UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de un sistema cortafuegos modular basado en XDP

Autor: Iván Gallardo Romero
Tutor: Luis de Pedro Sánchez
Ponente: Jorge Enrique López de Vergara Méndez

noviembre 2020

Iván Gallardo Romero
Desarrollo de un sistema cortafuegos modular basado en XDP

Iván Gallardo Romero
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia y amigos por su apoyo incondicional durante todos los años dedicados a esta carrera. Su apoyo ha sido muy importante para mí durante las decisiones más difíciles. También me gustaría dar las gracias a mis amigos y compañeros de la universidad con los que he pasado muchas horas tanto dentro como fuera de la misma. Han sido un gran apoyo durante la carrera.

En particular, quiero destacar el apoyo de mi tutor Luis De Pedro Sánchez por la ayuda que me ha proporcionado para la elaboración de este proyecto y el desarrollo de esta aplicación.

También quiero tener un recuerdo para Serhii Pimenov dado que me ha ayudado en el desarrollo con su librería de código abierto, Metro 4, para el desarrollo de la interfaz web.

Y por supuesto a Alejandro Romero del Campo por su trabajo de clasificación de flujos utilizando técnicas de aprendizaje automático que me ha permitido añadir un ejemplo de programa externo de detección de huellas en el cortafuegos.

Resumen

En un mundo cada vez más conectado, con el crecimiento exponencial del volumen de datos en la red, los ataques en internet, como los ataques de denegación del servicio, son cada vez más frecuentes.

Actualmente es de interés el desarrollo de herramientas de seguridad modulares que, por una parte, permitan añadir fácilmente aportaciones de la comunidad de desarrolladores y, por otra, sean sencillas de utilizar por parte del usuario.

La aportación de este trabajo de fin de grado cosiste en el diseño y desarrollo de un cortafuegos, que permitirá, por un lado, que los programadores añadan sus propias soluciones de forma muy sencilla y, por otro lado, a los usuarios gestionar el cortafuegos fácilmente.

En primer lugar, se analizarán las soluciones y sistemas de cortafuegos existentes en el mercado con el fin de detectar sus fortalezas y debilidades. El propósito es diseñar una aplicación que supla estas carencias y sea verdaderamente útil para el usuario.

Una vez realizado el estudio, se procederá a diseñar la aplicación utilizando el mejor sistema para filtrar paquetes. Entre las distintas posibilidades disponibles actualmente, XDP es una alternativa muy interesante puesto que permite que se ejecute código en el controlador de tarjeta de red. La ventaja de este sistema es poder eliminar paquetes a alta velocidad y con menos recursos, algo muy importante cuando se trata de detener un ataque. La principal desventaja es la complejidad del desarrollo y comunicación debido a la necesidad de utilizar lenguajes de bajo nivel en el funcionamiento interno y de alto nivel en la interfaz.

Por último, se realizarán pruebas de rendimiento y de validación con el fin de comprobar que todos los requisitos se satisfacen y el cortafuegos es viable. Esto es muy importante, ya que es una aplicación corriendo en una zona crítica del sistema.

Palabras clave

Cortafuegos, metodologías ágiles, filtrado de paquetes, XDP, eBPF, Flask Python, ataques de internet, API REST, desarrollo web

Abstract

Nowadays, as the volume of data on the internet increases exponentially, cyber attacks such as distributed denial-of-service, are becoming increasingly common.

The development of modular security tools that, on the one hand, allow easy addition of contributions from the developer community and, on the other hand, are user-friendly is currently of interest.

The contribution of this end-of-degree project consists in the design and development of a firewall, which will allow, on the one hand, programmers to add their own solutions in a very simple way and, on the other hand, users to manage the firewall with ease.

First, the existing firewall solutions and systems on the market will be analysed in order to identify their strengths and weaknesses. The purpose is to design an application that will address these shortcomings and be truly useful to the user.

Once the study is completed, the application will be designed using the best system for filtering packages. Among the various possibilities currently available, XDP is a very interesting alternative since it allows the code to be executed using the network card driver. The advantage of this system is its ability to eliminate packages at high speed and with less resources, which is very important when it comes to stopping an attack. The main disadvantage is the complexity of the development and communication due to the need to use low level languages in the internal functioning and high level languages in the interface.

Finally, performance and validation tests will be carried out in order to verify that all requirements are met and the firewall is viable. This is very important, since it is an application running in a critical area of the system.

Keywords

Firewall, agile methodologies, packet filtering, XDP, eBPF, Flask Python, cyber attacks, API REST, web development

Índice

Listas

Lista de algoritmos

Lista de códigos

Lista de cuadros

Lista de ecuaciones

Lista de figuras

Lista de tablas

Lista de cuadros

1

Introducción

En esta sección contiene el marco general del proyecto, empezando por la motivación que ha llevado a realizar la aplicación y explicando los objetivos principales de la misma.

1.1. Motivación

Actualmente es de interés [1] el desarrollo de herramientas de seguridad modulares que permitan aportaciones de la comunidad de desarrolladores. Con el aumento exponencial del volumen y uso de datos en la red, los ataques de internet, como por ejemplo los ataques de denegación del servicio distribuidos, también llamados **DDOS** por sus siglas en inglés *Distributed Denial of Service*, son cada vez más frecuentes. [2].

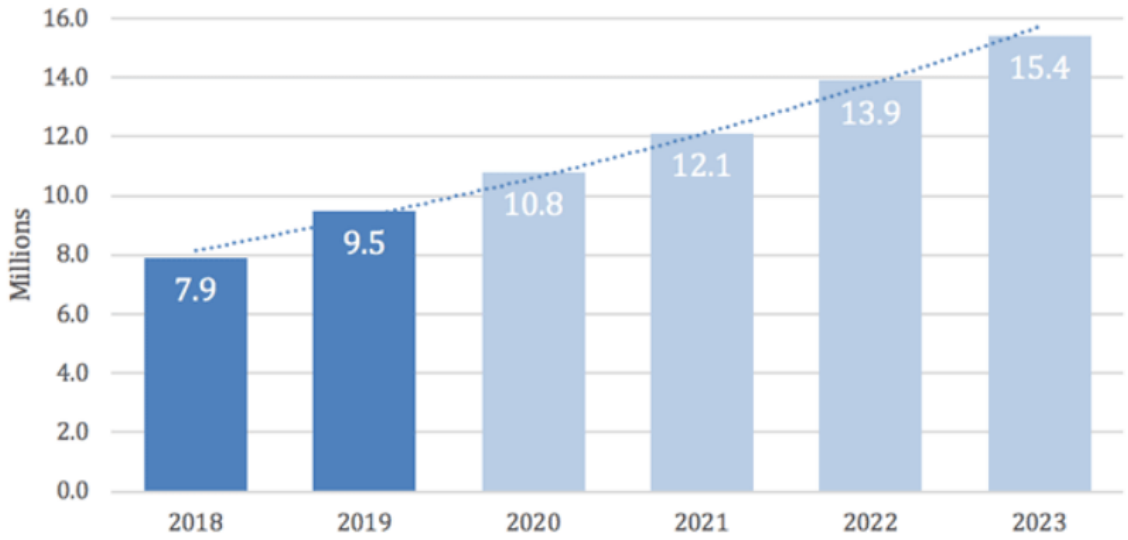


Figura 1.1: Análisis de Cisco del historial y las predicciones totales de ataques DDoS [3].

El abaratamiento de los servidores dedicados, compartidos, y las instancias en la nube [5] ha permitido que cada vez más usuarios sin conocimientos avanzados en informática son capaces de crear servidores para uso recreativo o, incluso, ganar dinero con la publicidad [6] de su contenido y subscripciones [2]. Estos servidores son, por ejemplo:

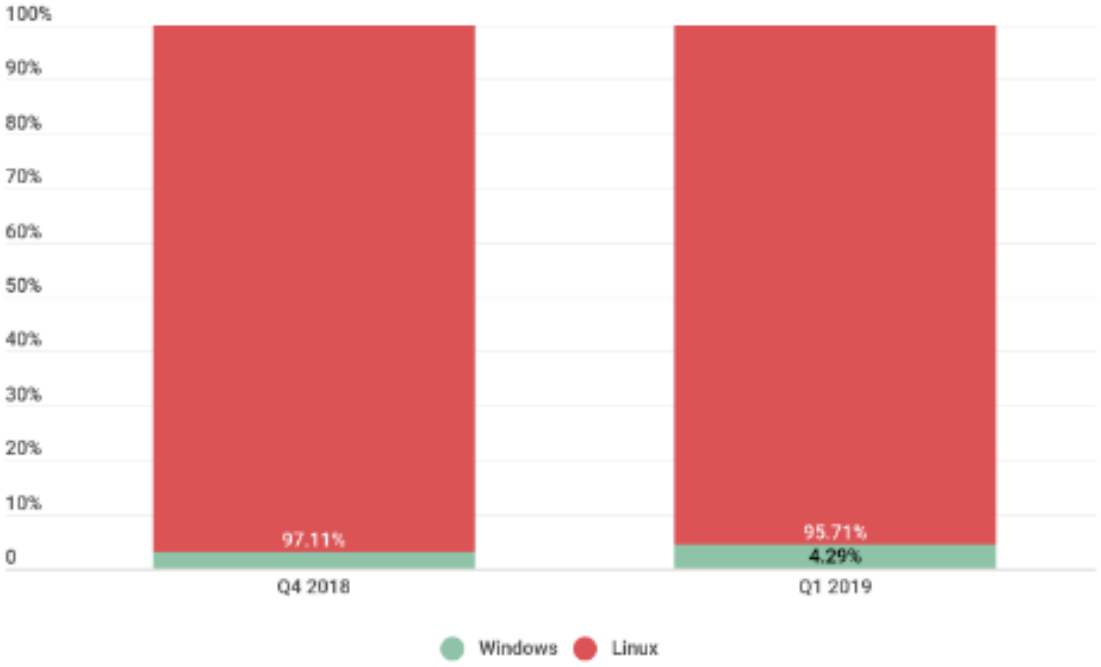


Figura 1.2: Análisis de Kaspersky sobre el sistema operativo más utilizado para atacar en 2018 y 2019. [4].

- Servidores de juegos, como Minecraft [7].
- NextCloud y ownCloud (Servidor de archivos en internet) [8].
- Mattermost es un servicio de chat en línea de código abierto y autohospedable [9].
- Semillas de torrents y descargador de torrents por web [10].
- Servidores de Discord y TeamSpeak 3 (chat y habla en directo) [11].
- Servidor web, de correo, etc. . .

La mayoría de los usuarios que crea este tipo de aplicaciones no cuenta con el conocimiento suficiente como para proteger la máquina y tampoco es una buena opción comprar un cortafuegos avanzado debido a su alto precio. Algunas empresas ofrecen en el alquiler de sus servidores más baratos un servicio de protección básico incluido [12], pero en la mayoría de las ocasiones este no cubre todos los ataques a la máquina, no muestra información sobre el ataque, no puedes personalizar la protección y tampoco garantizan la eficacia. Otras empresas, no incluyen protección en sus servidores [13], lo que conlleva un riesgo alto debido a las facilidades que existen para ejecutar un ataque. Las empresas que no ofrecen protección pueden ser de interés para el usuario debido a su localización, para reducir la latencia, u otras características.

Una de las mayores empresas especializadas en detener ataques de red, CloudFlare [14], no utiliza hardware avanzado para realizar el filtrado de paquetes, sino que está en cada uno de sus servidores. Esto les permite aumentar su capacidad de detección y bloqueo a medida que su infraestructura se

hace más grande [15]. El cortafuegos de Facebook también aplica esta estrategia [16].

1.2. Objetivos

Una vez analizada la motivación podemos exponer los objetivos principales que perseguirá la aplicación que se desarrolle con este trabajo.

O-1.– Crear un cortafuegos que permita controlar y mostrar el tráfico de la máquina.

O-2.– La interfaz cortafuegos deberá ser sencilla para que la puedan utilizar todos los usuarios.

O-3.– El cortafuegos deberá permitir a la comunidad de desarrolladores aportar soluciones para detectar huellas y ataques de tráfico de forma sencilla.

O-4.– El cortafuegos deberá permitir bloquear el tráfico de un ataque de forma sencilla, tanto a la comunidad de desarrolladores como al propio usuario.

O-5.– El cortafuegos deberá ser compatible con Linux, el principal sistema operativo al que atacan.

O-6.– El sistema deberá poder bloquear los paquetes maliciosos en el menor tiempo posible y sin perjudicar el uso de CPU del servidor en caso de ataque.

1.3. Organización de la memoria

El presente documento tiene como propósito la descripción del plan de proyecto, con el fin de que el lector pueda conocer las diferentes fases en las que se ha llevado a cabo el proyecto y el motivo de las decisiones que se han tomado para crear el producto final. En el documento se pueden distinguir los siguientes capítulos:

- **Estado de arte:** En esta sección se hablará sobre las soluciones que existen actualmente en el mercado mostrando las diferencias, similitudes, ventajas y desventajas con respecto a la que será desarrollada.
- **Diseño:** Dentro de este capítulo se describirán con detalle los objetivos de la aplicación y las características generales del sistema por subsistemas para una mejor comprensión. También se detallarán los requisitos funcionales y no funcionales que debe cumplir la aplicación y, por último, se mostrarán las pantallas de la aplicación y cómo se interactúa con ellas.
- **Desarrollo:** En este apartado se explicará cómo funciona la aplicación de forma interna, los problemas encontrados y las soluciones planteadas.
- **Integración, pruebas y resultados:** En esta sección se mostrará la integración del cortafuegos en Linux y el uso de la interfaz en los navegadores, así como las pruebas realizadas a la aplicación.
- **Conclusiones:** Por último, este apartado engloba las conclusiones del trabajo realizado y el trabajo futuro sobre este proyecto.

Por último, en los **apéndices** se proporcionará un manual de instalación de la aplicación y un manual para el programador. También estará disponible un cronograma con la planificación del proyecto, el GitHub y una demostración en tiempo real con ejemplos.

2

Estado del arte

A lo largo de este capítulo se van a estudiar las diferentes soluciones que existen para filtrar, controlar y bloquear el tráfico en una máquina, principalmente con el fin de detener los ataques DDOS.

2.1. Análisis de las diferentes soluciones

Con el fin de que la aplicación que se pretende desarrollar sea útil para los usuarios, en este apartado se van a estudiar las soluciones que existen actualmente y desventajas con respecto a la aplicación.

2.1.1. Proveedores de alojamiento con contafuegos

En la actualidad, la mayoría de los proveedores de alojamiento y alojamiento en la nube proporcionan un cortafuegos [17] con el fin de proteger su infraestructura o a los otros clientes, como en el caso en el cual el servidor sea compartido. Otros proveedores no ofrecen esta protección incluida. [18]

Proveedores sin protección

En algunos casos un cliente puede verse obligado a seleccionar una empresa que no incluye protección contra ataques debido a las leyes de protección de datos o la localización del servidor para reducir la latencia, por ejemplo.

