UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

DEPARTAMENTO DE LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

AUDITORÍA AUTOMATIZADA BASADA EN UN SISTEMA DE DETECCIÓN DE VULNERABILIDADES Y EN LA EXPLOTACIÓN CONTROLADA DE AMENAZAS SOFTWARE

Automated auditing based on a vulnerability detection system and on the exploitation controlled of software threats

Álvaro García Fernández

Supervisado por:

Cristina Alcaraz Trello

Fco. Javier López Muñoz

Resumen

El número de malware crece tan rápido como el número de vulnerabilidades que pueden ser explotadas con propósitos maliciosos. Cada vez, estos malwares son mas peligrosos y sofisticados suponiendo una amenaza real para los usuarios.

De esta problematica nace la idea del desarrollo de este trabajo, dotar al usuario de una herramienta capaz de detectar, analizar y buscar de forma automática, vulnerabilidades en el sistema con el fin de avisar ante cualquier posible fallo, para que, de una forma fácil pueda defenderse, realizando comprobaciones sobre el estado de su hardware, los servicios abiertos o escaneando la red y modificando el firewall, sin tener conocimientos previos sobre ciberseguridad.

Palabras clave

Ciberseguridad, Auditoría, Automatización, Malware, Firewall, Python, Android, UPnP, aprendizaje profundo.

Abstract

The number of malware threats, as well as the number of vulnerabilities which can be exploited with malicious purposes in consumer software are increasing in the recent years. It is important that users know the level of exposure of their devices in order to allow them to act against those risks, thus neutralizing or reducing their effects as much as possible.

The goal of this study is to provide users with a tool which can inform them of the possible risks their devices could be exposed to, automating the exploit of well known vulnerabilities in a controlled way as well as detecting them in real time. This way users would be allowed to kill with processes which could be originating the risk. Moreover, this tool could sniff network packets in order to detect anomalies in the network traffic.

Keywords

Cibersecurity, Audit, Automatization, Malware, Firewall, Python, Android, UPnP, Machine Learning.

Agradecimientos

“Para aprender a volar es necesario aprender primero a estar de pie y caminar y correr y escalar y bailar; uno no puede volar hacia el vuelo.”

–Friedrich Nietzsche

Agradecer a Cristina Alcaraz Tello su tutorización y sin cuya implicación, este proyecto no hubiese sido posible, y sobre todo gracias a sus ideas y a su modo de hacer las cosas.

Agradezco también a mis amigos Sergio Nunes Díaz, Elias Ibn Ziaten Cerezo, Carlos Gamero Muñoz, Alberto Reyes Martín y Daniel Pozo Escalona por ser una fuente inagotable de inspiración y hacerme ver con otras perspectivas los problemas.

Por último agradezco a toda mi familia, y a Julia Aguilar Muñoz, por su indiscutible apoyo.

Índice:

Índice de figuras

1. Introducción

Para comprender el objetivo del presente trabajo, es necesario observar el escenario actual de la sociedad, la cual, cada vez más, está orientada hacia el mundo tecnológico. Este mundo tecnológico cojea de un pilar fundamental, la seguridad.

La seguridad informática es un campo de estudio que practica defender los sistemas electrónicos, redes y los datos de ataques maliciosos. Existen numerosos tipos de ataques *(figura 1)*. Entre estos ataques el mas usado y de los que mas problemas ha causado está el **"ransomware"**, ataque que impide a los usuarios acceder a su sistema o a sus archivos personales y que exige el pago de un rescate. En mayo de 2016, se descubrió el primer ataque de este tipo, **"PETYA"** *[18]*, que sobrescribía el ***resgitro de arranque maestro*** **(MBR)** para que los equipos infectados no pudiesen arrancar el sistema operativo, posterior a este ataque y mas reciente encontramos el **"wannacry"** que infectó telefónica y ocasionó un ataque masivo de mas de 300.000 máquinas. Una de las formas más fácil de infectar a los usuarios es haciendo uso de la explotación de vulnerabilidades.

Una de las formas más fácil de infectar a los usuarios es haciendo uso de la explotación de vulnerabilidades que, son en esencia, un fallo no subsanado que puede permitir el acceso de malware en los diferentes dispositivos de un usuario, y de hecho se cree que el ransomware anteriormente nombrado *(wannacry)*, utilizó una vulnerabilidad llamada **"EternalBlue"** *[24]*, para conseguir el acceso al sistema.

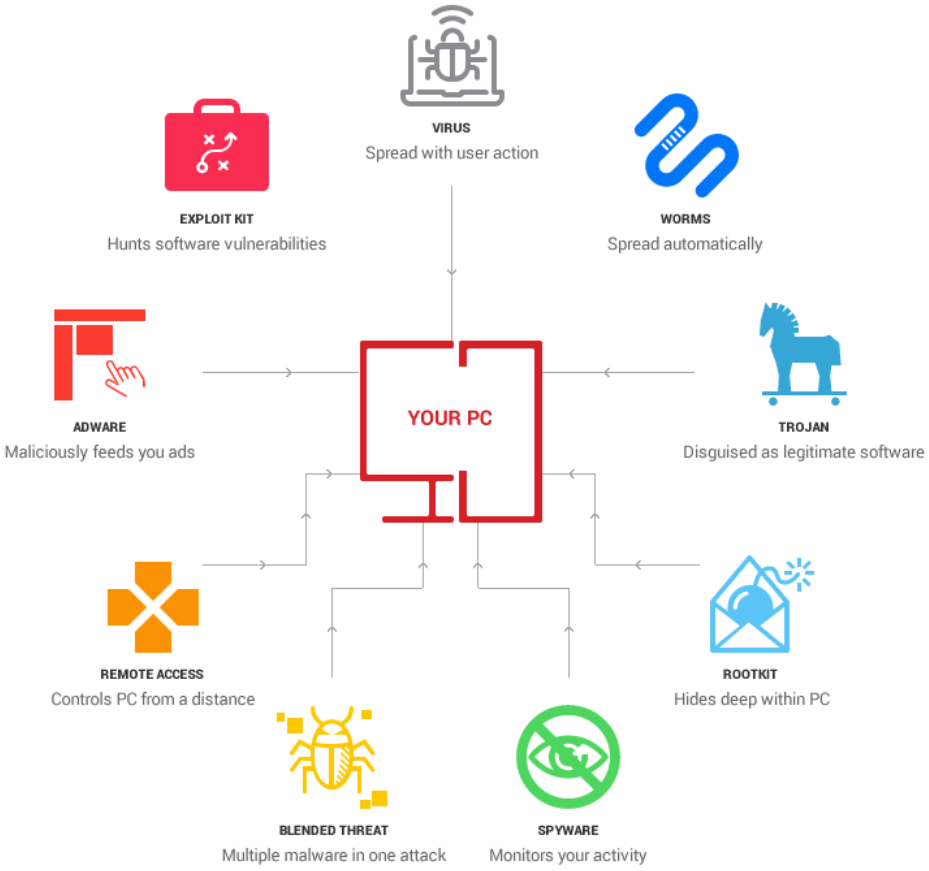

Figura 1: Tipos de malware - *[8]*

Para evitar el problema nombrado anteriormente, existen herramientas tanto a nivel de usuario, como a nivel de expertos en la materia, no obstante, y dado que el objetivo del trabajo es un público no especializado nos centraremos en el primer caso.

Los programas antivirus sirven a los usuarios de una herramienta capaz de neutralizar software potencialmente malicioso, que pueda incurrir en fallos graves del sistema. Estos programas brindan una solución sencilla al público no especializado. Sin embargo, no muestran al usuario la realidad total, es decir no indica las vulnerabilidades del dispositivo o el estado de la red del mismo, que para el usuario podría ser interesante.

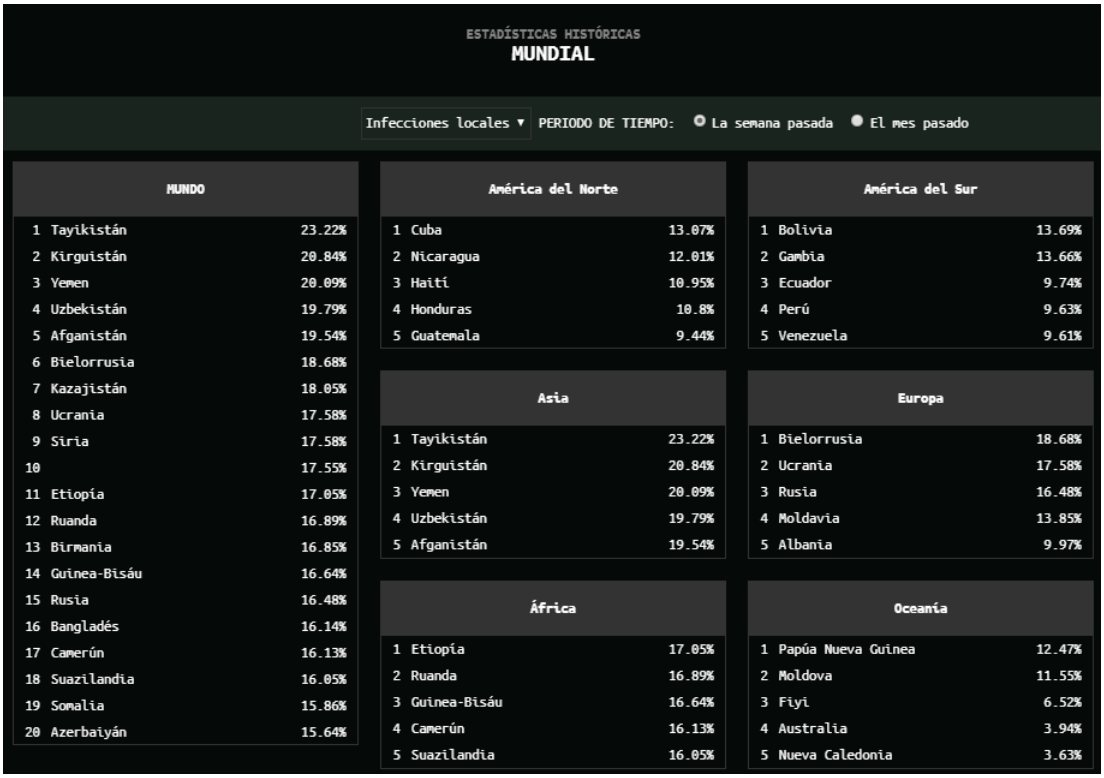

Figura 2: Infecciones por países - *[10]*

Concluyendo, existe una brecha digital entre las personas que trabajan en el sector y las que, hacen un uso normal de las tecnologías sin conocer su funcionamiento, y desconociendo todos los peligros a los que se expone, pues diariamente se producen multitud de ataques *(figura 2)*.

La intención y el objetivo del trabajo es dotar a estos usuarios de una herramienta sencilla que les permita conocer con el máximo grado de detalle posible las posibles vulnerabilidades del dispositivo, así como realizar una serie de operaciones para que puedan protegerse contra posibles ataques, de una forma sencilla.

2. Estado del arte

Durante la época de los primeros computadores analógicos, **Grace Murray Hopper** descubrió el que sería el primer ***bug*** de la historia. En aquel caso la culpa del error de la máquina con la que trabajaba Grace Murray fue solo un pequeño bicho que se había colado en las entrañas del sistema, sin embargo, actualmente denominamos ***bug*** a cualquier error o defecto en el *software* o *hardware* que hace que un programa o componente no funcione como debiese. Estos errores pueden ser la causa de la existencia de lo que se entiende en el mundo de la seguridad informática como vulnerabilidad, que pueden ser aprovechadas por los denominados ***exploits***, fragmentos de código o acciones, con el objetivo de provocar un comportamiento no deseado del sistema.

Figura 3: Ciclo de vida del software - *[2]*

Afortunadamente, el desarrollo *"software"* ha variado, y ahora, el ciclo de vida del mismo *(figura 3)* posee varias fases entre las que podemos destacar dos de ellas: **pruebas** y **mantenimiento**.

Durante la **fase de pruebas**, se ejecuta el software y se le realizan pruebas de diversa índole con el objetivo de detectar los posibles errores del software antes del lanzamiento del mismo, sin embargo, muchos de los fallos son descubiertos posteriormente cuando el desarrollo del software se ha concluido por lo que será durante la **fase de mantenimiento** donde se deberán corregir las vulnerabilidades que se encuentren tras el lanzamiento del mismo. Estas vulnerabilidades pueden ser descubiertas por los propios desarrolladores o por personas ajenas, que alertan a los encargados del mismo para que puedan corregirla

Figura 4: Ciclo de vida de las vulnerabilidades

Las vulnerabilidades tienen un ciclo de vida al igual que el software *(figura 4)*. Si descubrimos una vulnerabilidad debemos alertar a la empresa responsable para que esta lance un parche de seguridad en el menor tiempo posible. Posteriormente, esta vulnerabilidad sera informada a través del ***Computer Emergency Response Team*** **(CERT)**

para que posteriormente se publique un *exploit* que verifique la vulnerabilidad. Sin embargo, este ciclo no siempre se cumple y los problemas surgen cuando el lanzamiento del parche es posterior a su publicación en los CERT, dejando a los usuarios del *software*, vulnerables ante posibles ataques que exploten dicha vulnerabilidad.

Las vulnerabilidades son clasificadas en función de su peligrosidad, para esto existe el ***Common Vulnerability Score System*** **(CVSS)** que mide con una puntuación el impacto de la vulnerabilidad, de esta forma esclarecemos 3 tipos de peligrosidad:

- baja: 0-3.9
- media: 4-6.9
- alta: 7-10

Las vulnerabilidades más peligrosas son las llamadas ***zero day*** las cuales han sido o no reportadas pero no parcheadas, exponiendo así a los usuarios.

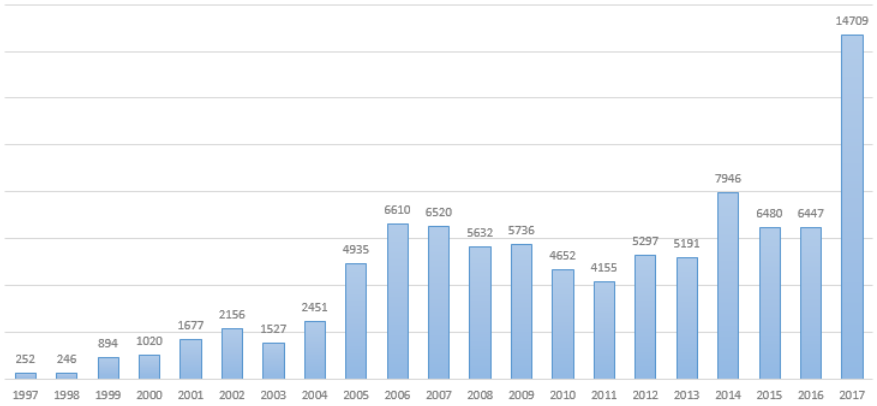

Figura 5: Vulnerabilidades reportadas a lo largo de los años - *[26]*

En la actualidad, existen numerosos programas que avisan a los usuarios sobre las amenazas existentes en sus dispositivos. Estos programas escanean el sistema de ficheros, entre otras acciones, en busca de patrones comunes que siguen los programas maliciosos identificándolos para su posterior neutralización. Sin embargo, estos programas antivirus se quedan en el análisis y no muestra al usuario los fallos o vulnerabilidades reportadas, que pueda padecer el dispositivo del usuario. Para esto existen otro tipo de software, que se encarga de escanear concretamente vulnerabilidades como pueden ser:

- **Nessus:** herramienta que detecta numerosos fallos de seguridad en base a plugins o módulos externos de pruebas que se van actualizando periódicamente.
- **Nmap:** escáner para auditorías de seguridad en red. Permite escanear servicios TCP, UDP, ICMP, RPC, etc. así como el S.O de la máquina remota.
- **OpenVas:** escáner de vulnerabilidades, muy similar a Nessus pero de carácter libre.

No obstante estas herramientas requieren de conocimientos avanzados en el campo de la informática para su correcto uso, de forma que, para el usuario no especializado, sería imposible usar.

Es por esto, por lo que nace la idea de una aplicación multifunción que permita al usuario gestionar los procesos de su sistema operativo, así como conocer en todo momento los servicios abiertos o monitorizar el estado de su sistema, además de controlar las vulnerabilidades a las que se encuentra expuesto o los ataque a los que es vulnerables, contando con una interfaz gráfica amigable para que cualquier usuario pueda acceder a conocer el nivel de seguridad de su dispositivo en todo momento.

3. Descripción del proyecto

3.1. Retrato y metodología

El proyecto constará de una aplicación que sirva al usuario para monitorizar sus dispositivos, con el objetivo de conocer en todo momento el estado del mismo, así como las vulnerabilidades o ataques a los que esta comprometido mediante auditorías y herramientas de análisis de comportamiento del sistema, con el fin de poder ejercer acciones contra estos minimizando o neutralizando su efecto y haciendo uso de interfaces intuitivas y agradables que minimicen las barreras intelectuales necesarias de forma que cualquier usuario pueda hacer uso de la aplicación.

Se hará uso de una metodología **SCRUM** con el objetivo de realizar un desarrollo iterativo para identificar posibles errores de concepto durante la realización del proyecto, de forma que los cambios no introduzcan una complejidad alta en el mismo.

3.2. Objetivos

La aplicación contará con las siguientes funcionalidades:

- **Información del estado "hardware" del sistema:** el usuario podrá saber en todo momento el estado de los componentes de su sistema, y en base a las características funcionales de la CPU, memoria virtual, discos de almacenamiento, batería (si posee), sistema operativo y versión y usuarios del sistema.
- **Comprobación de puertos:** se indicará al usuario los puertos abiertos por aplicaciones de forma que este pueda realizar acciones contra aplicaciones que no considere de confianza tales como matar el proceso.

- **Información sobre los procesos del sistema:** se indicará al usuario los procesos que se estan ejecutando actualmente en el sistema así como su *process ID* (PID).

- **Monitorización del tráfico de red:** el usuario podrá ejecutar análisis de la red del sistema con el objetivo de identificar paquetes, que mediante técnicas de **"machine learning"** (en español, aprendizaje automático) serán procesados para indicar si estos son anómalos o no con respecto al tamaño, puerto y procedencia del paquete.

- **Comprobación de sistemas de intranet:** el usuario será capaz de ver los dispositivos de su red que estén conectados a través de ***Universal Plug and Play* (UPnP)** al punto de acceso, ejecutando si es posible las acciones disponibles para cada dispositivo pudiendo descubrir la bondad de estas, dando como resultado una ejecución correcta de la acción o un fallo en caso de que la acción no pueda llevarse a cabo. Esta función es extendida por la auditoría para comprobar si los dispositivos son o no seguros con el objetivo de indicárselo al usuario.

- **Sistema de manejo remoto de firewall:** el usuario podrá realizar acciones sobre el hardware del sistema tales como encenderlo, apagarlo, crear copias de seguridad, restaurar versiones y crear o eliminar reglas.

- **Auditoría:** realizara un escáner en busca de "malware" en el dispositivo tanto a nivel de ficheros como a nivel de los procesos que este el sistema ejecutando el susodicho momento.

3.3. Casos de uso

En esta sección se localizan los casos de usos, en ellos, se pueden identificar dos actores principales:

- **Usuario**
- **Sistema**

El usuario interactúa con el sistema mediante la ejecución de los casos de uso que se definen en el diagrama presente en la figura 6.

El *National Institute of Standards and Technology* (NIST) propone diversos estándares para la especificación de casos de uso dependiendo del campo de actuación. Los casos de uso provistos posteriormente se basan en plantillas de estos estándares modificadas convenientemente para el tema que aquí se tratan y hacen uso de las buenas prácticas de ingeniería para la descripción de los mismos.

