 Marzo 2017].
- [38] «Web de E2guardian,» [En línea]. Available: <http://e2guardian.org/cms/index.php>. [Último acceso: 4 Marzo 2017].
- [39] D. J. W. Andrew S. Tanenbaum, Redes de Computadoras, 5ª edición ed., Pearson (ISBN 978-607-32-0817-8), 2012, pp. 587-588.

ANEXO I: CONTENIDO HTML

En este anexo vamos a recoger el código empleado para crear la página en HTML que se muestra al usuario cuando intenta acceder a un sitio bloqueado:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title> Banned! </title>
  </head>
  <body>
    <div align="center">
      <p><h1>¡Has sido bloqueado por el Homemade Firewall!</h1></p>
      
      <p><h2>Sistema creado con Raspberry Pi</h2></p>
    </div>
  </body>
</html>
```

ANEXO II: SCRIPTS

Aquí vamos a poner el contenido de los scripts que hemos creado. El primero de todos será permitir actualizar una de base de datos de lista negras de páginas:

Creamos un script en /usr/local/bin llamado “dbupdate” con el siguiente código:

```
#!/bin/bash
cd /home/pi/Downloads

if [ -f /home/pi/Downloads/shallalist.tar.gz ];
then
    sudo rm -rf shallalist.tar.gz
    echo "El archivo existe actualmente"
    sudo rm -rf shallalist.tar.gz
    sudo rm -rf BL
else
    echo "descargando archivo"
    sudo wget http://www.shallalist.de/Downloads/shallalist.tar.gz
    tar -xzf shallalist.tar.gz
    sudo cp BL -R /var/lib/squidguard/db
    echo "archivo copiado satisfactoriamente"
    cd /var/lib/squidguard/db
    sudo chmod -R 755 /var/lib/squidguard/db/BL
    sudo chown -R proxy:proxy /var/lib/squidguard/db/BL
    sudo /usr/bin/squidGuard -C all
    echo "compilado"
    cd /home/pi/Downloads
    sudo rm -rf shallalist.tar.gz
    sudo rm -rf BL
    echo "eliminado"
fi
```

Lo metemos en el fichero crontab, después de darle permisos, para que compruebe si existen actualizaciones cada día 1 del mes.

```
crontab -e
```

```
0 0 1 * * dbupdate
```

Script de aviso por correo electrónico:

Creamos el fichero de comprobación llamado “check”

```
#!/bin/bash

cp /var/log/snort/alert /home/pi/fichero/alert
diff --suppress-common-lines -N /home/pi/fichero/alert /home/pi/fichero/anterior.txt > /home/pi/fichero/diferencias.txt

if [ -s /home/pi/fichero/diferencias.txt ];
then
echo "Alertas" | mail -s "Atención!" tfgraspberry@gmail.com < /home/pi/fichero/diferencias.txt
fi
mv /home/pi/fichero/alert /home/pi/fichero/anterior.txt
```

Lo introducimos en crontab para que se ejecute una vez al día a las 00:00 de cada día

```
0 0 * * * check
```

Script de comprobación de carga de tiempos. Lo llamaremos “load_time”

```
#!/bin/bash

estadistica () {
    local site=$1
    echo $site echo ${site} | sed -n 's/./-/gp'
    curl -w '
Lookup time:¥t%{time_namelookup} s
Connect time:¥t%{time_connect} s
Pretransfer time:¥t%{time_pretransfer} s
Starttransfer time:¥t%{time_starttransfer} s
Size download:¥t%{size_download} bytes
Speed download:¥t%{speed_download} bytes/s

Total time:¥t%{time_total} s ' -o /dev/null -s $site
    echo
}

for i in ${@}; do
    estadistica $i
done
```