Proveedores que ofrecen protección gratuita

En la mayoría de estos casos, al ser una protección gratuita y centrada en proteger la infraestructura del proveedor más que el cliente podemos encontrar numerosas desventajas:

- No se puede administrar ni configurar el cortafuegos. [19]
- No se muestran estadísticas ni información del ataque con el fin de prevenirlo. [12]
- No filtran todos los ataques, sobre todo a nivel de la capa de aplicación. [20]

- En algunos casos, entre la detección del ataque y su mitigación puede producirse una demora que conlleva una pérdida momentánea del servicio. [21]
- Analizan el tráfico de internet, pero no el que proviene, por ejemplo, de una red privada. [21]
- En caso de ataque, algunas empresas desvían el tráfico hacia la infraestructura de mitigación. Esto conlleva un aumento de la latencia. [21]

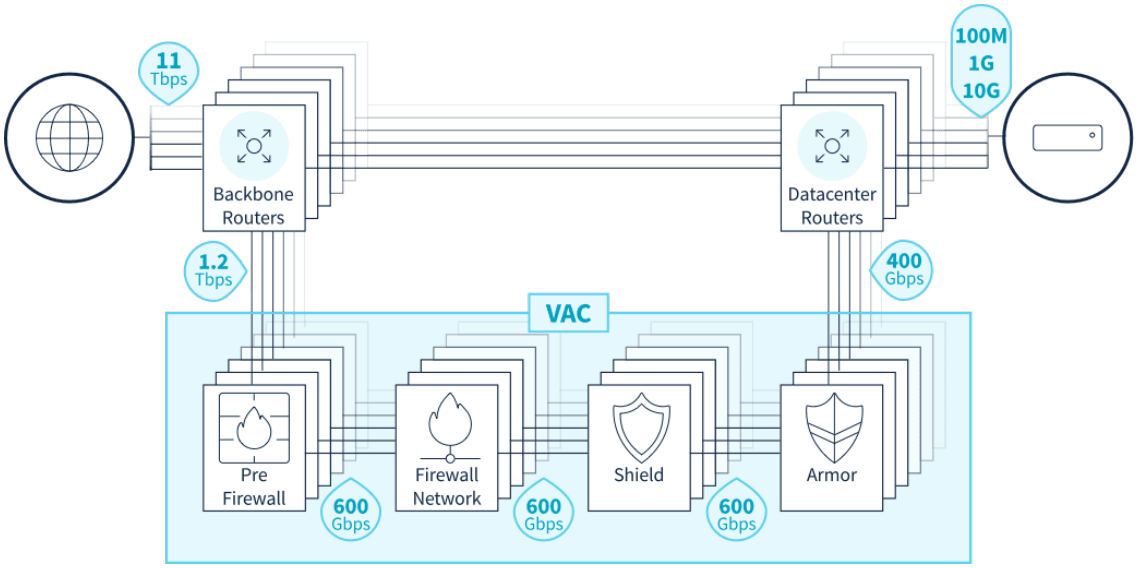


Figura 2.1: Tecnología anti-DDoS de OVH. [21].

En la figura se muestra el cortafuegos de uno de los mayores proveedores de servicios en la nube, OVH [21]. En sus paquetes más básicos no es posible configurar ninguna regla ni ver información del ataque. Este sistema envía una parte del tráfico enviado por los routers con el fin de analizar la dosmilésima parte del tráfico. Por tanto, no se activa de manera inmediata. Si el ataque coincide con alguna de sus firmas el tráfico desde la red pública hasta el servidor, sin analizar la red privada, pasa por una serie de cortafuegos llamados VAC. Este sistema es común a todos los servidores y, al activarse, el tráfico pasará por los cortafuegos del VAC aumentando la latencia. Además, no da información del ataque con el fin de prevenirlo.

Por tanto, es de interés crear una aplicación que permita añadir de forma muy sencilla soluciones personalizadas para detectar huellas y tráfico malicioso y de información sobre el tráfico de la máquina. También, será fácilmente administrable y estará en el propio servidor. Muchas empresas, como Cloudflare [15] o Facebook [16] prefieren tener un filtrado de paquetes en cada servidor debido a las numerosas ventajas, como ahorrarse los costes de hardware externo y aumentar la capacidad de mitigación a medida que la infraestructura crece. También para evitarse retrasos o fugas de seguridad.

2.1.2. Aplicaciones similares

Hoy en día existen numerosas aplicaciones que analizan, filtran, mitigan y controlan el tráfico. En esta sección se mostrarán estas soluciones.

Principales cortafuegos de Linux

Aplicaciones como IPtables [22], IPFire [23] o Suricata [24] son potentes cortafuegos para Linux, pero para utilizarlos es necesario tener altos conocimientos sobre la red. Por ejemplo, en IPtables para bloquear un conjunto muy grande direcciones del protocolo de internet (en inglés *Internet protocol* o **IP**) hace falta crear un conjunto hash [25] para aumentar la velocidad de la búsqueda y no ralentizar el tráfico en el servidor significativamente. Los comandos, ajustes avanzados hacen que un usuario sin grandes conocimientos de la red no los pueda usar.

Cortafuegos en Windows más sencillos de utilizar

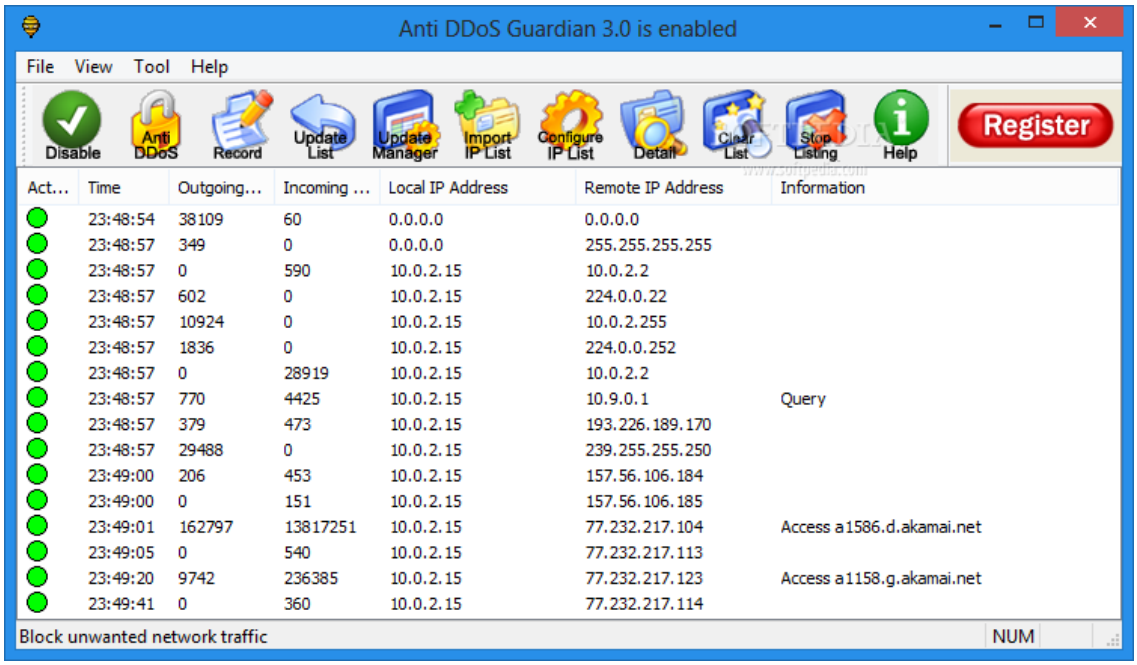


Figura 2.2: Anti DDoS de BeeThink. [26].

Algunas empresas como FortGuard [27] BeeThink [26] vender cortafuegos compatibles con Windows capaces de mostrar el tráfico del servidor, bloquear el tráfico y limitarlo. Su principal propósito es detener ataques de denegación del servicio. Además de su elevado coste y la incompatibilidad con el sistema operativo Linux, principal blanco de los ataques, las principales desventajas son, por una parte, la imposibilidad de que desarrolladores aporten sus propias soluciones y, por otro, la poca eficacia de bloquear los ataques limitando el tráfico.
Por tanto, la aplicación deberá permitir a los usuarios que no tienen grandes conocimientos de informática utilizarla y a los desarrolladores añadir soluciones personalizadas de forma sencilla. El sistema deberá ser compatible con las últimas soluciones contra ataques, como por ejemplo detectar huellas y compararlas con firmas de ataques.

2.2. Técnicas de bloqueo de paquetes

Ser capaz de descartar paquetes muy rápidamente es vital para detener ataques en la red. Por tanto, esta sección analizará las técnicas para descartar paquetes en Linux. Se tratará de escoger la mejor ya que técnica tiene sus ventajas y limitaciones.

2.2.1. Análisis de las diferentes técnicas

Actualmente, en Linux, se utiliza Netfilter [28] para interceptar, bloquear y manipular paquetes. Este proyecto no solo ofrece componentes para interactuar con los paquetes, sino que también ofrece herramientas y librerías.

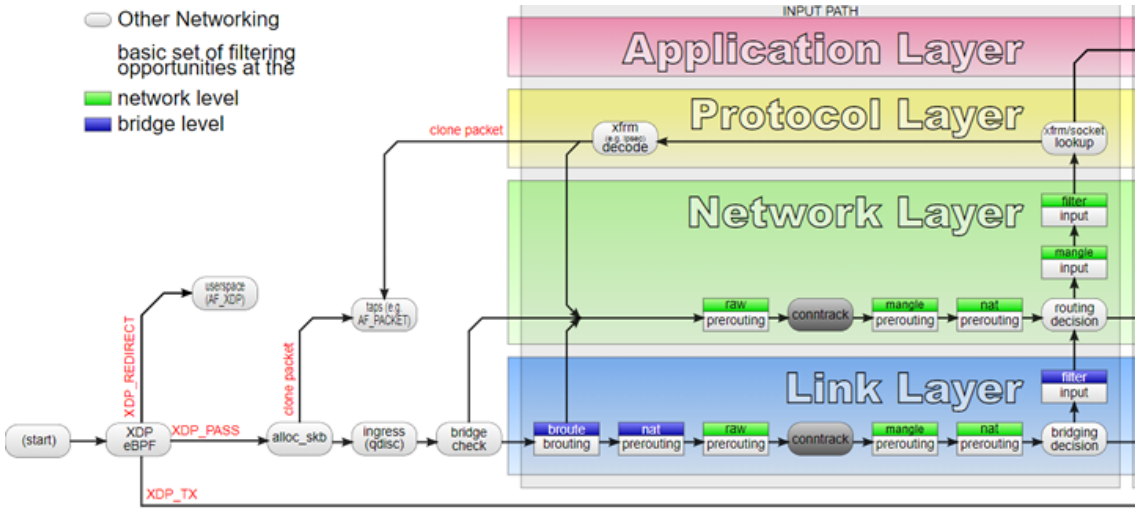


Figura 2.3: Diagrama de flujo de datos de entrada de Netfilter. [28].

IPtables ha sido la herramienta principal para implementar cortafuegos y filtros en Linux durante muchos años [29]. Con el crecimiento de las reglas de configuración y bloqueo se creó la herramienta IPset [30], que permitió comprimir estas reglas en una tabla hash con el fin de aumentar la velocidad de respuesta a medida que las conexiones a internet aumentaban. Por desgracia, este aumento no fue suficiente y actualmente se siguen investigando nuevos avances. Con esta investigación surgió el filtro de paquetes de Berkeley (en inglés *Berkeley Packet Filter* o **BPF**) [31], una tecnología que permite analizar el tráfico de red sin procesar. También ha surgido ruta de datos exprés (en inglés eXpress Data Path o **XDP**) [32], una ruta de datos de alto rendimiento basada en BPF que permite decidir al usuario el destino del paquete antes de ser procesado por el núcleo. Estas nuevas herramientas están actualmente en continuo desarrollo y evolución. En la figura 2.3 se puede observar el flujo de datos de entrada de paquetes de Netfilter. La principal idea es que cuanto antes se descarte el paquete, menos recursos se necesitarán para el procesamiento. Más adelante, en este documento, se analizarán las diferentes formas que hay para bloquear un paquete, por ejemplo, en la capa de aplicación.

3

Diseño

Esta sección comienza analizando los objetivos principales y las características generales de la aplicación. A continuación, se describen los subsistemas de la aplicación, detallando su funcionalidad y relación correspondiente. Por último se especifican los requisitos funcionales y no funcionales del sistema.

3.1. Descripción y objetivos de la aplicación

Se pretende que la aplicación creada en XDP sea transparente para el usuario y solo se muestre la aplicación web que permitirá mostrar información del tráfico que pasa por el servidor a tiempo real. También tiene por objetivo bloquear tráfico y ayudar mitigar ataques de denegación del servicio.
La aplicación debe tener las siguientes características:

- La interfaz deberá ser sencilla de utilizar.
- Mostrar información en tiempo real del tráfico permitiendo la compatibilidad con IP e IPV6, así como los principales protocolos de red.
- Bloquear direcciones, puertos y protocolos.
- Limitar el tráfico, analizarlo y filtrarlo.
- Proteger la interfaz web del cortafuegos por medio de una contraseña maestra almacenada de forma segura.
- Deberá ser muy sencillo para otros desarrolladores añadir otros programas mediante la Interfaz de programación de aplicaciones de transferencia de estado representacional, en inglés *application programming interface representational state transfer* o **API REST**.
- Ofrecer un buen rendimiento.

3.2. Arquitectura de la aplicación

En sección se encuentran los diagramas dónde se explica el diseño de la arquitectura de la aplicación. El primer diagrama tiene mayor nivel de abstracción que el segundo.

Visión general

A nivel general (Figura 3.1), los paquetes llegan al núcleo de Linux donde se encuentra la aplicación XDP enganchada. Esta aplicación, mediante una lista de exclusiones que comparte con el programa del espacio de usuario en C es capaz de descartar paquetes muy rápidamente, ya que no pasan por la pila de red el sistema. Esta aplicación en C está a su vez conectada mediante memoria compartida con la interfaz en Python y pueden compartirse reglas entre sí. Por último, mediante la API REST se podrán conectar otras aplicaciones que permitirá, de forma muy sencilla, controlar el cortafuegos.

Visión detallada

En concreto (Figura 3.2), los paquetes del servidor llegan a la tarjeta de red y llegan al programa XDP sin pasar por la pila de red el sistema. Por cada unidad central de procesamiento (conocida por las siglas **CPU**, del inglés: *Central Processing Unit*), el programa XDP cargado en el núcleo analiza mínimamente el paquete y si está bloqueado, según las listas de excusión en el mapa BPF, lo descarta. Por otro lado, si el paquete se acepta se recogen las estadísticas y muestras correspondientes y se deja continuar por el núcleo hasta su destino. Estas estadísticas se recogen mediante mapas BPF en la aplicación del espacio de usuario en C. En general, se utiliza un mapa BPF de tipo clave-valor por cada tipo de exclusión y un mapa BPF de tipo evento del núcleo de Linux para las estadísticas y muestras. Una vez en el espacio usuario, el programa en C suma todas las muestras de todas las CPU, controla el tráfico y se comunica con la interfaz en Python mediante memoria compartida. La interfaz permite controlar el cortafuegos y mediante la API REST otros desarrolladores pueden controlar el cortafuegos.

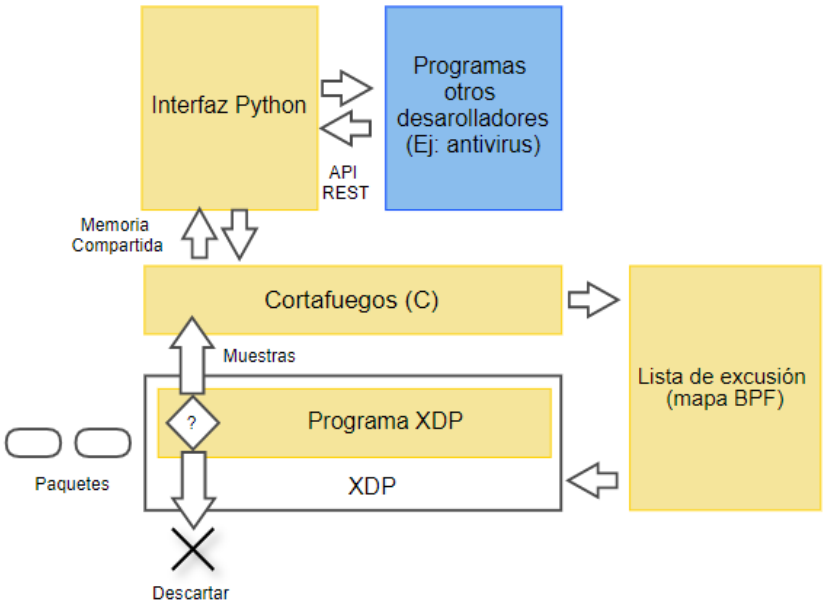


Figura 3.1: Arquitectura de la aplicación en general.