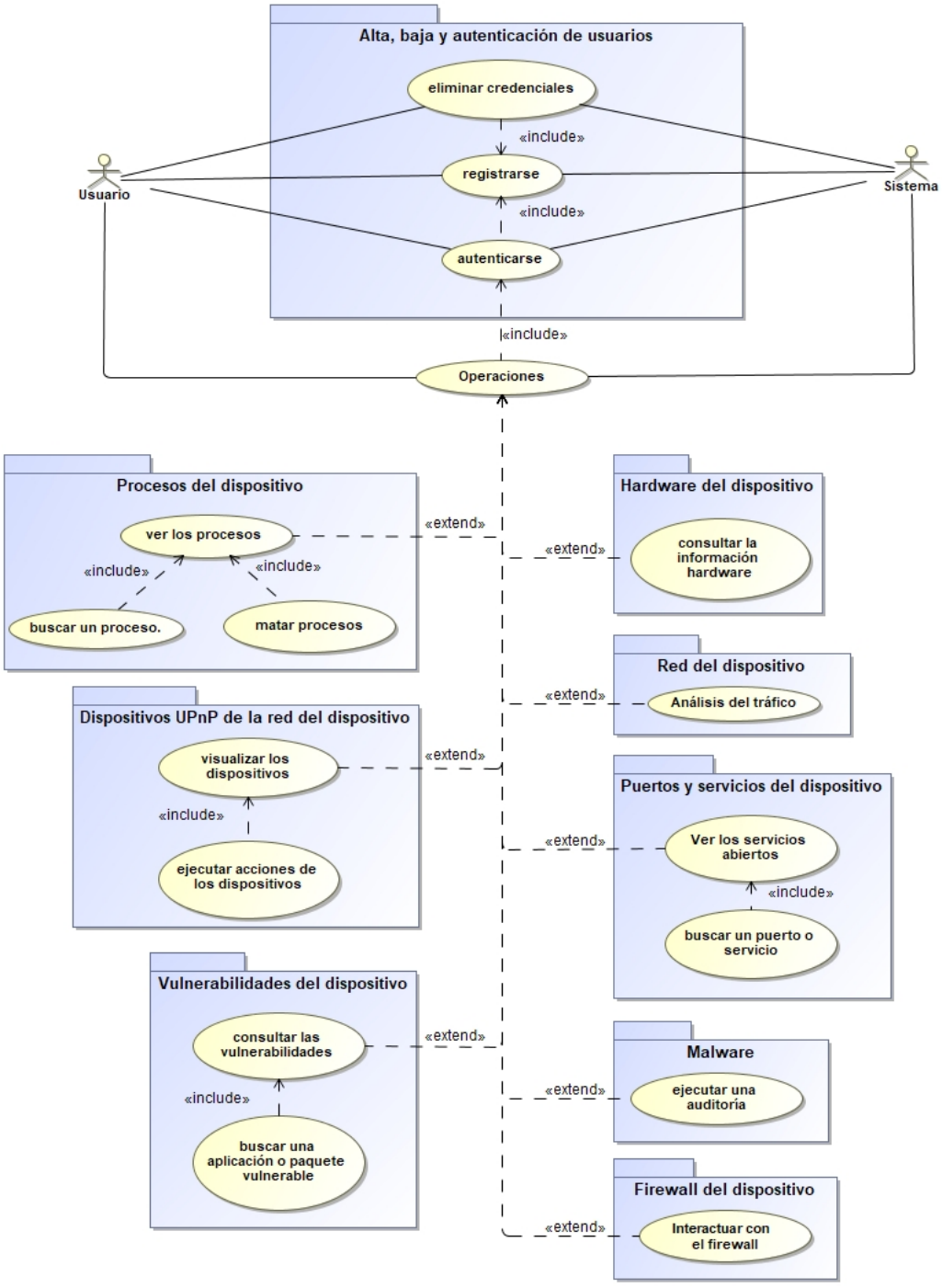

Figura 6: Diagrama de casos de uso del sistema

3.3.1. El usuario podrá registrarse en el sistema.

Descripción	El usuario debe ser capaz registrarse en el sistema para su posterior acceso.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario pulsará sobre el botón usuarios. 2. El sistema mostrará la ventana de registro de credenciales. 3. El usuario escribirá las credenciales. 4. El usuario pulsará el botón crear. 5. El sistema guardará las credenciales.	
Escenario alternativo	
5. El sistema no guardará las credenciales debido a que ya existen. 6. El sistema indicará al usuario el error.	
Postcondiciones	1. El usuario habrá registrado las credenciales en el sistema.

3.3.2. El usuario podrá eliminar credenciales del sistema.

Descripción	El usuario debe ser capaz el eliminar credenciales del sistema.
Prioridad	Media
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario pulsará sobre el botón usuarios. 2. El sistema mostrará la ventana de registro de credenciales. 3. El usuario escribirá las credenciales. 4. El usuario pulsará el botón eliminar. 5. El sistema eliminará las credenciales.	
Escenario alternativo	
5. El sistema no eliminará las credenciales debido a que no existen. 6. El sistema indicará al usuario el error.	
Postcondiciones	1. El usuario habrá eliminado las credenciales en el sistema.

3.3.3. El usuario podrá autenticarse en el sistema.

Descripción	El usuario debe ser capaz de autenticarse en el sistema.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario accederá al sistema. 2. El sistema pedirá las credenciales al usuario. 3. El usuario escribirá las credenciales. 4. El usuario pulsará el botón “conectar”. 5. El sistema verificará al usuario.	
Escenario alternativo	
5. El sistema no verificará al usuario debido a un error en las credenciales. 6. El sistema indicará al usuario el error.	
Postcondiciones	1. El usuario habrá accedido al sistema.

3.3.4. El usuario podrá consultar la información hardware de su dispositivo.

Descripción	El usuario debe ser capaz de ver en todo momento información relevante acerca de su hardware tales como sus discos duros, memoria, CPU, etc.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”hardware”. 2. El sistema mostrará el hardware del dispositivo.	
Postcondiciones	1. El usuario habrá visualizado el hardware del dispositivo.

3.3.5. El usuario podrá ver los procesos de su dispositivo.

Descripción	El usuario debe ser capaz de visualizar los procesos que se ejecutan en su dispositivo.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”Procesos”. 2. El sistema mostrará los procesos ejecutados por su dispositivo.	
Postcondiciones	1. El usuario habrá visualizado los procesos de su dispositivo.

3.3.6. El usuario podrá buscar un proceso.

Descripción	El usuario debe ser capaz de buscar un proceso en la pantalla en la que se visualizan los procesos del dispositivo.
Prioridad	Media
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”Procesos”. 2. El sistema mostrará los procesos ejecutados por su dispositivo. 3. El usuario escribirá en la barra de búsqueda el proceso a buscar 4. El sistema mostrará el proceso buscado.	
Postcondiciones	1. El usuario habrá encontrado el proceso buscado.

3.3.7. El usuario podrá matar procesos del dispositivo.

Descripción	El usuario debe ser capaz de matar un proceso ejecutado en su dispositivo si lo cree conveniente en tiempo real.
Prioridad	Media
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”Procesos”. 2. El sistema mostrará los procesos ejecutados por su dispositivo. 3. El pulsará sobre el proceso que quiera matar 4. El sistema matará el proceso ejecutado en el dispositivo.	
Escenario alternativo	
4. El sistema no matará el proceso debido a que no tiene los permisos necesarios para hacerlo. 5. El sistema indicará al usuario el error.	
Postcondiciones	1. El sistema habrá matado el proceso seleccionado.

3.3.8. El usuario podrá consultar los servicios o puertos abiertos de su sistema.

Descripción	El usuario debe ser capaz de visualizar los puertos o servicios que su dispositivo tiene abiertos en tiempo real.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”Puertos”. 2. El sistema mostrará los puertos abiertos de su dispositivo.	
Postcondiciones	1. El usuario habrá visualizado los puertos de su dispositivo.

3.3.9. El usuario podrá buscar un puerto o servicio del dispositivo.

Descripción	El usuario debe ser capaz de buscar un puerto en la pantalla en la que se visualizan los puertos abiertos del dispositivo.
Prioridad	Media
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”Puertos”. 2. El sistema mostrará los puertos abiertos de su dispositivo. 3. El usuario escribirá en la barra de búsqueda el puerto a buscar 4. El sistema mostrará el puerto buscado si esta abierto en el	
Postcondiciones	1. El usuario habrá encontrado el puerto buscado.

3.3.10. El usuario podrá consultar las vulnerabilidades a las que esta expuesto.

Descripción	El usuario debe ser capaz de comprobar las vulnerabilidades a las que se encuentra expuesto su dispositivo.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”Vulnerabilidades”. 2. El sistema mostrará las vulnerabilidades del dispositivo.	
Postcondiciones	1. El usuario habrá visualizado las vulnerabilidades de su dispositivo.

3.3.11. El usuario podrá buscar una aplicación o paquete vulnerable.

Descripción	El usuario debe ser capaz de buscar un paquete o aplicación vulnerable de su dispositivo.
Prioridad	Media
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”Vulnerabilidades”. 2. El sistema mostrará las vulnerabilidades de su dispositivo. 3. El usuario escribirá en la barra de búsqueda el paquete o aplicación a buscar 4. El sistema mostrará la aplicación o paquete buscado.	
Postcondiciones	1. El usuario habrá encontrado el paquete o aplicación buscado.

3.3.12. El usuario podrá visualizar los dispositivos de intranet inseguros.

Descripción	El usuario debe ser capaz de visualizar los dispositivos que usan el protocolo UPnp que sean inseguros y estén dentro de la red del dispositivo .
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción ”dispositivos upnp”. 2. El sistema mostrará los dispositivos UPnP de la red.	
Postcondiciones	1. El usuario habrá visualizado los dispositivos UPnP de la red.

3.3.13. El usuario podrá ejecutar acciones de los dispositivos UPnP.

Descripción	El usuario debe ser capaz de ejecutar acciones en los dispositivos UPnP encontrados.
Prioridad	Media
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar en la pantalla de los dispositivos UPnP.
Escenario principal	
1. El usuario pulsará sobre el dispositivo sobre el que quiera realizar la acción 2. El sistema mostrará las acciones disponibles. 3. El usuario pulsará la acción a ejecutar. 4. El sistema mostrara la pantalla para introducir los datos requeridos por la acción a ejecutar. 5. El usuario introducirá los datos 6. El usuario pulsara ejecutar 7. El sistema ejecutará la acción	
Escenario alternativo	
7. El sistema no fallará al ejecutar la acción debido a algún problema con el dispositivo o los datos introducidos. 8. El sistema indicará al usuario el error.	
Postcondiciones	1. El usuario habrá ejecutado la acción de un dispositivo UPnP de su red.

3.3.14. El usuario podrá ejecutar un análisis de su trafico de red.

Descripción	El usuario debe ser capaz de comprobar el estado de su red mediante la ejecución de un análisis, para ver si esta o no expuesto a algún tipo de ataque así como detectar anomalías en el tráfico.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción “Red”. 2. El sistema lanzará la ejecución de un análisis de la red. 3. El sistema mostrará los resultados del análisis	
Postcondiciones	1. El usuario habrá visualizado el estado de la red en la que esta el dispositivo.

3.3.15. El usuario podrá interactuar con el *firewall* de su sistema.

Descripción	El usuario debe ser capaz de interactuar con el firewall de su sistema.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opcion ”Firewall”. 2. El sistema mostrará el estado del firewall y las opciones disponibles.	
Postcondiciones	1. El usuario habrá visualizado el estado del firewall de su sistema y visto las opciones disponibles.

3.3.16. El usuario podrá ejecutar una auditoría del sistema

Descripción	El usuario debe ser capaz de ejecutar una auditoría que muestre los procesos y ficheros que puedan incluir algun tipo de malware en el sistema.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Precondiciones	1. El usuario debe estar autenticado en el sistema.
Escenario principal	
1. El usuario pulsará sobre la opción Auditoría. 2. El sistema lanzará un escáner completo. 3. El sistema mostrará el resultado del análisis.	
Postcondiciones	1. El usuario habrá visualizado si existe malware en su dispositivo.

3.4. Arquitectura

La aplicación a desarrollar constará de dos partes basándose en el paradigma **cliente-servidor** *(figura 7)*, siendo el servidor el dispositivo de escritorio el cual el usuario desee monitorizar y el cliente la aplicación móvil que se comunicará con el servidor.

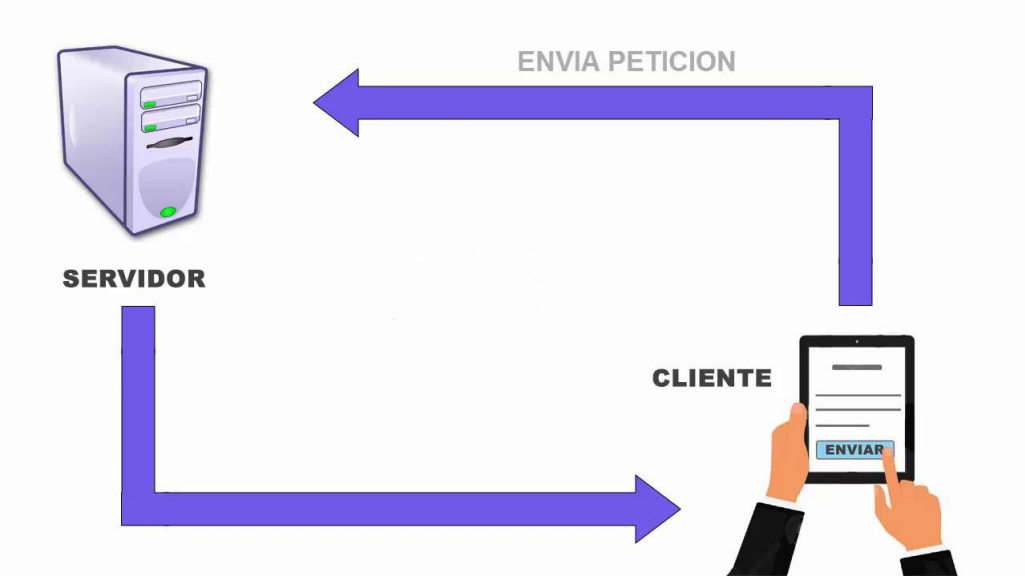

Figura 7: Paradigma cliente-servidor

- **El servidor** será la parte mas importante de la aplicación y la que tenga el grueso del trabajo. Se compondrá de una interfaz que ayudará al usuario para iniciar correctamente el servidor en su dispositivo, que una vez iniciado indicará al usuario los parámetros necesarios para su posterior conexión con el cliente, y esperará acciones a ejecutar.
- **El cliente** constituirá la parte visible de la aplicación que será la que el usuario finalmente use. Esta parte se comunicará con el servidor permitiendo la ejecución de todas las funcionalidades de una forma transparente.

3.5. Comunicación

La comunicación entre ambas partes del sistema se realizará mediante el paso de mensajes en formato ***JavaScript Object Notation*** **(JSON)**, debido a la facilidad de intercambio de datos.

El inicio de la conexión se realizará en cuatro pasos principales *(figura 8)*:

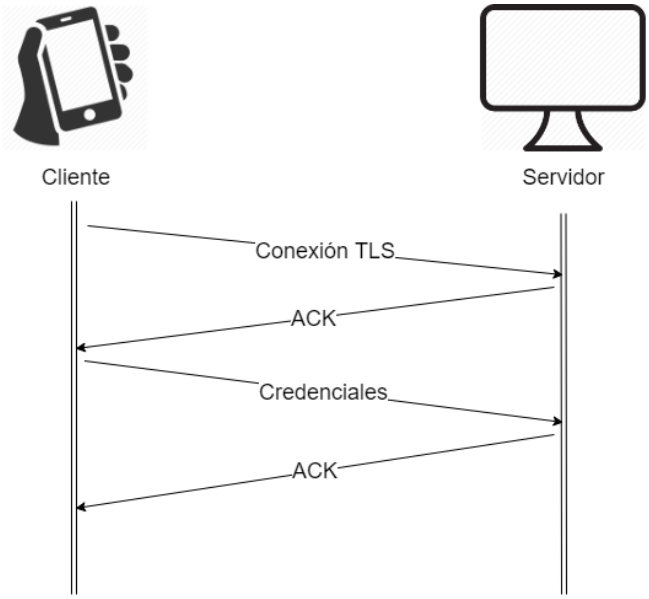

Figura 8: Conexión cliente-servidor

1. **Conexión *Transport Layer Security* (TLS):** este paso engloba todo lo necesario para la conexión segura haciendo uso de TLS y de todas sus fases *(figura 9)*.
2. **Ack:** una vez terminado el intercambio y comprobación de certificados durante la fase anterior el servidor manda una comprobación de que todo se ha realizado correctamente.

3. **Credenciales:** son enviadas a través del cliente para su posterior verificación en el servidor.

4. **Ack:** una vez verificada la identidad del cliente el servidor envia la confirmación de que las credenciales son correctas y se finaliza el comienzo de la conexión.

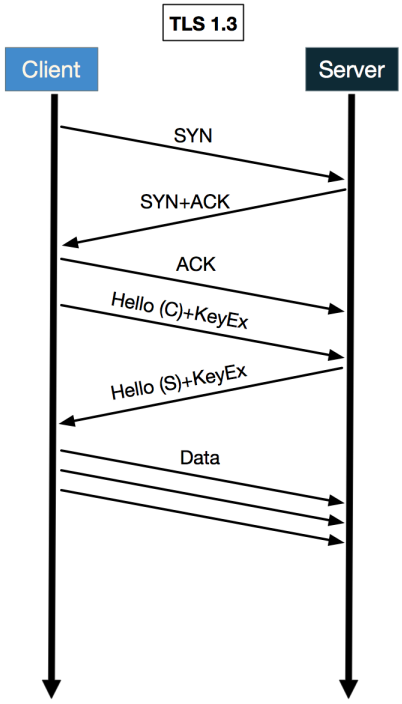

Figura 9: Conexión TLS

Un fallo producido en cualquiera de los pasos descritos anteriormente daría como resultado un cierre inmediato de la conexión para prevenir posibles conexiones inseguras.

3.6. Seguridad

La seguridad de la aplicación puede abordarse desde dos puntos diferentes:

- **Seguridad a nivel de protocolo:** ambas partes se comunican mediante el paso de mensajes en formato **JSON**. Estos mensajes pasan a través de la red bajo el protocolo TLS en la versión mas alta posible, junto con el uso de **certificados X.509** para el cifrado de los mensajes.
- **Seguridad a nivel de aplicación:** se regulará mediante el uso de credenciales que permitirán autenticar un cliente con un servidor determinado de forma que sean requeridas a la hora de realizar la comunicación y con el objetivo de que nadie que use la aplicación pueda acceder a los dispositivos de otros usuarios.

4. Servidor

4.1. Tecnologías

El servidor se encuentra desarrollado en su mayoría en el lenguaje de programación **python** haciendo uso también de otros como **shellscript**. Además se hace uso del lenguaje **C** para el uso de ciertas librerías como pcap. Por último durante su desarrollo numerosos paquetes que serán expuestos en la sección 12.1.

Para la construcción de esta aplicación se ha hecho uso de varias herramientas que, por su relevancia requieren cierta explicación, para ello, y a continuación, se listan estas herramientas con una explicación mas detallada:

- **Pcap**: esta herramienta dota al software desarrollado de la capacidad de identificar paquetes entrantes y salientes de la red del dispositivo. [`https://www.tcpdump.org/manpages/pcap-filter.7.html`]
- **Scikit-learn:** es una gran libreria python desarrollada para el aprendizaje automático y cuyo uso se integra en la aplicación para el análisis de los paquetes de red y su clasificación en anomalías.
- **Vulners:** se trata de una aplicación web que integra una API REST, a través de la cual nuestra aplicación es capaz, una vez identificado el software y version instalados en el sistema, preguntar si ese software tiene alguna vulnerabilidad publicada en este repositorio gratuito, que se nutre de otros repositorios famosos de vulnerabilidades tales como `https://packetstormsecurity.com/` o `https://www.exploit-db.com/`

- **YARA:** con esta herraienta podemos escribir en un lenguaje propio unas reglas para identificar malware y a través de las cuales nuestro software es capaz de analizar tanto procesos como ficheros. Se hace uso de un repositorio público de reglas yara `https://github.com/Yara-Rules/rules`

4.2. Clases

A continuación se definen las clases que modelan el sistema servidor mediante la figura visual que representa la clase y un texto explicativo que complementa la figura, con la finalidad de que la lectura sea lo mas fácil y limpia posible.