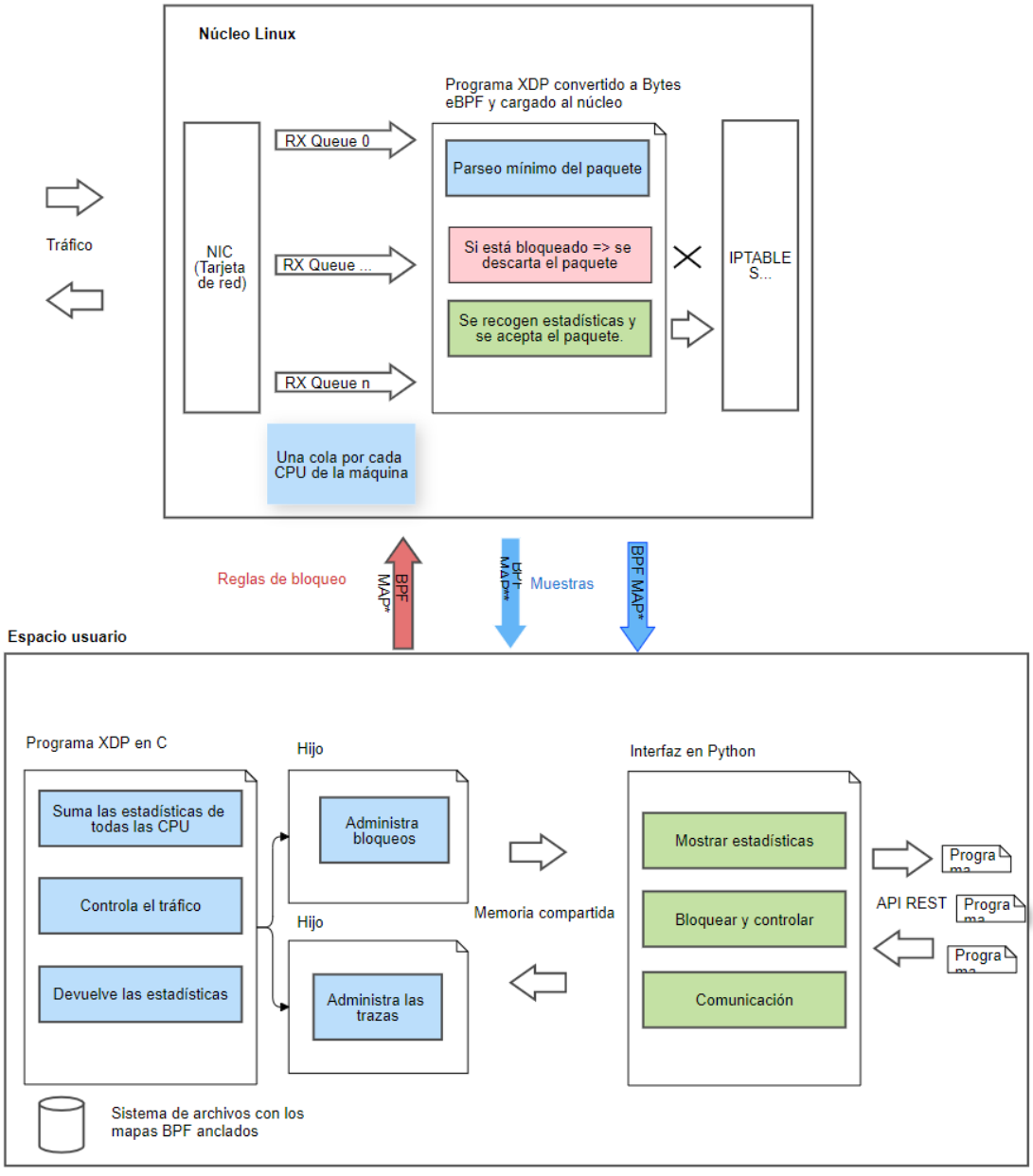


Figura 3.2: Arquitectura de la aplicación en específico. En amarillo lo desarrollado en este proyecto.

Decisiones con respecto al rendimiento

Una aplicación de tipo cortafuegos requiere detener de forma rápida los paquetes y el tráfico malicioso sin consumir excesivos recursos en la máquina. Por tanto, se han tomado las siguientes decisiones de diseño:

Por una parte, se utilizará XDP para la construcción del cortafuegos debido a que es la herramienta más rápida para bloquear paquetes en Linux. Estos estudios se han puesto en el apartado desarrollo del documento.

Por otra, con el fin de librar de carga las zonas críticas, se han tomado las siguientes decisiones:

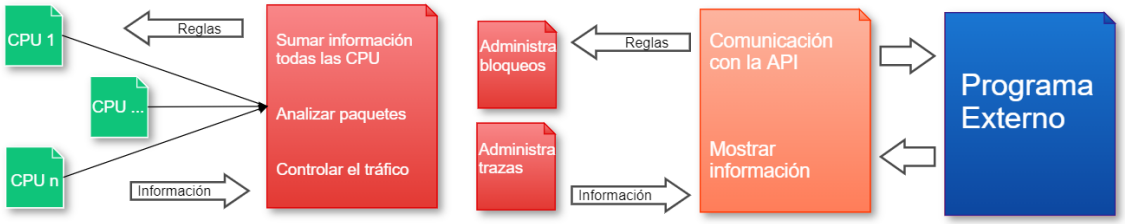


Figura 3.3: Vista del diseño del rendimiento de la aplicación.

En primer lugar, si se considera el núcleo como una zona crítica y se utilizan operaciones atómicas debe protegerse la memoria compartida. Esto implicaría aumentar considerablemente la carga del núcleo. Para evitarlo, se tomarán las estadísticas para cada procesador y se sumarán en el espacio de usuario. Además, el núcleo solo analizará el paquete mínimamente, comprobará los bloqueos y enviará la información al espacio usuario. La finalidad es no cargar el núcleo para que los bloqueos sean más rápidos y los paquetes fluyan con normalidad.

En segundo lugar, la aplicación de espacio usuario estará programada en C debido su excelente rendimiento. La aplicación principal sumará las estadísticas de cada CPU, analizará más en detalle cada paquete e impondrá las restricciones del tráfico. También se encargará de comunicarse y enviar información a la interfaz web. Esta aplicación, a su vez, tendrá dos subprocesos uno para administrar las reglas de bloqueo y otro para enviar y gestionar las trazas. Este último solo estará en ejecución cuando se activen las trazas.

Por último, la interfaz web se escribirá en Python y permitirá crear reglas, mostrar información y se encargará de la comunicación con los programas externos.

En resumen, se ha cargado el núcleo lo menos posible trasladando toda la carga a la aplicación en el espacio usuario en C.

3.3. Características generales del sistema: subsistemas

La aplicación cuenta con requisitos interrelacionados que han sido englobados en una serie de subsistemas para su mejor comprensión. Estos subsistemas a su vez pueden depender entre sí.

3.3.1. Subsistema de autenticación

El subsistema de autenticación es el menos extenso de la aplicación, pero uno de los más importantes ya que engloba la funcionalidad referente al manejo de la sesión, y autenticación en la API REST. Permite al usuario o programa acceder a la aplicación. Por una parte, inicio de sesión se realiza mediante una contraseña maestra guardada con HASH que se configura la primera vez que se instala la aplicación. En caso de pérdida o cambio de contraseña es necesario hacer una reinstalación del cortafuegos. Por otra, en la API se realiza mediante una llave que puede regenerar el usuario. Finalmente, se podrá cerrar la sesión e incluso apagar el cortafuegos y todos sus componentes liberándolos de forma correcta.

3.3.2. Subsistema de análisis de tráfico

El subsistema de análisis de tráfico engloba toda la funcionalidad referente al rastreo, seguimiento y generación de muestras del tráfico de la máquina. Es uno de los subsistemas principales de la aplicación ya que permite presentar el tráfico de la capa de internet con el fin de identificar las amenazas de red para, posteriormente, limitar su impacto. En concreto, se proporcionará información de los paquetes totales, paquetes por segundo, KBytes totales y Mbits por segundo de cada dirección IP en tiempo real. Esta información se podrá actualizar automáticamente o a petición del usuario y se podrán buscar direcciones. También contiene la funcionalidad para descargar los datos con el objetivo de tener una huella del ataque. Por último, se podrán limpiar los datos para eliminar información anticuada y redundante.
Este subsistema, también, será el encargado de recoger las trazas de tráfico para que mediante el subsistema de comunicación se pueda obtener una muestra de tráfico en formato PCAP.

Subsistema de información en tiempo real

El subsistema de información en tiempo real muestra información más detallada sobre los paquetes que recibe la máquina. Tiene la funcionalidad de analizar los protocolos de la capa de transporte más utilizados por una dirección y mostrar el historial en tiempo real. Es compatible con los protocolos ICMP, UDP y TCP. De estos dos últimos muestra el puerto de origen y destino.

3.3.3. Subsistema de bloqueo de tráfico

El subsistema de bloqueo de tráfico faculta la funcionalidad de poder bloquear y desbloquear una determinada dirección IPv4 o IPv6 durante un tiempo determinado. También hace posible la restricción de protocolos, en concreto IP, ICMP, UDP y TCP y en estos dos últimos se podrá bloquear un determinado puerto durante un periodo de tiempo. Para facilitar las tareas este subsistema cuenta con un buscador de direcciones, puertos y protocolos bloqueados y la posibilidad de exportarlos al ordenador para tener una copia.

3.3.4. Subsistema de control del tráfico

Con el fin de controlar el tráfico y mitigar o limitar el alcance de los ataques hacia la maquina o servidor este subsistema puede restringir el tráfico hacia el servidor. Para ello, se podrán limitar los paquetes por segundo o los Mbits/s. Si se incumplen las restricciones durante un periodo determinado de tiempo, se bloquea automáticamente el tráfico durante otro periodo de tiempo definido por el usuario. Si se dejan de incumplir las restricciones antes de que termine el lapso de tiempo no se bloqueará.

3.3.5. Subsistema de configuración

El guardado de los límites y el bloqueo se delega en el sistema de configuración, que deberá asegurar la persistencia y actualización de los datos en tiempo real. Además, para no acumular demasiada información es posible configurar la eliminación del registro de estadísticas para que se haga cada lapso de tiempo.

3.3.6. Subsistema de comunicación

El subsistema de comunicación permite aumentar la funcionalidad del cortafuegos mediante el uso de una API REST de forma muy sencilla y aprovechando las ventajas de XDP. En concreto, la API es capaz de realizar las siguientes funcionalidades:
Por un lado, mostrar información del tráfico, esto incluye, las estadísticas principales (fuente, destino, paquetes y tamaño del tráfico), las direcciones IP bloqueadas, los puertos bloqueados y los protocolos que han sido bloqueados.
Además, permite la gestión de las trazas de tráfico en formato PCAP para recoger información en tiempo real de los paquetes. Este sistema puede habilitarse y deshabilitarse, si no se usa, con el fin de gastar menos recursos.
Por otro lado, mediante el uso de este subsistema se puede bloquear y desbloquear el tráfico, aportando la dirección IP, protocolo o puerto y el tiempo de bloqueo.

Por último, la API incluye la opción de limpiar las estadísticas. Todas las opciones están protegidas mediante el sistema de autenticación y usan los otros sistemas para obtener la funcionalidad.

3.4. Requisitos funcionales

En esta sección se van a dividir entre los diferentes subsistemas los requisitos funcionales que debe cumplir la aplicación.

3.4.1. Subsistema de autenticación

RF-1.– Inicio de sesión mediante contraseña: Los dueños o administradores de la máquina deben poder acceder al cortafuegos en todo momento. Para ello, se deberá introducir una contraseña maestra.

RF-2.– Creación inicial de la contraseña: Al instalar el cortafuegos por primera vez se deberá poder crear una contraseña maestra.

RF-3.– Creación de una llave para la API: El sistema deberá ser capaz de generar una clave en caso de que no exista o permitir al usuario crearla.

RF-4.– Generación de una nueva llave para la API: El usuario podrá generar una nueva llave aleatoria.

RF-5.– Cerrar la sesión: El sistema deberá permitir el cierre de sesión del usuario.

3.4.2. Subsistema de análisis de tráfico

RF-6.– Mostrar una tabla con estadísticas: La aplicación debe de proveer al usuario de una tabla con información sobre el tráfico, en concreto, la dirección IP fuente y destino, y, los paquetes totales, paquetes por segundo, KBytes totales y Mbits por segundo de cada tupla. Esta tabla estará paginada con el fin de no mostrar excesiva información.

- **RF-6.1.– Búsqueda en la tabla:** En la tabla se podrá buscar por dirección IP de destino o fuente.
- **RF-6.2.– Ordenación de la tabla:** Por defecto la tabla estará ordenada por Kbytes pero se podrá ordenar por los paquetes por segundo, Mbits/s y paquetes totales.
- **RF-6.3.– Guardar la información de la tabla :** El usuario podrá descargar la información de la tabla en formato CSV para tener una huella del tráfico de la máquina y las direcciones entrantes y salientes.
- **RF-6.4.– Actualizar la tabla :** El sistema tendrá un botón para poder actualizar la tabla.
- **RF-6.5.– Actualizar la tabla automáticamente:** El usuario podrá marcar la opción que de los datos de la tabla se actualicen automáticamente.
- **RF-6.6.– Vaciar las estadísticas :** Con el fin de proporcionar legibilidad la aplicación contará con un sistema que permitirá vaciar las estadísticas para borrar los registros antiguos.

RF-7.– Recoger trazas: El sistema deberá recoger una traza de los últimos 60 segundos de tráfico en formato PCAP.

Subsistema de información en tiempo real

RF-8.– Protocolos más utilizados de nivel de aplicación: Para cada tupla (dirección fuente y destino) este subsistema deberá mostrar un histórico en tiempo real con los protocolos a nivel de aplicación (TCP, UDP, ICMP) que predominen en esa conexión.

RF-9.– Mostrar información sobre protocolos: Para los protocolos que lo admitan se mostrará el puerto origen y destino.

3.4.3. Subsistema de bloqueo de tráfico

RF-10.– Bloquear protocolos: El sistema deberá permitir el bloqueo de los protocolos IPv4, IPv6, ICMP, TCP y UDP por un determinado tiempo.

- **RF-10.1.– Selección del protocolo:** El usuario podrá seleccionar el protocolo a bloquear.
- **RF-10.2.– Selección del tiempo:** El usuario podrá seleccionar el tiempo que desea bloquear el protocolo.
- **RF-10.3.– Mostrar protocolos bloqueados:** El sistema mostrará los protocolos bloqueados actualmente y el tiempo que queda para el desbloqueo.
- **RF-10.4.– Buscar protocolos bloqueados:** La aplicación permitirá buscar los protocolos bloqueados actualmente.
- **RF-10.5.– Descargar lista de protocolos bloqueados:** La aplicación permitirá descargar una copia en CSV de los protocolos bloqueados.
- **RF-10.6.– Desbloquear protocolos:** El usuario podrá desbloquear un protocolo.

RF-11.– Bloquear puertos: La aplicación deberá permitir bloquear puertos.

- **RF-11.1.– Comprobación del puerto:** El sistema comprobará el puerto introducido para no excederse de los límites.
- **RF-11.2.– Selección del tiempo:** El usuario podrá seleccionar el tiempo que desea bloquear el protocolo.
- **RF-11.3.– Mostrar puertos bloqueados:** El usuario podrá ver los puertos bloqueados actualmente y el tiempo que queda para el desbloqueo.
- **RF-11.4.– Buscar puertos bloqueados:** La aplicación permitirá buscar los puertos bloqueados.
- **RF-11.5.– Descargar lista de puertos bloqueados:** La aplicación permitirá descargar una copia en CSV de los puertos bloqueados.
- **RF-11.6.– Desbloquear puertos:** El usuario podrá desbloquear puertos.

RF-12.– Bloquear direcciones IP: La aplicación permitirá bloquear una dirección IP.

- **RF-12.1.– Compatibilidad con IPV6:** El sistema de bloqueo admitirá direcciones IPV6.
- **RF-12.2.– Comprobar dirección:** La aplicación comprobará que lo introducido es una dirección IPV4 o IPV6.
- **RF-12.3.– Selección del tiempo:** El usuario podrá seleccionar el tiempo que desea bloquear el protocolo.
- **RF-12.4.– Mostrar direcciones bloqueadas:** El sistema mostrará las direcciones bloqueadas actualmente y el tiempo que queda para el desbloqueo.
- **RF-12.5.– Buscar direcciones bloqueadas:** La aplicación permitirá buscar los protocolos bloqueadas actualmente.

RF-12.6.– Descargar lista de direcciones bloqueadas: La aplicación permitirá descargar una copia en CSV de las direcciones bloqueadas.

RF-12.7.– Desbloquear direcciones: El usuario podrá desbloquear una dirección.

3.4.4. Subsistema de control del tráfico

RF-13.– Limitar los paquetes por segundo: El sistema deberá permitirle al usuario bloquear los paquetes por segundo de una dirección si incumplen un determinado límite durante un tiempo definido por el usuario.

RF-14.– Limitar los MBits/s: La aplicación deberá permitirle al usuario bloquear los Mbits/s de una dirección si incumplen un determinado límite durante un tiempo definido por el usuario.

3.4.5. Subsistema de configuración

RF-15.– Actualizar y guardar datos del sistema de control: Permitirá actualizar los límites de los paquetes por segundo, Mbits/s y el tiempo de incumplimiento, así como, el tiempo de bloqueo.

RF-16.– Programar eliminación automática: El usuario tendrá la opción de definir un lapso de tiempo para que el sistema elimine los registros de las estadísticas.

RF-17.– Comprobación de los datos: La aplicación comprobará que los datos introducidos sean válidos.

RF-18.– Apagar el cortafuegos: El sistema deberá poder apagarse y liberar los recursos adecuadamente.

3.4.6. Subsistema de comunicación

RF-19.– Mostrar información del tráfico: Mediante la API REST se podrá obtener la información del tráfico de la máquina.

RF-19.1.– Obtener las estadísticas del tráfico: Se podrán obtener las estadísticas del tráfico con la dirección fuente y destino, los paquetes y el tamaño en formato JSON.

RF-19.2.– Obtener las direcciones IP bloqueadas: Se podrán obtener las direcciones IP bloqueadas y el tiempo en formato JSON.

RF-19.3.– Obtener los puertos bloqueados: Se podrán obtener los puertos bloqueados y el tiempo en formato JSON.

RF-19.4.– Obtener los protocolos bloqueados: Se podrán obtener los protocolos bloqueados y el tiempo en formato JSON.

RF-20.– Gestionar las trazas: Permite gestionar la recogida de las trazas de tráfico mediante la API.

RF-20.1.– Iniciar la recogida de trazas: La API permitirá iniciar la recogida de trazas.

RF-20.2.– Obtener una muestra: Una vez iniciada la recogida de trazas se podrá obtener los últimos 60 segundos de muestra de tráfico en tiempo real.

RF-20.3.– Parar la recogida de trazas: Permite parar la recogida de trazas mediante la API con el fin de ahorrar recursos.

RF-21.– Bloquear el tráfico: Permite bloquear el tráfico mediante la API.

RF-21.1.– Bloquear una dirección IP: Bloquear una dirección IPV4 o IPV6 durante un determinado tiempo en segundos mediante la API.

RF-21.2.– Bloquear un puerto: Bloquear un puerto durante un determinado tiempo en segundos mediante la API.

RF-21.3.– Bloquear un protocolo: Bloquear un protocolo durante un determinado tiempo en segundos mediante la API.

RF-22.– Desbloquear el tráfico: Permite desbloquear el tráfico mediante la API.

RF-22.1.– Desbloquear una dirección IP: Desbloquear una dirección IPV4 o IPV6 durante un determinado tiempo en segundos mediante la API.

RF-22.2.– Desbloquear un puerto: Desbloquear un puerto durante un determinado tiempo en segundos mediante la API.

RF-22.3.– Desbloquear un protocolo: Desbloquear un protocolo durante un determinado tiempo en segundos mediante la API.

RF-23.– Eliminar las estadísticas: Permite limpiar las estadísticas mediante la API.

3.5. Requisitos no funcionales

En esta sección se muestran los requisitos no funcionales de la aplicación.

RNF-1.– Interfaz sencilla: La interfaz debe ser sencilla e intuitiva para que la pueda utilizar un usuario sin grandes conocimientos.

RNF-2.– Interfaz con diseño adaptativo: La interfaz debe ser sencilla e intuitiva para que la pueda utilizar un usuario sin grandes conocimientos.

RNF-3.– Portabilidad de la interfaz: La interfaz estará disponible en los navegadores con HTML5 más utilizados como Google Chrome, Internet Explorer, Mozilla Firefox, Safari y Opera, con JavaScript y cookies habilitados.

RNF-4.– Rendimiento: El sistema deberá tener un buen rendimiento, principalmente en el bloqueo de paquetes, pero también deberá utilizar la menor CPU posible y tener unos buenos tiempos de respuesta. En cuanto al procesamiento de paquetes debe de ser en tiempo real para que no se vea afectado el tráfico que sale y entra en la máquina.

RNF-5.– Seguridad: La aplicación deberá identificar a un administrador de la máquina legitimo para que el cortafuegos no sea modificado por terceros. La contraseña se cifrará.

3.6. Pantallas

Con el fin de conocer la estructura de la aplicación y cómo se navega entre las diferentes pantallas en este apartado se van a mostrar estas pantallas y los resultados de interactuar con las mismas. Para ello, el sistema deberá estar ya instalado siguiendo el manual de instalación disponible en: Manual de instalación . Este manual, además, permite configurar la contraseña del cortafuegos.

3.6.1. Pantalla de inicio de sesión

La primera página que se muestra es la de inicio de sesión. Una vez introducida la contraseña y pulsado el botón de entrar (1) se podrá acceder a la página principal del cortafuegos. En caso de que

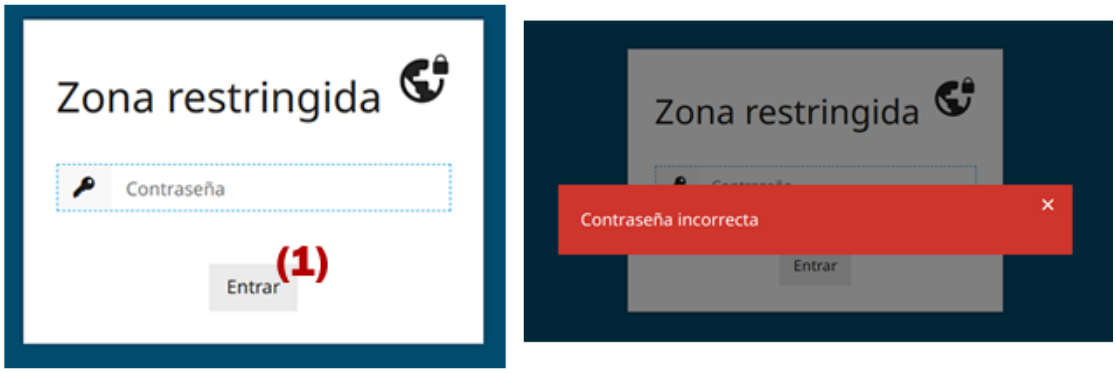


Figura 3.4: En esta figura se muestra la pantalla de inicio de sesión del cortafuegos.

la contraseña no sea la correcta se mostrará un mensaje de error al usuario en rojo y no podrá acceder al cortafuegos hasta introducir la contraseña correcta.

3.6.2. Menús de navegación

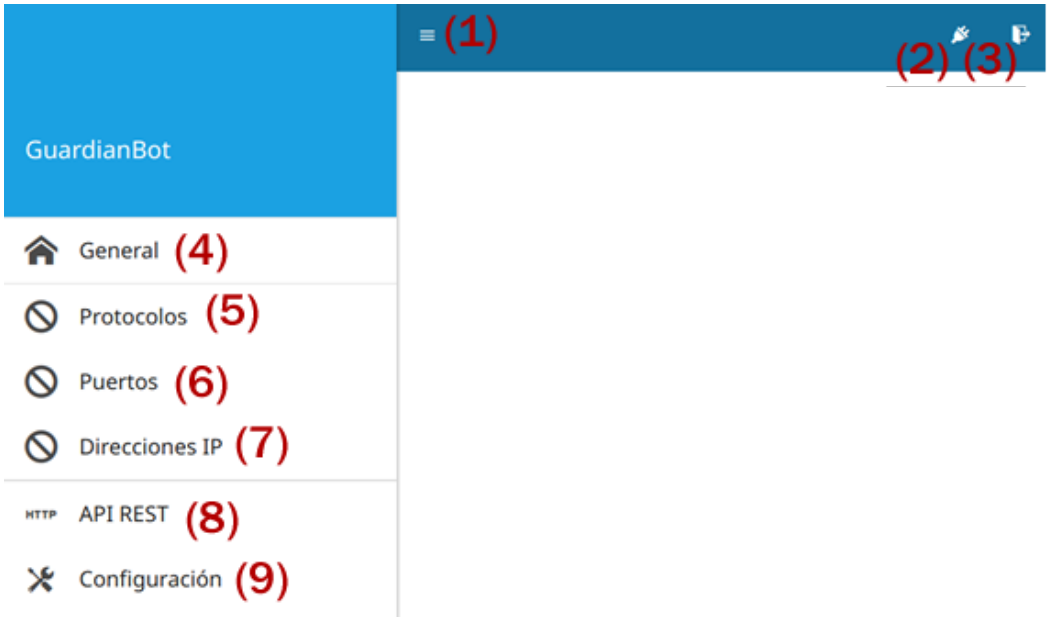


Figura 3.5: Menús de navegación de la aplicación.

Una vez dentro de la aplicación, en todas las pantallas, podemos observar dos menús de navegación. El primero permite ocultar/mostrar el menú lateral en pantallas pequeñas (1), ir a la página para apagar el cortafuegos (2) o cerrar la sesión (3).
Por otra parte, el menú lateral está preparado para ocultarse automáticamente en pantallas pequeñas. Este menú contiene las opciones principales de la aplicación: mostrar las estadísticas generales (4), acceder a la página para bloquear protocolos (5), puertos (6), direcciones IP (7), mostrar información

y administrar la llave de la API (8) y acceder a la página de configuración (9).

3.6.3. Pantalla de información del tráfico

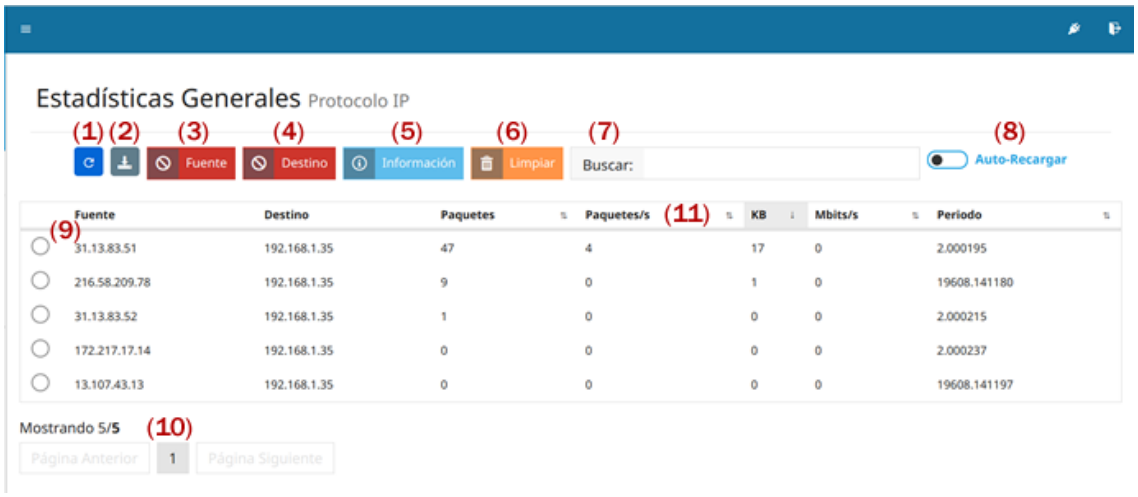


Figura 3.6: Estadísticas generales de la aplicación.

La pantalla de estadísticas generales presenta el tráfico de la capa de internet mediante una tabla. Esta tabla aparece paginada y se puede cambiar de página mediante los botones (10). En esta tabla se muestra la dirección fuente y destino y los paquetes por segundo (11), KBytes y los Mbits/s se pueden ordenar pulsando sobre ellos en la cabecera. Por defecto, se ordenan por Kbytes totales transferidos. Al seleccionar una traza (9) se pueden realizar diferentes funciones en la tabla:

- Bloquear la dirección fuente (3)
- Bloquear la dirección destino (4)
- Mostrar más información (5)

Además, hay unos botones que permiten realizar las siguientes funciones en la tabla:

- Botón para actualizar y recargar la tabla (1)
- Botón para descargar un archivo en formato CSV con los datos de la tabla (2)
- Botón para vaciar los datos antiguos de la tabla (6). Mostrará un mensaje de éxito en caso de que la operación tenga éxito.
- Cuadro de búsqueda dinámico que permite al usuario buscar por una dirección fuente o destino (7)
- Botón tipo interruptor que permite actualizar la tabla automáticamente en tiempo real (8)

Si el usuario no selecciona ningún registro de la tabla y pulsa alguna de las acciones que requieren seleccionar uno, se mostrará un mensaje de error (2) o, por el contrario, un mensaje de éxito (1).

En cuanto al botón de mostrar más información (5) se abrirá una ventana dentro de la página con la información del registro.

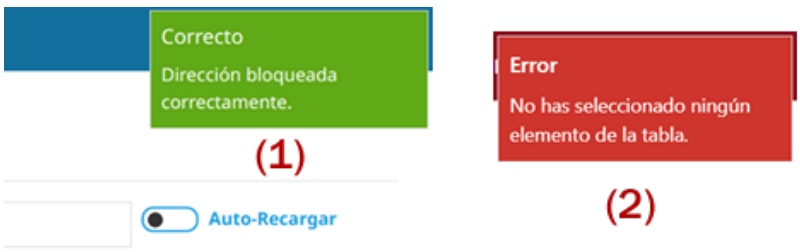


Figura 3.7: Mensaje de error en las estadísticas.

Figura 3.8: Muestra la información del registro.

Esta ventana que se abre por encima de la tabla muestra la información en tiempo real de la tabla, pero además la amplia analizando los protocolos a nivel de aplicación actualmente más utilizados y dentro de estos, si el protocolo lo permite, da el puerto de origen y destino. Esta tabla guarda el historial de los protocolos y actualmente es compatible con ICMP, TCP y UDP. Esta información se actualiza automáticamente. La ventana se puede maximizar, minimizar o cerrar mediante los botones (1). También se puede cambiar el tamaño dentro de la pestaña.

3.6.4. Pantalla para bloquear protocolos

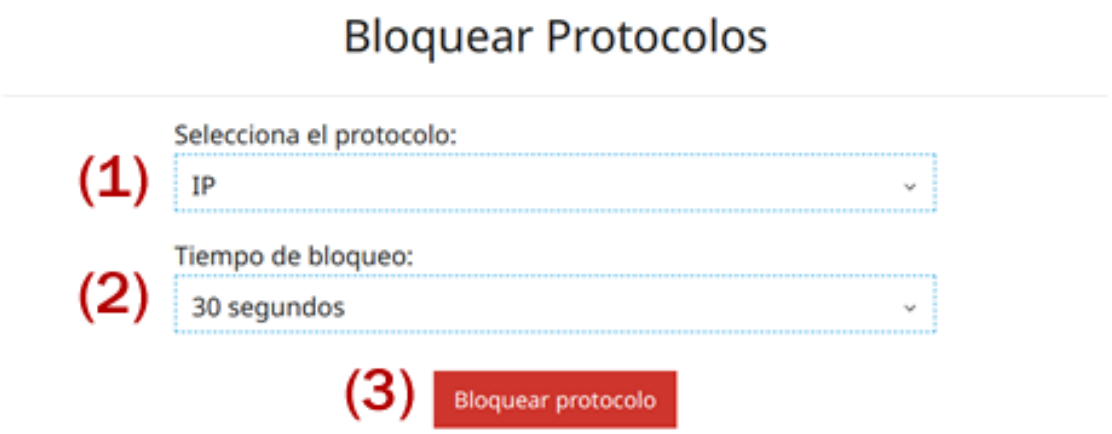


Figura 3.9: En esta figura se muestra la pantalla de bloqueo de protocolos primera parte.

Para bloquear un protocolo primero se selecciona uno de los disponibles en el desplegable (ICMP, UDP, TCP, IP) en (1) y, después, se selecciona el tiempo de bloqueo deseado. Por último, se deberá pulsar el botón de bloquear y saldrá en la lista de bloqueos activos.