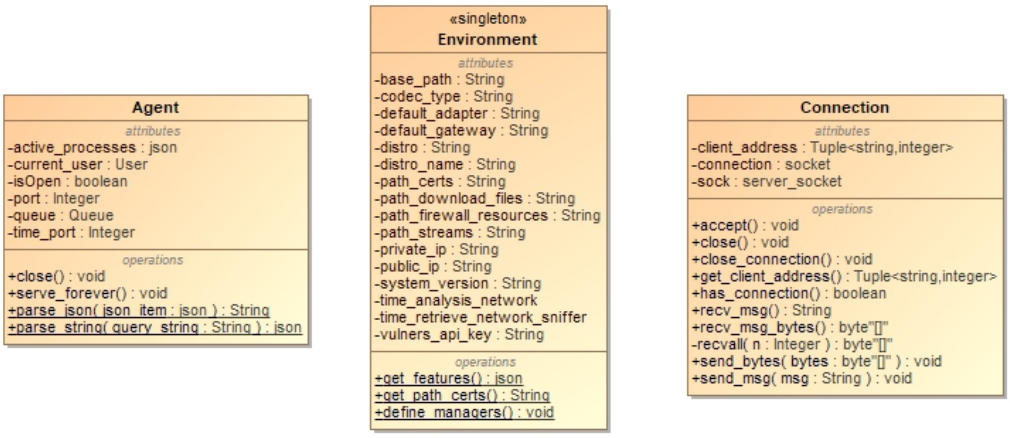

Figura 10: Clases Agent, Environment y Connection

- **Agent:** se encargará de la estructura básica del servidor y de atender las peticiones entrantes así como de la autenticación de la comunicación. *(Figura 10)*.
- **Environment:** se trata de una clase que implementa el *patrón singleton* y a través de la cual, cuando se instancia se generan los ficheros necesarios para el correcto funcionamiento del software. Además, contiene variables donde se guardan las

distintas rutas a ficheros que maneja el software así como los objetos principales del mismo. *(Figura 10)*.

- **Connection:** es la clase que se encargará de gestionar la conexión, es decir, una API a alto nivel que esta implementada con sockets TCP y a través de la cual podemos enviar y recibir mensajes de una forma mas sencilla y abstracta. *(Figura 10)*.

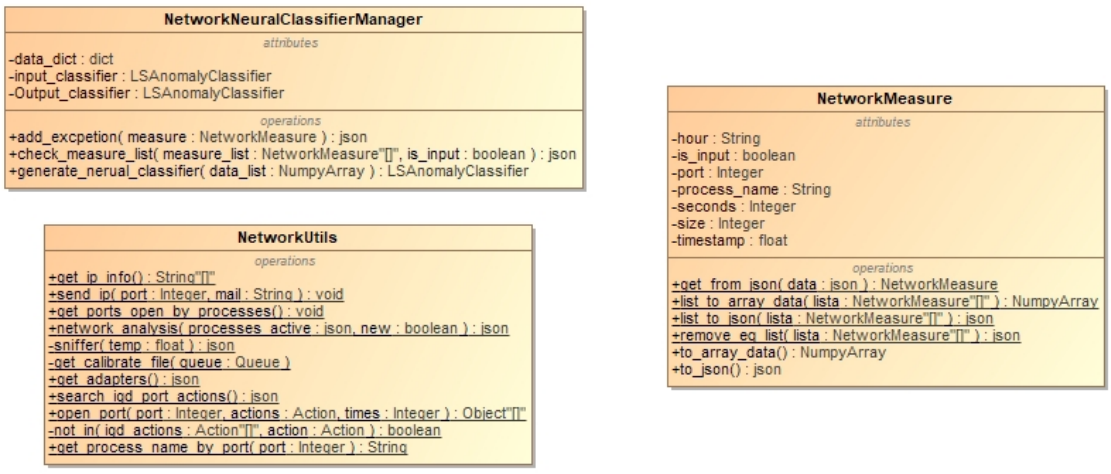

Figura 11: Clases que modelan las utilidades de red del sistema

- **NetworkNeuralClassifierManager:** es el encargado de analizar los paquetes de la red encapsulados en forma de NetworkMeasure y dictar si se tratan o no de anomalias de acuerdo a los parametros estipulados con anterioridad y haciendo uso de técnicas de machine learning. *(Figura 11)*.

- **NetworkMeasure:** esta clase encapsula un paquete de red de una forma mas abstracta con el fin de poder clasificar bien estas medidas. *(Figura 11)*.

- **NetworkUtils:** engloba un conjunto de métodos que ayuda al software en las operaciones que tienen algo que ver con la red o servicios del sistema. *(Figura 11)*.

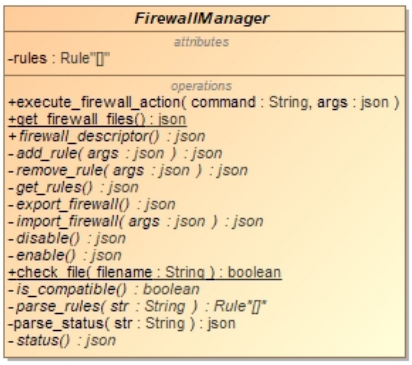

Figura 12: Firewall manager

- **FirewallManager:** esta clase es un middleware entre el software y el firewall del sistema operativo, que mas adelante será definido de forma mas específica. *(Figura 12)*.

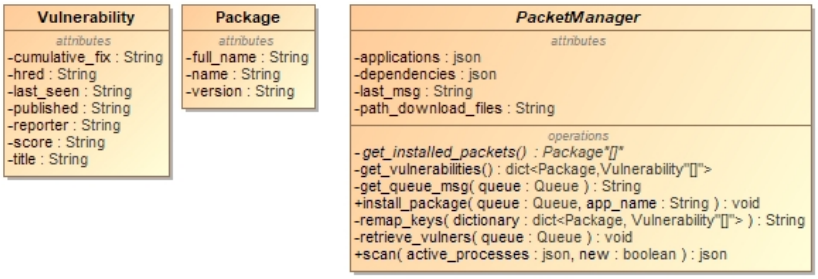

Figura 13: Packet manager

- **PacketManager:** clase que permite al software recopilar información de una forma rápida sobre los paquetes o software instalados en el sistema operativo así como

si alguno de ellos posee alguna vulnerabilidad publicada en repositorios públicos. *(Figura 13)*.

- **Vulnerability:** engloba a alto nivel el significado de vulnerabilidad y contiene todos los atributos que provee la base de datos con la que trabaja el software. *(Figura 13)*.

- **Package:** clase que abstrae distintos aspectos de un paquete o programa tales como su nombre y su versión. *(Figura 13)*.

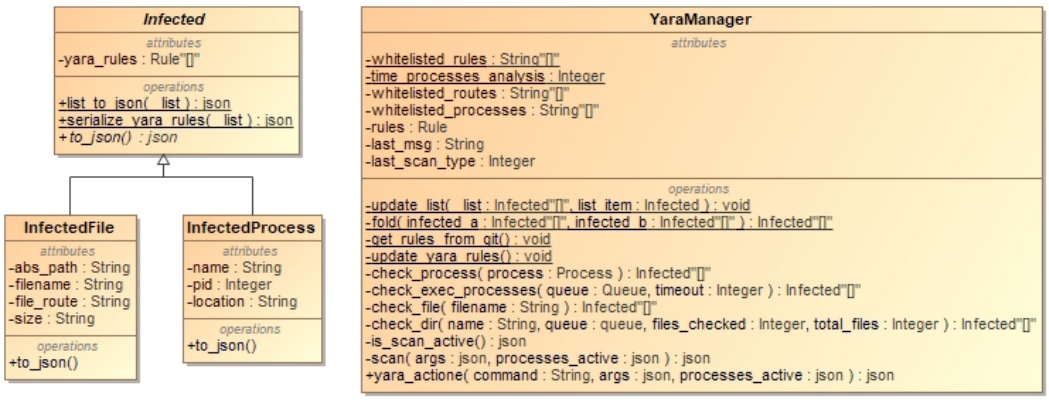

Figura 14: Yara manager

- **YaraManager:** dota al software de una capa mas abstracta a través de la cual interactuar con el escáner **YARA**. *(Figura 14)*.

- **Infected:** superclase que abstrae lo que podemos definir como infección y cuyo atributo principal son las reglas YARA que cumple. *(Figura 14)*.

- **InfectedFile:** engloba a los ficheros que son analizados por el escáner y dan positivo en alguna de las reglas. *(Figura 14)*.

- **InfectedProcess:** engloba a los procesos en memoria que son analizados por el escáner y dan positivo en alguna de las reglas. *(Figura 14)*.

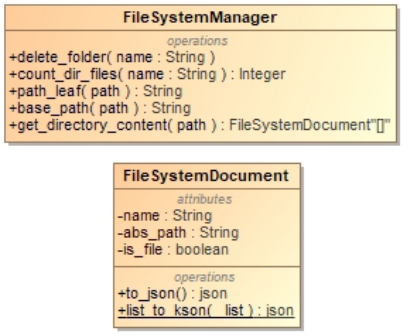

Figura 15: Filesystem manager

- **FileSystemManager:** conjunto de funciones que ayudan al sistema a tener interacción con todos los ficheros del sistema operativo en el que esta funcionando. *(Figura 15)*.
- **FileSystemDocument:** abstracción de los elementos que contiene el sistema de ficheros. *(Figura 15)*.

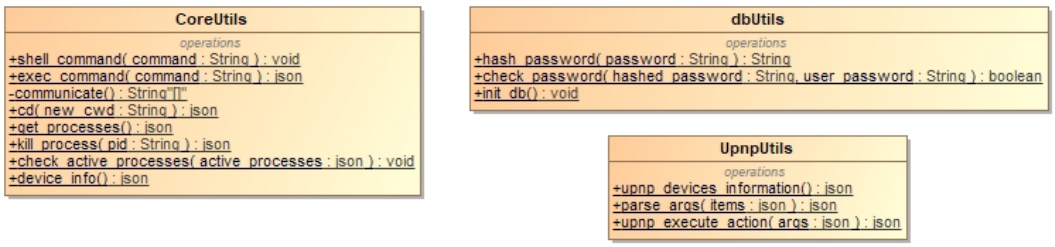

Figura 16: Otras clases

- **CoreUtils:** conjunto de funciones que dotan al software de herramientas para, de forma rápida interactuar con el sistema a bajo nivel.
- **dbUtils:** funciones que ayudan a la base de datos a su propia encriptación. *(Figura 16)*.
- **UpnpUtils:** abstracción que permite al sistema identificar e interactuar con los dispositivos que estén conectados a la red del sistema a través del protocolo UpNp. *(Figura 16)*.

4.3. Multiplataforma

Debido a que la implementación a bajo nivel cambia según el sistema operativo, se ha diseñado una arquitectura basada en el polimorfismo y herencia facilitando la adaptación al medio en el que se esta ejecutando. Concretamente esto se implementa en la clase **FirewallManager** (*figura 12*) y en la clase **PacketManager** (*figura 13*).

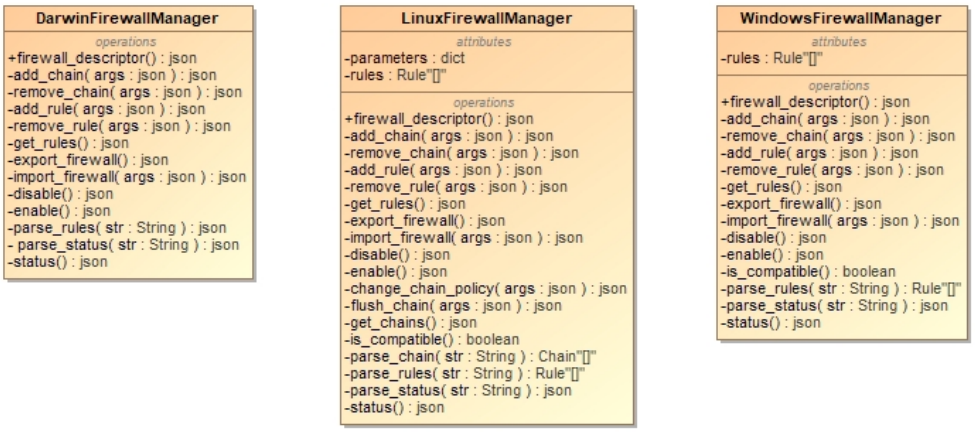

Figura 17: Multiplataforma de la clase FirewallManager

Las clases que sostentan e implementan la superclase FirewallManager quedan reflejadas en el diagrama anterior *(figura 17)*. Para cada sistema operativo, es necesario una subclase, que sea capaz de implementar a bajo nivel las llamadas adecuadas al sistema, con el fin de comunicarse con el tipo de firewall que implementan:

- **Windows:** firewall de windows.
- **Linux:** iptables.
- **MacOs X:** pfctl.

Así, el diagrama de clases correspondiente a la superclase anteriormente nombrada quedaría *(figura 18)*:

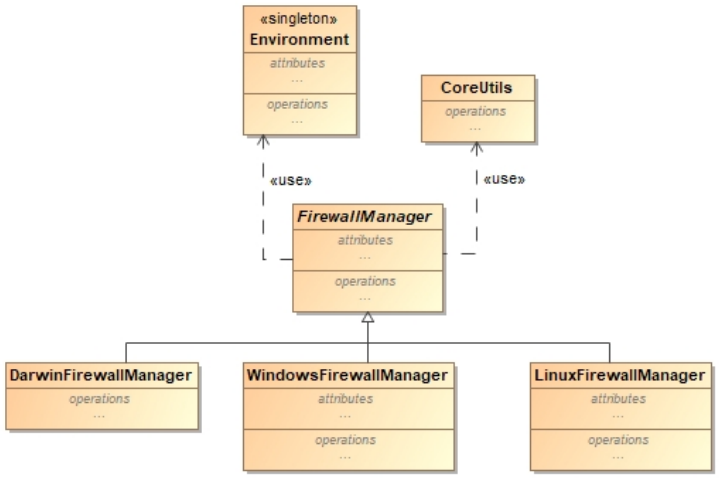

Figura 18: Diagrama de clases de FirewallManager

De la misma forma ocurre con la clase PacketManager y cuya implementación es realizada por las siguientes clases *(figura 19)*:

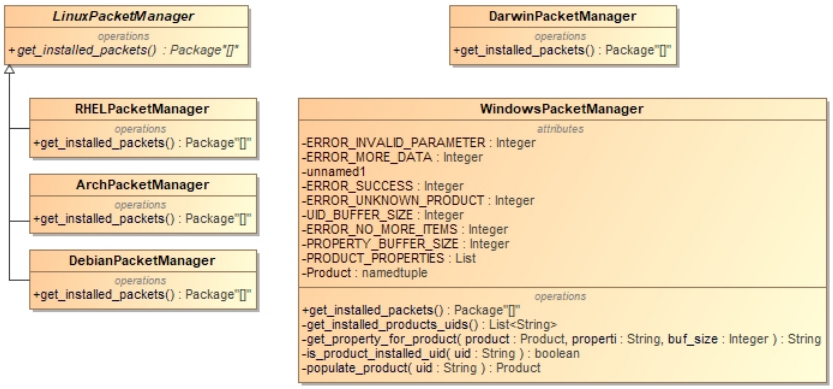

Figura 19: Multiplataforma de la clase PacketManager

De nuevo, cada una de estas clases implementa los comandos específicos de cada sistema operativo, con el objetivo de recolectar los datos para el correcto funcionamiento del sistema, quedando finalmente el diagrama de clases para PacketManager de la siguiente forma *(figura 20)*:

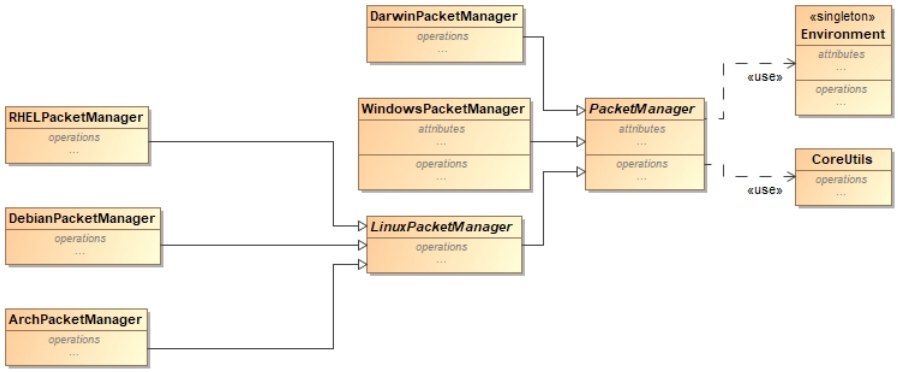

Figura 20: Diagrama de clases de PacketManager

4.4. Modelo

El modelo general del sistema se sostenta de dos clases principales, la clase **“Agent”** y la clase **“Environment”** (*figura 10*). De esta forma el modelo mas general del sistema es representado por el siguiente diagrama *(figura 21)*:

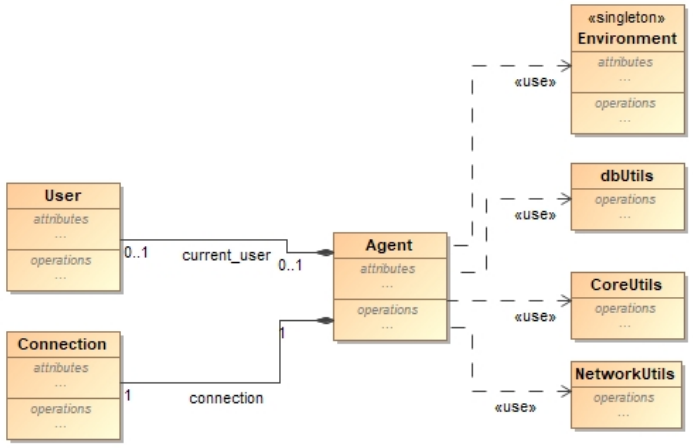

Figura 21: Diagrama de clases del sistema general

A través de este diagrama podemos observar, que es esta clase, **Agent**, la encargada de manejar a alto nivel las peticiones y redirigir adecuadamente para generar la respuesta adecuada. Esta clase hace uso de **“Environment”**, anteriormente nombrada y cuyo diagrama de clases es representado de la siguiente forma *(figura 22)*:

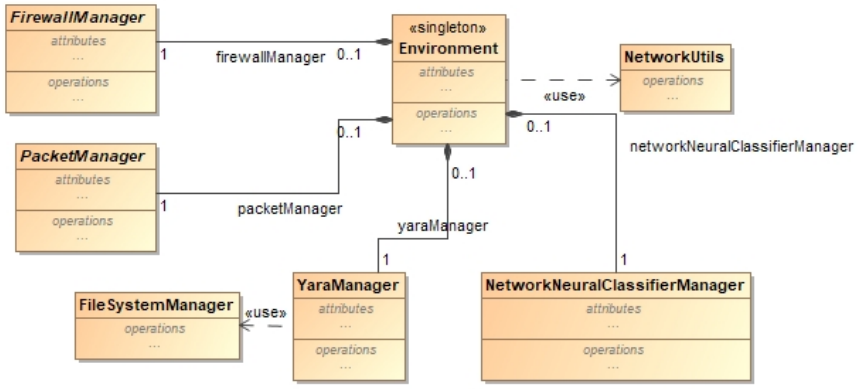

Figura 22: Diagrama de Environment

4.5. API

Debido a que el servidor de comunica con el cliente a través del paso de mensajes y utilizando JSON, se podrían crear diferentes clientes para todas las plataformas, es por ello por lo que la arquitectura del cliente se ha desarrollado de acuerdo a la siguiente documentación.