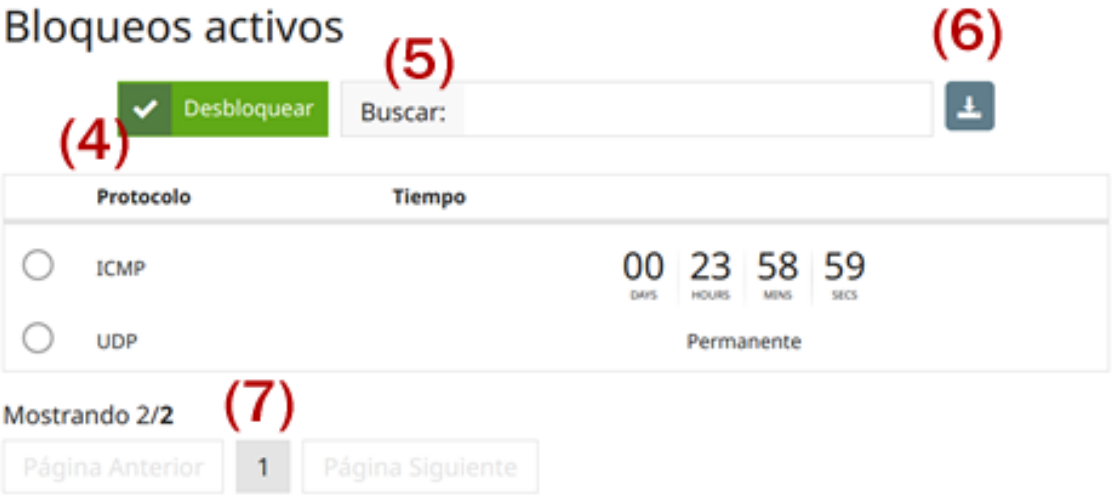


Figura 3.10: En esta figura se muestra la pantalla de bloqueo de protocolos segunda parte.

A continuación, se muestra una tabla con los bloqueos activos actualmente. La información que se muestra en esta taba es el protocolo y un contador con el tiempo que falta para que se desbloqueé. Para desbloquear un protocolo antes de que finalice el tiempo se pulsará el botón de desbloqueo (4). Si no se selecciona ningún protocolo se mostrará un mensaje de error, en caso contrario se actualizará la lista eliminando el protocolo desbloqueado. También es posible buscar protocolos (6) y descargarse una copia en CSV de los protocolos bloqueados (6). Por último, la tabla permite navegar mediante los botones (7).

3.6.5. Pantalla para bloquear puertos

Figura 3.11: En esta figura se muestra la pantalla de bloqueo de puertos.

Los puertos se pueden bloquear de forma similar a los protocolos, en este caso el sistema comprobará que el puerto es válido, es decir, que este comprendido entre el número 0 y el 65535. Una vez introducido un puerto correcto y el tiempo por el que se bloqueará, se deberá pulsar el botón (1). La página también mostrará una tabla paginada con los bloqueos activos y se podrá desbloquear seleccionando uno y usando el botón de desbloquear (2). Si no se selecciona ningún registro al pulsar este botón saltará un mensaje de error. Por último, es posible descargarse una lista en formato CSV con un registro de los bloqueos activos (3).

3.6.6. Pantalla para bloquear direcciones IP

Figura 3.12: En esta figura se muestra la pantalla de bloqueo de direcciones IP.

De una forma similar al bloqueo de puertos y protocolos funciona el bloqueo de direcciones IP. Al bloquear la dirección (2), una vez introducido el tiempo y la dirección se comprobará que sea una dirección IPV6 o IPV4 correcta, si esto no es así se mostrará en rojo como se observa en la imagen (1). Además se muestran las direcciones en una tabla paginada y se pueden desbloquear (3), buscar (4) y descargar una copia (5).

3.6.7. Pantalla de información sobre la API

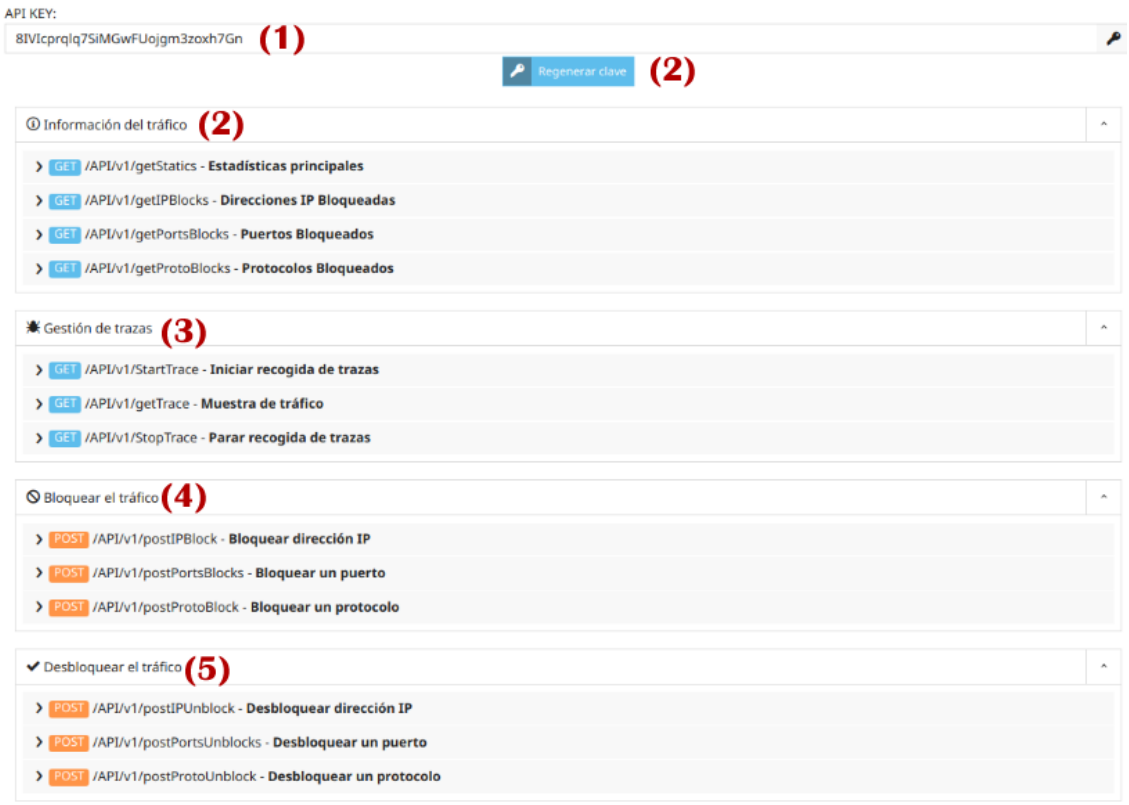


Figura 3.13: En esta figura se muestra la pantalla de información de la API.

Por un lado, esta pantalla permite mostrar (1) y modificar (2) la clave de la API. Por otro lado, se muestra información general sobre el uso de la API REST. En concreto, se divide en las siguientes opciones:

- **Información del tráfico (2):** Contiene todas las funciones de la API encargadas de proporcionar información sobre el tráfico del servidor.
  - **Estadísticas principales:** Para obtener las estadísticas del protocolo IP mediante un archivo JSON.
  - **Direcciones IP bloqueadas:** Muestra cómo obtener una lista en formato JSON con las direcciones IP bloqueadas mediante el uso de la API.
  - **Puertos bloqueados:** En este caso, muestra cómo usar la API para conseguir en formato JSON los puertos bloqueados.
  - **Direcciones IP bloqueadas:** Muestra cómo obtener en formato JSON una lista con las direcciones IP bloqueadas mediante la API.
- **Gestión de trazas (3):** Contiene cómo se utilizan las funciones de la API para gestionar la recogida de trazas en la máquina.
  - **Iniciar recogida de trazas:** Muestra cómo obtener la recogida de trazas mediante la API.
  - **Muestra de tráfico:** Muestra cómo obtener una huella de los últimos 60 segundos de tráfico en el servidor en formato PCAP.
  - **Parar recogida de trazas:** Muestra cómo parar la recogida de trazas mediante la API.
- **Bloquear tráfico (4):** Contiene toda la información necesaria para bloquear el tráfico.
  - **Bloquear dirección IP**: Muestra cómo bloquear una dirección IP mediante la API.

- **Bloquear puertos:** Muestra cómo bloquear puertos mediante la API.
- **Bloquear protocolos:** Muestra cómo bloquear protocolos mediante la API.

- **Desbloquear tráfico (5):** Contiene todas las funciones referentes al uso de la API para desbloquear tráfico.
  - **Desbloquear dirección IP:** Muestra cómo desbloquear una dirección IP mediante la API.
  - **Desbloquear puertos:** Muestra cómo desbloquear puertos mediante la API.
  - **Desbloquear protocolos:** Muestra cómo desbloquear protocolos mediante la API.
- **Control de tráfico:** Contiene la información sobre cómo limpiar la tabla de estadísticas principales de la API.

Además, esta página muestra cómo se debería realizar la solicitud a la API, usando los métodos GET (2) o POST (4). Para obtener más información basta con desplegar la función de la API deseada, por ejemplo, como se muestra a continuación (6). Cada función de la API estará descrita brevemente (7), con los parámetros que hay que introducir (8) y un ejemplo sobre cómo utilizarla (9). Por último, se describirá la respuesta de la API al realizar la solicitud (10).

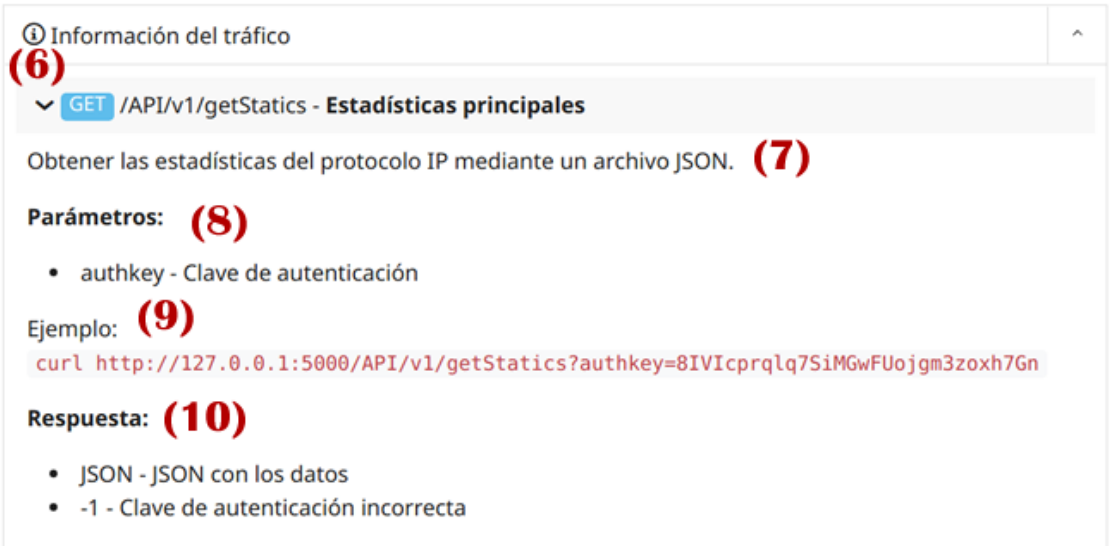


Figura 3.14: En esta figura se muestra la pantalla de información ampliada de la API.

3.6.8. Pantalla de configuración

En la pantalla de configuración el usuario puede editar las limitaciones de tráfico de la máquina y los ajustes generales del cortafuegos. Por un lado, es posible limitar los paquetes por segundo (1) y/o los Mbits por segundo (2) durante un determinado periodo de tiempo (4) si se incumple la restricción durante (3).Por otro, también es posible vaciar la tabla de registro de las estadísticas del tráfico mediante la opción (5). Todos los tiempos están en segundos y si se introduce el número 0 la opción quedará desactivada. Si se introduce información incorrecta, por ejemplo, un límite sin tiempo de bloqueo, el cortafuegos ignorará esa configuración. Para finalizar, se pulsará el botón de guardar(6) la configuración.

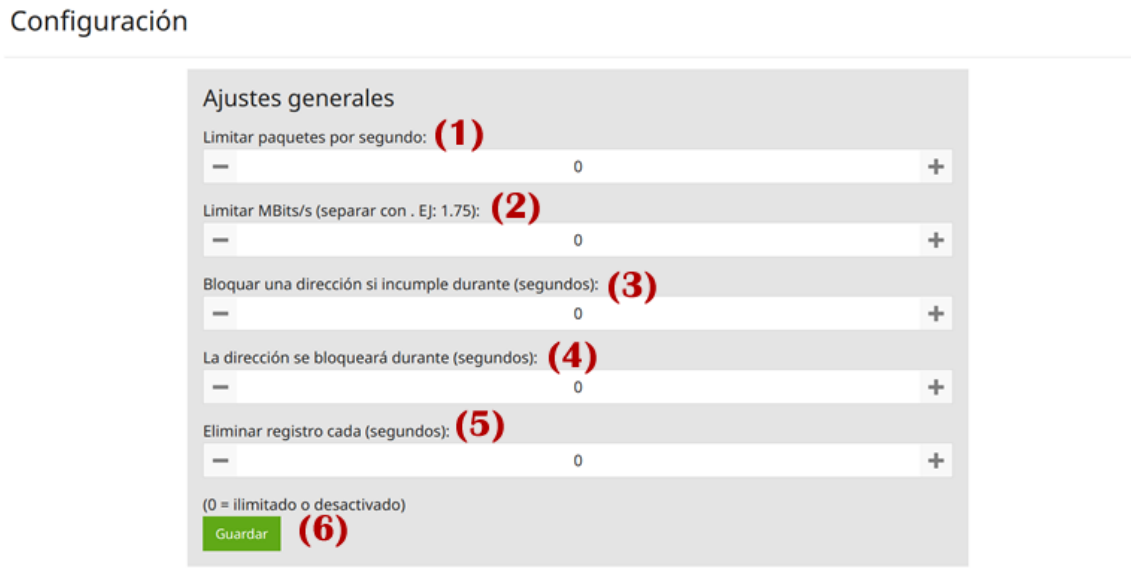


Figura 3.15: En esta figura se muestra la pantalla de configuración del cortafuegos.

3.6.9. Pantalla de apagado del cortafuegos

Figura 3.16: En esta figura se muestra la pantalla de apagado del cortafuegos.

Por último, es posible apagar el cortafuegos, liberar y apagar todos sus componentes mediante el botón de apagado (1) o volver a la página anterior mediante el botón (2). Una vez apagado el cortafuegos, será necesario inicializarlo manualmente cómo se indica en el Manual de instalación.

4

Desarrollo

En esta sección se explica en profundidad los conceptos y elementos empleados para el desarrollo del cortafuegos.

4.1. Bloqueo de paquetes

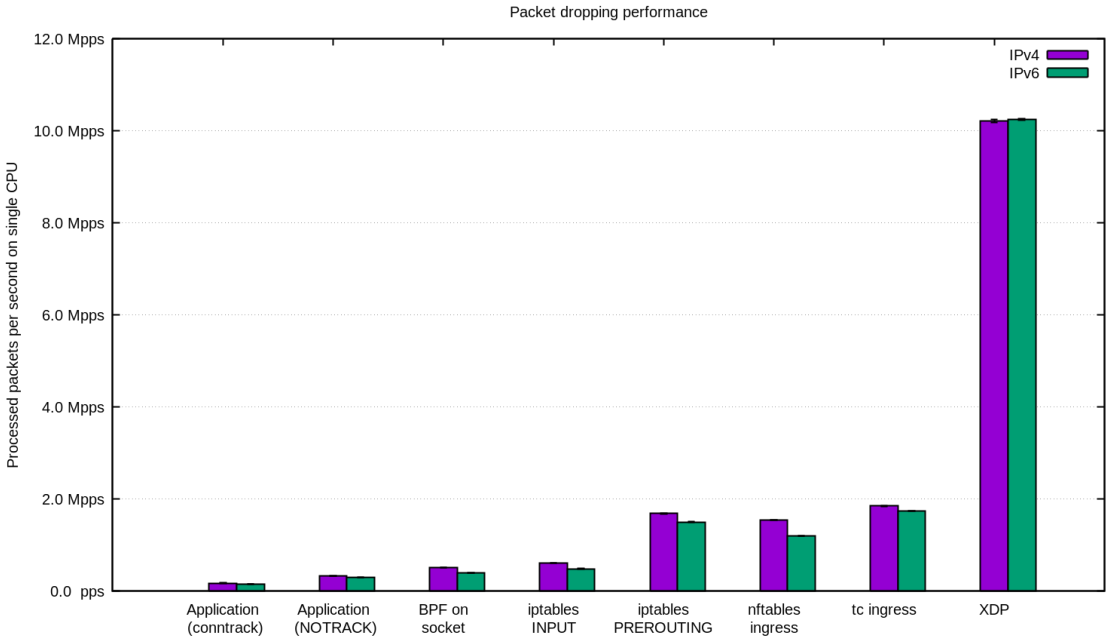


Figura 4.1: Comparación métodos de eliminación de paquetes de Cloudflare. [33].

Linux tiene gran cantidad de formas para filtrar paquetes [33], cada una con sus diferentes características de rendimiento y facilidad de uso. En general, como se puede observar en la gráfica, el rendimiento para filtrar IPV6 es menor debido a que los paquetes son ligeramente más grandes. En la gráfica se puede observar el rendimiento de las diferentes formas de filtrar paquetes en Linux:

- Descartar paquetes a nivel de aplicación: entregar paquetes a una aplicación e ignorarlos en el código del espacio de usuario.
- Limitar el número de conexiones limitando las entradas del CONNTRACK (establecimiento de conexión).

- Filtro BPF clásico que filtrará los paquetes antes de que los reciba la aplicación.
- Filtrar con IPtables antes del enrutado.
- Filtrar paquetes, antes del pre-enrutado.
- Eliminar con Nftables antes de CONNTRACK (establecimiento de conexión) es peor que IPtables filtrando paquetes aunque Nftables promete ser más rápido que IPtables. Una de las razones es la carencia de especulaciones sobre saltos indirectos [34].
- Eliminar paquetes antes del pre-enrutado mediante las entradas tc (control de tráfico).
- Usar eXpress Data Path para eliminar el paquete.

Como conclusión se puede observar en la gráfica que cuanto menor sea el procesamiento del paquete antes de descartarlo mejor es el rendimiento, pero, por lo general, más complicado es para el programador. En concreto hay una forma de bloquear paquetes que destaca: XDP.

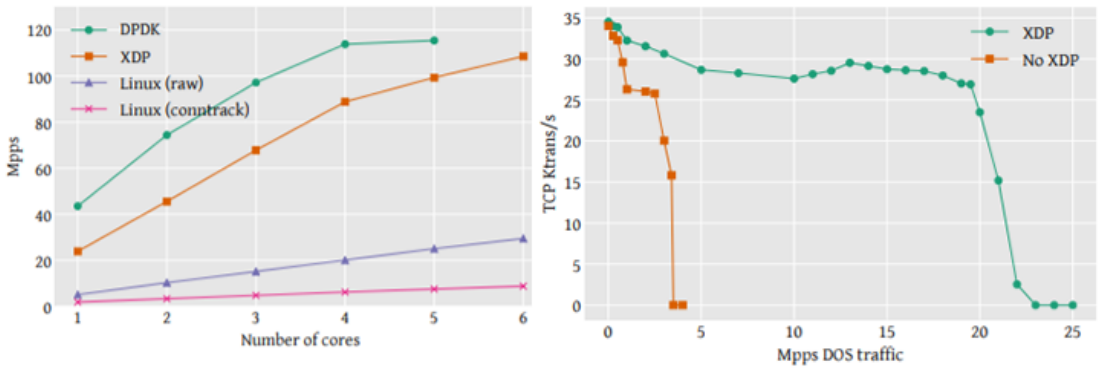


Figura 4.2: A la izquierda el rendimiento de XDP con los paquetes bloqueados y a la derecha contra un ataque DoS. [35].

Estos experimentos de XDP [35] se han realizado en un solo núcleo, para ilustrar la situación donde el tráfico legítimo tiene que competir por los mismos recursos de hardware que el tráfico del atacante. Se puede observar que en la gráfica de la izquierda sin el filtro XDP, el rendimiento cae rápidamente, reduciéndose de la mitad a 3 Mpps (millones de paquetes por segundo) y a cero en poco menos de 5 Mpps de tráfico de ataque. Sin embargo, con XDP en su lugar, el rendimiento de las conexiones TCP es estable hasta los 19,5 Mpps de tráfico de ataque, después vuelve a caer rápidamente.
En las dos gráficas se observa como el rendimiento de XDP es considerablemente mayor al uso de otros métodos. En concreto, en la gráfica de la derecha se puede observar cómo XDP es capaz de detener más conexiones TCP cuando la máquina está bajo un ataque DDOS que otros métodos.

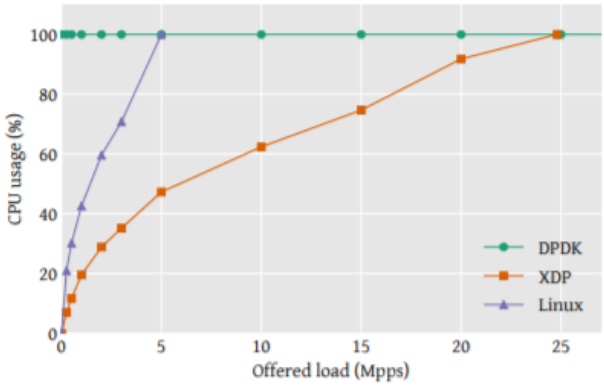


Figura 4.3: Rendimiento de XDP por uso de CPU. [35].

También es importante destacar el uso de CPU en el escenario de eliminación de paquetes. Estos datos, también tomados sobre una sola CPU, muestran como el rendimiento de XDP es mucho mayor a un paquete ya procesado por Linux.

4.2. Comunicación interna

En cuanto a la comunicación, por una parte, se ha optado por usar los mapas BPF (ver más información sobre mapas BPF en el anexo manual del programador) para la comunicación **entre el núcleo y el espacio usuario**. En concreto, se han usado mapas de tipo hash para el paso de información de las listas de bloqueo, como direcciones IP o puertos y mapas de tipo eventos del núcleo de Linux para pasar las trazas e información. Para la comunicación entre **Python y C** se ha optado por usar memoria compartida debido a su eficiencia, las escasas dependencias necesarias para su implementación, la compatibilidad y el tiempo que hay que emplear para su implementación. Al tener que comunicar una gran cantidad de datos se han descartado las tuberías.

Figura 4.4: Comunicación interna de la aplicación.

En la figura se puede observar como el programa XDP en el núcleo se comunica con la aplicación del espacio usuario mediante los dos tipos de mapas. Estos datos no necesitan ningún control de

concurrencia debido a que los datos se toman en el núcleo por cada procesador y se suman en el espacio de usuario, liberando de carga al núcleo. Las reglas se escriben en el espacio de usuario y se leen en el núcleo sin necesidad de controlar la concurrencia y, por tanto, ahorrando recursos.
Por otra parte, la aplicación C y Python se comunican mediante dos memorias compartidas, la primera alerta a la interfaz de usuario de que hay nuevos datos y la segunda transmite esos grandes datos.

4.3. API REST

Para permitir que otros desarrolladores puedan incluir sus huellas y soluciones en el cortafuegos se han estudiado varias posibilidades. Una API (Interfaz de programación de aplicaciones) es el conjunto de protocolos y herramientas que se usarán para permitir esta funcionalidad. Entre ellas, se ha seleccionado API REST debido a que destaca por su escalabilidad, por su flexibilidad y portabilidad e independencia. Además, es sencillo de construir y adaptar y tiene un escaso consume de recursos. La principal desventaja con respecto a SOAP es que este resulta complejo a la hora de utilizar. La transferencia de estado representacional (en inglés representational state transfer, REST) usa directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos.

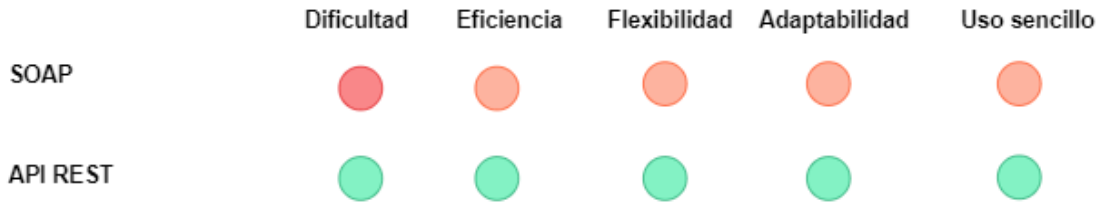


Figura 4.5: Comparación intercambio de datos externos.

4.4. Persistencia de datos

En cuanto a la persistencia de datos se ha decidido guardar la información en ficheros debido a la sencillez y de la información presente en esta. La ventaja de esto es que el usuario no necesita instalar dependencias adicionales ni configurarlas y, además, es más rápido para el desarrollador. Se han descartado las bases de datos relacionales debido a que todos los datos son independientes y simples. La coherencia de los datos no es un factor excesivamente importante debido a que a información del cortafuegos no es crítica.

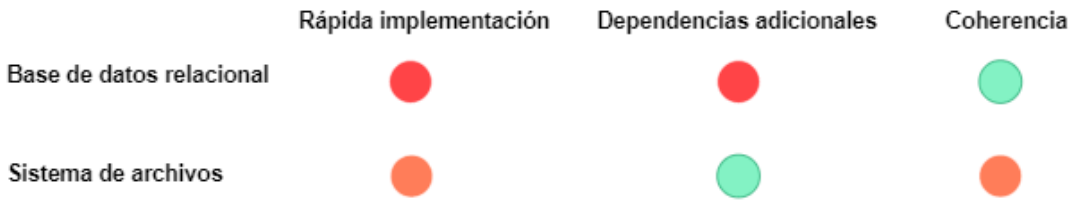


Figura 4.6: Comparación persistencia de datos.

4.5. Metodología de desarrollo

Con el objetivo de asegurar la calidad del proyecto se ha utilizado una metodología ágil. En concreto, se ha realizado un desarrollo iterativo que permite dividir la planificación en diversos bloques temporales e ir obteniendo resultados importantes y usables desde las primeras iteraciones. Esto permite que cada fase del trabajo sea una pequeña entrega parcial. Cada iteración está definida por los requisitos que la componen. En cuanto al tiempo, en los anexos se puede ver un diagrama de Gantt con toda la planificación. De manera regular, entre una iteración y otra se han comprobado y mitigado todos los problemas encontrados.

Iteración 1 - Mostrar información

Requisito funcionales				
1	2	3	4	5
6.1	6.2	6.3	6.4	6.5

Tabla 4.1: Esta tabla muestra los requisitos de la iteración 1.

En la primera iteración ha sido la que más tiempo ha llevado realizar debido a que se ha implementado la base de la aplicación, tanto de la interfaz (menú de usuario, diseño...) como los programas XDP del núcleo y espacio usuario. Además, se han implementado en esta iteración lo referente al sistema de autenticación y de análisis de tráfico que permite mostrar las primeras estadísticas en el cortafuegos.

Iteración 2 - Trazas e información avanzada

Requisito funcionales		
7	8	9

Tabla 4.2: Esta tabla muestra los requisitos de la iteración 2.

La segunda iteración se ha mejorado el sistema que muestra la información del tráfico añadiendo datos en tiempo real sobre los protocolos más usados. También se ha implementado el mecanismo para poder recoger trazas de los paquetes y obtener muestras en tiempo real.

Iteración 3 - Bloquear paquetes

Requisito funcionales				
10	10.1	10.2	10.3	10.4
10.5	10.6	11	11.1	11.2
11.3	11.4	11.5	11.6	12
12.1	12.2	12.3	12.4	12.5
12.6	12.7			

Tabla 4.3: Esta tabla muestra los requisitos de la iteración 3.

En la tercera iteración se implementan los requisitos que sirven para bloquear direcciones IP, IPV6, puertos y protocolos en el cortafuegos. Así como mostrar su información, asegurar su persistencia, controlar el tiempo de bloqueo, etc... Lo que ha llevado más tiempo en la implementación de esta iteración han sido desarrollar las pequeñas funcionalidades asociadas a mostrar los bloqueos.

Iteración 4 - Control del tráfico y configuración

Requisito funcionales				
13	14	15	16	17
18				

Tabla 4.4: Esta tabla muestra los requisitos de la iteración 4.

En esta iteración se han implementado los requisitos referentes a la configuración del cortafuegos, como por ejemplo al borrado automático de los datos cada cierto tiempo o la posibilidad de apagar el cortafuegos desde la interfaz. También se han desarrollado el control de tráfico para que sea posible poner una limitación de velocidad.

Iteración 5 - API REST

Requisito funcionales				
19	19.1	19.2	19.3	19.4
20	20.1	20.2	20.3	21
21.1	21.2	21.3	22	22.1
22.2	22.3	23		

Tabla 4.5: Esta tabla muestra los requisitos de la iteración 5.

Por último, con el fin de permitir que otros desarrolladores puedan acoplar sus soluciones en esta iteración se ha implementado todo lo referente a la API REST. Desde la interfaz en la que se muestra la información hasta todas las funcionalidades de bloqueo, control, gestión y muestras.

5

Integración, pruebas y resultados

En este capítulo se va a mostrar cómo se ha realizado la integración de la aplicación en los dispositivos de los usuarios y algunas de las pruebas que se han realizado sobre la aplicación.

Para las pruebas se ha aplicado la técnica de evaluación heurística. El objetivo de esta técnica es tener una primera toma de contacto con la aplicación, para verificar tanto la calidad como la usabilidad de esta, así como identificar posibles errores en la aplicación desarrollada y así poder solventarlos.
El procedimiento que se ha llevado a cabo ha sido el testeo por un usuario externo. Este usuario no tenía conocimiento alguno de la aplicación. El usuario ha comprobado todas las funcionalidades que ofrece la aplicación, así como la observación de la usabilidad, documentación y calidad del producto a probar. Además, se ha comprobado que la aplicación cargada en el núcleo no tenía ningún problema, como por ejemplo un uso excesivo de memoria o bucles infinitos que podrían dejar colgado al núcleo. También se ha comprobado la correcta liberación de memoria en C. Por último, se ha lanzado la aplicación en producción y se han hecho pruebas de estrés creando flujos de tráfico mediante la herramienta perf de Linux. La aplicación ha pasado todas las pruebas en Ubuntu, Linux y funciona en los principales navegadores adaptándose al tamaño de la pantalla.

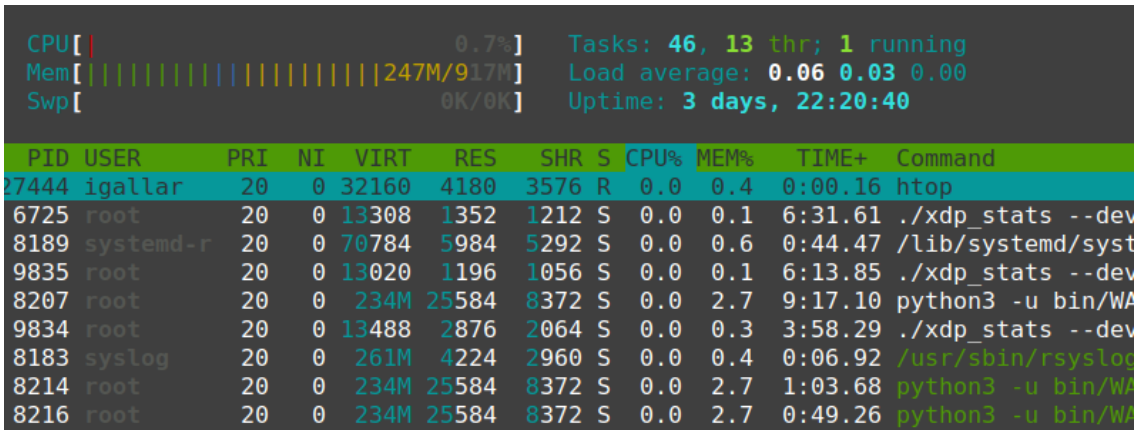


Figura 5.1: Se puede observar como la aplicación apenas usa un 6 % de la memoria del ordenador y muy poco procesador, con un tiempo en línea de 3 días.

6

Conclusiones y trabajo futuro

En este capítulo de la memoria se darán unas conclusiones sobre el aprendizaje que ha supuesto este proyecto y un planteamiento de trabajo futuro para mejorar la aplicación creada.