Antes de exponer las diferentes peticiones posibles, explicaremos la estructura de la petición ya que todas son iguales. La petición al servidor se estructura de la siguiente forma:

```
{
   "command":
   "args":
}
```

El atributo comando sera un tipo *String* que indicará el comando que necesitemos que el servidor realice, y el atributo *args* contendrá los parámetros necesarios para generar la respuesta adecuada a la petición. A continuación, se exponen las peticiones y respuestas posibles:

1. **cd:** cambia de directorio dentro del servidor. Devuelve si se ha cambiado o no de directorio y el directorio actual.

   Petición:

```
{
  "command": "cd"
  "args": String
}
```

   Respuesta:

```
{
  "status": boolean
  "data": String
}
```

2. **ps:** devuelve los procesos en ejecución del sistema.

   Petición:

```
{
  "command": "ps"
}
```

Respuesta:

```
{
    "status" : boolean
    "data" :
        [
            {
                "pid": Integer
                "name": String
            },
        ]
}
```

3. **kill:** mata el proceso cuyo pid se pasa por argumentos y devuelve si se ha podido matar o no y un mensaje de información.

   Petición:

```
{
  "command": "kill"
  "args": pid_proceso
}
```

   Respuesta:

```
{
  "status": boolean
  "data": String
}
```

4. **ports:** devuelve una lista con todos los puertos abiertos del sistema y el proceso asociado a dicho puerto.

   Petición:

```
{
  "command": "ports"
}
```

Respuesta:

```
{
    "status" : boolean
    "data" :
        [
            {
                "port": number
                "processes":
                    [
                        {
                            "pid": number
                            "name": str
                        },
                    ]
            }
        ]
}
```

5. **HWinfo:** genera información relevante acerca del hardware del sistema y la devuelve.

   Petición:

```
{
  "command": "Hwinfo"
}
```

Respuesta:

```
{
    "status" : boolean
    "data" :
        {
            "system":
                {
                    "platform": String
                    "system_users": [String]
                }
            "cpu":
                {
                    "processor": String
                    "architecture": String
                    "cores": String
                    "threads": String
                    "usage": [String]
                }
            "virtual_memory":
                {
                    "total": String
                    "available": String
                    "used": String
                    "used_percent": String
                }
            "disks":
                [
                    {
                        "device_name":
                            {
                                "mountpoint": String
                                "format": String
                                "features": String
                                "total": String
                                "used": String
                                "free": String
                                "used_percent": String
                            }
                    },
                ]
            "battery":
                {
                    "percent": String
                    "remaining_time": String
                    "power": String
                }
        }
}
```

6. **network analysis:** lanza un analisis sobre la red del sistema y devuelve si ha terminado o no y el resultado del análisis.

   Petición:

```
{
  "command": "network analysis"
}
```

   Respuesta:

```
{
  "status": boolean
  "data": String or [NetworkMeasure]
}
```

7. **upnp devices:** devuelve los dispositivos upnp cpnectados a la red del sistema asi como sus interfaces y acciones posibles.

   Petición:

```
{
  "command": "upnp devices"
}
```

Respuesta:

```
{
    "status" : boolean
    "data" :
        [
            {
                "location": str
                "name": str
                "services:"
                    [
                        {
                            "id": str
                            "actions:"
                                [
                                    {
                                        "name": str
                                        "args_in":
                                        "args_out":
                                        "url": str
                                    }
                                ]
                        }
                    ]
            }
        ]
}
```

8. **upnp exec:** ejecuta una acción upnp de un dispositivo.

   Petición:

```
{
  "command": "upnp exec"
  "args": {
            "location" : String
            "service" : String
            "action" : String
            "args_in" :
    }
}
```

Respuesta:

```
{
    "status" : boolean
    "data" : args_out
}
```

9. **vulners:** ejecuta un análisis en búsqueda de vulnerabilidades del dispositivo y devuelve el resultado.

   Petición:

```
{
  "command": "vulners"
}
```

   Respuesta:

```
{
  "status": boolean
  "data": String or [Vulnerability]
}
```

10. **yarascan scan:** realiza un escáner de acuerdo a las reglas yara.

    Petición:

```
{
  "command": "yarascan scan"
  "args": {
        "scan_type" : Integer
        "directory" : String
        "filename" : String
        }
}
```

Respuesta:

```
{
    "status" : boolean
    "data" : [Infected]
    "scan_type" : integer
}
```

11. **yarascan is active:** devuelve si hay o no un escáner yara activo en el sistema.

    Petición:

```
{
  "command": "yarascan is active"
}
```

    Respuesta:

```
{
  "status": boolean
}
```

12. **get folder content:** devuelve una lista con los elementos de un directorio.

    Petición:

```
{
  "command": "get folder content"
  "args": String
}
```

    Respuesta:

```
{
  "status": boolean
  "data": [FileSystemDocument]
}
```

Debido a la naturaleza de la comunicación con la característica que proporciona el sistema para la comunicación con el firewall, y a que, es la llamada a esta misma la que proporciona las funciones que se implementan para el sistema operativo en el que se este ejecutando el sistema servidor, se ha decidido exponer solo la llamada a la susodicha función siendo expuesta a continuación.

Petición:

```
{
  "command": "firewall descriptor"
}
```

Respuesta:

```
{
  "status": boolean
  "data": [Funciones]
}
```

Donde **"[Funciones]"**, es una lista compuesta por las funciones implementadas, teniendo la siguiente estructura:

```
{
"name": String,
"args": {
    arg1: type_arg1,
    arg2: type_arg2,
    ... : ...
    },
}
```

El atributo **"name"** es el nombre de la función, y la composición de la palabra "firewall" más el nombre de la susodicha, componen el comando que lanzaría la acción. El atributo **"args"** comprende los argumentos que debe recibir la función, así como sus tipos, para poder ser ejecutada.

4.6. Interfaz

La interfaz esta realizada con la biblioteca gráfica **WxPython** debido a su versatilidad, sencillez y la posibilidad de ser multiplataforma. La aplicación se compone de dos ventanas, la principal en la cual el usuario puede iniciar el servidor además de otras funciones (*figura 23*) y la ventana de gestión de usuarios en las que podrá crear y eliminar usuarios que tienen acceso al sistema (*figura 24*)

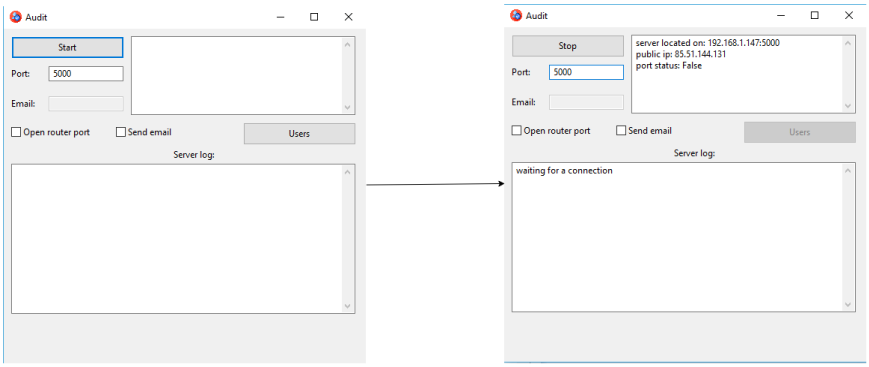

Figura 23: Ventana principal del servidor

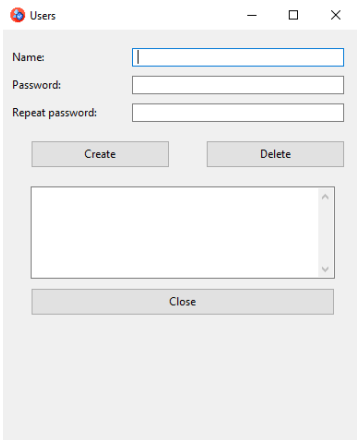

Figura 24: Ventana gestión de usuarios

5. Cliente

5.1. Tecnologías

El cliente se encuentra desarrollado para la plataforma android mediante el lenguaje de programación **java** y el sistema de dependencias **gradle** y el lenguaje de etiquetas **XML** a través del cual se especifican las interfaces en esta plataforma. Para la construcción de la aplicación se ha optado por una arquitectura basadas en tareas asíncronas que posibilitan la comunicación con el servidor de forma que el paso de mensajes no resulte costoso en términos de eficiencia de energía para el dispositivo. Así mismo y debido, a que la aplicación se basa en el tratamiento de los datos recibidos por el servidor en formato JSON, la aplicación se basa en interfaces que realizan acciones muy concretas, a través de una única clase denominada *"Connection"*, es por esto, y por simplificar su exposición, por lo que se ha optado por la realización de un diagrama de actividades de la aplicación en detrimento de la creación de un modelo de clases. Además, las interfaces están creadas de acuerdo a los estilos establecidos para la plataforma **android** e integrados en el sistema mediante el patrón *Modelo Vista Controlador* (MVC) que tan bien integrado esta con la susodicha plataforma.

5.2. Diagrama de actividades

Los siguientes diagramas muestran todas las acciones que podemos llevar a cabo a través de las diferentes actividades de las que se compone la aplicación, a excepción de la acción de regresar a la actividad anterior, debido a que es posible efectuarla en cualquier momento.

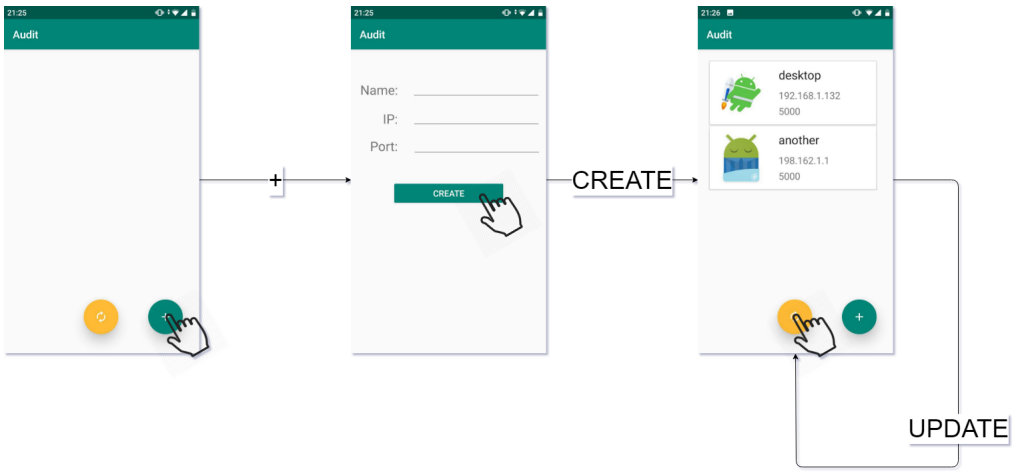

Figura 25: Diagrama actividad: Creación de dispositivos

El proceso para añadir un nuevo dispositivo en la aplicación *(figura 25)*, se compone de dos actividades diferentes, siendo la primera de ellas la actividad principal, es decir, la ventana **"Home"** de la aplicación y la segunda la actividad en la cual podemos introducir los datos para añadir el nuevo dispositivo. Estos datos son el nombre, dirección IP del dispositivo y puerto a través del cual se realiza la conexión y son obligatorios para su creación.

La conexión de la aplicación con el dispositivo *(figura 26)*, se realiza desde la actividad principal de la aplicación, en la que pulsaremos sobre un dispositivo previamente registrado, y una vez dentro, pasaremos a una nueva actividad que contendrá los datos del dispositivo accedido, y nos dará opciones tales como eliminarlo, actualizarlo o editar sus parámetros.

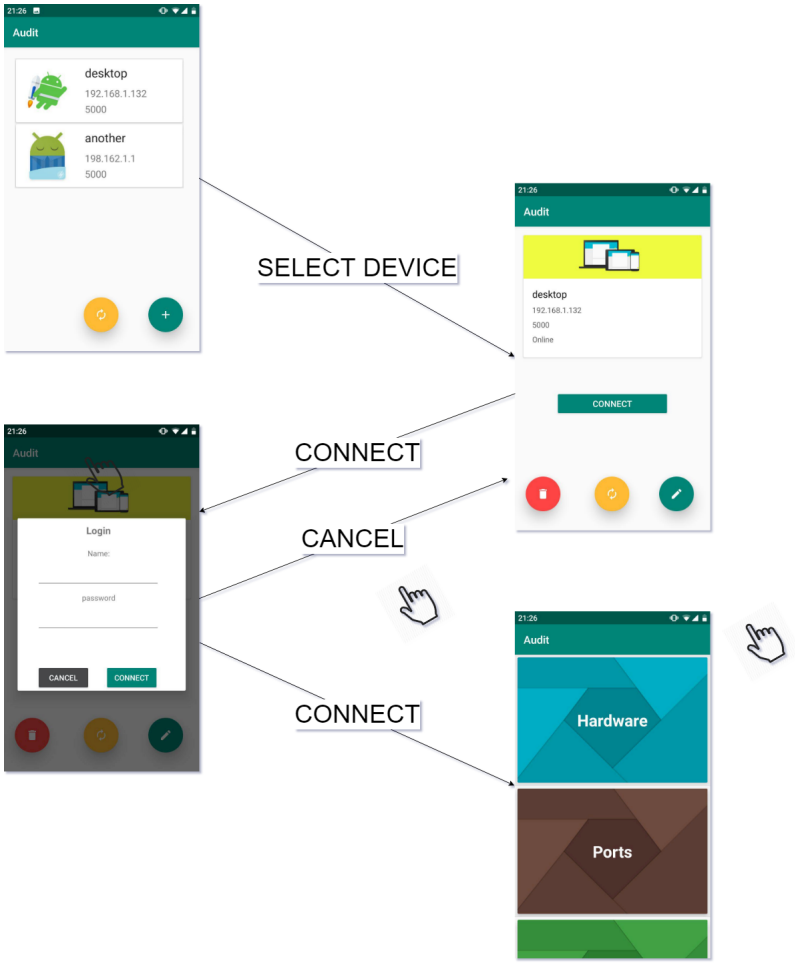

Figura 26: Diagrama actividad: Creación de dispositivos

Si el dispositivo se encuentra encendido, podremos acceder a él mediante el botón **“conectar”**, que abrirá un diálogo en el que deberemos ingresar nuestras credenciales. Si son correctas accederemos al menú principal de acciones del dispositivo, que nos permitirá interactuar con el mismo mediante la ejecución de diferentes tareas, que han sido definidas previamente en los objetivos de este trabajo.

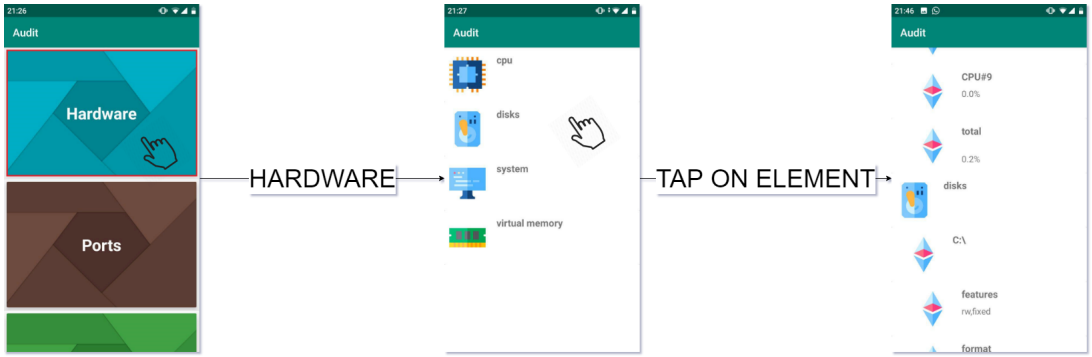

Figura 27: Diagrama actividad: Hardware

Si en el menú de acciones seleccionamos la opción **“Hardware”** *(figura 27)*, abriremos una actividad en la cual podremos ver todos los componentes del sistema, así como su estado en una agradable vista de árbol mediante la cual, podemos seleccionar los elementos expandiéndolos, para ver toda la información.

No es posible ver todo el hardware del equipo, esto es, debido a que esta función esta implementada en el servidor de una forma común a todos los sistemas operativos de escritorio en los que puede ejecutarse, y debido a que las llamadas al sistema son complicadas en numerosas ocasiones, se ha prescindido de otro tipo de información que podría haber sido relevante, tales como la temperatura del procesador, placa base, memoria. Tampoco es posible recuperar información de la gráfica debido a como se implementan los drivers de esta en los diferentes sistemas.

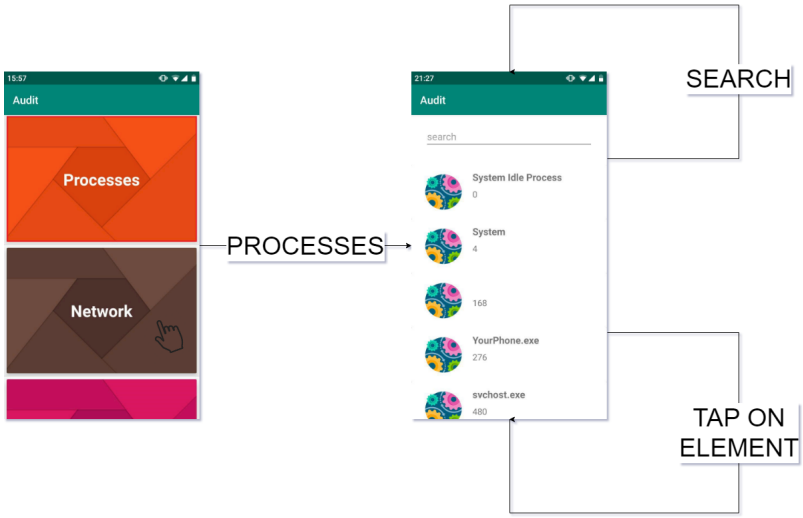

Figura 28: Diagrama actividad: Procesos

Si elegimos la opción **“Processes”** *(figura 28)*, pasaremos a una actividad que nos mostrará todos los procesos que se están ejecutando actualmente en el equipo y si pulsamos sobre ellos, mataremos el proceso en el dispositivo. Esta función se implementa en el servidor gracias a librerías escritas en C que son capaces de, mediante punteros en memoria y llamadas al sistema recoger los procesos en ejecución del sistema.

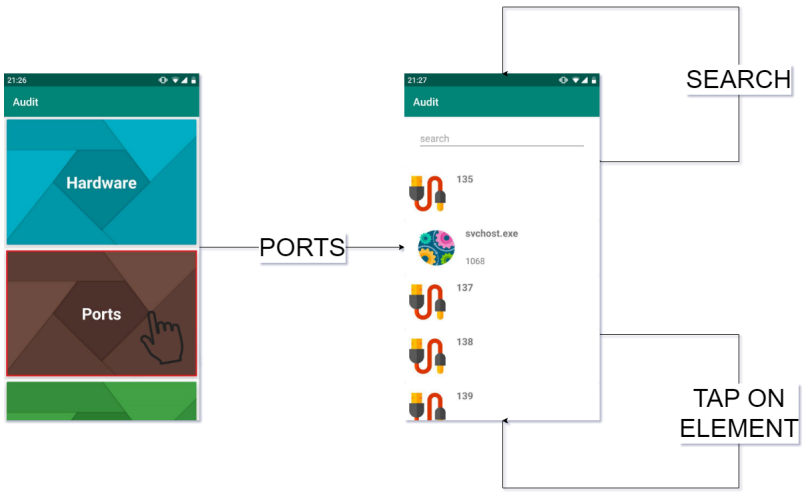

Figura 29: Diagrama actividad: Puertos del sistema

Otra opción disponible es **“Ports”** *(figura 29)*, que nos permitirá visualizar los servicios actualmente abiertos en el sistema así como indicándonos el proceso que lo esta usando además de filtrar por número de puerto, y poder matar el proceso si así lo deseamos. La parte mas interesante de esta función es sin duda conocer el proceso que lo usa pues de esta forma, el usuario si detecta alguna anomalía en ese puerto puede conocer que software esta provocando ese comportamiento anómalo en el sistema de una forma fácil e intuitiva.