Conclusiones

La principal idea de este proyecto ha sido crear un cortafuegos muy sencillo de utilizar para el usuario y con la capacidad de que un desarrollador pueda incorporar sus propias huellas y configuraciones fácilmente. Esto puede ser complejo ya que hay que tener especial cuidado con el código que se inserta en el núcleo, una parte crítica del sistema, y se le añade la dificultad de comunicar la aplicación del núcleo con la del espacio usuario y con la interfaz escrita en un lenguaje de alto nivel.
La elaboración de este proyecto me ha permitido apreciar las diferentes fases de un proyecto de programación, como la investigación, el diseño y la metodología de desarrollo para construir una aplicación funcional en el menor tiempo posible. También he comprendido la importancia del uso de una metodología ágil con el fin de mejorar la calidad y productividad dividiendo el proyecto en pequeñas partes.
Esta aplicación combina el bajo nivel con el alto nivel y usa tecnologías novedosas, como XDP, que hace que los errores no sean tan fáciles de resolver debido al pequeño tamaño de la comunidad de desarrolladores y las constantes actualizaciones al tratarse de una tecnología nueva.
Por último, se ha construido una aplicación cortafuegos para Linux completamente funcional y que cumple con todos los requisitos funcionales y no funcionales. Esta aplicación es capaz de utilizar una tecnología novedosa a bajo nivel, en el núcleo de Linux y, utilizando el alto nivel proporciona una interfaz sencilla para el usuario y facilita el desarrollo de soluciones de otros programadores mediante el uso de una API REST. El diseño de la interfaz se adapta al tamaño de la pantalla.

Trabajo futuro

En cuanto al trabajo futuro se pretende traducir la aplicación a otros idiomas, como el inglés, con el fin de que pueda ser utilizada por más personas. Por otro lado, se quiere añadir al repositorio GitHub las huellas y soluciones aportadas por otros desarrolladores para que puedan ser utilizadas por toda la comunidad. También es posible mejorar la aplicación creando nuevas soluciones de detección de huellas, actualizándola y mejorando su compatibilidad con todos los sistemas operativos.

Bibliografía

[1] E. Ribas, “Qué es api rest y por qué debes de integrarla en tu negocio,” visitado: 05/11/2020. [Online]. Available: https://bit.ly/2TTgi07

[2] Minebox, “Tienda oficial de minebox,” visitado: 05/11/2020. [Online]. Available: https://tienda.minebox.es/

[3] P. Nicholson, “Five most famous ddos attacks and then some,” visitado: 05/11/2020. [Online]. Available: https://www.a10networks.com/blog/5-most-famous-ddos-attacks/

[4] Kaspersky, “Ataques ddos en el primer trimestre de 2019,” visitado: 05/11/2020. [Online]. Available: https://securelist.lat/ddos-report-q1-2019/88828/

[5] A. W. Services, “Precios amazon aws,” visitado: 05/11/2020. [Online]. Available: https://aws.amazon.com/es/pricing/

[6] G. LLC, “Historias de éxito co google adsense,” visitado: 05/11/2020. [Online]. Available: https://www.google.com/intl/es-419/adsense/start/success-stories/

[7] 40ServidoresMC, “Lista de servidores minecraft creadas por usuarios inexpertos,” visitado: 05/11/2020. [Online]. Available: https://www.40servidoresmc.es/

[8] C. González, “Cómo crear tu propia nube para almacenar datos en casa,” visitado: 05/11/2020. [Online]. Available: https://bit.ly/3evskWJ

[9] I. QNAP Systems, “Guía del usuario de mattermost,” visitado: 05/11/2020. [Online]. Available: https://www.qnap.com/es-es/how-to/tutorial/article/gu%C3%ADa-del-usuario-de-mattermost/

[10] d. Alejandro, “Montar un servidor casero de descargas torrent,” visitado: 05/11/2020. [Online]. Available: https://blog.desdelinux.net/montar-un-servidor-casero-de-descargas-torrent/

[11] H. Gustavo, “Cómo crear un servidor teamspeak 3 – guía completa,” visitado: 05/11/2020. [Online]. Available: https://www.hostinger.es/tutoriales/crear-servidor-ts3-teamspeak-3

[12] O. SAS, “Proteja sus servidores dedicados de los ataques ddos,” visitado: 05/11/2020. [Online]. Available: https://www.online.net/es/dedicated-server/ddos-arbor

[13] S. ALMAZENA INTERNET, “Almazena cloud españa,” visitado: 05/11/2020. [Online]. Available: https://www.almazena.com/

[14] I. Cloudflare, “Cloudflare protección anti-ddos,” visitado: 05/11/2020. [Online]. Available: https://www.cloudflare.com/ddos/

[15] CloudFlare, “Meet gatebot - a bot that allows us to sleep,” visitado: 05/11/2020. [Online]. Available: https://blog.cloudflare.com/meet-gatebot-a-bot-that-allows-us-to-sleep/

[16] N. V. S. F. T. team, “Xdp: 1.5 years in production. evolution and lessons learned.” visitado: 05/11/2020. [Online]. Available: https://bit.ly/32fXgpn

[17] D. Delony, “The best ddos protection hosting: Who’s the best for your site,” visitado: 05/11/2020. [Online]. Available: https://www.whoishostingthis.com/compare/ddos-protection/

[18] I. CloudVPS, “Openstack for public and private clouds | cloudvps,” visitado: 05/11/2020. [Online]. Available: https://www.cloudvps.com/

[19] Hostinger, “Tecnología web de última generación.” visitado: 05/11/2020. [Online]. Available: https://www.hostinger.es/tecnologia

[20] Arsys, “Seguridad en hosting y almacenamiento,” visitado: 05/11/2020. [Online]. Available: https://www.arsys.es/infraestructura/seguridad

[21] OVH, “Tecnología anti-ddos: Pre-firewall y firewall de red - ovh,” visitado: 05/11/2020. [Online]. Available: https://www.ovh.es/anti-ddos/tecnologia-anti-ddos.xml

[22] G. N. Purdy, *Linux iptables Pocket Reference: Firewalls, NAT y Accounting*, ser. 1. The name of the publisher, 2004, vol. 1, explicación funcionamiento IPTables.

[23] IpFire, “Ipfire - the open source firewall,” visitado: 05/11/2020. [Online]. Available: https://www.ipfire.org/

[24] Suricata, “Suricata - open source ids,” visitado: 05/11/2020. [Online]. Available: https://suricata-ids.org/

[25] H. V. Styn, “Advanced firewall configurations with ipset,” visitado: 05/11/2020. [Online]. Available: https://www.linuxjournal.com/content/advanced-firewall-configurations-ipset

[26] Beethink, “Antiddos,” visitado: 05/11/2020. [Online]. Available: http://www.beethink.com

[27] FortGuard, “Fortguard - anti-ddos,” visitado: 05/11/2020. [Online]. Available: https://fortguard.com/

[28] P. N. A. Harald Welte, “The netfilter proyect,” visitado: 05/11/2020. [Online]. Available: https://www.netfilter.org/

[29] T. Graf, “Why is the kernel community replacing iptables with bpf?” visitado: 05/11/2020. [Online]. Available: https://cilium.io/blog/2018/04/17/why-is-the-kernel-community-replacing-iptables/

[30] AspenCore, “Accelerating network packet processing in linux,” visitado: 05/11/2020. [Online]. Available: https://www.embedded.com/accelerating-network-packet-processing-in-linux/

[31] T. kernel development community, “The kernel linux documentation bpf,” visitado: 05/11/2020. [Online]. Available: https://www.kernel.org/doc/html/latest/bpf/index.html

[32] ——, “The kernel linux documentation xdp,” visitado: 05/11/2020. [Online]. Available: https://www.kernel.org/doc/html/latest/networking/af_xdp.html

[33] C. Marek Majkowski, “How to drop 10 million packets per second,” visitado: 05/11/2020. [Online]. Available: https://blog.cloudflare.com/how-to-drop-10-million-packets/

[34] RedHat, “Nftables explanation,” visitado: 05/11/2020. [Online]. Available: https://www.youtube.com/watch?v=9Zr8XqdET1c

[35] T. Høiland-Jørgensen, *The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel*, 2018, https://bit.ly/32fVDYE.

[36] T. kernel development community, “Bpf type format (btf),” visitado: 05/11/2020. [Online]. Available: https://www.kernel.org/doc/html/latest/bpf/btf.html

[37] E. Dumazet, “net: filter: Just in time compiler,” visitado: 05/11/2020. [Online]. Available: https://lwn.net/Articles/437884/

7

Glosario, acrónimos y definiciones

BPF El filtro de paquetes de Berkeley es una tecnología utilizada en ciertos sistemas operativos para los programas que necesitan, entre otras cosas, analizar el tráfico de red. Permite enviar, recibir y filtrar paquetes de la capa de enlace sin procesar.

eBPF Se trata de la versión extendida de BPF que, entre otras cosas, proporciona un intérprete de usuario para BPF con la implementación de nuevas librerías y los mapas BPF.

Clang Capa de presentación de compilador un para los lenguajes de programación C, C++, Objective-C y Objective-C++.

LLVM Es la arquitectura interna de un compilador para los lenguajes de programación C, C++, Objective-C y Objective-C++.

ELF Fichero que es producto de la compilación de un programa BPF con clang.

XDP Es una ruta de datos de alto rendimiento basada en eBPF.

DOS Un ataque de denegación del servicio (en inglés DOS) es un tipo de ataque que causa que una máquina o servicio sea inaccesible para los usuarios legítimos.

DDOS Es un ataque de denegación del servicio pero distribuido desde un gran número de ordenadores. Mucho más difícil de detener y cuyo impacto es mayor.

NIC Se trata de la tarjeta de red, también conocida como interfaz de red física.

Netfilter Es una comunidad de desarrolladores de software que han creado un marco de trabajo disponible en el núcleo de Linux que permite interceptar y manipular paquetes de red.

Iptables Es una utilidad que permite dar instrucciones al cortafuegos del núcleo de Linux.

VPS Se trata de un servidor privado virtual cuyo hardware puede ser compartido.

PCAP Es el formato de un fichero con una captura de paquetes de red.

API Permite la comunicación entre dos aplicaciones de software a través de un conjunto de reglas y protocolos.

API REST Es una API que respeta el protocolo HTTP para su implementación.

Apéndices

A

Manual de instalación

A.1. Instalación de dependencias

Las principales dependencias son libbpf, llvm, clang y libelf. El compilador LLVM + clang convierte el código en C en bytes BPF y lo almacena en un archivo ELF (libelf).

A.1.1. Paquetes necesarios

Paquetes en Fedora

Si su distribución de linux es Fedora necesitará los siguientes paquetes:

sudo dnf install clang llvm

sudo dnf install elfutils-libelf-devel libpcap-devel perf

Fedora limita la cantidad de memoria del núcleo, lo que puede interferir con los mapas BPF. Hay que ejecutar este comando para aumentar el límite:

ulimit -l 1024

Paquetes en openSUSE

sudo zypper install clang llvm libelf-devel libpcap-devel

Paquetes en Debian/Ubuntu

sudo apt install clang llvm libelf-dev libpcap-dev gcc-multilib build-essential

Instalación de Python y pip

sudo apt install python3 sudo apt install python3-pip

A.1.2. Instalación

En este apartado se describe la instalación del cortafuegos.
Cambiar de usuario a root.

sudo su

Descarga una copia local del repositorio git.

git clone https://github.com/igallar98/GuardianBot.git
cd GuardianBot

Instala los requisitos de python:

pip3 install -r requirements.txt

Compilación y creación de la contraseña maestra.

bash install.sh

Para iniciar asegúrese de que está en super usuario (sudo su).

bash start.sh

Accede a http://[IP]:4020 y usa la contraseña configurada en la instalación. La dirección IP puede ser la privada o la pública del servidor.

Para iniciar en modo manual, en caso de que quiera seleccionar la interfaz de forma manual o haya un error en el script de inicio automático:

cd xdp
sudo mount -t bpf bpf /sys/fs/bpf/
ulimit -l unlimited
sudo ./xdp_loader –auto-mode –dev [INTERFAZ] –force –progsec xdp_pass
sudo ./xdp_stats –dev [INTERFAZ] &
cd ../interface
sudo python3 run.py &
Cambiar [INTERFAZ] por la interfaz que desea usar.

B

Manual del programador

En este anexo se detalla la documentación técnica de la aplicación y se detalla estructura de la aplicación, su entorno de desarrollo y las forma de depurar y crear trazas con el fin de que futuros programadores puedan trabajar en su ampliación o modificación. Para trabajar en el proyecto es necesario tener instaladas las utilidades que se detallan en el manual de instalación y conocer cómo se utiliza la aplicación mediante el manual de usuario. Si el programador desea añadir su propio sistema de detección de huellas será necesario utilizar la API REST.

B.1. Estructura interna de la aplicación

El repositorio del proyecto se distribuye de la siguiente manera:

common Contiene las librerías con las funciones y estructuras más usadas, como por ejemplo la memoria compartida o de análisis de paquetes.

headers Copia de las cabeceras del núcleo de Linux para asegurar la compatibilidad.

interface Interfaz de usuario en Python y Flask.

/**app**/**static** CSS, JS y otros archivos estáticos de la web.

/**app**/**templates** Plantillas en HTML5 con el diseño base de la aplicación.

xdp Programa XDP que se cargará en el núcleo y el espacio usuario escrito en C.

libbpf Submódulo con la librería libbpf para aseguridad compatibilidad con la versión.

install.sh Script para instalar la aplicación y su primera configuración.

start.sh Script para iniciar la aplicación.

B.2. Desarrollo interno: Aplicación XDP

El archivo xdp_prog_kern.c contiene el código que se cargará en el **núcleo**. El compilador se encargará de convertir el código en C restringido en bytes BPF. Este programa tiene una única secuencia XDP que en función de el parámetro devuelto (XDP_PASS, XDP_DROP) se aceptará o bloqueará el paquete.

Para **analizar los paquetes** y obtener la información de los protocolos se utilizan las funciones definidas en /**common/parsing/parsing_helpers.h**.

Al programar en la aplicación que se enganchará en el núcleo es muy importante controlar el gasto de memoria porque el núcleo limita la memoria con el fin de protegerse ante posibles ataques o perder rendimiento. Por tanto, se deben evitar variables sin usar o listas y estructuras de gran tamaño.

Los **mapas eBPF** también tienen esta limitación, por tanto su tamaño es limitado. Estos mapas se usan para comunicar el núcleo con el espacio usuario. Los tipos de mapas más importantes que utiliza la aplicación son:

BPF_MAP_TYPE_HASH Mapa hash de clave/valor. Para acceder y modificar la información es necesario bloquear la sección critica mediante el uso, por ejemplo, de semaforos.

BPF_MAP_TYPE_PERCPU_HASH Mapa hash pero por CPU. Es necesario sumar en el espacio usuario todos los valores de los paquetes analizados.

Tipo ARRAY En vez de hash es posible crear mapas de tipo lista. La clave es el índice de la lista y los elementos no se pueden eliminar.

Código B.1: Muestra como se suman las estadísticas en un mapa eBPF por CPU

```
1  long values[nr_cpus];
2
3  ret = bpf_map_lookup_elem(map_fd, &next_key, values);
4
5  for (i = 0; i < nr_cpus; i++) {
6      sum += values[i];
7  }
```

Para **compilar** la aplicación se ha creado un Makefile que compilará el programa y el cargador. Para cargar el programa hay que ejecutar el comando:

sudo ./xdp_loader –help

Esto mostrará las diferentes opciones para ejecutar la aplicación y seleccionar la interfaz a la que se aplicará el filtro y análisis.

Para **depurar** el programa se puede utilizar la función bpf_trace_printk que imprime una traza en /sys/kernel/debug/tracing/trace_pipe

B.2.1. Compilación y estructura de un programa

A lo largo de este apartado se estudiará el funcionamiento interno y la construcción de un programa XDP.

Los programas eBPF se realizan generalmente en C y constan de dos archivos:

- _kern.c que contiene el código fuente que se ejecutará en el núcleo como código bytes eBPF.
- _user.c que incluye el programa de espacio de usuario y el cargador del código eBPF al núcleo.

Actualmente GCC carece de compatibilidad con eBPF, por tanto, se utiliza LLVM+clang que es capaz de generar el código en bytes eBPF en un archivo ELF objeto. Las secciones ELF de este archivo se usan para distinguir definiciones de mapas eBPF, utilizados para compartir información y código ejecutable. Cada sección contiene, generalmente, una única función y sirve para separar el programa con el fin de poder seleccionar la sección o secciones que quieren ser cargadas.