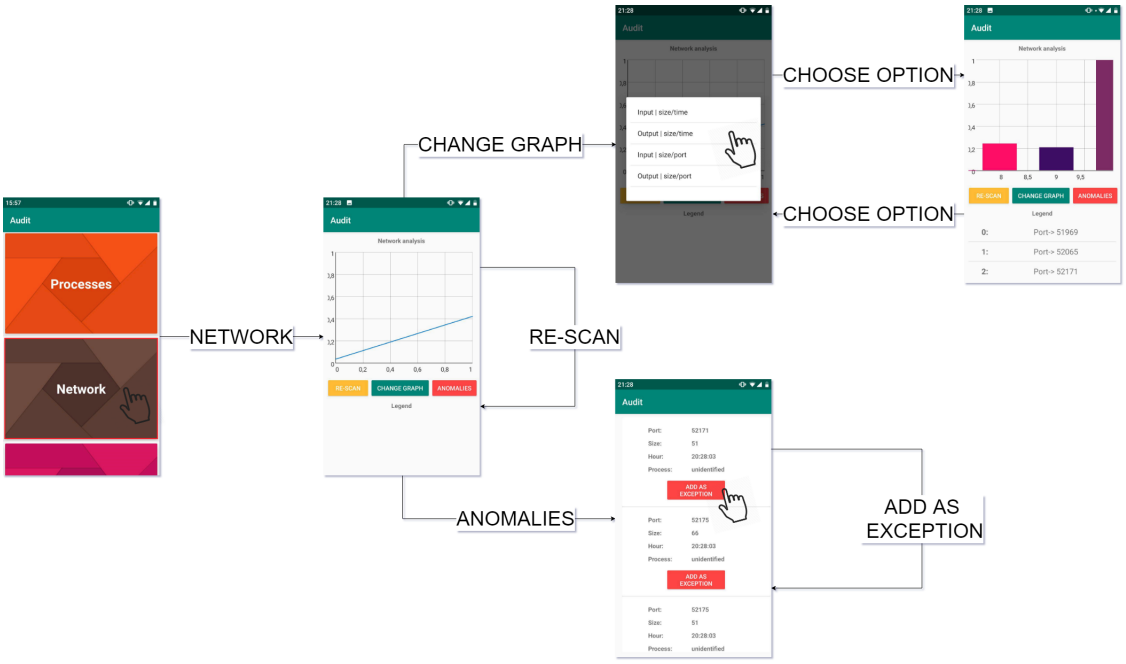

Figura 30: Diagrama actividad: Análisis de red

Una de las opciones mas interesantes que tenemos es **“Network”** *(figura 30)*, que nos permitirá analizar la red durante un corto periodo de tiempo, mostrándonos información del análisis mediante los siguientes diagramas:

- **Input/Size:** tráfico de entrada en función del tamaño y el tiempo.
- **Input/Port:** tráfico de entrada en función de los puertos y el tiempo.
- **Output/Size:** tráfico de salida en función del tamaño y el tiempo.
- **Output/Port:** tráfico de salida en función de los puertos y el tiempo.

Esta opción también avisará sobre comportamientos anómalos en la red, escaneando los paquetes procedentes del resultado del previo ofreciendo la posibilidad al usuario de marcarlas como excepciones.

Gracias a la potencia de la librería pcap que nos permite realizar un seguimiento del trafico de la red y al machine learning, concretamente a las ***Support Vector Machine*** **(SVM)** también conocidas como máquinas de vectores soportes, se ha conseguido crear una función útil, interesante y sobre todo potente par que cualquier usuario pueda ir acomodando la aplicación conforme al uso que le da al dispositivo.

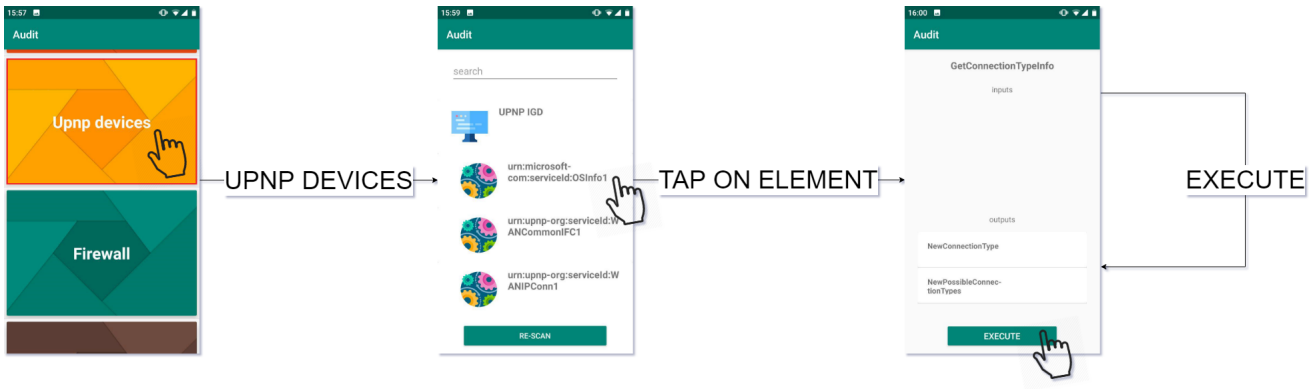

Figura 31: Diagrama actividad: Dispositivos UPnP

La opción **“Upnp devices”** *(figura 31)* nos permitirá visualizar dispositivos conectados a la red mediante este protocolo con el objetivo de explotar de forma controlada los servicios que estos ofrecen ejecutando las acciones disponibles para el dispositivo en cuestión.

Los dispositivos que utilizan el protocolo UPnP son en su mayoría inseguros, debido a que el propósito de este es, la facilitación al usuario de tener que realizar acciones para que el software domótico o accesible desde internet y que se encuentre en la red donde se ubica el dispositivo pueda funcionar correctamente. Concluyendo, esta función se enmarca en el contexto de que el usuario sea capaz de identificar que dispositivos posee que puedan dar lugar a ataques por parte del exterior debido a su inseguridad.

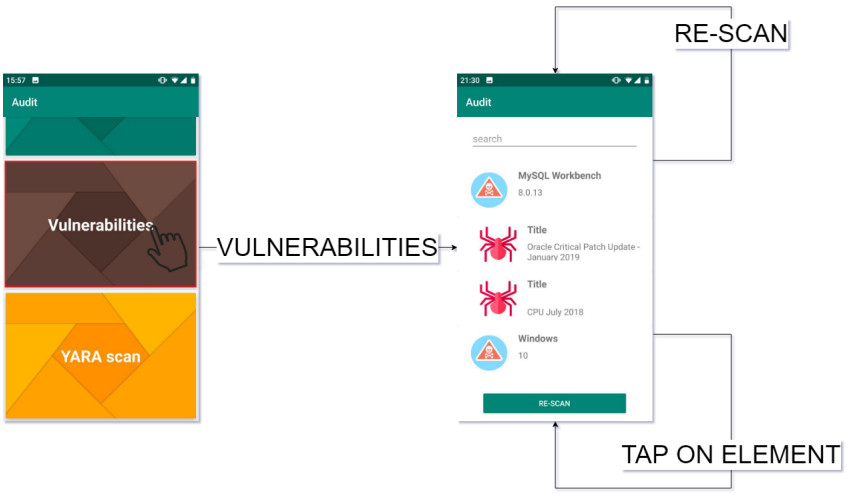

Figura 32: Diagrama actividad: Vulnerabilidades

Otra de las funciones es **“Vulnerabilities”** *(figura 32)*, nos permite analizar los programas instalados en el equipo y reconocibles por el sistema operativo, es decir, todo

aquel software que el sistema operativo es consciente de su instalación, en busca de vulnerabilidades reportadas en repositorios públicos de forma que el usuario sea consciente de los riesgos a los que esta expuesto. Las vulnerabilidades son mostradas junto a toda la información disponible de la misma en la base de datos y repositorios públicos donde se comprueban estas vulnerabilidades.

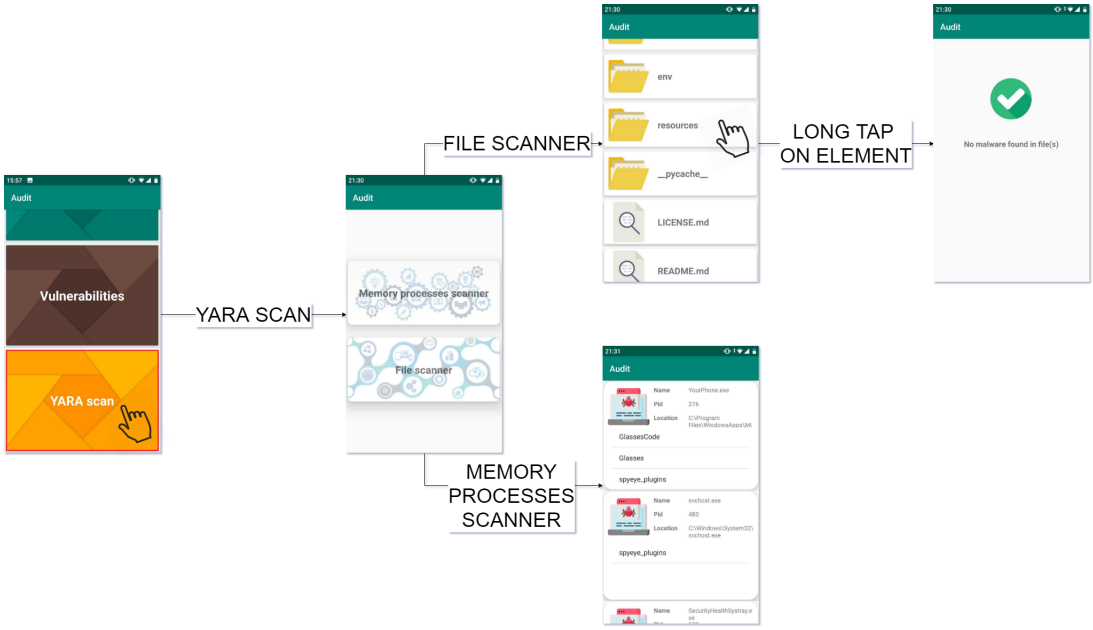

Figura 33: Diagrama actividad: Escáner YARA

Por ultimo, la función **“YARA scan”** *(figura 33)* que, por un lado, puede lanzar un escáner sobre un fichero o un directorio completo, y por otro lado, puede lanzar el análisis sobre los procesos que se están ejecutando en memoria. Una vez analizados los resultados son mostrados en una lista en la que se indican las reglas YARA en las que ha dado positivo.

Podemos entender YARA como una tecnología que a raíz de un volcado de los datos de un proceso o fichero, identifica patrones de acuerdo a reglas escritas en este lenguaje y que se han definido previamente, no obstante, y aunque útil y potente, YARA es también una herramienta que puede ser evadida por malware específico.

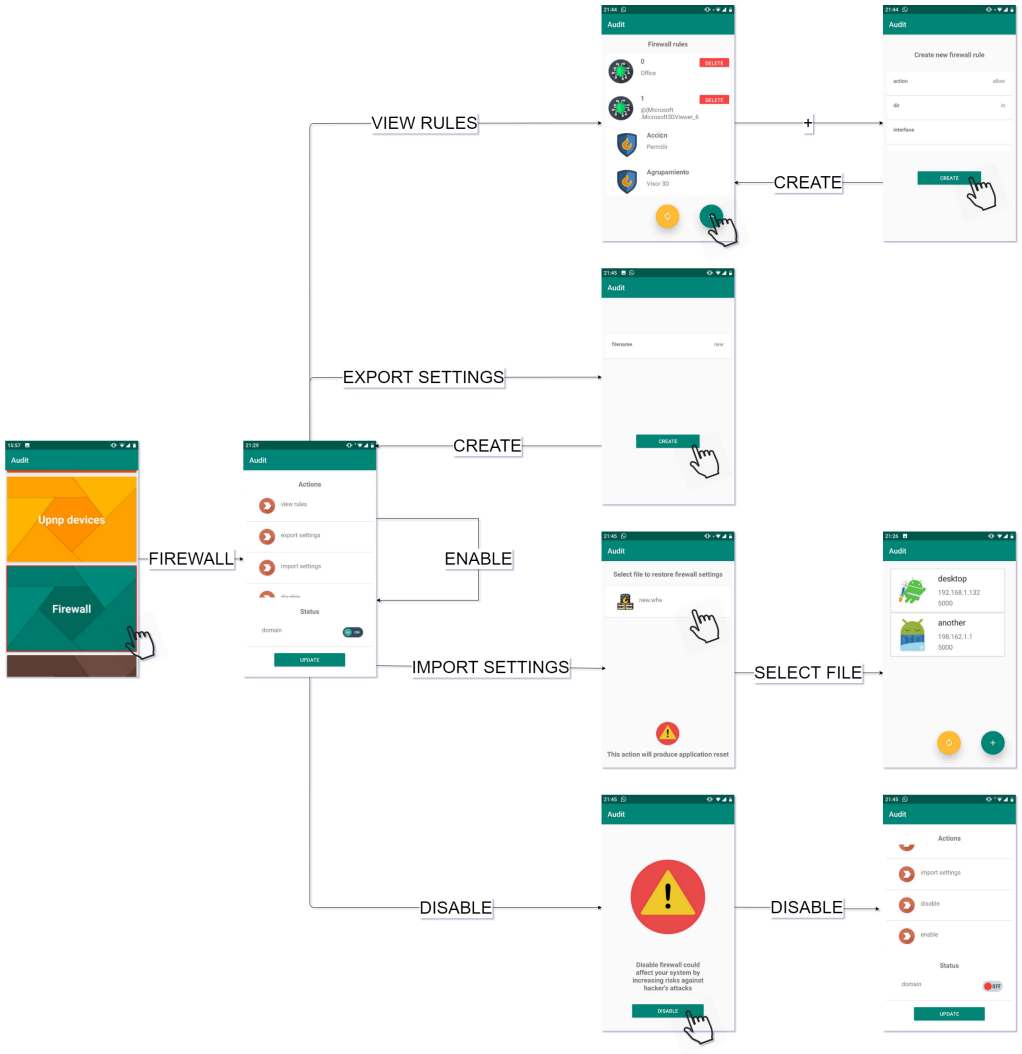

Figura 34: Diagrama actividad: Firewall

La opción **“Firewall”** *(figura 34)* nos permitirá comunicarnos con el firewall de nuestro sistema pudiendo realizar, dependiendo del sistema operativo, todas las tareas que nos brinda, es decir podemos exportar su estado actual, importar una copia anterior, crear o eliminar reglas, así como activarlo y desactivarlo, además de crear cadenas etc.

5.3. Arquitectura de la aplicación móvil

La aplicación se sustenta sobre la clase **“Connection”** anteriormente nombrada, que provee al sistema de una capa de alto nivel para que este pueda comunicarse de forma sencilla y segura con el dispositivo al que desee el usuario conectarse mediante el uso de sockets TCP y mediante TLS integrando en la aplicación un almacén de certificados seguros y reconocibles para la aplicación servidor, protegido por una contraseña. Además, el sistema se compone de otras clases a través de las cuales los datos en formato JSON pueden convertirse a objetos con el objetivo de diseñar interfaces y adaptadores que faciliten el uso del usuario final. En definitiva podemos afirmar que no se realiza ninguna operación con los datos facilitados por el servidor en el mismo dispositivo sino, que se trata de una aplicación capaz de transformar y realizar las acciones disponibles por el servidor de forma sencilla y transparente para el usuario.

5.4. Diseño y estilo

El diseño de la aplicación esta fuertemente influenciado por **“material design”**, que es la guía de estilos por excelencia en la plataforma android. Además se ha enfocado el diseño de la misma a ser lo mas intuitiva posible con el objetivo de que todo el mundo, pueda utilizarla sin ningún problema. También se han utilizado iconos creados por vectores y cuyos autores quedan retribuidos en la sección 12.2.

6. Problemas surgidos y soluciones

La toma de requisitos y la especificación de casos de uso, tienen lugar en un momento muy temprano del desarrollo software, con el fin de obtener una correcta especificación entre lo que se pide y el producto a desarrollar. No obstante en estas fases tan tempranas se tiende a pensar que el desarrollo puede ser mas fácil de lo que luego, realmente es. Este proyecto no ha estado exento de estos errores y por consiguiente en esta sección se exponen los principales problemas ocurridos durante el desarrollo del mismo así como las soluciones que se implementaron para el correcto funcionamiento del sistema.

6.1. Seguridad de la comunicación

En primera instancia y atendiendo a los casos de uso, se especifico la creación de un formulario de acceso para usuarios para que no todo el mundo desde su cliente pudiese conectarse a ordenadores de los cuales no era propietario. No obstante, en un proyecto orientado a la seguridad informática, la comunicación de la transmisión de los datos entre el cliente *(aplicación móvil)*, y el servidor *(dispositivo)*, debía estar cifrada.

Finalmente se opto por la creación de certificados autofirmados para cifrar la comunicación mediante TLS, como se expuso anteriormente en la sección 3.6. Una vez la comunicación estuviese cifrada no había problema en que los datos pudiesen ser obtenidos mediante técnicas de **analizamiento de paquetes *(Sniffer)*** ya que no circulaban en texto plano, y el cifrado no era derivable *(figura 35)*.

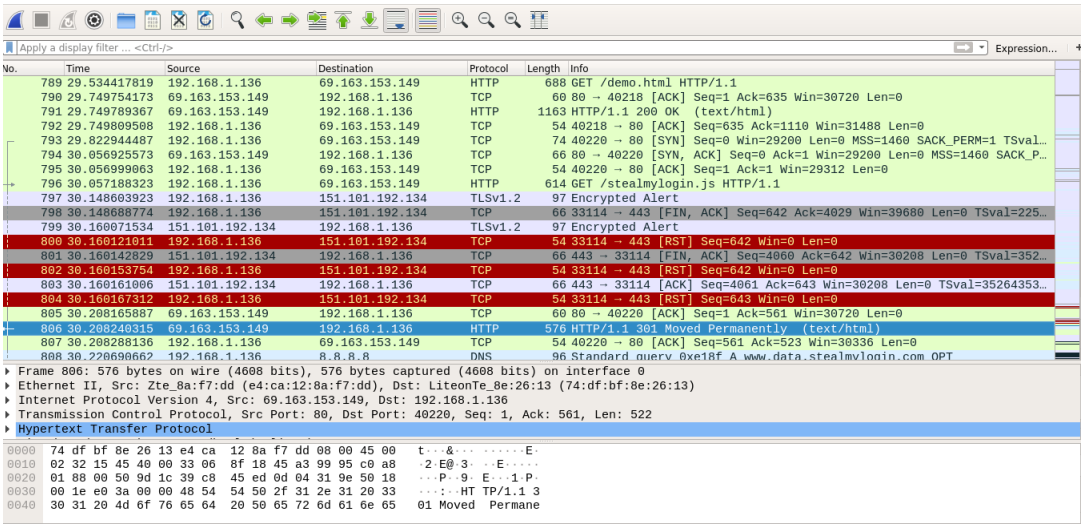

Figura 35: Análisis de paquetes usando **“wireshark”** *[5]*

6.2. Análisis de vulnerabilidades

En el momento en el que se especifico este requisito, no se hizo de la forma mas correcta. No se pensó con suficiente detalle como se podría escanear el dispositivo buscando software instalado cuya versión tuviese una vulnerabilidad reportada en los **“CERT”** lo que llevó a una falsa sensación de sencillez.

Cuando se empezó a definir la arquitectura de esta característica surgió el primer problema, la multiplataforma. Un requisito fundamental era que la aplicación pudiese ser ejecutada en el mayor número posible de sistemas operativos de escritorio *(Windows, Mac OS, Linux)*. Este problema fue abarcado desde distintas perspectivas:

1. **Creación de un “middleware”:** este nuevo middleware se encargaría de la comunicación entre el sistema operativo y bases de datos de “exploit” como es **“Exploit database”** *[19]*.

2. **Creación de un servicio aislado:** este servicio atendería peticiones y sería un proyecto diferente únicamente dedicado a buscar en la ***National Vulnerability Database*** **(NVD)** “CVSS” que reportasen vulnerabilidades sobre el software encontrado en el dispositivo.

3. **Usar la API de “Vulners”:** que ahorraría trabajo y hacia la tarea por nosotros mandándole las especificaciones del software encontrado.

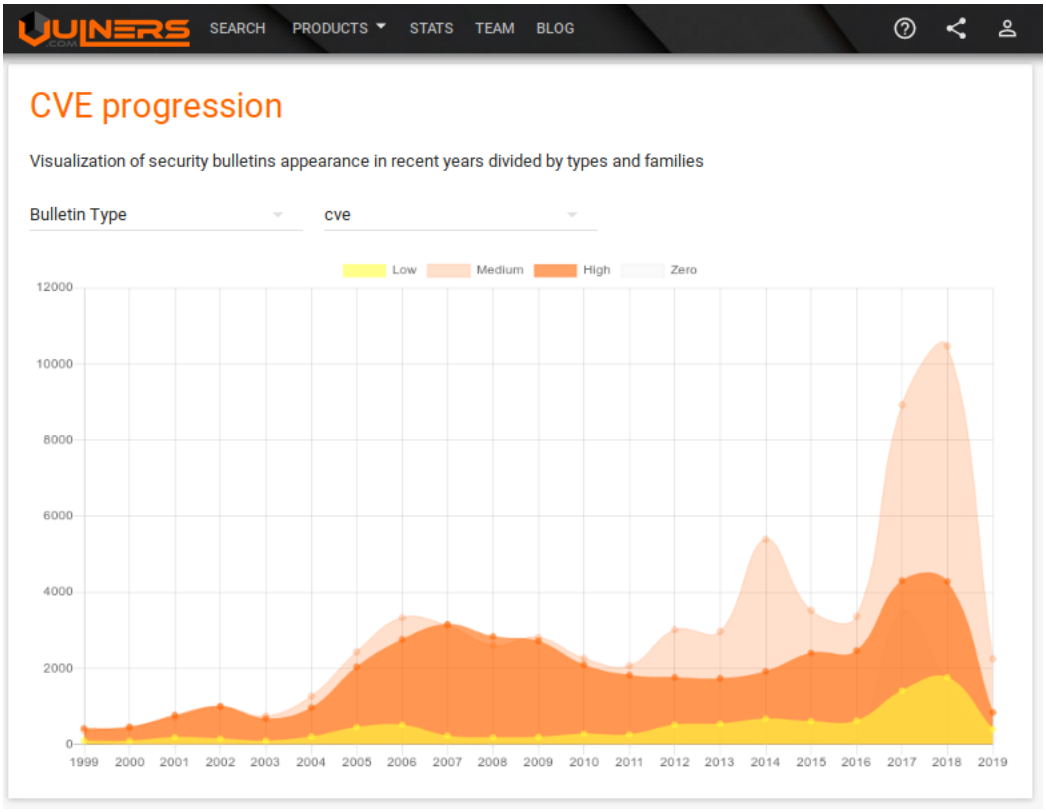

Figura 36: Página de **“Vulners”** *[23]*

Finalmente y como esta expuesto y explicado anteriormente en la sección 4.1 se optó por el uso de “Vulners” *(figura 36)*.

Cualquiera de las soluciones nombradas anteriormente tenía otro problema, recabar información acerca del software instalado en el dispositivo. De esta forma nos encontramos los siguientes problemas:

- **La multiplataforma:** debíamos implementar mecanismos para recabar información en todos los sistemas operativos donde, según los requisitos debía funcionar el producto, lo que introdujo una serie de complicaciones internas.
- **La escasez de comunicación con "Windows":** este sistema operativo no ofrece mecanismos sencillos para comunicarse externamente con el debido su naturaleza privativa.
- **La fragmentación de Linux:** existen tantas distribuciones de este sistema operativo que sería imposible abarcarlas todas, ya que existen muchos gestores de paquete diferentes. Es por eso por lo que finalmente nos centramos en 3 tipos, **"DPKG", "PACMAN" y "RPM"**.

El primero de estos problemas fue atacada mediante técnicas de desarrollo software como la herencia y la creación de interfaces para crear una capa transparente para el sistema como se expone en la sección 4.3.

En cuanto al problema con "Windows" finalmente se decidió obtener solo información del registro del sistema operativo acerca del software instalado, es decir, no se tenia consciencia de paquetes instalados que no estuviesen especificados en el registro anteriormente nombrado.

Figura 37: Gestores de paquetes soportados *[20]*

Finalmente para solucionar el problema de la fragmentación de “Linux” con los gestores de paquetes, se redujeron los sistemas operativos soportados por esta característica del proyecto, a aquellas distribuciones cuyos gestores de paquetes, fuesen algunos de los definidos anteriormente. *(figura 37)*.

6.3. Comunicación con el “firewall”

La comunicación con el firewall no podía ser directa, de nuevo una vez mas debido a la multiplataforma y a la diferencia fundamental entre los 3 tipos de firewall que este proyecto abarca, **“firewall de windows”, “Iptables” y “pfctl”**.

Para solventar este problema, y tal y como se explica en la sección 4.3, se optó por la creación de interfaces que creasen una capa de alto nivel, trasparente para el sistema. No obstante y debido a que cada “firewall” tiene sus limitaciones y/o funciones, se optó por reducir el número de opciones disponibles de cara al usuario final, y la creación de un método, que el cliente pudiese invocar, para que el servidor le informase sobre las funciones disponibles, y como llamar a su ejecución.

Figura 38: opciones del comando **“netsh”**

En el caso de **“Iptables”** y **“pfctl”**, son ejecutables desde una consola del sistema lo que resultaba mas o menos sencillo la comunicación del “firewall” con el sistema del producto. No obstante, el **“firewall de windows”** y una vez más debido a su naturaleza privativa es algo mas complejo, y para el manejo de este se decidió utilizar un comando de este sistema llamado **“netsh”** que permite una comunicación mas directa con el mismo *(figura 38)*.

6.4. Análisis de la red

Para hacer un análisis de la red debemos ser capaces de obtener los paquetes que circulan por la red en la que esta el dispositivo haciendo uso de **“sniffers”**, anteriormente nombrado en la sección 6.1.

La librería principal usada para la realización y procesamiento de los paquetes de red es **“pcap”** y esta escrita en C. Por suerte, existen librerías para python que, aunque dependen del sistema operativo, enlazan este lenguaje de programación con llamadas a la susodicha librería facilitándonos la tarea de la obtención de los paquetes anteriormente nombrados. El problema de estas librerías es la escasez de documentación, que una vez solventada mediante la lectura del código nos lleva a un problema un poco mas grave, la ley.

Debido a que el proyecto no sabe en que condiciones va ser utilizado y la naturaleza ilícita de realizar estas técnicas en redes no propietarias, se ha decidido que el análisis de la red en busca de anomalías sea solo sobre los paquetes entrantes y salientes del dispositivo en el que se ejecuta la aplicación.

7. Plan de pruebas

A continuación se exponen los casos de prueba establecidos para el proyecto con el fin de satisfacer las necesidades del mismo mediante su posterior validación (sección 7.2). El formato establecido para la definición del plan de pruebas, tanto los casos de prueba, como los de validación son tablas basadas en los estándares incluidos en las publicaciones 800-115 [14], y 800-53A [15] del **NIST**, en el que se especifica el ID del paquete a evaluar así como su nombre, objetivos y los planes de evaluación del paquete.

7.1. Casos de prueba

PR-US	**Alta, baja y autenticación de usuarios**	
	Objetivo a evaluar	registro, autenticación y eliminación de los usuarios.
	US-CV-1	comprobación por parte de un usuario con conocimientos del proyecto del correcto funcionamiento de la característica.
	US-CV-2	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas de integración	

PR-FN-HW	Función “hardware” del sistema	
	Objetivo a evaluar	veracidad de los datos resultados de la ejecución de la característica y sencillez de uso de la misma.
	HW-CV-1	validación de los datos resultados de la ejecución de la característica por parte de un usuario con conocimientos técnicos del proyecto.
	HW-CV-2	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas unitarias	

PR-FN-PR	Función “processes” del sistema	
	Objetivo a evaluar	veracidad de la información recibida como resultado de la ejecución de la función que obtiene los procesos que se encuentran ejecutados en el sistema, el correcto funcionamiento de la búsqueda de los mismos y su posible finalización así como la sencillez de uso.
	PR-CV-1	validación de la información recibida de obtener los procesos que se encuentran en estado de ejecución.
	PR-CV-2	comprobación del correcto funcionamiento de la acción de búsqueda.
	PR-CV-3	comprobación del correcto funcionamiento de la acción de finalización del proceso seleccionado.
	PR-CV-4	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas unitarias	

PR-FN-PU	Función “ports” del sistema	
	Objetivo a evaluar	veracidad de la información recibida como resultado de la ejecución de la función que obtiene los puertos y servicios abiertos en el sistema, el correcto funcionamiento de la búsqueda de los mismos y su la posible finalización de los procesos que los poseen, así como la sencillez de uso.
	PU-CV-1	validación de la información recibida de obtener los puertos y servicios que se encuentran abiertos.
	PU-CV-2	comprobación del correcto funcionamiento de la acción de búsqueda.
	PU-CV-3	comprobación del correcto funcionamiento de la acción de finalización del proceso que posee el servicio.
	PU-CV-4	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	■ Entrevista con usuarios no expertos ■ Pruebas unitarias	

PR-FN-VU	Función “Vulnerabilities” del sistema	
	Objetivo a evaluar	grado de efectividad de la búsqueda de vulnerabilidades y sencillez de uso.
	VU-CV-1	validación de la información recibida resultante de la ejecución del análisis en el sistema servidor.
	VU-CV-2	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas unitarias	

PR-FN-UP	Función “UPnP” del sistema	
	Objetivo a evaluar	funcionalidad de la característica y sencillez de uso.
	UP-CV-1	validación de los dispositivos identificados como UPnP dentro de la red del sistema servidor.
	UP-CV-2	comprobación de la ejecución satisfactoria de acciones ofertadas por los dispositivos identificados.
	UP-CV-3	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas unitarias	

PR-FN-NW	Función “network” del sistema	
	Objetivo a evaluar	funcionalidad de la característica y sencillez de uso.
	NW-CV-1	comprobación de la correcta visualización de los datos recibidos del análisis para todas las formas disponibles de visualización.
	NW-CV-2	verificación de la detección de anomalías y aprendizaje del sistema.
	NW-CV-3	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas unitarias	

PR-FN-FW	Función “firewall” del sistema	
	Objetivo a evaluar	funcionalidad de la característica, integración de las acciones disponibles según sistema operativo y sencillez de uso.
	FW-CV-1	verificación de los permisos necesarios para poder ejecutar la acción.
	FW-CV-2	comprobación del funcionamiento de las distintas opciones ejecutables en función del sistema operativo.
	FW-CV-3	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas de integración	

PR-FN-MA	Función “malware scan” del sistema	
	Objetivo a evaluar	verificación del análisis de malware del dispositivo y sencillez de uso.
	MA-CV-1	comprobación del correcto funcionamiento del escáner de ficheros.
	MA-CV-2	comprobación del correcto funcionamiento del escáner de procesos en memoria.
	MA-CV-3	uso de la característica por parte de usuarios ajenos al proyecto con bajo nivel en el uso de sistemas informáticos para evaluar la sencillez de la misma.
	Herramientas y métodos de evaluación	
	▪ Entrevista con usuarios no expertos ▪ Pruebas unitarias	

7.2. Validación de los casos de prueba

La incorporación de cada una de las características contempladas en el plan de prueba, se han realizado mediante pruebas de integración continua para la comprobación de la robustez y correcto funcionamiento del sistema en conjunto. A continuación se detallan las validaciones individuales para los paquetes anteriormente descritos.

<table>
<tr><th>PR-US</th><th colspan="3">Alta, baja y autenticación de usuarios</th></tr>
<tr><td rowspan="5"></td><td>Objetivo a evaluar</td><td colspan="2">registro, autenticación y eliminación de los usuarios.</td></tr>
<tr><td>US-CV-1</td><td>pruebas de integración realizadas por un usuario del proyecto comprobando la robustez de la característica implementada.</td><td>Grado de éxito: A</td></tr>
<tr><td>US-CV-2</td><td>evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.</td><td>Grado de éxito: M</td></tr>
<tr><td colspan="3">Comentarios y recomendaciones</td></tr>
<tr><td colspan="3">■ US-CV-2: eliminar usuarios del sistema requiere conocer las credenciales del usuario a eliminar.</td></tr>
</table>

PR-FN-HW	Función “hardware” del sistema		
	Objetivo a evaluar	veracidad de los datos resultados dela ejecución de la característica y sencillez de uso de la misma.	
	HW-CV-1	pruebas unitarias realizados por un usuario del proyecto comprobando la veracidad de la información resultante de la ejecución de la característica así como su correcto funcionamiento.	**Grado de éxito:** A
	HW-CV-2	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A

PR-FN-PR	Función “processes” del sistema (1)		
	Objetivo a evaluar	veracidad de la información recibidacomo resultado de la ejecución de lafunción que obtiene los procesos quese encuentran ejecutados en el siste-ma, el correcto funcionamiento de la-búsqueda de los mismos y su posi-ble finalización así como la sencillezde uso.	
	PR-CV-1	realización de pruebas unitarias que comprueban que los procesos encontrados son, exactamente los procesos que se ejecutan en el sistema en el momento llamar a la susodicha función.	**Grado de éxito:** A
	PR-CV-2	comprobación de la robustez de la acción de búsqueda.	**Grado de éxito:** A
	PR-CV-3	creación de pruebas unitarias que intentan la finalización de un proceso obtenido mediante la llamada a esta característica.	**Grado de éxito:**M

PR-FN-PR	Función “processes” del sistema (2)		
	PR-CV-4	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A
	Comentarios y recomendaciones		
	▪ **PR-CV-3:** el proceso seleccionado no siempre finaliza, debido a que, o bien no se tienen los permisos necesarios, o bien el proceso ya no se encuentra en ejecución en el momento de la selección.		

PR-FN-PU	Función “ports” del sistema (1)		
	Objetivo a evaluar	veracidad de la información recibida como resultado de la ejecución de la función que obtiene los puertos y servicios abiertos en el sistema, el correcto funcionamiento de la búsqueda de los mismos y su la posible finalización de los procesos que los poseen,así como la sencillez de uso.	
	PU-CV-1	creación de pruebas unitarias que comprueban que los servicios abiertos encontrados son, exactamente los servicios abiertos en el momento llamar a la susodicha función.	**Grado de éxito:** A
	PU-CV-2	comprobación de la robustez de la acción de búsqueda.	**Grado de éxito:**M
	PU-CV-3	realización de pruebas unitarias que intentan la finalización de un proceso que posee el servicio abierto.	**Grado de éxito:**M

PR-FN-PU	Función “ports” del sistema (2)		
	PU-CV-4	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A
	Comentarios y recomendaciones		
	▪ **PU-CV-2:** solo se puede buscar por número de puerto no por proceso que posee el puerto. ▪ **PU-CV-3:** el proceso que posee un servicio no siempre finaliza, debido a que, o bien no se tienen los permisos necesarios, o bien el servicio ya no esta abierto en el momento de ejecución de la acción.		

PR-FN-VU	Función “Vulnerabilies” del sistema		
	Objetivo a evaluar	grado de efectividad de la búsqueda de vulnerabilidades y sencillez de uso.	
	VU-CV-1	análisis del grado de vulnerabilidades recogidas en diferentes sistemas y versiones.	**Grado de éxito:**M
	VU-CV-2	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A
	Comentarios y recomendaciones		
	▪ **VU-CV-1:** no todas las vulnerabilidades son recogidas, debido a problemas con caracteres en otro formato que compongan el nombre y o la versión de la aplicación instalada.		

PR-FN-UP	Función “UPnP” del sistema		
	Objetivo a evaluar	funcionalidad de la característica y sencillez de uso.	
	UP-CV-1	comprobación física de igualdad entre los dispositivos identificados y los dispositivos instalados en la red del sistema servidor .	**Grado de éxito:** A
	UP-CV-2	pruebas unitarias para la ejecución de las acciones ofertadas por los dispositivos.	**Grado de éxito:**M
	UP-CV-3	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A
	Comentarios y recomendaciones		
	▪ **UP-CV-2:** una minoría de las acciones no pueden ser invocadas con éxito debido a incompatibilidades entre los tipos en el lenguaje de programación establecidos y los tipos soportados por estos dispositivos y protocolos.		

PR-FN-NW	Función “network” del sistema		
	Objetivo a evaluar	funcionalidad de la característica y sencillez de uso.	
	NW-CV-1	pruebas de comprobación de la visualización de los gráficos para la representación de los datos.	**Grado de éxito:** A
	NW-CV-2	Verificación mediante test unitarios del aprendizaje del sistema tras la indicación y formación previa del modelo que las gestiona.	**Grado de éxito:** A
	NW-CV-3	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A

PR-FN-FW	Función “firewall” del sistema		
	Objetivo a evaluar	funcionalidad de la característica y sencillez de uso.	
	FW-CV-1	pruebas unitarias de acceso a la característica para comprobar su ejecución en función de los permisos de ejecución.	**Grado de éxito:** A
	FW-CV-2	pruebas de integración de todas las funcionalidades implementadas por la característica según el sistema operativo para verificar el correcto funcionamiento de la misma en su totalidad.	**Grado de éxito:** A
	FW-CV-3	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A
	Comentarios y recomendaciones		
	▪ **FW-CV-2:** tras la ejecución de diversas funciones se requiere del reinicio de la aplicación debido a microcortes producidos por cambios de directiva en el firewall del dispositivo servidor.		

PR-FN-MA	Función “malware scan” del sistema		
	Objetivo a evaluar	verificación del análisis de malwaredel dispositivo y sencillez de uso.	
	MA-CV-1	creación de pruebas unitarias con ficheros maliciosos y libres de malware.	**Grado de éxito:**M
	MA-CV-2	creación de pruebas unitarias con entornos controlados y procesos malware en ejecución.	**Grado de éxito:**M
	MA-CV-3	evidencias obtenidas por parte de usuarios ajenos al proyecto sobre la sencillez de la característica implementada.	**Grado de éxito:** A
	Comentarios y recomendaciones		
	▪ **MA-CV-1:** la comprobación de malware funciona con las reglas yara disponibles, lo que significa que el malware que no se contemple en estas reglas no será detectado. ▪ **MA-CV-2:** debido a la protección de diversos procesos, es mas o menos sencillo que el escáner tenga falsos positivos.		

8. Conclusiones

El desarrollo de la aplicación ha enfrentado numerosos retos, sobretodo de cara a la implementación debido a la intencionalidad de facilitar al usuario el uso de la misma con el objetivo de que, cualquier usuario pueda estar un poco mas cerca del mundo de la ciberseguridad, sin entrar en detalles excesivos. Uno de los retos mas grande ha sido sin duda la programación y desarrollo de la multiplataforma y todo el **"middleware"** desarrollado para que la aplicación, pueda comunicarse con cada sistema operativo soportado, para dar soporte a la función de control de firewall, que a mi parecer es de las mas interesantes, pues poder controlarlo desde el teléfono es algo totalmente novedoso, y que no implementa ninguna otra aplicación, permitiendo al usuario toda la potencia de manejo del mismo en el bolsillo.

Otra de las funciones mas interesantes es sin duda poder comprobar el software instalado en busca de vulnerabilidades publicadas, para que el usuario, sea informado y actualice el software para suplir la vulnerabilidad.

La funcionalidad mas potente del software es sin duda el uso de la potencia de YARA, a través de la cual podemos escanear todos los procesos que se encuentran en ejecución, pudiendo encontrar malware oculto en el sistema en forma de programas conocidos, así como los ficheros del sistema. La gran ventaja de YARA radica en que se usa una base de reglas que se actualiza constantemente, pero también el usuario podría crear sus propias reglas dotando a esta función de una enorme flexibilidad.

En pocas palabras, la ciberseguridad y sobre todo el tratamiento y reconocimiento de anomalías y amenazas es un tema siempre en boca de todos debido a que todo el mundo conoce los riesgos de la computación moderna. Por todo ello pienso que el desarrollo e investigación de este trabajo es bastante satisfactorio sobretodo, por que da al usuario corriente la posibilidad sin mucho esfuerzo de estar informado acerca del estado de su dispositivo de la forma mas detallada posible para que sea entendible por ellos. De ningún modo este desarrollo ha pretendido convertirse en una herramienta de análisis forense pues, para combatir el malware, y en general las amenazas a las que cada día los usuarios somos mas vulnerables, existen otras herramientas mucho mas potentes y especificas para gente del sector.

Para finalizar, decir que la ciberseguridad es un campo del conocimiento muy amplio, y que debido a que existe mucha gente creando, desarrollando malware, y explotando vulnerabilidades, requiere de un gran esfuerzo personal, ya que lo que puedes ver hoy, mañana seguramente habrá mutado y por tanto debemos esforzarnos constantemente por estar al tanto de todo lo que ocurre.

9. Trabajo futuro

Gracias al desarrollo de una **API** en la parte del servidor, la escalabilidad del software es bastante buena, pues facilita la explotación de la misma puediendo crear clientes para otros dispositivos no incluidos en este trabajo tales como la plataforma IOS o simplemente clientes de escritorio.

Me gustaría resaltar que existen numerosos estudios que hablan acerca de la detección del malware mediante el uso de redes neuronales convolucionales y machine learning. Hubiese sido un punto interesante que haber abordado en este trabajo, sin embargo, y debido a que no todos los dispositivos tienen la capacidad de hacer cálculos que pueden llegar a ser muy complejos dependiendo de lo que se quisiese analizar se ha decidió no incluir.

También se podrían mejorar y optimizar las funciones implementadas por el software, corrigiendo y subsanando errores, comentarios y en definitiva mejoras expuestas en la sección 7.2

Con un mayor tiempo de desarrollo se hubiese posibilitado el añadir notificaciones a la aplicación cliente de forma que la aplicación pudiese ejecutar tareas periódicas informando de los resultados, en caso de ser peligrosos, al usuario.

Finalmente debido al campo de investigación del presente trabajo, la escalabilidad de la aplicación puede orientarse en muchos caminos pudiendo dar lugar a una herramienta mucho mas potente y aprovechando todo el potencial de la informática moderna.

10. Glosario

Siglas:

API *Application Programming Interface*. 6, 53

JSON *JavaScript Object Notation*. 40

MBR *resgitro de arranque maestro*. 11

MVC *Modelo Vista Controlador*. 64

SVM *Support Vector Machine*. 70

TLS *Transport Layer Security*. 40

UPnP *Universal Plug and Play*. 20

APK *Android Application Package*. 120

CERT *Computer Emergency Response Team*. 16

CVSS *Common Vulnerability Score System*. 17

FSF *Free Software Foundation*. 140

NIST *National Institute of Standards and Technology*. 21

NVD *National Vulnerability Database*. 77

PID *process ID*. 20

Nomenclatura:

API conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. 6, 53, 105

bug error o un defecto en el software o hardware que hace que un programa funcione incorrectamente. 15

certificados X.509 estándar UIT-T para infraestructuras de claves públicas. X.509 especifica, entre otras cosas, formatos estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación. 42

exploit fragmento de software, datos, comandos o acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado. 4

firewall parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. 37

JSON formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente. 40, 105

machine learning subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. De forma más concreta, se trata de crear programas capaces

de generalizar comportamientos a partir de una información suministrada en forma de ejemplos.. 20

MBR primer sector de un dispositivo de almacenamiento de datos, como un disco duro.. 11, 105

MVC arquitectura software que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.. 64, 105

patrón singleton patrón de diseño que permite restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.. 44

pcap interfaz de una aplicación de programación para captura de paquetes.. 43

SCRUM nombre con el que se denomina a los marcos de desarrollo ágiles caracterizados por adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto, basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados y por el solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada.. 19

Sniffer programa de captura de las tramas de una red de computadoras.. 75

SVM modelos de aprendizaje supervisado que se asocian con algoritmos que analizan datos para su clasificación, regresión y análisis.. 70, 105

TLS protocolo criptográfico que proporciona comunicaciones seguras a través de una red, comunmente internet. 40, 105

UPnP conjunto de protocolos de comunicación que permite a periféricos en red, como computadoras personales, impresoras, pasarelas de Internet, puntos de acceso Wi-Fi y dispositivos móviles, descubrir de manera transparente la presencia de otros dispositivos en la red y establecer servicios de red de comunicación, compartición de datos y entretenimiento. 20, 105

YARA herramienta que ayuda a la búsqueda y detección de malware así como su clasificación a través de descripciones malware basadas en patrones textuales o binarios.. 47

zero day vulnerabilidad para la cual no se crearon parches o revisiones y que se emplea para llevar a cabo un ataque. 17

11. Bibliografía

[1] Abien Fred M. Agarap. *A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification*. https://arxiv.org/pdf/1801.00318.pdf (último acceso 13 Febrero 2019).

[2] Ciclodevida. *Ciclo de vida del software*. http://ciclodevida.net/del-software (último acceso 14 Octubre 2018), 2018.

[3] CSIRT. *Herramientas escaneadoras y detectoras de vulnerabilidades*. https://www.csirtcv.gva.es/es/paginas/herramientas-escaneadoras-y-detectoras-de-vulnerabilidades.html (último acceso 13 Octubre 2018), 2018.

[4] Andrea Fortuna. *Finding malware on memory dumps using Volatility and Yara rules*. https://www.andreafortuna.org/dfir/finding-malware-on-memory-dumps-using-volatility-and-yara-rules/ (último acceso 6 Marzo 2019), 2018.

[5] Wireshark Foundation. *Go Deep*. https://www.wireshark.org/ (último acceso 28 Octubre 2018), 2019.

[6] freebsd.org. *PFCTL documentation*. https://www.freebsd.org/cgi/man.cgi?query=pfctl(8)&sektion= (último acceso 1 Marzo 2019).

[7] Google. *Material design*. https://material.io/design/ (último acceso 3 Febrero 2019).

[8] Heimdal. *How Every Cyber Attack Works – A Full List*. https://heimdalsecurity.com/blog/cyber-attack/ (último acceso 16 Febrero 2019), 2017.

[9] Iptables. *Man page of IPTABLES*. http://ipset.netfilter.org/iptables.man.html (último acceso 27 Diciembre 2018).

[10] Karpesky. *CIBERAMENAZA MAPA EN TIEMPO REAL*. https://cybermap.kaspersky.com/es/stats (último acceso 22 Marzo 2019), 2019.

[11] Microsoft. *How to use the "netsh advfirewall firewall"*. https://support.microsoft.com/en-us/help/947709/how-to-use-the-netsh-advfirewall-firewall-context-instead-of-the-netsh (último acceso 10 Noviembre 2018).

[12] Yuval Nativ. *the zoo*. https://github.com/ytisf/theZoo (último acceso 9 Noviembre 2018).

[13] Optical networks. *Tipos de ataques informáticos y previsiones para el 2019*. https://www.optical.pe/tipos-de-ataques-informaticos-y-previsiones-para-el-2019/ (último acceso 16 Febrero 2019), 2018.

[14] NIST. *Technical Guide to Information Security Testing and Assessment, NIST Special Publication 800-115*. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf (último acceso 12 Abril 2019), 2008.

[15] NIST. *Assessing Security and Privacy Controls in Federal Information Systems and Organizations, Building Effective Assessment Plans, NIST Special Publication 800-53A (revision 4),*. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-53ar4.pdf (último acceso 12 Abril 2019), 2014.

[16] Pypcap. *analisis de paquetes de red*. https://pypcap.readthedocs.io/en/latest/ (último acceso 19 Febrero 2019).

[17] scikit learn. *Herramienta de machine learning para python*. https://scikit-learn.org/stable/documentation.html (último acceso 27 Febrero 2019).

[18] SecureList. *Informe KSN: Ransomware en 2016-2017*. https://securelist.lat/ksn-report-ransomware-in-2016-2017/85207/ (último acceso 11 Abril 2019), 2017.

[19] Offensive security. *Exploits for Penetration Testers*. https://www.exploit-db.com/ (último acceso 16 Diciembre 2019), 2019.

[20] Tecmint. *5 Best Linux Package Managers*. https://www.tecmint.com/linux-package-managers/ (último acceso 7 Enero 2019), 2016.

[21] Read the docs. *documentación de librerias*. https://readthedocs.org/ (último acceso 23 Marzo 2019).

[22] VirusTotal. *Herramienta de detección de malware*. https://github.com/VirusTotal/yara-python (último acceso 2 Marzo 2019), 2016.

[23] Vulners. *Vulnerability Data Base*. https://vulners.com/ (último acceso 14 Diciembre 2018), 2019.

[24] Welivesecurity. *Un año después: el exploit EternalBlue registra mayor actividad ahora que durante el brote de WannaCryptor*. https://bit.ly/2KPqelz (último acceso 23 Marzo 2019), 2018.

[25] YARA. *The pattern matching swiss knife for malware researchers*. https://virustotal.github.io/yara/ (último acceso 8 Enero 2019).

[26] Miguel Ángel Mendoza. *Vulnerabilities reached a historic peak in 2017*. https://www.welivesecurity.com/2018/02/05/vulnerabilities-reached-historic-peak-2017/ (último acceso 13 Octubre 2018), 2018.

12. Anexo I. Atribuciones

12.1. Dependencias y paquetes

12.1.1. Paquetes python:

- upnpclient https://github.com/flyte/upnpclient
- wxpython https://wxpython.org/
- ifaddr https://github.com/pydron/ifaddr
- hurry.filesize https://pypi.org/project/hurry.filesize/
- dpkt https://pypi.org/project/dpkt/
- requests http://docs.python-requests.org/en/master/
- psutil https://github.com/giampaolo/psutil
- pcap-ct https://pypi.org/project/pcap-ct/
- pypcap https://github.com/pynetwork/pypcap
- distro https://github.com/nir0s/distro
- netifaces https://github.com/al45tair/netifaces
- vulners https://github.com/vulnersCom/api
- sqlalchemy https://www.sqlalchemy.org/
- scikit-learn https://scikit-learn.org/stable/

- lsanomaly https://github.com/lsanomaly/lsanomaly
- numpy https://www.numpy.org/
- GitPython https://github.com/gitpython-developers/GitPython
- yara-python https://github.com/VirusTotal/yara-python

12.1.2. Dependencias gradle:

- Conscrypt https://github.com/google/conscrypt
- Spinkit https://github.com/ybq/Android-SpinKit
- Tree View https://github.com/bmelnychuk/AndroidTreeView
- Graph View http://www.android-graphview.org/

12.2. Iconos e imágenes

- https://www.flaticon.com/authors/smashicons
- https://www.flaticon.com/authors/freepik
- https://www.flaticon.com/authors/smalllikeart
- https://www.flaticon.com/authors/roundicons
- https://www.flaticon.com/authors/vectors-market
- https://www.flaticon.com/authors/surang
- https://www.flaticon.com/authors/pixel-perfect
- https://www.flaticon.com/authors/pixel-buddha

13. Anexo II. Manual de usuario e instalación

Contenido:

13.1. Instalación del servidor I. Prerrequisitos

Se necesita un dispositivo de escritorio con alguno de los sistemas operativos soportados y tener instalado python 3.5+, pip y git.

13.1.1. Prerrequisitos Linux

Distribuciones basadas en Debian

Se necesita instalar las siguientes dependencias:

1. python-dev
2. gtk (preferiblemente la versión 3)
3. gstreamer
4. gstreamer-plugins-base
5. glut
6. libwebkitgtk
7. libjpeg
8. libpng
9. libtiff
10. libsdl
11. libnotify
12. libsm
13. libpcap-dev

Para ello ejecutamos en una consola como superusuario los siguientes comandos:

```
1. sudo apt install make gcc libgtk-3-dev libwebkitgtk-dev libwebkitgtk-3.0-dev libgstreamer-gl1.0-0 freeglut3 freeglut3-dev python-gst-1.0 python3-gst-1.0 libglib2.0-dev ubuntu-restricted-extras libgstreamer-plugins-base1.0-dev

2. sudo apt install libpcap-dev
```

Ahora podemos instalar WxPython ejecutando:

```
1. pip install wxpython
```

Distribuciones basadas en Arch linux

Ejecutamos en una consola los siguientes comandos:

```
1. pacman -S libpcap
2. pacman -S python-wxpython
```

13.1.2. Prerrequisitos Windows

Se necesita tener PCAP instalado en el sistema, para ello es posible descargarlo del siguiente enlace:

```
1. https://nmap.org/npcap/dist/npcap-0.99-r7.exe
```

13.2. Instalación del servidor II. Instalar

Clonamos el repositorio haciendo uso de git

```
1. git clone https://github.com/alvarogf97/Audit
```

Nos movemos a la carpeta donde se ha clonado el repositorio y ejecutamos

```
1. pip install -r requirements.txt
```

Finalmente ejecutamos la aplicación

```
1. python start.py
```

13.3. Instalación del servidor III. Compilar

Es posible compilar la aplicación y generar un ejecutable portable para el sistema operativo en el que se haga la susodicha tarea. Para ello instalamos **pyinstaller**

```
1. pip install pyinstaller
```

Una vez instalado, en la carpeta donde se ha clonado el repositorio nos dirigimos a una carpeta cuyo nombre es **“builder”** y dependiendo de nuestro sistema operativo ejecutamos el shell script que encontramos. Se creará una carpeta **“dist”** donde podremos encontrar el ejecutable.

13.4. Instalación del cliente

El cliente esta programado para ser ejecutado en la plataforma android. Para poder instalarlo necesitamos o bien compilar nosotros mismos la *Android Application Package* (APK) o bien descargar la apk ya compilada e instalarlo en el dispositivo android como una aplicación cualquiera.

13.4.1. Compilar APK

Primero necesitamos clonar el repositorio. Para ello ejecutamos

1. git clone `https://github.com/alvarogf97/Client_Audit`

Usando Gradle

Necesitamos tener instaladas las siguientes herramientas:

1. Android SDKManager
2. Gradle
3. SDK de la versión a la que vamos a compilar

Nos movemos a la carpeta donde hemos clonado el repositorio y ejecutamos el siguiente comando

1. gradle assemble

Nos creara una carpeta en la que podremos encontrar la aplicación compilada.

Usando android studio

Abrimos el proyecto clonado usando android studio. A continuación seguimos los siguientes pasos

1. Click en el menú desplegable de la barra de herramientas
2. Seleccionamos **“Editar configuración”**
3. Click sobre el símbolo **“+”**
4. Seleccionamos **“Gradle”**
5. Elegimos el módulo como proyecto de Gradle
6. En **“Tareas”** seleccionamos **“assemble”**
7. Pulsamos el boton de **“Run”**

Se habrá creado una carpeta en la ruta ProjectName\app\build\outputs\apk donde encontraremos la apk.

13.4.2. Descargar APK compilada

Podemos encontrar la APK en el apartado **“release”** del repositorio donde se encuentra publicado el código de la misma. El enlace es el siguiente

1. `https://github.com/alvarogf97/Client_Audit/releases`

13.5. Uso del servidor

13.5.1. Creando un usuario en el servidor

Pulsaremos sobre el botón **“Users”** *(figura 39)*. Una vez hemos llegado a la interfaz de usuarios *(figura 40)* procederemos a rellenar el formulario con los datos del nuevo usuario y posteriormente pulsaremos sobre el botón **“Create”**. Finalmente en el cuadro de diálogo nos aparecerá un mensaje que nos informará si el usuario a podido ser creado o si ya existe.

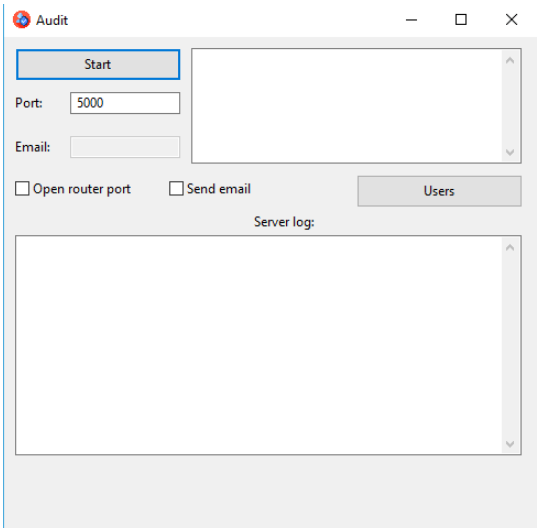

Figura 39: Ventana principal

13.5.2. Eliminando un usuario en el servidor

En la ventana Usuarios *(figura 40)* rellenamos el formulario con un usuario existente que queramos eliminar y pulsamos sobre el botón **“Delete”**.

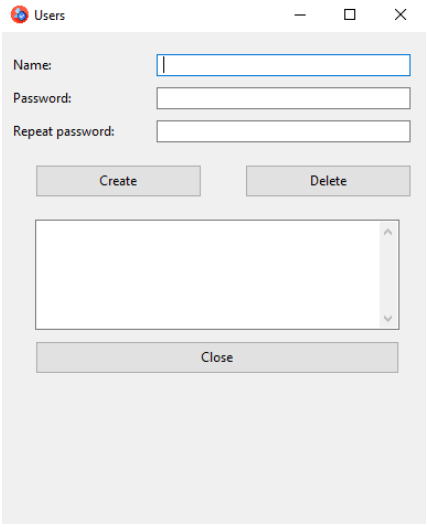

Figura 40: Ventana de Usuarios

13.5.3. Encendiendo el servidor

En la ventana principal *(figura 39)* escribiremos el puerto en el que queremos que se inicie el servidor, y escribiremos el correo electrónico al que queramos enviar la información para la conexión si hemos marcado la casilla correspondiente. Además podemos marcar la casilla **“Open router port”** que abrirá la conexión hacia el exterior si es posible. Una vez realizado todo lo anterior pulsamos sobre el botón **“Start”** que iniciará las comprobaciones pertinentes y generará si es necesario el sistema de ficheros y posteriormente nos informará de que el servidor se encuentra encendido así como sus datos de conexión. Para parar el servidor pulsaremos el mismo botón que para iniciarlo que ahora tendrá la etiqueta **“Stop”**.

13.6. Uso del cliente

13.6.1. Creando un nuevo dispositivo

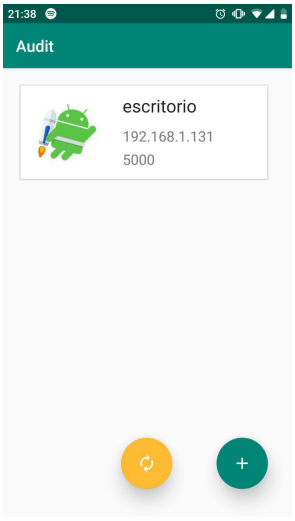

Figura 41: Actividad principal

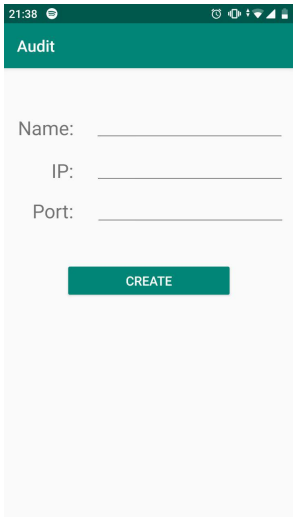

Figura 42: Actividad creación dispositivo

Partimos de la ventana principal *(figura 41)*. A continuación pulsamos sobre el botón **“+”** en la esquina inferior derecha lo que nos llevará a la ventana para la creación de un nuevo dispositivo *(figura 42)*. Finalmente rellenamos el formulario y pulsamos sobre el botón **“create”** que, si los datos son correctos nos devolverá a la actividad principal con el nuevo dispositivo correctamente creado.

13.6.2. Eliminando un dispositivo

Desde la ventana principal *(figura 41)*, pulsamos sobre cualquiera de los dispositivos creados para acceder a la ventana de gestión del dispositivo *(figura 43)* en la cual pulsaremos sobre el botón de color rojo con el símbolo de una papelera. Una vez pulsado volveremos a la ventana de principal y habremos borrado correctamente el dispositivo.

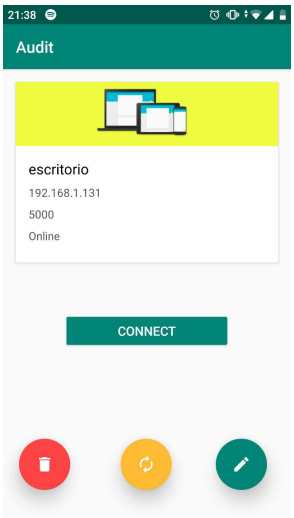

Figura 43: Actividad dispositivo

13.6.3. Editando un dispositivo

En la ventana de gestión del dispositivo *(figura 43)* pulsaremos sobre el botón de color verde de la esquina inferior derecha y con el símbolo de un lápiz. A continuación nos encontraremos en la ventana para la edición con el mismo formulario que en la ventana *(figura 42)*. Finalmente rellenamos los datos y pulsaremos sobre el botón **“update”**.

13.6.4. Accediendo a un dispositivo

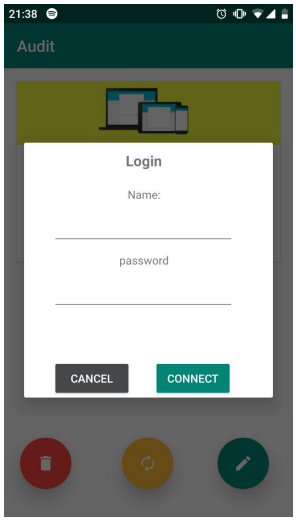

Figura 44: Actividad dispositivo

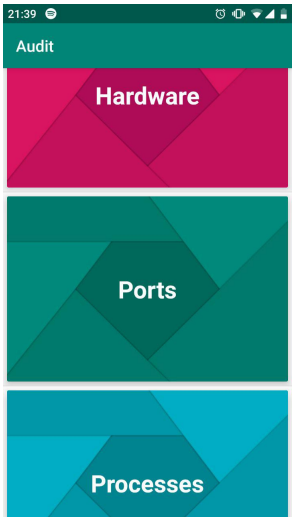

Figura 45: Actividad menú dispositivo

Una vez nos encontramos en la ventana de gestión del dispositivo *(figura 43)* pulsaremos sobre el botón **"connect"** y se nos abrirá un diálogo en el que debemos introducir las credenciales *(figura 44)*. Finalmente pulsamos sobre el botón **"connect"** y si las credenciales introducidas son válidas accederemos al menú del dispositivo *(figura 45)*.

13.6.5. Usando la función “Hardware”

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“Hardware”** que nos conducirá a la ventana de visualización de los componentes y estado de los mismos que posee el dispositivo *(figura 46)*. Una vez en esta ventana, podemos pulsar sobre cualesquiera de los parámetros para obtener información acerca de ellos.

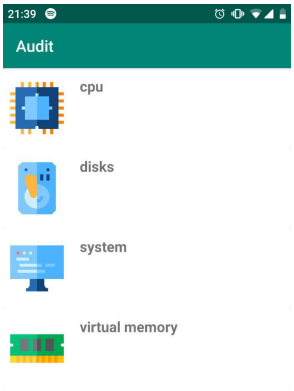

Figura 46: Actividad hardware

13.6.6. Usando la función “Ports”

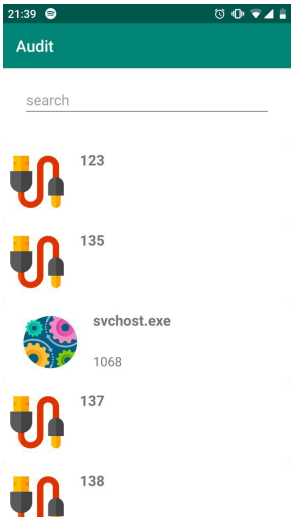

Figura 47: Actividad Ports

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“Ports”** que nos conducirá a la ventana de visualización de los servicios abiertos en el dispositivo *(figura 47)*. Una vez en esta ventana, podemos pulsar sobre cualesquiera de los puertos encontrados para conocer que proceso es el que esta usando el susodicho puerto, pudiendo matar el procesos si pulsamos sobre él. También podemos mediante la barra de búsqueda encontrar un puerto que empiece por tal numeración.

13.6.7. Usando la función “Processes”

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“Processes”** que nos conducirá a la ventana de visualización de los procesos que se encuentran actualmente en ejecución el dispositivo *(figura 48)*. En esta ventana podremos usar la barra de búsqueda para encontrar procesos, o pulsar sobre cualesquiera de los procesos para matarlo.

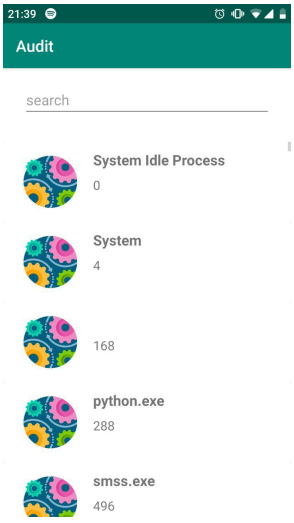

Figura 48: Actividad Processes

13.6.8. Usando la función “Network”

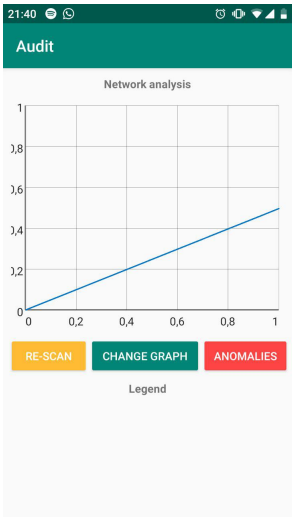

Figura 49: Actividad Network

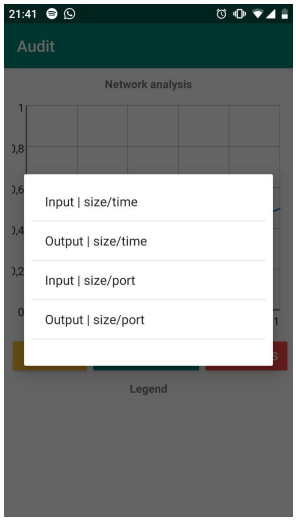

Figura 50: Cambiando gráfico Network

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“Network”** que nos conducirá a la ventana del análisis de red *(figura 49)*. En esta ventana podremos visualizar el resultado del escáner de los paquetes de la red pudiendo, cambiar el tipo de visualización eligiendo entre las diferentes formas *(figura 50)* al puslar sobre el botón **“change graph”**.

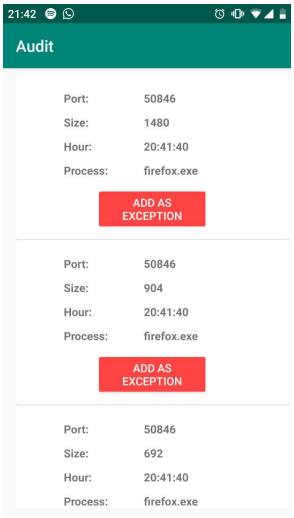

Figura 51: Actividad Network Anomalies

Si pulsamos sobre **“anomalies”** abriremos una ventana en la que podremos ver si existen anomalías los paquetes escaneados *(figura 51)* pudiendo añadir las posibles anomalías como excepciones al pulsar sobre el botón **“add as exception”** para que el programa vaya aprendiendo de nuestro entorno.

13.6.9. Usando la función “Upnp devices”

Figura 52: Actividad Upnp devices

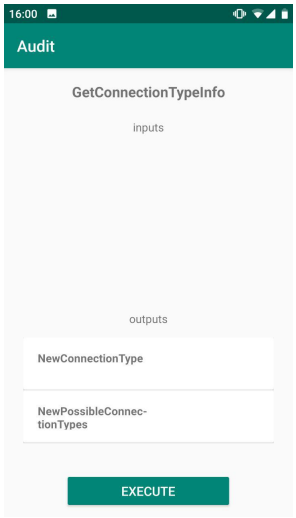

Figura 53: Actividad Upnp action

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“Upnp devices”** que nos conducirá a la ventana de los dispositivos upnp *(figura 52)*. En esta ventana podremos visualizar los diferentes dispositivos de nuestra red que hacen uso de este protocolo. Además al pulsar sobre ellos desplegaremos la lista de servicios y acciones que nos ofertan. Si pulsamos sobre alguna de las acciones abriremos la ventana de ejecución de la acción *(figura 53)* en la cual podremos rellenar los datos para el correcto funcionamiento de la misma, si es que posee, para finalmente pulsar sobre el botón **“execute”** que invocará la acción en el dispositivo y devolverá en el cuadro de diálogo de la parte inferior los atributos resultados.

13.6.10. Usando la función “Firewall”

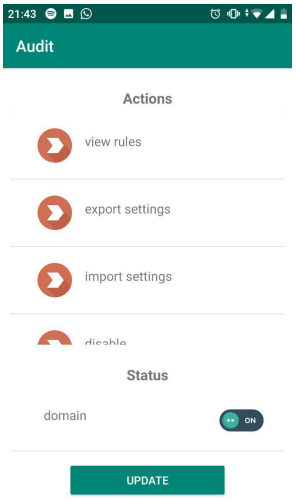

Figura 54: Actividad Firewall

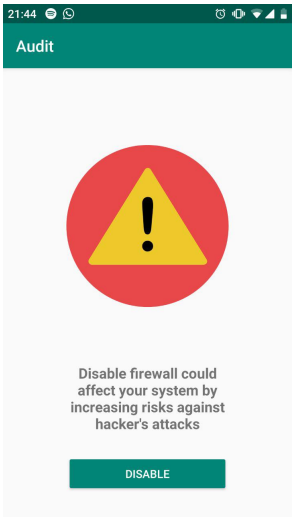

Figura 55: Actividad Firewall desactivar

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“Firewall”** que nos conducirá a la ventana principal del firewall del dispositivo *(figura 54)*. En esta ventana podemos ver una lista con las acciones disponibles y en la parte inferior de la misma el estado actual del firewall, el cual podemos activar si pulsamos sobre la acción **“enable”** y desactivar. Para esto último pulsaremos la opción **“disable”** que nos moverá a una nueva ventana *(figura 55)*, que nos indicará el riesgo que conlleva la realización de esa acción y tras pulsar el botón **“disable”** volveremos a la venta anterior comprobando que el estado del firewall ha sido cambiado.

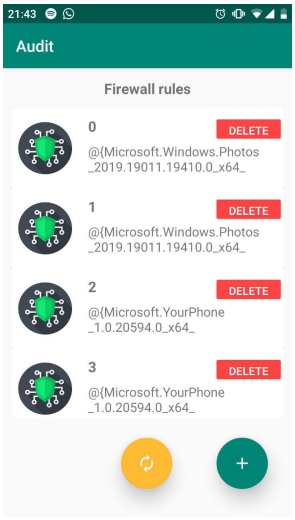

Figura 56: Actividad Firewall reglas

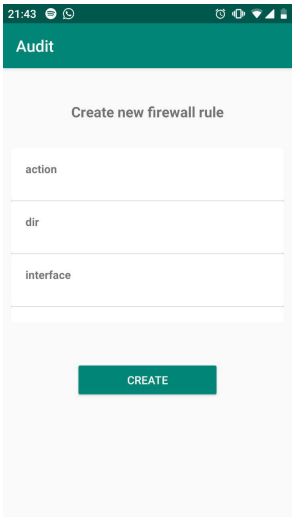

Figura 57: Actividad Firewall crear regla

Si seleccionamos la opción **“view rules”** abriremos una nueva ventana *(figura 56)* en la que encontraremos todas las reglas que existen en el firewall del dispositivo. Podemos eliminar estas reglas si pulsamos sobre el botón **“delete”** de la susodicha regla. Además si pulsamos sobre el botón verde con el símbolo **“+”** de la esquina inferior derecha, abriremos una ventana *(figura 57)* en la que podremos crear una nueva regla y añadirla al dispositivo, para lo cual rellenamos las casillas del formulario que necesitemos y pulsamos sobre el botón **“create”**.

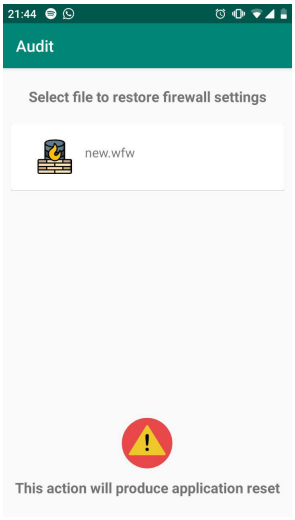

Figura 58: Actividad Firewall importar

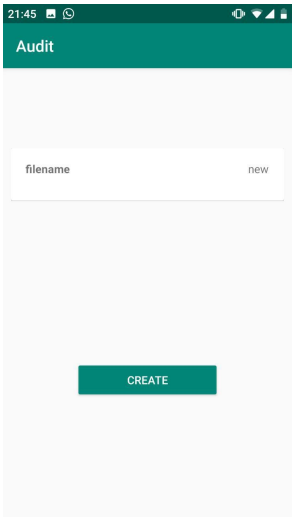

Figura 59: Actividad Firewall exportar

Podremos guardar copias de seguridad del estado actual del firewall si seleccionamos la opción **"export settings"** que nos abrirá una nueva ventana *(figura 59)*. En esta nueva ventana escribiremos el nombre con el que queremos guardar el fichero donde se almacenará el estado y pulsamos sobre el botón **"create"** que generará el fichero si no existe otro fichero con ese nombre en las copias de seguridad del firewall. Así mismo, también podremos importar las configuraciones del firewall que hayamos guardado seleccionado la opción **"import settings"**. Tras seleccionar esta opción se abrirá una ventana en la que podremos seleccionar las copias anteriores que hayamos guardado *(figura 58)*. Para esto último solo tenemos que tocar la copia que queremos restaurar y posteriormente se reiniciará la aplicación para que el cambio surja efecto.

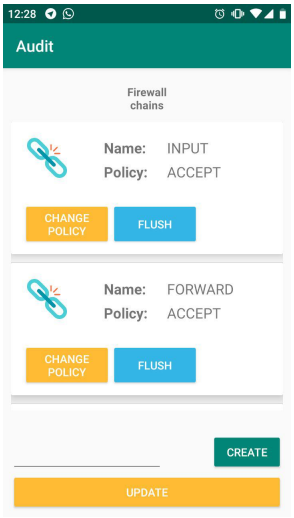

Figura 60: Actividad Firewall chains

Finalmente, si nuestro dispositivo usa un sistema operativo Linux, podremos pulsar sobre la opción **“view chains”** que nos conducirá a una actividad *(figura 60)* en la cual, podremos crear nuevas cadenas, eliminar las cadenas creadas, cambiar la política de las cadenas por defecto del sistema y vaciar las reglas de todas las cadenas. Para realizar todas estas funciones solo tenemos que pulsar el botón correspondiente a esa acción.

13.6.11. Usando la función “Vulnerabilities”

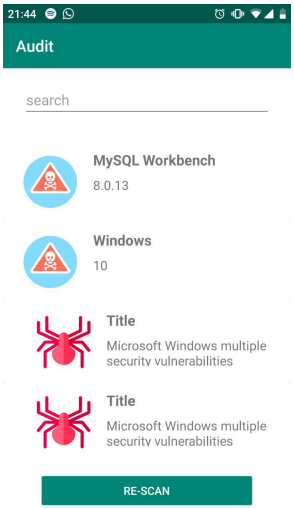

Figura 61: Actividad Vulnerabilities

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“Vulnerabilities”** que nos conducirá a una ventana *(figura 61)*, en la que podremos observar las vulnerabilidades que posee nuestro dispositivo en cuanto a las aplicaciones instaladas en el mismo. Podemos buscar aplicaciones vulnerables mediante el buscador de la parte superior. Además podemos pulsar sobre los elementos para obtener información mas detalladas de estas vulnerabilidades.

13.6.12. Usando la función “YARA scan”

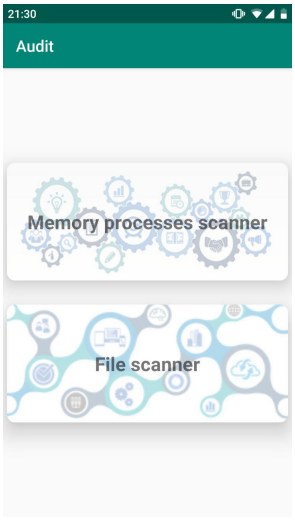

Figura 62: Actividad YARA scan

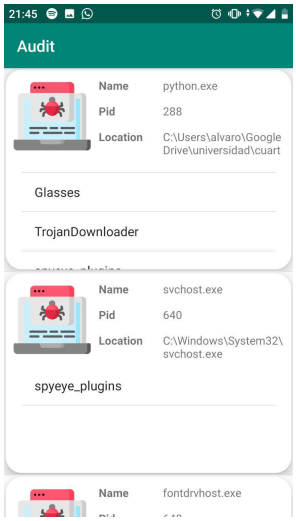

Figura 63: Actividad YARA results

Desde la actividad de menú del dispositivo *(figura 45)*, pulsaremos sobre la opción **“YARA scan”** que abrirá una nueva ventana *(figura 62)*, en la que tenemos dos opciones disponibles. Si pulsamos sobre la opción **“memory processes scanner”**, se lanzará un escáner que analizará los programas en ejecución del dispositivo durante un corto periodo de tiempo y abrirá una ventana *(figura 63)*, mediante la cual podremos observar el resultado del escáner.

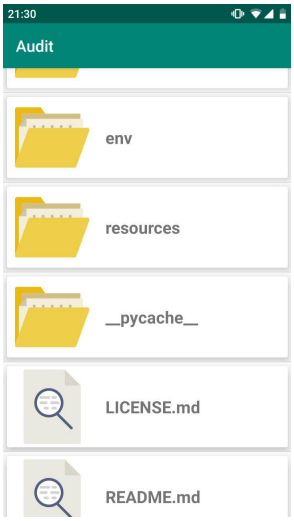

Figura 64: Actividad File scanner

Si elegimos la opción **“file scanner”**, abriremos el explorador de archivos del dispositivo *(figura 64)*. En este explorador podremos pulsar sobre los directorios para movernos por todo el sistema de ficheros y además, si pulsamos de forma prolongada sobre un fichero o directorio, se lanzará un escáner sobre los ficheros en cuestión, indicándonos cuando acabe, el resultado del mismo.

13.7. Licencia

Ambas aplicaciones están públicas en repositorios de **github**. En estos repositorios podemos encontrar los ficheros de licencia **“LICENSE.md”**.

La aplicación de escritorio esta disponible en el repositorio

```
1. https://github.com/alvarogf97/Audit
```

Esta aplicación esta licenciada bajo la **GPLv3** ofrecida por la *Free Software Foundation* (FSF) y cuyas condiciones pueden consultarse en el fichero anteriormente nombrado en el susodicho repositorio.

La aplicación móvil esta disponible en el repositorio

```
1. https://github.com/alvarogf97/Client_Audit
```

Esta aplicación esta licenciada bajo la licencia **MIT** y cuyas condiciones pueden consultarse también en el fichero anteriormente nombrado en el susodicho repositorio.