Para inspeccionar este código se puede usar: llvm-objdump -S objeto_elf_bpf.o

Código B.2: Muestra la estructura de un programa eBPF XDP con su correspondencia en código bytes eBPF.

```
1 //xdp_pass_kern.o: formato de archivo ELF64-BPF
2 //Desmontaje de la sección xdp:
3
4 SEC("xdp")
5 int xdp_pass(struct xdp_md *ctx) //xdp_prog_simple:
6 { //0: b7 00 00 00 02 00 00 00  r0 = 2
7     return XDP_PASS;//1: 95 00 00 00 00 00 00 00 exit
8 }
```

Se puede observar que XDP_PASS tiene un valor de dos, según el ENUM en el que esta definido. Estos valores por defecto que puede devolver la función, en concreto, XDP_ABORTED, XDP_DROP, XDP_PASS, XDP_TX, XDP_REDIRECT se utilizan para eliminar el paquete de la pila, aceptarlo o redirigirlo. La diferencia entre los dos primeros, ya que los dos eliminan el paquete, es que en el segundo activa un punto de rastreo que puede ser registrado y estudiado. SEC("xdp") define una sección.

B.2.2. Mapas BPF

Los mapas eBPF son mapas hash de clave/valor que permiten la comunicación entre el espacio de usuario y el kernel. Un programa que use estos mapas no puede ser cargado en el núcleo hasta que cada mapa es creado mediante una llamada a la función del sistema bpf(). Para identificar cada mapa se genera un descriptor el cual hay que reescribir en cada referencia del código ELF. La biblioteca

libbpf realiza esto de forma transparente al usuario.
El mapa no conoce la estructura de datos utilizada para el registro valor, que generalmente solo conoce el tamaño. Aunque el tipo de formato BFP permite obtener la estructura de datos mediante la información de depuración [36], es más sencillo sincronizarlos mediante una estructura de datos común.
El núcleo de Linux es capaz de ejecutar código de forma concurrente usando varios procesadores, por tanto, los datos de cada mapa eBPF se deben sumar en el espacio usuario o en el propio núcleo. Usar funciones atómicas en el núcleo es costoso ya que hay que restringir el acceso a memoria para evitar datos erróneos. Para usar una matriz de datos que almacene los valores por CPU basta utilizar BPF_MAP_TYPE_PERCPU_ARRAY trasladando así la carga al espacio usuario que deberá crear una función que sume los valores totales.
Para compartir los mapas entre varios programas se crea un archivo para cada mapa montado en un sistema de archivos especial de tipo bpf. Este sistema de archivos debe de ser montado de forma manual:

mount -t bpf bpf /sys/fs/bpf/

Código B.3: Muestra la definición de un mapa eBPF.

```
1 struct bpf_map_def SEC("maps") xdp_stats_map = {
2     .type = BPF_MAP_TYPE_ARRAY, // o BPF_MAP_TYPE_PERCPU_ARRAY
3     .key_size = sizeof(__u32),
4     .value_size = sizeof(struct datarec),
5     .max_entries = XDP_ACTION_MAX,
6 };
```

B.3. Cargador de eBPF

El cargador crea mapas a partir de su definición en ELF usando bpf(BPF_MAP_CREATE) y reescribe todas las referencias a mapas por los descriptores devueltos. Una vez que el código es convertido a bytes con el compilador pasa por un verificador que comprueba que el código es seguro y termina antes de acoplarse con el kernel.

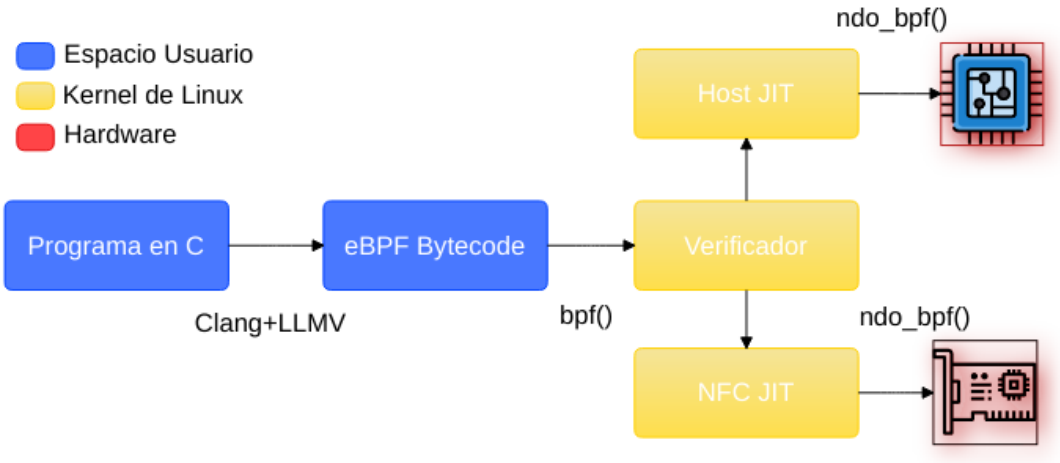


Figura B.1: En este esquema se muestra el flujo de compilación y carga de un programa eBPF.

Mediante la llamada a la función del sistema bpf(BPF_PROG_LOAD) el verificador comprueba fallos de compilación como que no haya bucles infinitos o problemas con las zonas de memoria, que podrían afectar gravemente a la seguridad o dejar colgado todo el núcleo.
El compilador en tiempo real (JIT) se introdujo en 2011 en el kernel de linux [37]. Este compilador traduce el código en bytes de eBPF a código ensamblador compatible con la plataforma elegida con el objetivo de mejorar el rendimiento. Finalmente, el código es inyectado en el núcleo directamente sobre los controladores del dispositivo. Algunos programas eBPF XDP puede ser cargados en el hardware del NIC si este lo soporta. En nuestro caso, permite llevar y administrar paquetes sin que pasen por la pila de red lo que conlleva un aumento de rendimiento.

B.4. Desarrollo de la interfaz en Python

En el lado del cliente la aplicación web se ha realizado utilizando CSS, JQUERY, JSON, JS y HTML5. Se ha utilizado la librería de código abierto Metro 4 cuya documentación se puede encontrar disponible en la web oficial. `https://metroui.org.ua/m4q-about.html`

El lado del servidor se ha implementado usando el framework Flask para Python. La persistencia de datos se realiza mediante.

B.5. Desarrollo externo: API REST

Si un desarrollador quiere implementar su propio sistema de detección y bloqueo de huellas o simplemente configurar el cortafuegos deberá hacer uso de la API REST. En esta sección se da un pequeño manual sobre cómo usarla.

GET /API/v1/getStatics - Estadísticas principales

Obtener las estadísticas del protocolo IP mediante un archivo JSON.

Parámetros:

- authkey - Clave de autenticación

Respuesta:

- JSON - JSON con los datos
- -1 - Clave de autenticación incorrecta

GET /API/v1/getIPBlocks - Direcciones IP Bloqueadas

Obtener las direcciones IP bloqueadas y el tiempo en formato JSON.

Parámetros:

- authkey - Clave de autenticación

Respuesta:

- JSON - JSON con los datos
- -1 - Clave de autenticación incorrecta

GET /API/v1/getPortsBlocks - Puertos Bloqueados

Obtener los puertos bloqueados y el tiempo en formato JSON.

Parámetros:

- authkey - Clave de autenticación

Respuesta:

- JSON - JSON con los datos
- -1 - Clave de autenticación incorrecta

GET /API/v1/getProtoBlocks - Protocolos Bloqueados

Obtener los protocolos bloqueados y el tiempo en formato JSON.

Parámetros:

- authkey - Clave de autenticación

Respuesta:

- JSON - JSON con los datos
- -1 - Clave de autenticación incorrecta

GET /API/v1/StartTrace - Iniciar recogida de trazas

Empezar a obtener trazas de tráfico.

Parámetros:

- authkey - Clave de autenticación

Respuesta:

- 0 - Correcto
- -1 - Clave de autenticación incorrecta

GET /API/v1/getTrace - Muestra de tráfico

Obtener los útimos 60 segundos de muestra de tráfico. Los datos se vacían cada 60 segundos.

Parámetros:

- authkey - Clave de autenticación

Respuesta:

- JSON - JSON con los datos
- 1 - No inicializado
- -1 - Error con la clave de autenticación incorrecta

GET /API/v1/StopTrace - Parar recogida de trazas

Detiene la recogida de trazas.

Parámetros:

- authkey - Clave de autenticación

Respuesta:

- 0 - Correcto
- -1 - Clave de autenticación incorrecta

POST /API/v1/postIPBlock - Bloquear dirección IP

Bloquear una dirección ip durante un determinado tiempo en segundos.

Parámetros:

- authkey - Clave de autenticación
- ip - Dirección IP
- time - Tiempo en segundos

Respuesta:

- 0 - Operación ejecutada correctamente
- 1 - Error
- -1 - Clave de autenticación incorrecta

POST /API/v1/postPortsBlocks - Bloquear un puerto

Bloquear un puerto durante un determinado tiempo en segundos.

Parámetros:

- authkey - Clave de autenticación
- port - Puerto
- time - Tiempo en segundos

Respuesta:

- 0 - Operación ejecutada correctamente
- 1 - Error
- -1 - Clave de autenticación incorrecta

POST /API/v1/postProtoBlock - Bloquear un protocolo

Bloquear un protocolo durante un determinado tiempo en segundos.

Parámetros:

- authkey - Clave de autenticación
- proto - Protocolo [IP, IPV6, ICMP, TPC, UDP]
- time - Tiempo en segundos

Respuesta:

- 0 - Operación ejecutada correctamente
- 1 - Error
- -1 - Clave de autenticación incorrecta

POST /API/v1/postProtoBlock - Bloquear un protocolo

Bloquear un protocolo durante un determinado tiempo en segundos.

Parámetros:

- authkey - Clave de autenticación
- proto - Protocolo [IP, IPV6, ICMP, TPC, UDP]
- time - Tiempo en segundos

Respuesta:

- 0 - Operación ejecutada correctamente
- 1 - Error
- -1 - Clave de autenticación incorrecta

POST /API/v1/postIPUnblock - Desbloquear dirección IP

Desbloquear una dirección IP.

Parámetros:

- authkey - Clave de autenticación
- ip - Dirección ip

Respuesta:

- 0 - Operación ejecutada correctamente
- 1 - Error
- -1 - Clave de autenticación incorrecta

POST /API/v1/postPortsUnblocks - Desbloquear dirección IP

Desbloquear un puerto.

Parámetros:

- authkey - Clave de autenticación
- port - Dirección puerto

Respuesta:

- 0 - Operación ejecutada correctamente

- 1 - Error
- -1 - Clave de autenticación incorrecta

POST /API/v1/postProtoUnblock - Desbloquear dirección IP

Desbloquear un protocolo.

Parámetros:

- authkey - Clave de autenticación
- protocol - Protocolo [IP, IPV6, ICMP, TPC, UDP]

Respuesta:

- 0 - Operación ejecutada correctamente
- 1 - Error
- -1 - Clave de autenticación incorrecta

POST /API/v1/makeClean - Limpiar la lista de direcciones

Limpiar y resetear la lista de direcciones IP.

Parámetros:

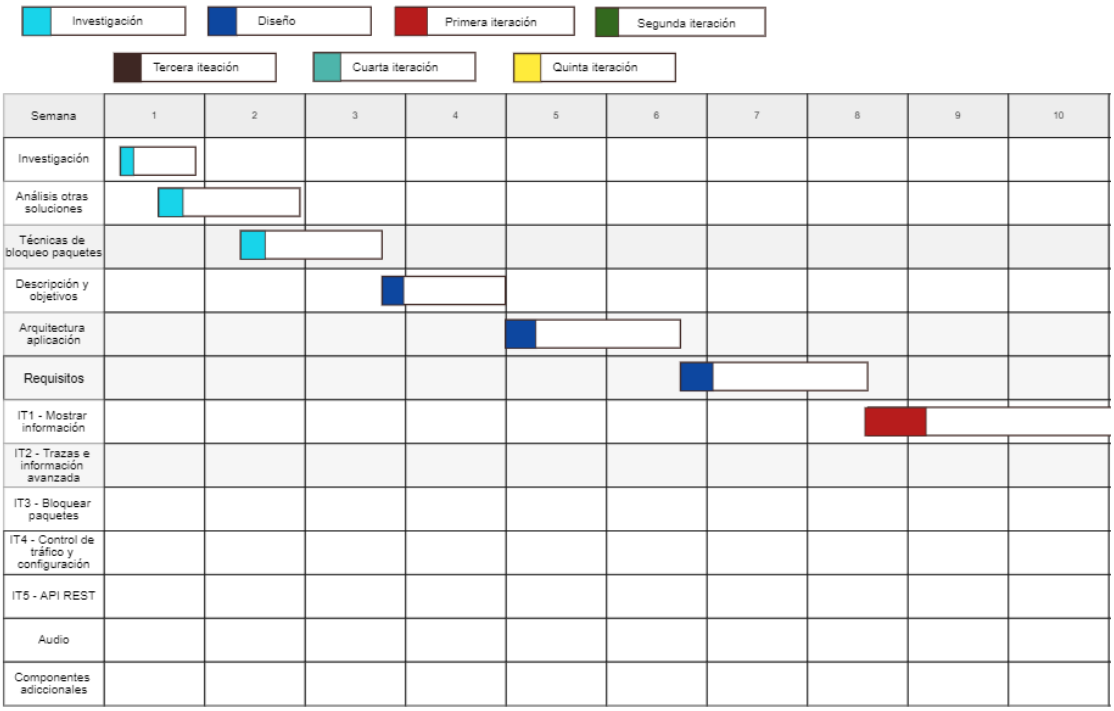
- authkey - Clave de autenticación

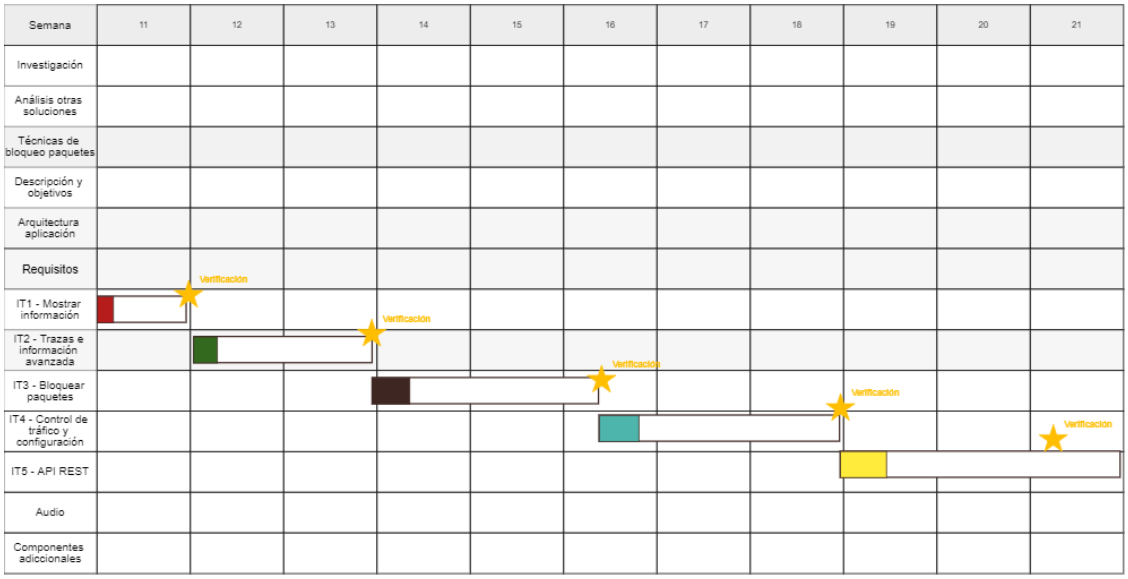
Respuesta:

- 0 - Operación ejecutada correctamente
- 1 - Error
- -1 - Clave de autenticación incorrecta

C

Diagrama de Gantt





D

GitHub, demostración y ejemplos

Se ha habilitado un repositorio de GitHub `https://github.com/igallar98/GuardianBot` y una demo en tiempo real `https://guardianbot.ga/`.
Para la construcción de la prueba en tiempo real se ha utilizado un servidor alojado Azure, la plataforma de servicios en la nube de Microsoft, así como un dominio y una red de distribución de contenidos para acelerar la carga y seguridad de la página.

Para la elaboración del ejemplo se ha utilizado la información del trabajo de fin de grado de Alejandro Romero del Campo. En concreto, se ha usado el árbol de decisión para la detección del escaneo de puertos. Estos árboles se han creado usando técnicas de aprendizaje automático mediante el programa Weka. Este ejemplo se puede encontrar en el archivo example2.py del GitHub oficial. En concreto, esto demuestra que es posible conectar un programa externo con una huella, por ejemplo, para detectar este tipo de ataques y bloquearlos. Todo ello usando la API REST del cortafuegos.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID