UNIVERSITAT DE BARCELONA

Trabajo de fin de grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

SERVIDOR ANTI-PHISHING

Autor: Arnau Gris Garcia

Tutor: Raúl Roca Cánovas
Realitzat a: Departamento de Matemáticas e Informática

Barcelona, 13 de junio de 2022

Resumen

En la actualidad, el phishing es uno de los ataques más realizados por el cibercrimen para extraer datos de forma maliciosa. Existen algunas empresas que ofrecen algún servicio como defensa para el phishing, pero el campo de las contramedidas contra el phishing aún está por explotar.

Por eso, en este proyecto se propone desarrollar un servicio que sea capaz de hacer que sea más difícil caer en un ataque de phishing mediante el análisis de los mensajes de correos y sus direcciones URL.

Resum

En l'actualitat el phishing és un dels atacs més realitzats pels ciberdelinqüents per extraure dades d'usuaris mail de forma maliciosa: Existeixen algunes empreses que implementen algun servei com a defensa al phishing, però aquest camp està encara per explotar.

És per aquest motiu que aquest projecte proposa desenvolupar un servei que sigui capaç de fer més difícil que un usuari caigui en atac de phishing, això es graciés a anàlisis dels missatges de correu i les seves adreces URL.

Abstract

Currently, phishing is one of the most common attacks carried out by cybercrime to maliciously extract data from email users. There are some companies that offer some services as defense against phishing, but the field of countermeasures against phishing is still undeveloped.

That is why this project proposes to develop a service that is capable of making it more difficult to fall into a phishing attack by analyzing email messages and their URL addresses.

Agradecimientos

“A mis padres y a toda mi familia, que siempre me han apoyado en los estudios y han hecho todo lo posible porque no me faltara de nada.”

“A mi pareja que me ha apoyado incondicionalmente en todos los momentos.”

“A mis compañeros del grado que han sido un pilar fundamental en esta experiencia universitaria, por todo lo que me han enseñado y todos los momentos que hemos compartido.”

“A mi tutor del TFG por orientarme y ayudarme en todo lo que ha sido posible.”

Índice

1. Introducción

Contexto

Hoy en día uno de los ataques más realizados en el ámbito del cibercrimen es el phishing, debido a la facilidad con la que se puede realizar este tipo de ataques y el gran potencial que tienen a la hora de perjudicar a un usuario. Este proyecto intenta reforzar la seguridad de los usuarios de correo frente al phishing mediante un servicio añadido al mismo correo.

Motivación

Durante mi estancia en el grado de ingeniería informática se me han presentado varias ramas, pero siempre me ha interesado el campo de la ciberseguridad, por ese motivo decidí contactar con mi tutor, Raúl, y realizar un proyecto relacionado con este tema. Al principio del proyecto no tenía muy clara la idea que iba a desarrollar en este proyecto, pero con la ayuda de mi tutor, encontramos una idea que me gustó y decidí investigar para ver si me podía gustar realmente. Finalmente, tras unos días, decidí que esta idea era muy interesante y que me gustaría trabajar en ella para el trabajo de fin de grado.

Otra gran motivación a la hora de ejecutar esta idea es la utilidad de este proyecto en nuestro día a día, ya que la seguridad de nuestros clientes mail es muy importante, puesto que cada día se envían millones de correos, por lo cual encuentro interesante el uso de este proyecto. También me motiva la experiencia que me aporta efectuar un proyecto de esta magnitud.

Objetivo principal

El objetivo principal de este trabajo de fin de grado consiste en crear una utilidad que ayude a los usuarios de correo a identificar si un correo puede ser considerado fraudulento o no, y así poder mejorar la seguridad frente a ataques de phishing.

Objetivos del proyecto

- Aprender como funciona el servicio mail.
- Investigar como desarrollar un servidor SMTP.
- Aprender en profundidad que es el phishing y como se utiliza para cometer cibercrimen.
- Adquirir experiencia en configuración de servidores remotos.
- Comprender los aspectos básicos de como funciona un firewall.
- Investigar en el tema de los patrones de diseño para realizar un código más limpio.
- Investigar como montar un servidor TCP en Java.

Funcionalidades del proyecto

- **Comunicación SMTP:** el servidor debe poder realizar una comunicación SMTP con el cliente, a la vez ha de extraer todos los datos del correo recibido.
- **Extractor de URL y dominios:** el servidor debe ser capaz de extraer todas las URL y sus correspondientes dominios de los correos recibidos.
- **Detector de URL escondidas:** el servidor debe ser capaz de detectar URL que no se muestran a primera vista en el correo, como las referencias HTML href.
- **Comprobador de blacklist y sistemas de reputación:** el servidor debe comprobar si las URL del correo se encuentran en alguna playlist o sistema de reputación.
- **Comprobador de recortadores:** el servidor ha de ser capaz de detectar si se ha utilizado algún sistema de recorte de URL.
- **Extractor de redirecciones:** el servidor ha de ser capaz de detectar si alguna URL redirige a otra, igual de alargar las URL recortadas.
- **Comprobador de dominios similares:** el servidor ha de incluir una funcionalidad que permita verificar si 2 dominios son similares, los dominios a revisar son establecidos por el usuario.
- **Detector de URL peligrosas:** el usuario establece dominios que no son de su interes y el servidor ha de detectar si se incluye alguno de estos dominios en el correo.
- **Generado de reporte:** el servidor ha de generar un reporte con todas las comprobaciones y enviarlo por correo al usuario que ha realizado la petición al servidor.

Planificación

La planificación de este proyecto se ha dividido en 3 grandes bloques: investigación, diseño e implementación, y redacción de la memoria. A continuación se muestra el diagrama de Gantt utilizado en esta planificación en la **Figura 1** y la **Figura 2**.

0	Investigación conceptos básicos ciberseguridad	DarkDhz	-	01/Nov	08/Jan
1	Investigación sobre protocolos mail y phishing	DarkDhz	-	08/Jan	01/Feb
2	Investigación de tecnologías para el proyecto	DarkDhz	-	01/Feb	22/Feb
3	Diseño de la aplicación	DarkDhz	-	22/Feb	03/Mar
4	Adquisición e instalación del entorno	DarkDhz	-	03/Mar	20/Mar
5	Desarrollo del proyecto	DarkDhz	-	20/Mar	29/May
6	Redacción de la memoria	DarkDhz	-	29/May	13/Jun

Figura 1: Tabla de planificación Gantt

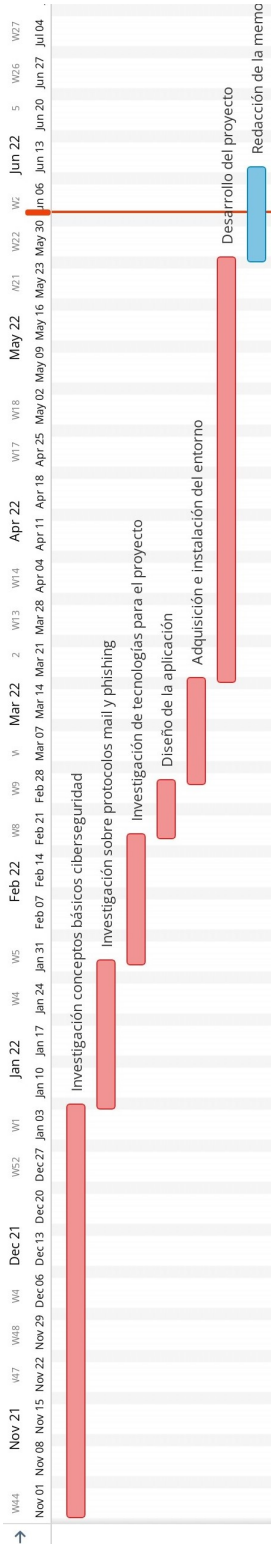

Figura 2: Diagrama de Gantt

Organización del documento

El documento se estructura en diferentes secciones:

- **Introducción:** En esta sección, como su propio nombre indica, se presenta una pequeña introducción del proyecto, tanto los objetivos de este como su planificación.
- **Tecnologías y herramientas utilizadas:** En esta sección se indican las tecnologías y herramientas empleadas para el desarrollo del proyecto.
- **Marco teórico:** En esta sección se introducen los conceptos básicos con gran relación con el proyecto.
- **Análisis:** En análisis se introduce la investigación realizada sobre la viabilidad de este proyecto.
- **Diseño:** En esta sección se expone el proceso de diseño del proyecto.
- **Implementación:** En implementación se explica con detalle como se ha implementado este proyecto.
- **Simulación de un caso:** En simulación de un caso se intenta demostrar con ejemplos la funcionalidad de esta aplicación.
- **Conclusiones y trabajo futuro:** En esta sección se exponen las conclusiones e ideas de ampliación del proyecto en el futuro.
- **Manual del proyecto:** En esta sección se explica detalladamente como instalar y configurar la aplicación paso a paso.
- **Apéndice:** Por último, en el apéndice se introducen conceptos teóricos relacionados con el proyecto.

2. Tecnologías y herramientas utilizadas

A continuación se introducen de forma resumida las herramientas y tecnologías empleadas para el desarrollo del proyecto.

2.1. Lenguajes de programacion

Java 8

Como lenguaje de programación para el proyecto se ha decidido emplear Java [1], esto es debido a la gran experiencia con Java por el desarrollador. También se ha decidido usar Java, ya que este permite compilar el mismo código en diferentes sistemas operativos gracias a la JVM (java virtual machine).

En una primera instancia el proyecto se empezó a desarrollar en Python 3, pero debido a los problemas de concurrencia que existen actualmente en Python se acabó cambiando a Java.

Para instalar java en Debian se pueden utilizar los siguientes comandos:

```
sudo apt-get update
sudo apt install -y default-jdk
sudo apt install -y default-jre
java --version
```

HTML

HTML (HyperText Markup Language) es el lenguaje más utilizado en el ámbito web, ya que se utiliza para desarrollar la mayoría de las páginas web. Este está implementado por todos los navegadores. En el proyecto se emplea HTML para generar el contenido de los mensajes de correo.

SQL

SQL (Structured Query Language) es un lenguaje de programación usado para manejar RDBMS (bases de datos relacionales), es decir, para modificar las tablas, añadir o eliminar datos, recuperar cierta información...

LaTeX

LaTeX es un editor de texto orientado a generar documentos con una calidad tipográfica. Se utiliza comúnmente para artículos y libros científicos. Se ha usado LaTeX para la redacción de este documento.

2.2. APIs

Dataformat YAML

Dataformat YAML es una API para java que nos facilita la lectura y edición de ficheros YAML desde línea de código, para incluirlo en el proyecto debemos añadir al pom.xml:

```
<dependencies>
    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-yaml</artifactId>
        <version>2.10.3</version>
    </dependency>
</dependencies>
```

JavaMail

JavaMail es una API para java que nos permite enviar correos de forma sencilla a través del protocolo SMTP, para incluir esta en nuestro proyecto debemos añadir lo siguiente al pom.xml:

```
<dependencies>
    <dependency>
        <groupId>com.sun.mail</groupId>
        <artifactId>javax.mail</artifactId>
        <version>1.6.2</version>
    </dependency>
</dependencies>
```

Java JDBC

Java JDBC es una API para Java que nos permite realizar consultas a una base de datos MySQL desde el código fuente, para incluir-lo en nuestro proyecto debemos añadir al pom.xml:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.29</version>
</dependency>
```

BlacklistMaster

Blacklist master [4] es una API HTTP RESTFUL que nos permite comprobar si un cierto dominio o dirección IP está en alguna de las blacklist o sistemas de reputación más reconocidos.

OneSimpleAPI

OneSimpleApi [5] es una API HTTP que incluye varias funcionalidades, una de ellas nos permite determinar el destino final de una URL.

2.3. Herramientas

Maven

Maven es una herramienta open-source que básicamente permite simplificar todos los procesos de compilación y ejecución de nuestro código, si no usamos maven tendríamos que analizar todo nuestro código para ver que necesitamos para compilarlo, maven nos facilita la vida en esto, ya que lo hace de forma automatizada mediante el uso del archivo pom.xml.

Para obtener maven en Debian podemos emplear estos comandos:

```
sudo apt-get update
sudo apt install maven
```

IntelliJ Idea

IntelliJ Idea [2] es un IDE o programa para el desarrollo de código fuente cuyo fin es el desarrollo de aplicaciones en Java, este ha sido desarrollado de tal forma que optimiza la productividad del desarrollador. Incluye funcionalidades como puede ser una 'Deep inteligence' que nos da sugerencias de como mejorar nuestro código para hacerlo más óptimo, incluye autocompletado de código, es decir, cuando nosotros escribimos IntelliJ nos sugiere opciones de autocompletado, incluso nos permite crear funciones prefabricadas para nuestras clases en Java.

Sublime Text

Sublime Text [3] es un editor de texto sofisticado, tiene funciones que facilitan la edición de textos como códigos fuente o archivos de configuración.

MySQL WorkBench

MySQL WorkBench es un programa o interfaz que nos permite realizar gestiones cómodamente de bases de datos MySQL.

Bitvise SSH Client

Bitvise SSH [6] es una herramienta que nos permite conectarnos a servidores remotos mediante el uso de los protocolos SSH y FTP.

Overleaf

Overleaf [7] es un editor LaTeX online que nos permite desarrollar documentos cómodamente y sin necesidad de instalar ningún programa.

Git

Git [8] es una herramienta open source de control de versiones para diferentes proyectos como puede ser un código Java.

Instagantt

Instagantt [9] es una página web que nos permite desarrollar fácilmente diagramas y tablas de Gantt.

LucidChart

LucidChart [10] es una herramienta online que permite crear diagramas, tablas de flujo y tablas de texto con mucha facilidad.

2.4. Otros

OpenSSL y Keystore

Para generar certificados autofirmados necesitaremos las herramientas OpenSSL [11] y KeyStore, keystore es una herramienta de Oracle, que como su propio nombre indica nos permite generar llaves criptográficas y certificados electrónicos (CA), OpenSSL tiene una función similar pero con un nivel de configuración mucho más amplio.

XML

XML es un lenguaje desarrollado para poder almacenar datos de forma visible, también incluye soporte para bases de datos.

3. Marco teórico

Para el desarrollo de un servidor SMTP que nos pueda ayudar a detectar el phishing cabe entender claramente un par de conceptos teóricos que se expondrán a continuación.

3.1. Protocolo SMTP

SMTP es un protocolo de la capa de aplicación que utiliza TCP para la transmisión del correo. SMTP, como casi todos los protocolos de esta capa, tiene 2 lados, el cliente y el servidor.

Este protocolo está definido en el RFC 5321 [12] y se encuentra ubicado en el puerto 25 por defecto y en el puerto 587 en el caso de SMTP con cifrado TLS/SSL.

SMTP por defecto restringe los mensajes a formato ASCII de 7 bits [13], pero en la actualidad esta decisión ha causado algunos problemas debido a la posibilidad de adjuntar archivos a los correos, por lo que los archivos multimedia adjuntos se deben codificar a ASCII y luego decodificado en binario al llegar al remitente.

Cabe remarcar que SMTP no utiliza servidores intermedios, por lo que si los 2 servidores, origen y destino, se encuentran en puntas opuestas del mundo, estos establecerán una conexión TCP directa, si el servidor destino no se encuentra disponible, el origen guarda el mensaje hasta que el destino esté disponible.

Para el envío de correos y asegurar la autenticidad se suele requerir autenticación SMTP, es decir, se debe incluir la dirección de correo y la contraseña para verificar si el usuario que envía el correo realmente está autorizado para hacerlo.

3.1.1. Ejemplificación del protocolo

Para realizar este ejemplo vamos a emplear el siguiente escenario: Arnau quiere enviar un correo a Raúl para felicitar su cumpleaños, podemos ver un ejemplo gráfico en **Figura 3**.

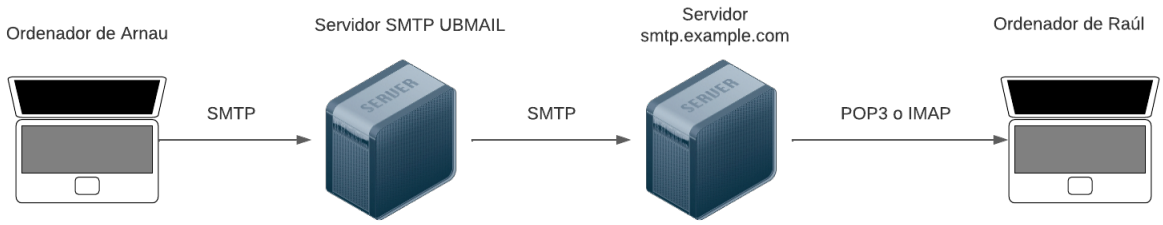

Figura 3: Comunicación SMTP entre Arnau y Raúl

1. Arnau accede a su aplicación de correo, en este caso UBMAIL, incluye la dirección electrónica de Raúl (raul@ejemplo.com), incluye un mensaje de felicitaciones y lo envía.
2. UBMAIL envía el mensaje de Arnau y lo coloca en la cola de correos.
3. El servidor UBMAIL abre una conexión TCP con smtp.ejemplo.com (servidor de Raúl).
4. El servidor smtp.ejemplo.com recibe el mensaje y lo incluye en el buzón de correo de Raúl.
5. Por último, Raúl abre su aplicación de correo electrónico y visualiza el mensaje.

3.1.2. Comandos básicos del protocolo

Los diferentes comandos del protocolo a tener en cuenta según el RFC 5321 [12] son:

- EHLO - Indica el nombre del cliente y que se ha conectado al servidor.
- MAIL FROM - El cliente indica la dirección de correo origen del correo.
- RCPT TO - El cliente nombra al destinatario del correo.
- DATA - El cliente empieza la transmisión de datos.
- RSET - El cliente solicita el reinicio del protocolo.
- VRFY/EXPN - El cliente pide verificar si el buzón de correo está disponible.
- NOOP - Mensaje usado para detectar si la conexión sigue activa mediante timeouts.
- QUIT - El cliente indica el fin de la conexión.

3.1.3. Códigos de respuesta básicos del protocolo

Los diferentes códigos del protocolo a tener en cuenta son:

- 220 - Servidor preparado, este código nos indica que el servidor está preparado para procesar el siguiente comando.
- 221 - Servicio cerrado, nos indica que la sesión actual se va a cerrar.
- 250 - OK, nos indica que el comando anterior ha sido aceptado.
- 354 - Envía el mensaje e indica su fin con '.' .
- 421 - No disponible, el servicio no se encuentra disponible o no es accesible.
- 450 - Error al procesar el mensaje.
- 451 - Error, el comando anterior se ha rechazado, ya que el servidor no lo puede procesar.
- 452 - Error, el servidor no puede procesar el comando porque no dispone de memoria suficiente.
- 455 - Error, el servidor no puede procesar el comando en estos momentos.
- 500 - El servidor no puede reconocer el comando.
- 501 - La dirección de correo es incorrecta.
- 502 - Este comando no está implementado.
- 503 - El servidor ha detectado una secuencia errónea de comandos.
- 521 - Este servidor no acepta mails.
- 541 - El correo no se puede entregar, seguramente por spam.

3.1.4. Ejemplo de comunicación Cliente-Servidor SMTP

En el log que se muestra a continuación podemos observar un ejemplo, extraído del programa realizado en este trabajo de campo, donde se puede ver como sería una comunicación satisfactoria con el protocolo SMTP, también se puede observar el diagrama de secuencia en la **Figura 4**.

```
S: 220 Hola buenas soy el servidor
C: EHLO DESKTOP-8F9DD86
S: 250 smtp.arnaugris.es Hello client.example.com
C: MAIL FROM:<from@gmail.com>
S: 250 OK
C: RCPT TO:<gopi.mani@xyz.com>
S: 250 OK
C: RCPT TO:<Maimsa.SF@xyz.com>
S: 250 OK
C: RCPT TO:<NEHA.SIVA@xyz.com>
S: 250 OK
C: RCPT TO:<Usha.B@xyz.com>
S: 250 OK
C: DATA
S: 354 OK
C: From: from@gmail.com
C: To: gopi.mani@xyz.com, Maimsa.SF@xyz.com
C: Cc: NEHA.SIVA@xyz.com
C: Message-ID: <428746855.1.1652902360972.JavaMail.DarkDhz@DESKTOP-8F9DD86>
C: Subject: Mail Subject
C: MIME-Version: 1.0
C: Content-Type: multipart/mixed;
C:  boundary="----=_Part_0_1618212626.1652902360949"
C:
C: ------=_Part_0_1618212626.1652902360949
C: Content-Type: text/html; charset=utf-8
C: Content-Transfer-Encoding: 7bit
C:
C: Esto es un correo de prueba
C:  https://www.ub.edu/cosas
C:  https://www.ubay.edu/cosas
C:  https://www.ub.com/cosas
C:  https://www.universitat.edu/cosas
C: ------=_Part_0_1618212626.1652902360949--
C: .
S: 250 OK
C: QUIT
S: 221 Bye
```

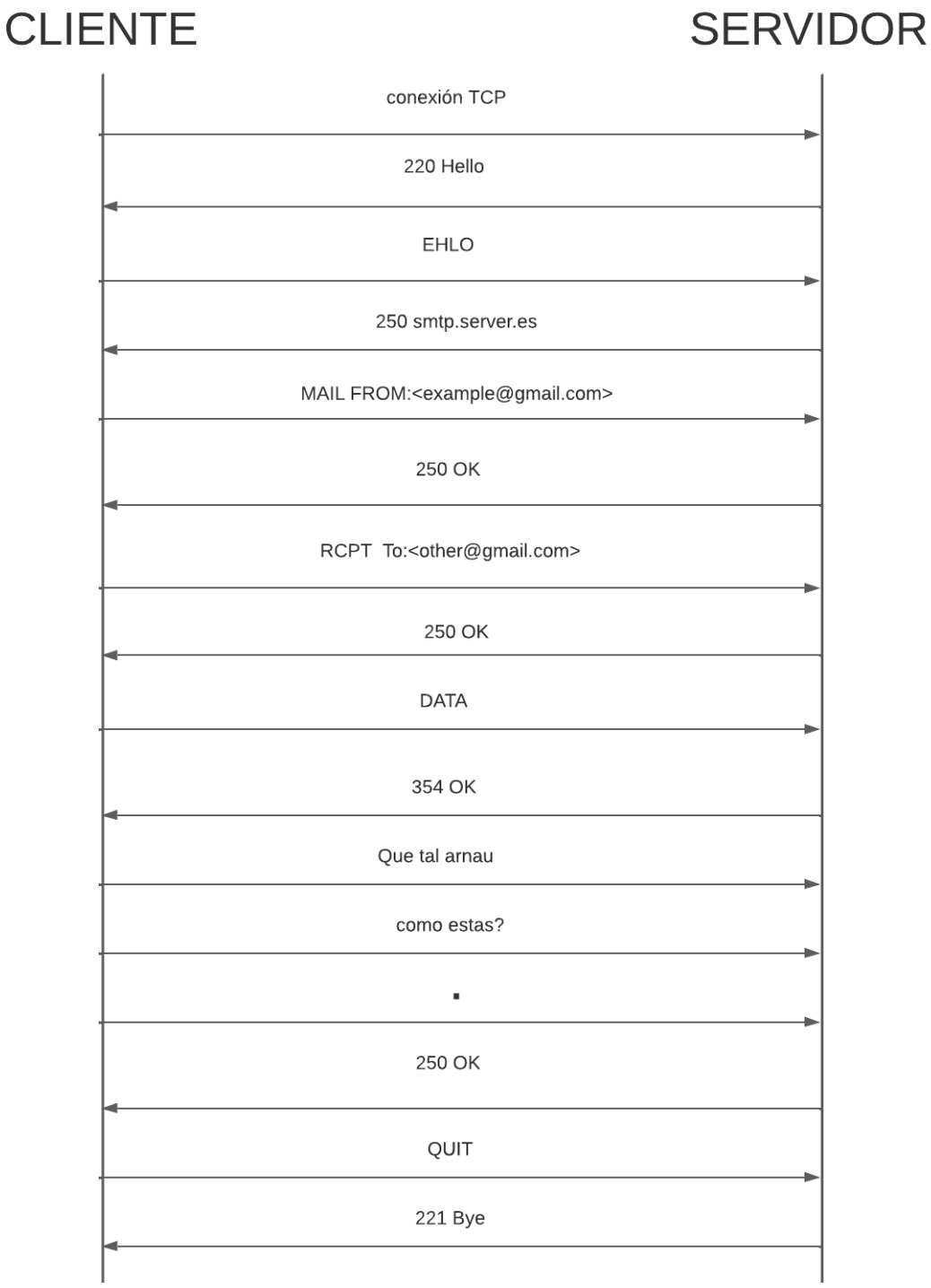

Figura 4: Diagrama de secuencia comunicación SMTP

3.2. Phishing

Este tipo de ciberataque es uno de los más comunes empleados por el cibercrimen [14], incluso con los conocimientos que tenemos sobre los correos falsos los usuarios del correo electrónico siguen cayendo en este tipo de engaños.

El Phishing es un ciberataque donde las víctimas son contactadas por correo electrónico, llamada telefónica o mensajes de texto, donde alguien simula que es una institución legítima para conseguir datos de valor sobre la víctima, como puede ser información bancaria o una cuenta de redes sociales. Esta información se puede usar para realizar robos o incluso realizar suplantación de identidad.

Existen varios tipos de phishing, los podemos encontrar resumidos en el apéndice.

3.2.1. Ejemplo de phishing

En la **Figura 5** podemos observar un ataque de phishing extraído de mi correo profesional, donde podemos ver que según el mail tenemos una cuenta de 188,050 € en bitcoin y que para retirar el saldo hemos de acceder a una web donde nos pide las credenciales de nuestro banco, podemos identificar que es phishing por varios motivos, entre ellos que el mail es demasiado bueno para ser verdad o que por ejemplo este no ha sido cifrado **Figura 6**. En la última figura mencionada observamos un icono de un candado rojo tachado, esto quiere decir que el mail no ha sido cifrado usando SSL o TLS.

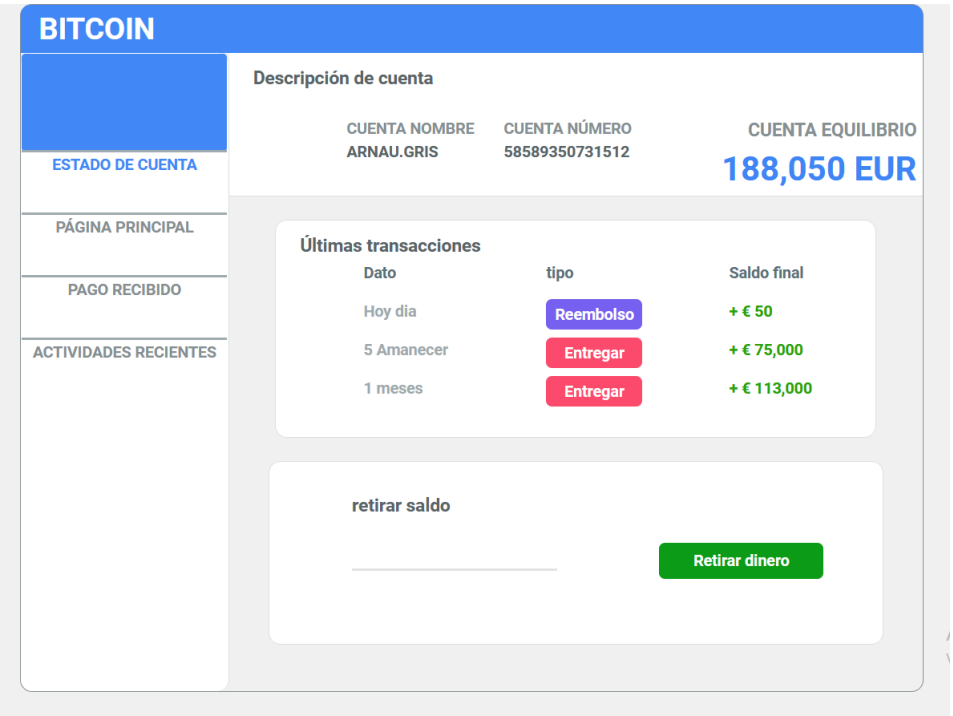

Figura 5: Mail phishing de tipo Email/Spam

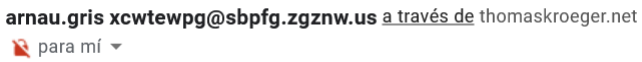

Figura 6: Mail no cifrado por el remitente

3.3. Contramedidas phishing

A continuación se muestran las medidas más comunes utilizadas para evitar el phishing:

Educación social

Esta contramedida es la más simple, pero a la vez la más efectiva, la base del phishing está en hacer caer a alguien humano en una trampa, por lo que si esta persona ha sido formada para no acceder a estos correos maliciosos será mucho más difícil que caiga en un ataque de phishing [16].

Sistemas de IA

Últimamente cada vez más se están implementando mecanismos de inteligencia artificial que permiten detectar si un correo es malicioso o no.

Lectores de correo

Los lectores de correo son programas cuya función consiste en leer el contenido de un correo, y mediante el formato de este y la forma en la que está este escrito, puede detectar si el correo es malicioso o no.

Filtros de spam

La mayoría de servidores de correo actuales implementan filtros anti-spam, por ejemplo cuando un correo se ha enviado sin cifrar se considera que puede ser peligroso. Al igual que las blacklist que guardan la reputación de ciertos dominios para detectar si estos son fiables o no.

4. Análisis

A continuación se muestra el análisis de viabilidad del proyecto. Dividiremos este análisis en tres partes viabilidad económica, viabilidad técnica y viabilidad de mercado.

4.1. Viabilidad económica

El primer paso del análisis consiste en explorar si realmente es factible el desarrollo del proyecto o no. Para ello vamos a observar 3 ámbitos:

Recursos de software

En cuanto al desarrollo del software se han utilizado herramientas open-source o con licencia gratuita. Al tratarse de un software realizado en Java, existen multitud de opciones con coste económico bajo para el desarrollo de este tipo de programas. Podemos decir entonces que el software en este proyecto no supondrá un gran impacto económico, ya que siempre que sea posible se emplearan licencias gratuitas.

Recursos de hardware

En el caso del hardware, el desarrollo del proyecto se puede realizar en un portátil de bajas prestaciones, ya que el desarrollo de software no requiere de un ordenador de gran rendimiento. Los costes en cuanto al hardware se encuentran en que necesitamos un alojamiento para el servidor de correo y otro para el servidor MySQL. Para ello se han investigado ciertos proveedores de estos servicios, como puedes ser el contratado en este proyecto OVH [31], que nos ofrece un servidor con prestaciones básica para nuestro proyecto, como podemos observar en la **Figura 7**.

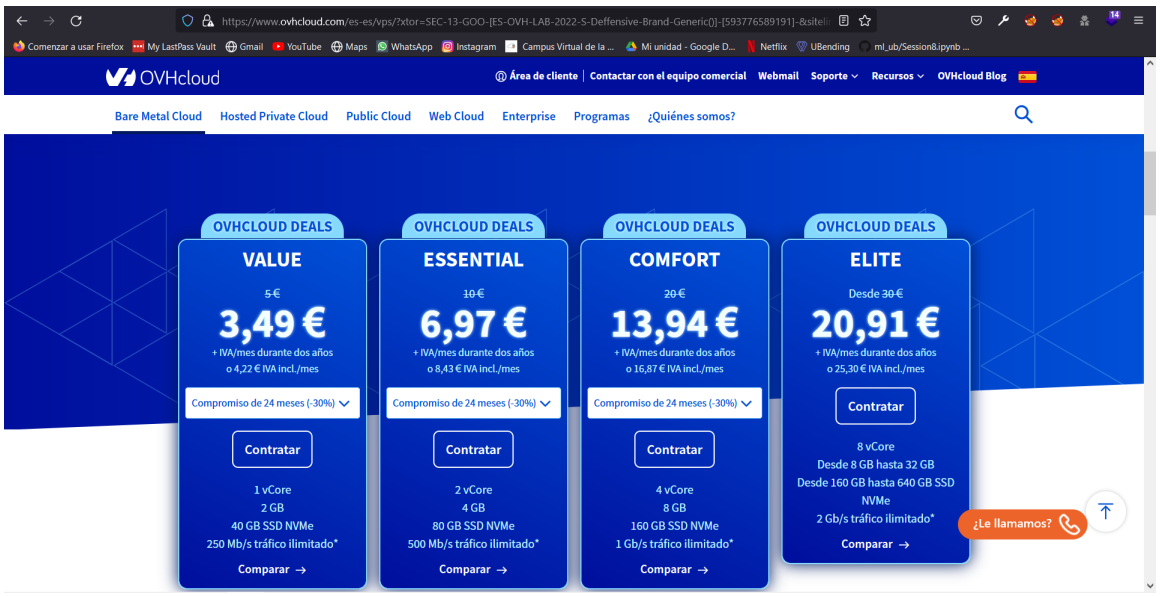

Figura 7: Gama económica hosting VPS en OVH

Este tipo de hosting al tratarse de un VPS (Virtual Private Server) nos permite implementar en el servicio de correo y el servicio MySQL. Este host está pensado para un caso de prueba donde no haya mucho tráfico, en el caso de utilizar el proyecto en un caso con gran cantidad de tráfico necesitaremos un host con más potencia.

En resumen, en el caso de esta implementación el hardware no supone un gran coste económico.

Recursos humanos

Al tratarse de un proyecto académico, el coste humano es bajo, esto es debido a que la misma persona se encarga del análisis, diseño, desarrollo y pruebas del proyecto. En el caso de desarrollarse el proyecto en un ámbito como puede ser una empresa, habría que tener en cuenta todo el personal implicado en el proyecto como su coste por hora.

Resumen

En resumen, según lo indicado en los 3 apartados anteriores, este proyecto es viable económicamente, ya que su desarrollo no supone un gran coste.

4.2. Viabilidad técnica

Debido a que las herramientas técnicas empleadas para el desarrollo se usan en multitud de proyectos de software, su uso no supondrá ningún riesgo. También los protocolos utilizados para el desarrollo están muy documentados mediante RFC (Request for Comments), por lo que podemos decir que el proyecto es viable técnicamente.

4.3. Viabilidad de mercado

En el caso de la viabilidad de mercado existen varias opciones para protegerse contra el phishing, entre ellas se encuentran:

- **sophos.com:** es una empresa encargada de formación humana y simulación de ataques de phishing, según ellos indican en su página web, refuerzan la superficie más expuesta a estos ataques, es decir los usuarios finales [17].
- **sofistic.com:** esta empresa ofrece productos que implementan inteligencia artificial en endpoints [18].

Como podemos observar, existen varias empresas que se dedican a hacer que los usuarios estén protegidos contra el phishing, pero ninguna de estas implementa sistemas de comprobación de URL o comprobaciones de dominios peligrosos, esto personalizado a cada cliente.

Podemos decir entonces que la protección contra el phishing es una área que aún está por explotar, por lo que este proyecto tiene cabida en el mercado.

5. Diseño

5.1. Actores

En este proyecto existen 3 tipos de usuarios objetivos, usuarios de mail, empleado de una empresa y administrador de la empresa, en la **Figura 8** se muestra el diagrama de actores correspondiente.

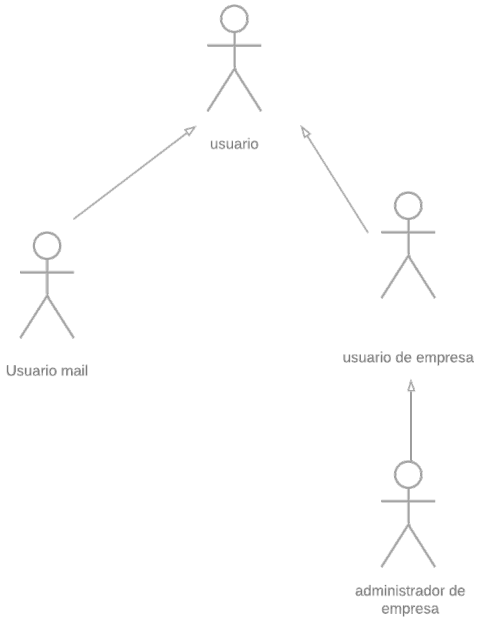

Figura 8: Diagrama de actores

Usuario mail

Este es un usuario que le da un uso personal al correo, es decir, lo utiliza para enviar y recibir correos de sus contactos o para informarse de ciertas promociones de alguna web. Este usuario usará el servidor del proyecto para comprobar que todos los correos que él recibe son seguros.

Empleado

Este es un tipo de usuario que utiliza un correo de empresa, está interesado en el uso de este proyecto para evitar el phishing dirigido, es decir, usara una configuración personalizada del servidor según los dominios que se consideren seguros y los que no para la empresa.

Administrador de la empresa

Este usuario utiliza el servidor mediante el correo de la empresa, a la vez se encarga de configurar las funcionalidades del servidor para que los empleados de la empresa estén seguros contra el phishing.

5.2. Casos de uso

En las **Tablas 1, 2, 3, 4, 5 y 6** se muestran los casos de uso de la aplicación.

CU-101: Generar reporte	
Descripción	El usuario recibe el reporte de un correo.
Actores	Usuario mail, empleado, administrador.
Precondiciones	1. El usuario tiene configurada la redirección mail. 2. El usuario recibe un correo.
Flujo	1. El usuario recibe el correo. 2. El cliente mail reenvía el correo al servidor. 3. El servidor genera el reporte. 4. El servidor envía el reporte por correo al usuario.
Postcondiciones	El usuario lee el reporte.

Cuadro 1: CU-101: Generar reporte

CU-102: Configurar servidor	
Descripción	El usuario configura el servidor.
Actores	Administrador.
Precondiciones	El usuario tiene acceso con privilegios al servidor.
Flujo	1. El usuario se conecta al servidor. 2. El usuario abre el archivo de configuración config.yml. 3. El usuario edita el archivo de configuración. 4. El usuario guarda los cambios y reinicia el servidor.
Postcondiciones	El servidor se inicia con la nueva configuración.

Cuadro 2: CU-102: Configurar servidor

CU-103: Añadir dominios a la base de datos	
Descripción	El usuario edita los dominios de la base de datos.
Actores	Administrador.
Precondiciones	El usuario tiene acceso con privilegios al servidor MySQL.
Flujo	1. El usuario se conecta al servidor MySQL. 2. El usuario selecciona la base de datos y la tabla domains. 3. El usuario añade mediante una query los dominios.
Postcondiciones	El servidor actualiza los dominios.

Cuadro 3: CU-103: Añadir dominios a la base de datos

CU-104: Eliminar dominios de la base de datos	
Descripción	El usuario elimina ciertos dominios de la base de datos.
Actores	Administrador.
Precondiciones	El usuario tiene acceso con privilegios al servidor MySQL.
Flujo	1. El usuario se conecta al servidor MySQL. 2. El usuario selecciona la base de datos y la tabla domains. 3. El usuario elimina mediante una query los dominios.
Postcondiciones	El servidor actualiza los dominios.

Cuadro 4: CU-104: Eliminar dominios de la base de datos

CU-105: Añadir dominios prohibidos a la base de datos	
Descripción	El usuario edita los dominios de la base de datos.
Actores	Administrador.
Precondiciones	El usuario tiene acceso con privilegios al servidor MySQL.
Flujo	1. El usuario se conecta al servidor MySQL. 2. El usuario selecciona la base de datos y la tabla banned. 3. El usuario añade mediante una query los dominios.
Postcondiciones	El servidor actualiza los dominios.

Cuadro 5: CU-105: Añadir dominios prohibidos a la base de dato

CU-106: Eliminar dominios prohibidos de la base de datos	
Descripción	El usuario elimina ciertos dominios de la base de datos.
Actores	Administrador.
Precondiciones	El usuario tiene acceso con privilegios al servidor MySQL.
Flujo	1. El usuario se conecta al servidor MySQL. 2. El usuario selecciona la base de datos y la tabla banned. 3. El usuario elimina mediante una query los dominios.
Postcondiciones	El servidor actualiza los dominios.

Cuadro 6: CU-106: Eliminar dominios prohibidos de la base de datos

5.3. Patrones de diseño

A continuación se exponen teóricamente los patrones de diseño empleados en este trabajo de fin de grado.

Singleton

Este patrón de diseño es usado en ingeniería del software para limitar el número de instancias de una clase, es decir, su objetivo principal es garantizar que solo exista una única instancia de una clase y el acceso a esta. También esta debe garantizar que si la instancia no se ha creado aun y varios hilos intentan acceder a ella, únicamente 1 hilo sea el encargado de generarlo. Un ejemplo de Singleton en Java sería el siguiente:

```
public class Levenshtein {

    // Singleton Instance
    private static volatile Levenshtein instance = null;

    private Levenshtein() {}

    public static Levenshtein getInstance() {
        // To ensure only one instance is created
        if (instance == null) {
            synchronized (Levenshtein.class) {
                if (instance == null) {
                    instance = new Levenshtein();
                }
            }
        }
        return instance;
    }
}
```

En el código que se muestra se puede observar que el constructor de la clase es 'private' por lo que no se puede acceder a él al instanciar la clase, observamos un método estático que nos devuelve la instancia Singleton de la clase, y que en caso de que la instancia no exista la crea, también este método garantiza que no haya 'race condition', se conoce como race condition cuando 2 o más hilos intentan acceder a un recurso al mismo tiempo, esto se debe al uso de la palabra reservada 'syncronized' que garantiza la exclusión mutua.

Factory method

Este patrón de diseño permite crear una interfaz para crear objetos en una superclase, mientras que en las subclases se decide el tipo de objeto que se va a crear [19]. Es decir, te permite generar un objeto sin tener que especificar la clase exacta de este objeto. Un ejemplo en Java de este patrón se muestra a continuación:

```
public Runnable createProxy(ProxyType type) throws IOException {
    ServerYaml server = ServerYaml.getInstance();
    switch (type)  {
        case NORMAL:
            return new Proxy(server.getIP(), server.getPort());
        case SSL:
            return new SSLProxy(server.getIP(), server.getSSlPort());
        case TLS:
            return new SSLProxy(server.getIP(), server.getTLSPort());
        default:
            return null;
    }
}
```

Este ejemplo nos permite crear un tipo de servidor, con TLS, SSL o sin cifrado, sin tener que indicar explícitamente la clase que queremos generar.

5.4. Diagrama de flujo

Una primera aproximación del flujo de la aplicación se muestra en la **Figura 9**.

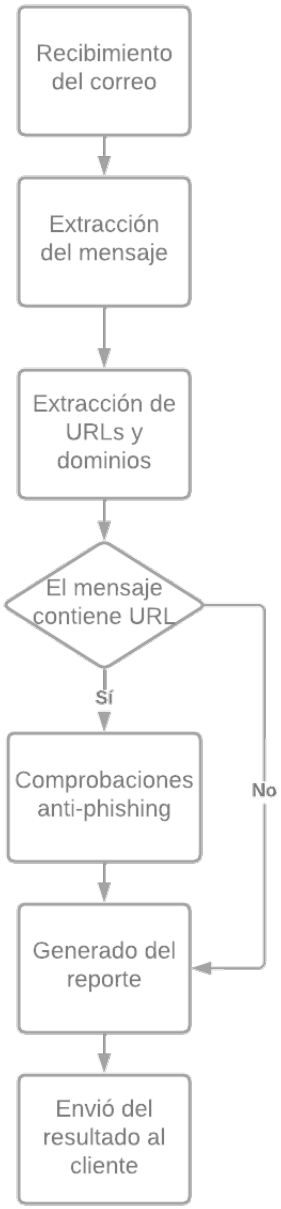

Figura 9: Diagrama de flujo del servidor

En la **Figura 10** se muestra en detalle el diagrama de flujo de las comprobaciones anti-phishing realizadas por el servidor.

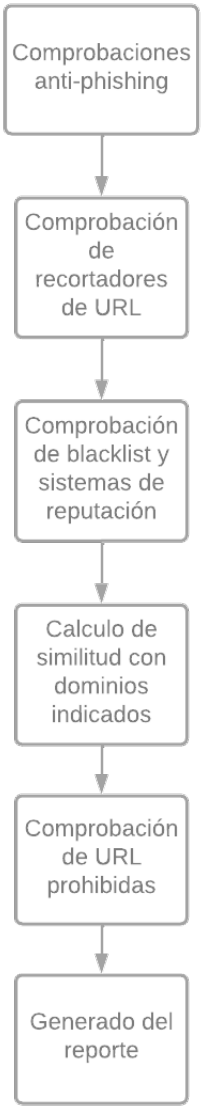

Figura 10: Diagrama de flujo de las comprobaciones

5.5. Arquitectura del sistema

En arquitectura del sistema se define el diseño de todos los elementos que se van a implementar en el proyecto.

5.5.1. Lectores de fichero YAML

En esta sección se muestra el diseño realizado para los lectores de fichero YAML **Figura 11**.

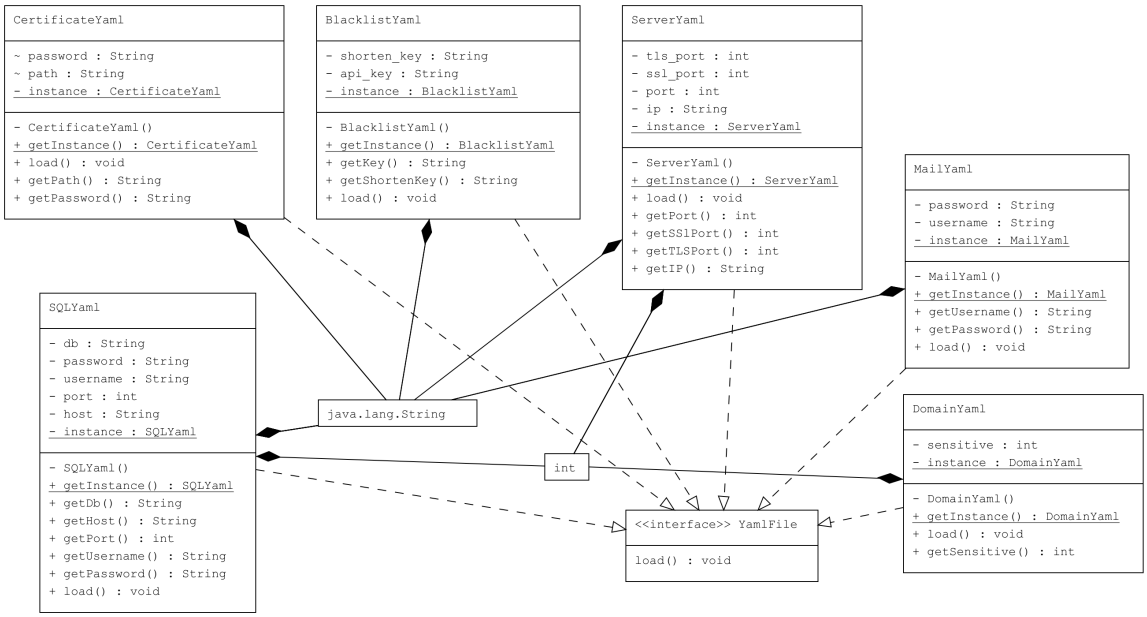

Figura 11: Diagrama de lectores de fichero

YamlFile

Define una interfaz para todos los lectores de ficheros YAML. El correspondiente diseño se muestra en **Figura 12**.

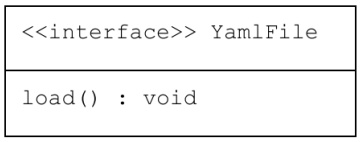

Figura 12: Diseño de la interfaz YamlFile

BlacklistYaml

Se encarga de extraer de la configuración toda la información relacionada con la blacklist. El correspondiente diseño se muestra en **Figura 13**.

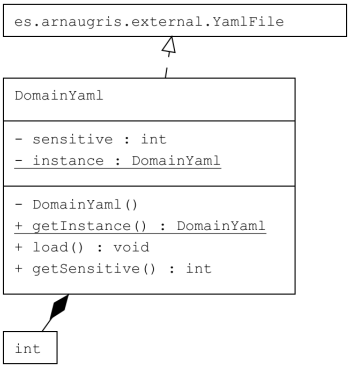

Figura 13: Diseño de la clase BlacklistYaml

CertificateYaml

Se encarga de extraer de la configuración toda la información relacionada con los certificados para el uso del servicio TLS. El correspondiente diseño se muestra en **Figura 14**.

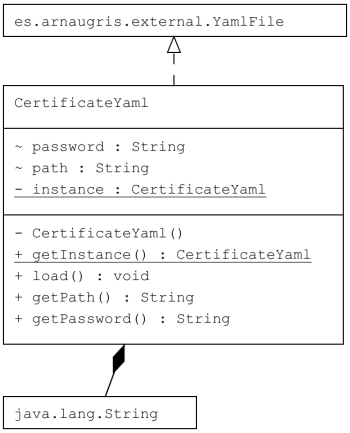

Figura 14: Diseño de la clase CertificateYaml

DomainYaml

Se encarga de extraer de la configuración toda la información relacionada con el algoritmo de detección de URL similares. El correspondiente diseño se muestra en **Figura 15**.

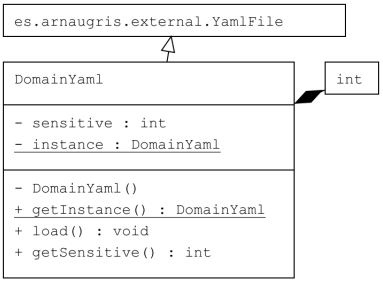

Figura 15: Diseño de la clase DomainYaml

MailYaml

Se encarga de extraer de la configuración toda la información relacionada con el mail que va a enviar el reporte. El correspondiente diseño se muestra en **Figura 16**.

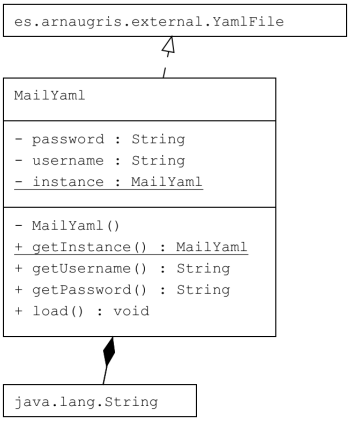

Figura 16: Diseño de la clase MailYaml

ServerYaml

Se encarga de extraer de la configuración toda la información relacionada con el servidor, es decir hostname y los diferentes puertos del servidor. El correspondiente diseño se muestra en **Figura 17**.

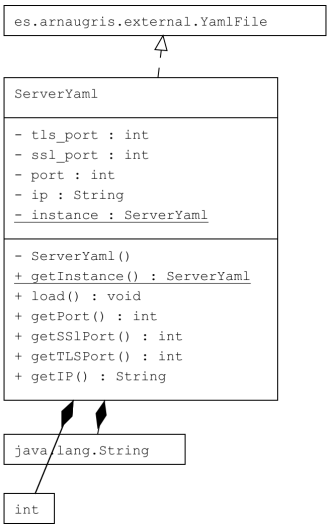

Figura 17: Diseño de la clase ServerYaml

SQLYaml

Se encarga de extraer de la configuración toda la información relacionada con la base de datos, su endpoint, su usuario de acceso y la base de datos en sí. El correspondiente diseño se muestra en **Figura 18**.

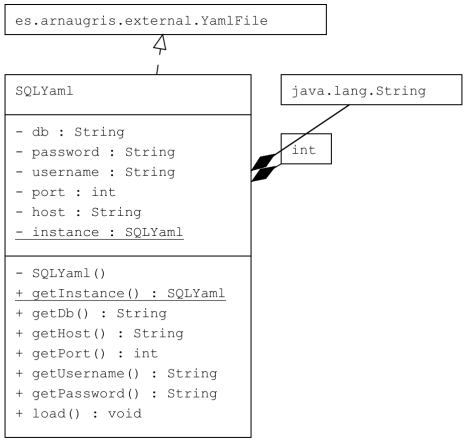

Figura 18: Diseño de la clase SQLYaml

5.5.2. Gestion de servidores

En esta sección se muestra el diseño utilizado para los gestores de servidores SMTP **Figura 19**.

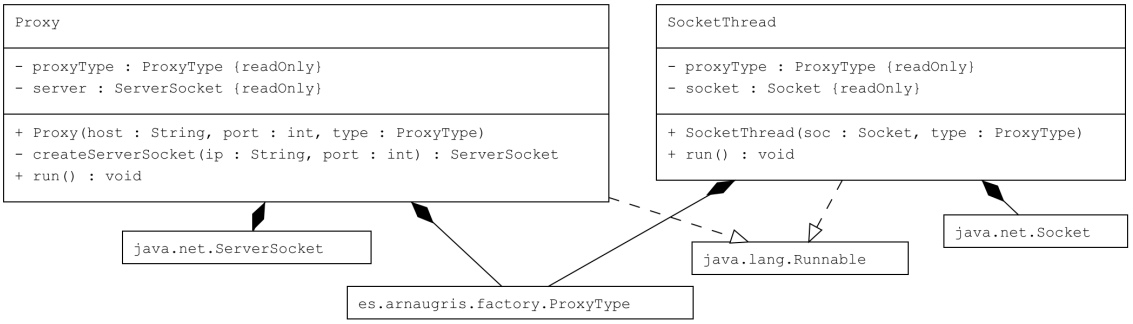

Figura 19: Diagrama de gestion de servidores

Proxy

Se encarga de generar el servidor de conexión TCP y de ir generando todos los hilos de las peticiones que se vayan realizando al servidor, gestiona tanto del servidor sin encriptado como el servidor TLS. El correspondiente diseño se muestra en **Figura 20**.

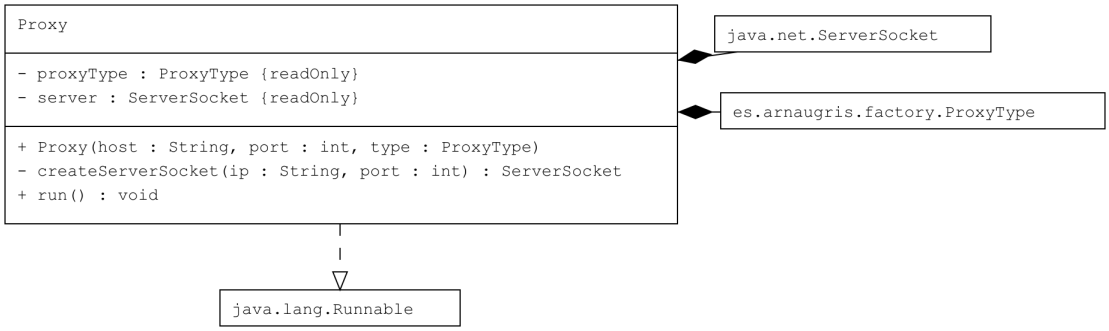

Figura 20: Diseño de la clase Proxy

SoketThread

Se encarga de gestionar los hilos de las peticiones TCP, así como generar los buffers de entrada y salida del socket de cada petición. El correspondiente diseño se muestra en **Figura 21**.

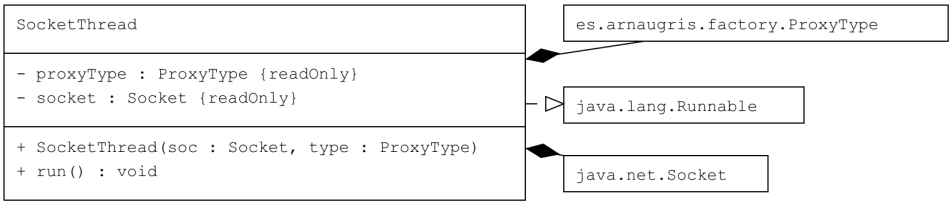

Figura 21: Diseño de la clase SoketThread

5.5.3. Gestión de protocolo SMTP

En esta sección se define el diseño para las clases encargadas de gestionar el protocolo SMTP, sus datos y el postprocesador del protocolo **Figura 22**.

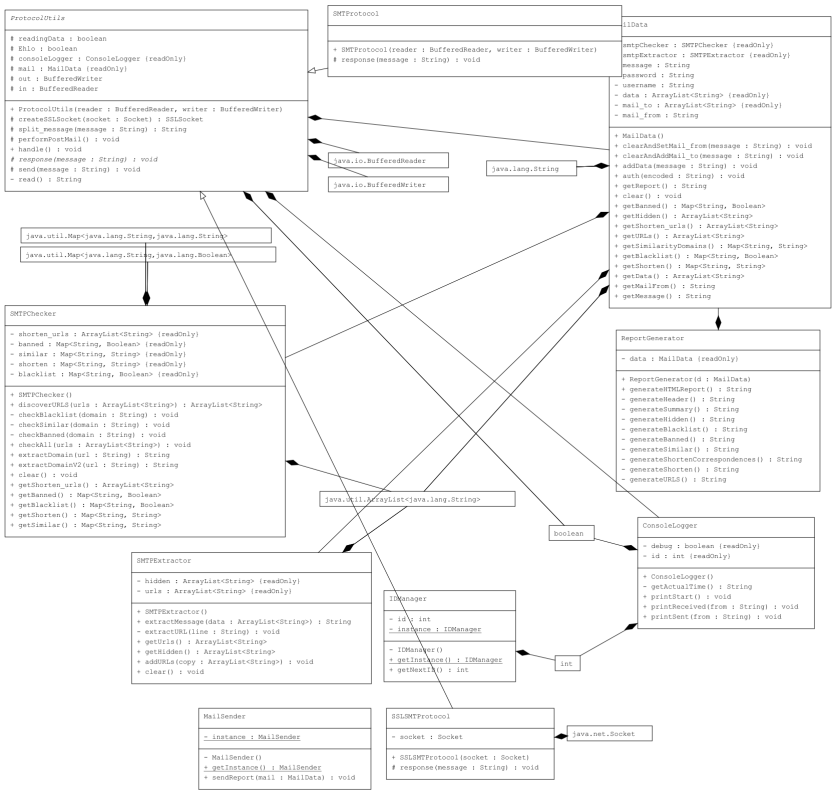

Figura 22: Diagrama de gestión de protocolo SMTP

ConsoleLogger

ConsoleLogger gestiona e imprime la información del servidor que se muestra por terminal. El correspondiente diseño se muestra en **Figura 23**.

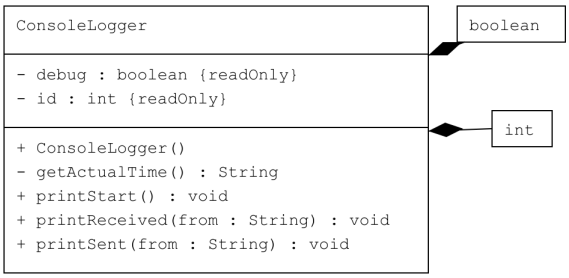

Figura 23: Diseño de la clase ConsoleLogger

IDManager

Esta clase se encarga de gestionar los ID de cada hilo del servidor, asegurando la exclusión mutua al generar un ID para cada hilo. El correspondiente diseño se muestra en **Figura 24**.

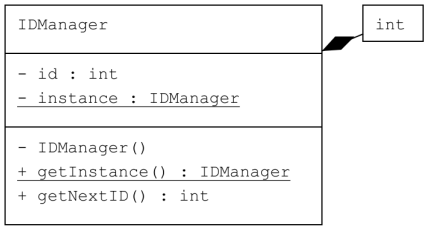

Figura 24: Diseño de la clase IDManager

MailData

MailData guarda toda la información que va recibiendo por el protocolo SMTP, esta se encarga de llamar a las entidades que realizan las comprobaciones y generan el reporte. El correspondiente diseño se muestra en **Figura 25**.

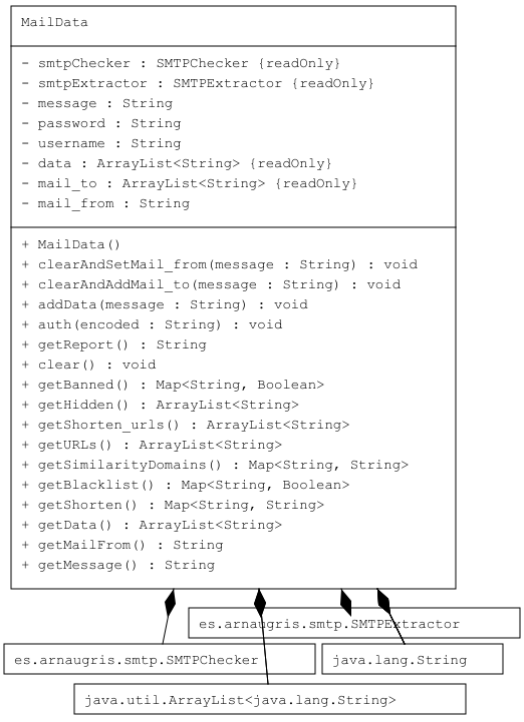

Figura 25: Diseño de la clase MailData

MailSender

MailSender es responsable de enviar el reporte por mail al cliente que lo solicita. El correspondiente diseño se muestra en **Figura 26**.

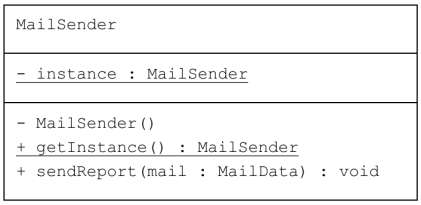

Figura 26: Diseño de la clase MailSender

ProtocolUtils

ProtocolUtils es una clase abstracta que define los métodos para enviar y recibir datos, también se encarga de ir leyendo todo lo que va recibiendo por el buffer de entrada. El correspondiente diseño se muestra en **Figura 27**.

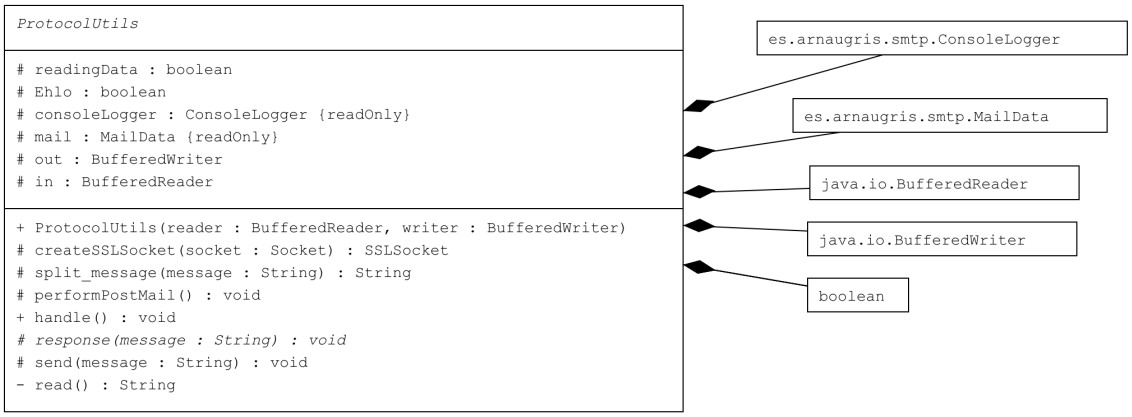

Figura 27: Diseño de la clase ProtocolUtils

ReportGenerator

ReportGenerator es el responsable de generar el reporte en formato HTML que posteriormente se enviara por mail al cliente. El correspondiente diseño se muestra en **Figura 28**.

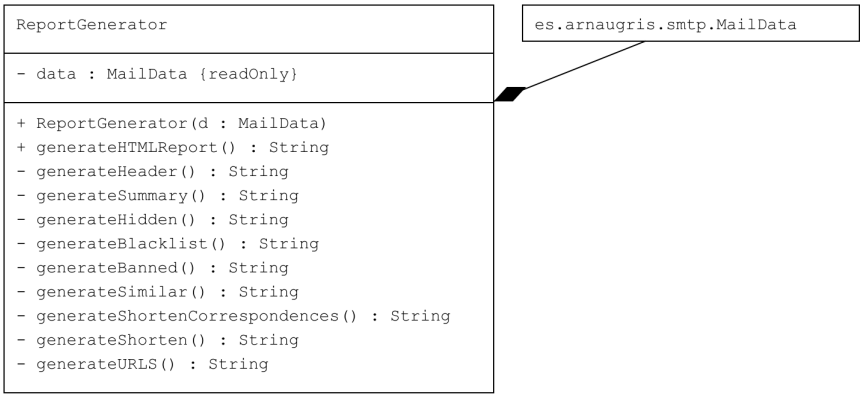

Figura 28: Diseño de la clase ReportGenerator

SMTPChecker

SMTPChecker se encarga de realizar todas las comprobaciones anti-phishing: blacklist, dominios similares, dominios baneados, descubrir redirecciones. El correspondiente diseño se muestra en **Figura 29**.

Figura 29: Diseño de la clase SMTPChecker

SMTPExtractor

SMTPExtractor es el responsable de extraer el mensaje recibido por el protocolo SMTP a la vez que va descubriendo las URL que se muestran en este. El correspondiente diseño se muestra en **Figura 30**.

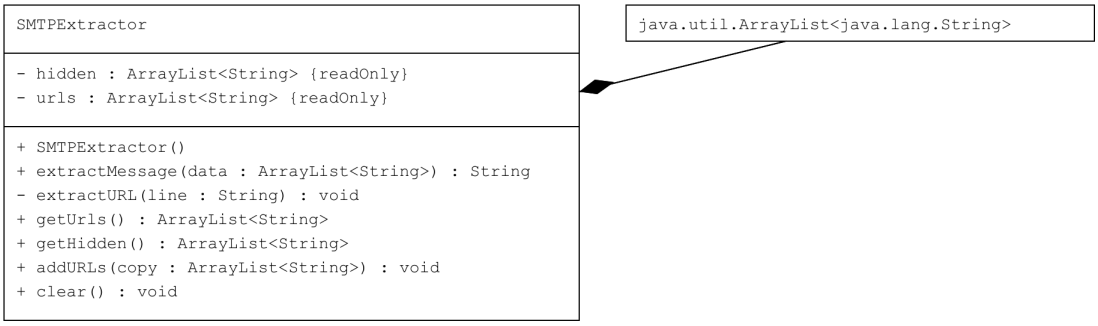

Figura 30: Diseño de la clase SMTPExtractor

SMTProtocol

SMTProtocol es el encargado de responder todos los comandos SMTP que va recibiendo por el buffer de salida. El correspondiente diseño se muestra en **Figura 31**.

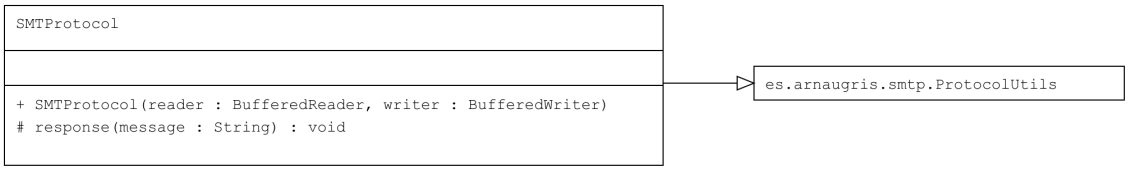

Figura 31: Diseño de la clase SMTProtocol

SSLSMTProtocol

SSLSMTProtocol es el encargado de responder todos los comandos SMTP en formato TLS que va recibiendo por el buffer de salida. El correspondiente diseño se muestra en **Figura 32**.

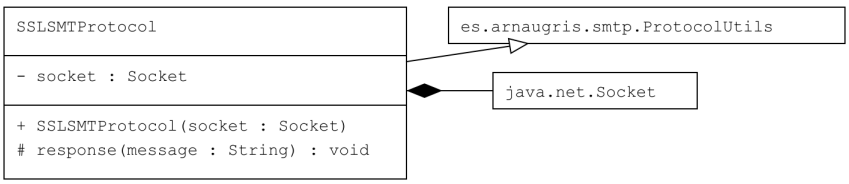

Figura 32: Diseño de la clase SSLSMTPProtocol

5.5.4. Gestores de MySQL

En esta sección se define el diseño para las clases encargadas de gestionar la base de datos MySQL **Figura 33**.

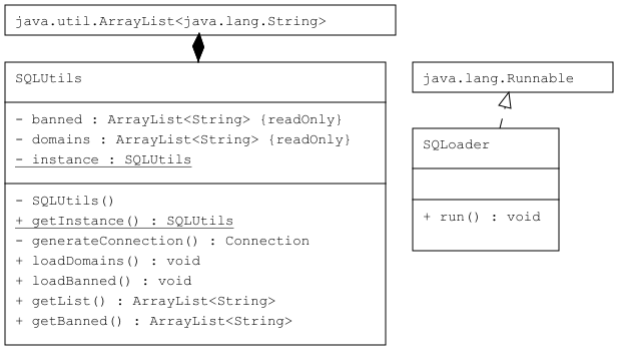

Figura 33: Diagrama de gestores de MySQL

SQLoader

SQLoader se encarga de cargar los datos de la base de datos y de ir recargando los datos cuando estos cambien. El correspondiente diseño se muestra en **Figura 34**.

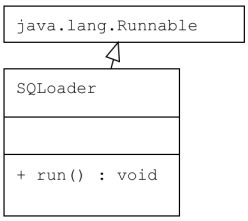

Figura 34: Diseño de la clase SQLoader

SQLUtils

SQLoader es responsable de realizar las conexiones y las consultas a la base de datos MySQL. El correspondiente diseño se muestra en **Figura 35**.

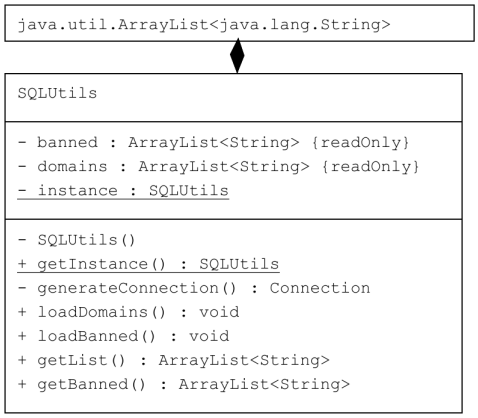

Figura 35: Diseño de la clase SQLUtils

5.5.5. Diagrama de clases

En la **Figura 36** se puede observar el diagrama de clases completo del servidor.

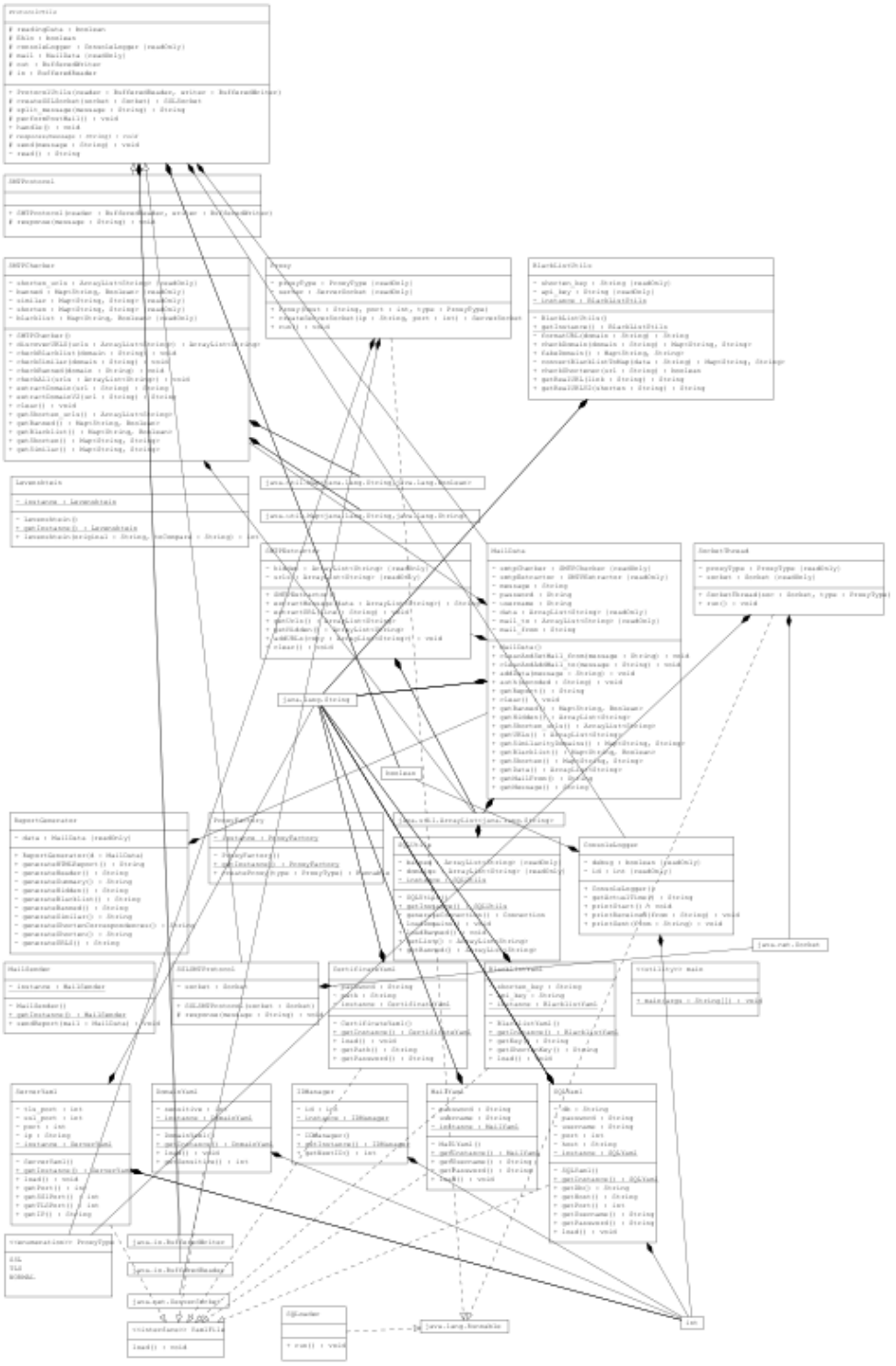

Figura 36: Diagrama de clases completo del servidor

6. Implementación

La idea base de la implementación consiste en que cuando recibimos un correo electrónico cualquiera, este se va a reenviar a una dirección mail configurada en nuestro servidor y posteriormente este servidor va a devolver vía SMTP un report del email recibido. En esta sección voy a entrar en los detalles más relevantes para desarrollar el código final, tanto como la configuración del VPS.

6.1. Configuración del Firewall UFW

Como se ha indicado previamente para el firewall del servidor nos hemos decantado por el uso de UFW. Para configurar este simplemente nos interesa permitir el acceso a ciertos puertos:

- 21 - Puerto FTP, este es necesario para intercambiar archivos con el servidor.
- 22 - Puerto SSH, este puerto nos permite conectarnos remotamente al servidor, es decir, a su terminal.
- 25 - Puerto SMTP, nuestro servidor SMTP por defecto estará alojado en el puerto 25.
- 465 - Puerto SMTP con TLS.

Para permitir la conexión externa a alguno de estos puertos debemos utilizar el siguiente comando:

```
sudo ufw allow <PUERTO>
```

Entonces nuestro firewall debe quedar similar a como se muestra en la **Figura 37**.

To	Action	From
--	------	----
22/tcp	ALLOW	Anywhere
3306	ALLOW	Anywhere
21	ALLOW	Anywhere
22	ALLOW	Anywhere
80	ALLOW	Anywhere
22/tcp	ALLOW	107.22.57.98
3306/tcp	ALLOW	107.22.57.98
1025	ALLOW	Anywhere
3	ALLOW	Anywhere
25	ALLOW	Anywhere
465	ALLOW	Anywhere

Figura 37: Tabla de reglas del firewall

6.2. Configuración DNS del servidor mail destino

Para reenviar los emails a nuestro servidor vamos a necesitar definir que el record MX de nuestro servidor tenga como destino nuestro servidor, podemos encontrar DNS explicado en detalle en la seccion **10.4 Protocolo DNS**. Para realizar esta tarea necesitamos tenemos un dominio propio para así modificar sus registros DNS, en este caso usaremos arnaugris.es. Primero debemos generar un registro de tipo A que apunte a la dirección IPv4 de nuestro servidor, en este caso usaremos el subdominio mail y la IPv4 54.36.191.29:

```
mail IN A 54.36.191.29
```

A continuación definiremos que nuestro registro MX es el servidor mail.arnaugris.es:

```
mail IN MX 1 mail.arnaugris.es.
```

Al realizar esta configuración cada vez que se envíe un correo a alguna dirección de correo @arnaugris.es, este correo pasara por nuestro servidor.

6.3. Configuración reenvío automático

A continuación se muestra los pasos realizados para configurar la redirección en GMAIL. Para configurar nuestro mail personal o profesional para que reciba los reportes debemos modificar el reenvío automático en nuestro cliente mail, en este ejemplo se muestra en el cliente mail GMAIL, para ello hemos creado una cuenta de mail llamada ubmailtesting@gmail.com. Primero debemos dirigirnos a los ajustes al apartado Reenvío y correo POP/IMAP, observaremos una sección llamada reenvío, en ella debemos incluir una dirección mail del subdominio que hemos configurado en el apartado anterior, en este caso usaremos postmaster@arnaugris.es. Al añadir-la se enviará un correo a nuestro servidor con un código de confirmación, debemos copiarlo para así verificar que queremos reenviar a esa dirección de correo, por último si todo ha funcionado correctamente la configuración debería aparecer como se muestra en la **Figura 38**.

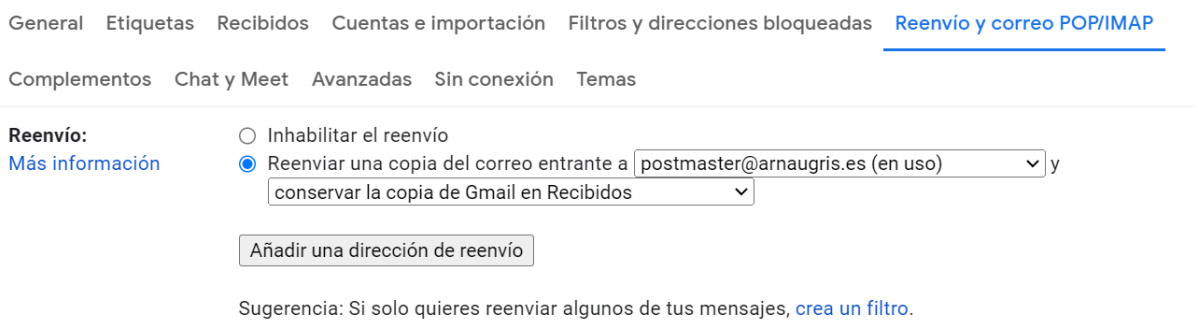

Figura 38: Configuración reenvío en GMAIL

En el caso de que usemos otros clientes de mail como Apple Mail o Microsoft Outlook, podemos encontrar como realizar la redirección en [21] y [22] respectivamente.

6.4. Creación del servidor en Java

Para abrir un servidor se ha utilizado la clase de Java ya implementada ServerSocket, esta nos permite abrir un socket en un cierto endpoint mediante el protocolo orientado a conexión TCP e ir aceptando las diferentes solicitudes que vayan llegando. En nuestro caso hemos utilizado un bucle while(true) para siempre estar aceptando las nuevas solicitudes:

```
public class Proxy extends ServerUtils implements Runnable {

    private final ServerSocket server;

    public Proxy(String host, int port) throws IOException {
        try {
            server = super.createServerSocket(host, port);
            System.out.println("Open server on " + host + ":" + port);
        } catch (IOException e) {
            throw new IOException("Cannot open server");
        }
    }

    public void run() {
        while(true) {
            try {
                Socket socket = server.accept();

                SocketThread st = new SocketThread(socket);

                Thread t = new Thread(st);
                t.start();
            } catch (IOException e) {
                System.out.println(e.getMessage());
                break;
            }
        }
    }
}
```

Para crear el endpoint se utiliza el método createServerSocket(), donde se le indica la IP y puerto donde queremos establecer el endpoint y este retorna nuestro ServerSocket:

```
private ServerSocket createServerSocket(String ip, int port) throws IOException {
        InetSocketAddress isa = new InetSocketAddress(ip, port);
        ServerSocket server = new ServerSocket();
        server.bind(isa, 50);
        return server;
    }
```

6.5. Gestión de concurrencia

Para gestionar varias peticiones SMTP al mismo tiempo se ha optado por el uso de la concurrencia mediante los Threads en Java, para ello se ha decidido que cada conexión TCP al servidor se tratara como un hilo independiente, a continuación se muestra un ejemplo de como se crea el hilo de la petición, el método start() se encarga de generarlo:

```
Socket socket = server.accept();
SocketThread st = new SocketThread(socket);
Thread t = new Thread(st);
t.start();
```

En cada hilo se extraen los buffers de entrada y salida para poder intercambiar los mensajes cliente-servidor, estos utilizan el charset UTF_8, ya que es el más usado por los clientes SMTP [20], a continuación podemos observar una implementación:

```
public void run() {
    try {
        final BufferedReader in = new BufferedReader(new InputStreamReader(socket.
    getInputStream(), StandardCharsets.UTF_8));
        final BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.
    getOutputStream(), StandardCharsets.UTF_8));

        SMTProtocol protocol_handler = new SMTProtocol(in, out);
        protocol_handler.handle();

    } catch (IOException e) {
        try {
            socket.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

6.6. Implementación del protocolo SMTP en Java

Tras crear el hilo de la comunicación se procede a realizar la comunicación SMTP, para ello se han adaptado las directrices del RFC 5321 [12]. Para empezar nuestro servidor envía el mensaje de bienvenida y a continuación va respondiendo a todos los comandos que el cliente va indicando:

```
public void handle() throws IOException {
    String readed;
    System.out.println("(" + this.getActualTime() +  " ID " + this.id + ") STARTED
    HANDLING NEW REQUEST");
    send("220 Hola buenas soy el servidor");

    while (true) {
        readed = this.read();

        if (readed == null) {
            throw new IOException("close socket");
        }

        response(readed);
    }

}
```

A continuación se muestra como se han implementado todos los comandos del protocolo y sus respuestas, podemos encontrar estos códigos explicados en la sección **3.1 Protocolo SMTP**:

```
private void response(String message) throws IOException {
    String opcode = split_message(message);

    if (opcode.equalsIgnoreCase("EHLO")) {
        if (this.Ehlo) {
            mail.clear();
        }
        this.send("250 ubmail");
        this.Ehlo = true;
    } else if (!this.Ehlo) {
        this.send("503 Invalid secuence of commands");
    } else if (opcode.equalsIgnoreCase("VRFY")) {
        this.send("250 OK");
    } else if (opcode.equalsIgnoreCase("MAIL")) {
        mail.clearAndSetMail_from(message);
        this.send("250 OK");
    } else if (opcode.equalsIgnoreCase("RCPT")) {
        mail.clearAndAddMail_to(message);
        this.send("250 OK");
    } else if (opcode.equalsIgnoreCase("RSET")) {
        this.send("250 OK");
        mail.clear();
    } else if (opcode.equalsIgnoreCase("NOOP")) {
        this.send("250 OK");
    } else if (opcode.equalsIgnoreCase("DATA")) {
        this.send("354 OK");
```

```
    } else if (opcode.equalsIgnoreCase(".")) {
        this.send("250 OK");
    } else if (opcode.equalsIgnoreCase("QUIT")) {
        this.send("221 Bye");
        // do the checks
        try {
            System.out.println("(" + this.getActualTime() +  " ID " + this.id + ") MAIL
    RECEIVED FROM " + mail.getMailFrom());
            performPostMail();
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }

        throw new IOException("close socket");
    } else {
        mail.addData(message);
    }
}
```

6.7. Implementación de comparadores de URLs

Los comparadores de URL simplemente sirven para detectar posibles URL maliciosas, el servidor por defecto tiene una lista con los dominios que se quiere comprobar, por ejemplo el dominio de una empresa, también existe una lista de dominios peligrosos 'banned'. Para comparar las distancias entre los dominios se ha implementado el algoritmo de Levenshtein. Antes de procesar las URL el servidor ha de identificar cuáles existen en el mensaje de correo, para ello se utiliza el boundary que nos indica donde empieza el texto de nuestro correo y donde finaliza y se va revisando uno a uno los elementos para identificar si existen URL.

A veces las URL se encuentran divididas en diversas líneas, cuando esto sucede se incluye el carácter -.al final de la línea. Entonces se extrae la URL hasta que este carácter ya no aparece o aparece algún indicador de fin de URL, como puede ser «"que nos indica que empieza otro elemento HTML o un espacio en blanco.

```
public String extractMessage() {
    String end_boundary, boundary = null;
    boolean reading = false;
    StringBuilder message = new StringBuilder();
    StringBuilder completeUrl = null;

    for (String line_string : data) {
        if (boundary != null) {
            if (reading) {
                if (line_string.contains(end_boundary)) {
                    reading = false;
                    boundary = null;
                } else {

                    String check = line_string.replaceAll("/r", "").replaceAll("/n", "")
    .replaceAll(" ", "");
                    String substring = "";

                    if (check.length() > 1) {
                         substring = check.trim().substring(check.length() - 1);
                    }

                    if (check.contains("http")  && completeUrl == null && substring.
    equalsIgnoreCase("=")) {

                        completeUrl = new StringBuilder();

                        String[] split = line_string.split(" ");
                        String word = split[split.length - 1];

                        word = word.replaceAll("/r", "").replaceAll("/n", "").replaceAll
    (" ", "").replaceAll("=3D", "=");

                        StringBuilder st = new StringBuilder(word);
                        st.deleteCharAt(st.length()-1);

                        completeUrl.append(st);

                    } else if (completeUrl != null){
```

```
                    if (line_string.contains(" ") || line_string.contains("<") ||
    line_string.contains("/r") || line_string.contains("/n")) {
                        String line = line_string.replaceAll(" ", "<");

                        completeUrl.append(line.split("<")[0]);
                        extractURL(completeUrl.toString());

                        completeUrl = null;
                    } else {
                        line_string = line_string.replaceAll("=3D", "=");

                        if (line_string.trim().substring(line_string.length()-1).
    equalsIgnoreCase("=")) {
                            StringBuilder st = new StringBuilder(line_string);
                            st.deleteCharAt(st.length()-1);
                            completeUrl.append(st);
                        } else {

                            completeUrl.append(line_string);
                        }
                    }
                }

                if (line_string.contains(" ") || line_string.contains("<") ||
    line_string.contains("/r") || line_string.contains("/n")) {
                    if (completeUrl == null) {
                        extractURL(line_string);
                    }
                }

                if (check.length() > 2 && substring.equalsIgnoreCase("=")) {
                    StringBuilder st = new StringBuilder(line_string);
                    st.deleteCharAt(st.length()-1);
                    line_string = st.toString();
                }

                message.append(line_string);

            }
        } else {
            if (line_string.contains(boundary)) {
                reading = true;
            }
        }
    } else if (line_string.contains("boundary")) {
        boundary = line_string.split("boundary=")[1].replaceAll("\"", "");
        end_boundary = boundary + "--";
    }
}
return message.toString();
}
```

A continuación, cuando hemos obtenido la lista de URL que pertenecen al correo nos interesa extraer solo el dominio de estas, ya que es lo que vamos a emplear para el comprobador, para ello utilizamos una función extractDomain() que separa los elementos de las URL. Esto se puede hacer gracias a que todas las URL siguen el mismo formato (<protocolo>://<dominio>/<carpeta>), entonces solo debemos eliminar todo lo anterior a la // y lo posterior a la primera /, a continuación se muestra el código encargado de realizar esta extracción:

```
private String extractDomainV2(String url) throws Exception {
        String[] split = url.split("//");

        if (split.length == 0) {
            throw new Exception("Not real domain");
        }

        URL uri = new URL(url);

        String check = uri.getHost();
        String substring;

        if (check.length() > 1) {
            substring = check.trim().substring(check.length() - 1);
            if (!substring.equalsIgnoreCase("=")) {
                return check;
            }
        }
        throw new IOException("not domain found");
    }
```

A continuación se muestra un ejemplo de funcionamiento, consideramos que hemos añadido a la lista de dominios a comprobar www.ub.edu y queremos bannear el dominio www.upc.edu. Entonces recibimos un correo con diferentes links, las URL que sean similares aparecerán como SIMILAR, las URL que son legítimas aparecerán como legitim link y por último las que no tengan ninguna similitud se indicaran así. Podemos ver el resultado de este ejemplo en la **Figura 39**.

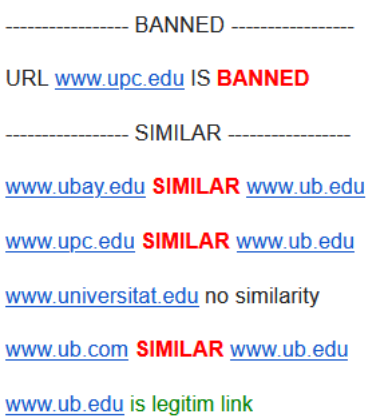

Figura 39: Resultado del comparador de URL

6.8. Implementación de la Blacklist

Para la implementación de la blacklist y sistemas de reputación hemos usado la API mencionada en la seccion **2.2 BlacklistMaster**, para usar esta API debemos crear una cuenta en la su web para así obtener una API_key **Figura 40**. Esta API en su versión gratuita solo nos permite 200 consultas por cuenta, esto es una limitación, pero en caso de un uso profesional se debería considerar adquirir un plan anual de BlackListMaster.

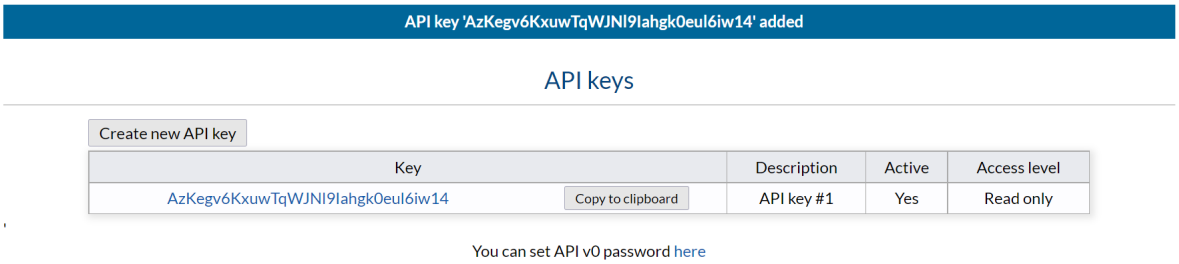

Figura 40: API key de blacklistmaster

Una vez generada nuestra llave blacklistmaster nos permite consultar al reputación de direcciones IPv4 y dominios con la API, para ello podemos usar consultas HTTP GET mediantes diferentes URL, un ejemplo de consulta de una URL 'peligrosa' se muestra en la **Figura 41**.

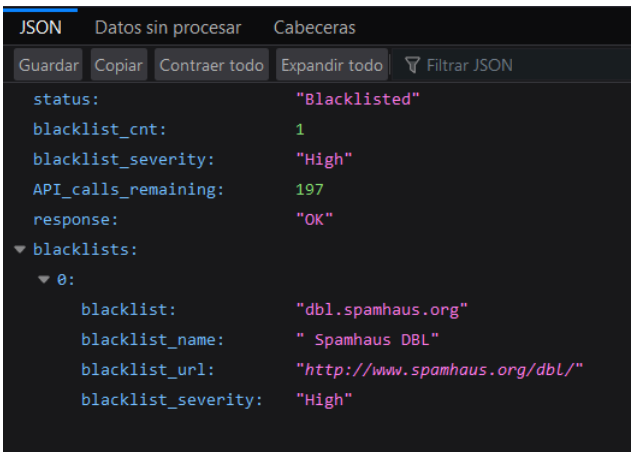

Figura 41: Consulta a la API de blacklistmaster

Para implementar la blacklist en Java, simplemente se ha realizado una petición HTTP a la API utilizando la API key y el dominio que se quiere consultar, se han extraído los datos de la respuesta HTTP en formato JSON, que por comodidad han sido transformados a una objeto MAP de Java:

```
private final String api_key = BlacklistYaml.getInstance().getKey();

private String formatURL(String domain) {
    return "https://www.blacklistmaster.com/restapi/v1/blacklistcheck/domain/" + domain
    + "?apikey=" + this.api_key;
}

public Map<String, String> checkDomain(String domain) throws IOException {

    StringBuilder result = new StringBuilder();

    URL url = new URL(formatURL(domain));

    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");

    try (BufferedReader reader = new BufferedReader(
            new InputStreamReader(conn.getInputStream()))) {
        for (String line; (line = reader.readLine()) != null; ) {
            result.append(line);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return convertBlacklistToMap(result.toString());
}

private Map<String, String> convertBlacklistToMap(String data) {
    data = data.substring(1, data.length()-1);
    String[] keyValuePairs = data.split(",");
    Map<String,String> map = new HashMap<>();

    for(String pair : keyValuePairs) {
        String[] entry = pair.split(":");
        map.put(entry[0].replaceAll("\"", ""), entry[1].replaceAll("\"", ""));
    }

    return map;
}
```

6.9. Implementación de descubridor de redirecciones y recortadores URL

Para comprobar si una URL realmente es el destino final, el servidor se conecta utilizando el protocolo HTTP a dicha URL, asi se obtiene la dirección real y se puede comprobar si hay una redirección.

```
ArrayList<String> copy = new ArrayList<>();

for (String uri : this.urls) {
    if (blackListUtils.checkShortener(uri) || uri.contains("ct.sendgrid.net")) {
        if (!uri.contains("ct.sendgrid.net")) {
            this.shorten_urls.add(uri);
        }

        String real_url;
        try {
            real_url = blackListUtils.getRealURL(uri);

            if (real_url == null) {
                real_url = blackListUtils.getRealURLV2(uri);

                if (real_url == null) {
                    continue;
                }

            }

            if (!uri.equalsIgnoreCase(real_url) && !real_url.equals("")) {
                this.shorten.put(uri, real_url);
                copy.add(real_url);
            }
        } catch (IOException ignored) {}
    }
}
```

En el código que se muestra se puede observar que primero se comprueba si la URL está utilizando un sistema de recorte de URL, posteriormente se identifica la URL a la que está redirigiendo y se añade a la lista de URL del servicio.

```
public String getRealURL(String link) {
    HttpURLConnection conn;
    try {
        URL inputURL = new URL(link);
        conn = (HttpURLConnection) inputURL.openConnection();
        conn.getHeaderFields();
        return conn.getHeaderField("Location");
    } catch(Exception e) {
        return null;
    }
}
```

En el caso de que este método no sea capaz de detectar la redirección, se utiliza otro método que realiza una consulta a la API onesimpleapi para determinar la URL final, en el caso de usar este método debemos registrarnos en onesimpleapi y generar una API key.

```
public String getRealURLV2(String shorten) throws IOException {
    String destiny = "https://onesimpleapi.com/api/unshorten?token=" + this.shorten_key
    + "&url=" + shorten;

    StringBuilder result = new StringBuilder();

    URL url = new URL(destiny);

    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");

    try (BufferedReader reader = new BufferedReader(
            new InputStreamReader(conn.getInputStream()))) {
        for (String line; (line = reader.readLine()) != null; ) {
            result.append(line);
        }
    } catch (Exception e) {
        return null;
    }
    return result.toString();
}
```

6.10. Ficheros de configuracion YAML

Para facilitar la configuración del servidor sin tener que compilar el código cada vez que se realiza un cambio, se ha añadido un archivo llamado config.yml que nos permite configurar ciertos parámetros para la ejecución del servidor, a continuación se muestra el contenido del archivo config.yml:

```
server:
  ip: "54.36.191.29"
  port: 25
  ssl_port: 465
  tls_port: 587
domain_check:
  sensitive: 5
blacklist:
  api_key: "AzKegv6KxuwTqWJNl9Iahgk0eul6iw14"
shorten:
  api_key: "64tNM9WaiGe6EX23cGSI6ZDfqRtqPgJlF866CAcW"
ssl:
  certificate_path: "/home/debian/java/crtf/ubmail.jks"
  pass: "ubmail"
mail:
  username: "postmaster@darkhorizon.es"
  password: "pswd"
mysql:
  host: "host"
  port: 3306
  username: "root"
  password: "psswd"
  database: "ubmail"
```

Como se puede observar, existen varios parámetros a configurar, entre ellos:

- server - nos permite configurar el endpoint del servidor. Indicamos la dirección IPv4 en el apartado IP, y los puertos en los que queremos establecer el servidor.
- domain_check - nos permite configurar la sensibilidad del comprobador de dominios similares.
- blacklist - se incluye la API key de blacklistmaster.
- shorten - se incluye la API key de onesimpleapi.
- ssl - nos permite modificar nuestro certificado para el uso de TLS/SSL.
- mail - configuración del mail para el envío del reporte.
- mysql - credenciales de la base de datos MySQL.

7. Simulación de un caso

Para simular el caso vamos a crear un escenario donde un estudiante de la UPC quiere robar las credenciales del campus de un usuario de la UB. Para ello este estudiante de la UPC genera un correo muy sencillo para ver si el otro cae en la trampa, podemos observar el correo en la **Figura 42**, este ataque seria de tipo phishing dirigido (Spear).

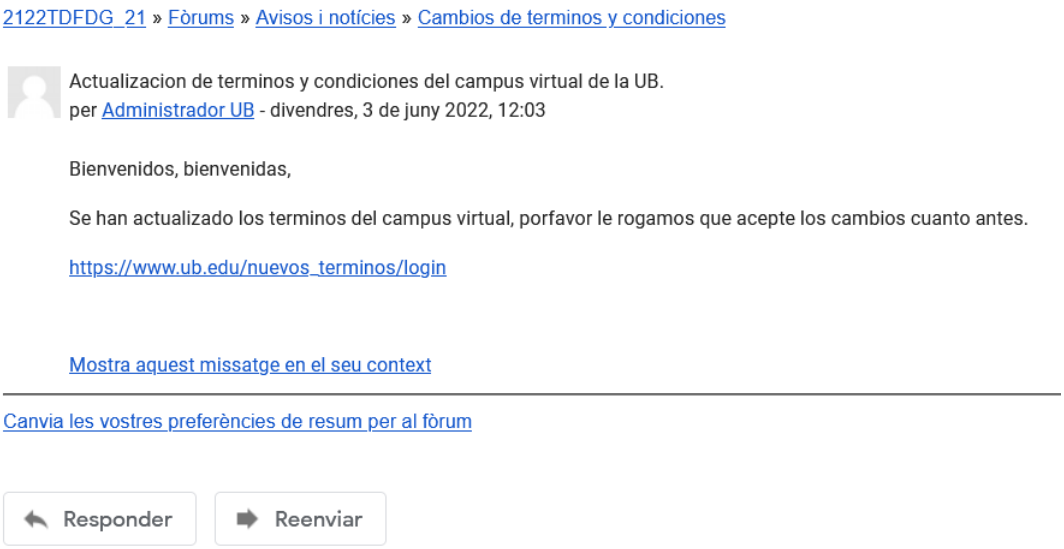

Figura 42: Mail malicioso

A primera vista el estudiante de la UB pensará que este email es legítimo, ya que como se puede observar el correo es muy similar a todos los que ha recibido del campus de la universidad, pero este correo oculta un link malicioso, la URL que se muestra como https://www.ub.edu/nuevos_terminos/login realmente está dirigida a https://www.upc.edu/login.

Por suerte el estudiante que está siendo atacado tiene configurada la redirección de correos a nuestro servidor, por lo que antes de abrir el correo comprueba el reporte correspondiente al email que ha recibido, la víctima puede observar en el reporte, que se muestra en la **Figura 43**, ciertas indicaciones que le llaman la atención:

- El cliente espera un mensaje de la UB, pero le parece sospechoso que la página de inicio de la UPC esté incluida en el correo y además no se muestra en el correo, sino que está oculta en un href.
- Parece que hay una URL en la blacklist, sendgrid.net, esta es peligrosa debido a que GMAIL la utiliza para redirigir a otras URL.

Entonces el cliente concluye que este correo no es seguro, sino que es una trampa de un estudiante de la UPC que le está intentando robar las credenciales de inicio del campus de la UB.

Gmail **Arnau Gris Garcia <arnau.gris@gmail.com>**

MAIL REPORT FROM arnau.gris@gmail.com

postmaster@darkhorizon.es <postmaster@darkhorizon.es> 10 de junio de 2022, 12:38
Para: arnau.gris@gmail.com

REPORT FROM ANTI PHISHING AG.ES

---------- SUMMARY ----------

NUMBER OF URLS: 12

HIDDEN URLS: 1

SHORTEN URLS: 0/12

BLACKLISTED DOMAINS: 4

BANNED DOMAINS: 1

---------- SIMILAR ----------

www.upc.edu **SIMILAR** www.ub.edu

sso.ub.edu is legitim link

u17063761.ct.sendgrid.net no similarity

www.ub.edu is legitim link

---------- REAL URLS ----------

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wfkh6m1noKwYo1HczC-2B4LHPcuR0aQJic9ellCeMypPeaQ-3D-3DfFIM_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4f5OlmlMP2kReS7IjnbKVqDT7r-2BeUtKnAAFQJ5Caf9CNkGksarNjJdp fX9mc6NvfZ1QMRecbfxb3KDJ8xaqezqBX5NHPstB9CLc58SXniZzlTbggjZ9mb-2BGnJ-2FPqoP7XexT-2F9drbirXlC5rHerTea3c-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i58ru3JCkEFT-2BF-2F9pfY2WgcDEjQFPrEhutik4EHymrryc0-3DCtOB_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4WWLnROJN5OpDMUHMLbkhxChgQAcPN0Wrg8BByoqkW1HM7c XYl55DvO6NWfpLBJPIoR0Y3n78SifQGZlkMhGGhTbgtUoV2V2NCv72OGEfwC 07lD9cJPIgq18zyBxKhz62UTNEfxZ106wV1apc0D02mU-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i58ru3JCkEFT-2BF-2F9pfY2WgcDC5OIT1-2FMgwcNg30QD2kChQ-3DnFrl_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4UIU0CBcuaqknv0i-2FnlmBg3JpEV77cFnyIoQW34gzqaITkU4EjMjNIs-2B6yKpU31Ig8W-2B-2Bz0jQY4Ml-2F1JEP1tnSSdFC-2BIvZdQtmsXFfaE1PWzVI2VY2KjLqMNq-2FThQJZXFSfO8xrjZYonejYq34Ak1g8-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wfIPfroMA3eZgTs5SYzqPpTsASAawq87UYMTRsidU9HotIpstMlSHa5cuJkroFKxoc-3Dwy-l_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4WaiSnGa2gllM9ytY-2FUj7HyWu8umGJTD-2BrcG9-2FzcHU2p528rNFv3-2F-2BFp2tZwvI8yB4VH31TWhJ4BKKqPmI0qL-2BgLc0Xw3sLsqBrtrrvCBdu92aD0Ao AmIFgMtLLtkMUS4EWQyqHfbPSvk18hbkmx-2BDs-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i581YS5as4SZkl-2BSxlb4oaeH6kBmaNWA4tBV2KWNazDnfw-3DD2ac_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4TETYoJsPXO7zvTkDln4hJ-2BBuBAhO0NJlDKBBP4ZwBo3j3OQWIW IMH7hzq4-2B6E57PNd4yp-2FbmelOLotHuXf4etcdqh61rmg-2FcScriGh6TMbG3X91p6s1lw8EDJgK t1n7rTi070-2FpBydCc11Mwgtyu8M-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wfIPfroMA3eZgTs5SYzqPpTsASAawq87UYMTRsidU9HotlpstMlSHa5cuJkroFKxoc-3D4sol_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4S25hX-2BTstfpWdAFhH3nHRRuY5bIb-2B1DU3yLz2z-2Bhb-2B675-2F6aqrcMO8yzEU0u1G1SsekkRg4PLsLeJ4Ho2CbsvLuQaMi1Z483uap-2BvBSIjFyv6MB-2BaGIRpOqlgYSQ9I3rPvHVmn8mWmLKQrmLuoh5pI-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i58juMGF2wYjpG4Q9gKcpUkOfyo6t1wNYIwRHVkVoLuw4s-3DoiLZ_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4URymcqEFB9EkmQPbPFqFJ3V-2FRE-2FPdHVjVyLdA3rzoXxlPQIYu3RO7Mny4Us-2F16dWKYvlwWZWg1Zfc5GxMqttDUOB48-2Ba-2BmIri3oHAQ647IkkXFI9Cf-2FqLSSigIE-2BNdPk3nme2tzUJ-2BrW-2F70129T2Po-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

URL https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i581YS5as4SZkl-2BSxlb4oaeH6kBmaNWA4tBV2KWNazDnfw-3DS8DZ_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4T4Je5tCvLHKjIcu7Qjpn5UXZlyoFxV2Qkry6-2FR0bl9IGt6Qqngn-2BQ6xKMa6iWadNGgIv-2FHEQ1mwWq-2B9bDKkThG66x-2BIv9UqMlapcWHTot33rQKMv1HuwFx0kTnD2vrqUOA6wVUX94CfeP-2Fg1qaMAus-3D IN REAL IS https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

---------- HIDDEN ----------

https://www.upc.edu/login NOT APPEARS AS MAIL TEXT

---------- BLACKLIST ----------

URL u17063761.ct.sendgrid.net IS **BLACKLISTED**

---------- BANNED ----------

URL www.upc.edu IS **BANNED**

---------- SHORTEN ----------

---------- URLS ----------

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wfkh6m1noKwYo1HczC-2B4LHPcuR0aQJic9ellCeMypPeaQ-3D-3DfFIM_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4f5OlmlMP2kReS7IjnbKVqDT7r-2BeUtKnAAFQJ5Caf9CNkGksarNjJdp fX9mc6NvfZ1QMRecbfxb3KDJ8xaqezqBX5NHPstB9CLc58SXniZzlTbggjZ9mb-2BGnJ-2FPqoP7XexT-2F9drbirXlC5rHerTea3c-3D

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i581YS5as4SZkl-2BSxlb4oaeH6kBmaNWA4tBV2KWNazDnfw-3DD2ac_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4TETYoJsPXO7zvTkDln4hJ-2BBuBAhO0NJlDKBBP4ZwBo3j3OQWIW IMH7hzq4-2B6E57PNd4yp-2FbmelOLotHuXf4etcdqh61rmg-2FcScriGh6TMbG3X91p6s1lw8EDJgK t1n7rTi070-2FpBydCc11Mwgtyu8M-3D

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i58juMGF2wYjpG4Q9gKcpUkOfyo6t1wNYIwRHVkVoLuw4s-3DoiLZ_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4URymcqEFB9EkmQPbPFqFJ3V-2FRE-2FPdHVjVyLdA3rzoXxlPQIYu3RO7Mny4Us-2F16dWKYvlwWZWg1Zfc5GxMqttDUOB48-2Ba-2BmIri3oHAQ647IkkXFI9Cf-2FqLSSigIE-2BNdPk3nme2tzUJ-2BrW-2F70129T2Po-3D

Gmail - MAIL REPORT FROM arnau.gris@gmail.com https://mail.google.com/mail/u/1/?ik=47214659d3&view=pt&search...

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i58ru3JCkEFT-2BF-2F9pfY2WgcDEjQFPrEhutik4EHymrryc0-3DCtOB_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4WWLnROJN5OpDMUHMLbkhxChgQAcPN0Wrg8BByoqkW1HM7c
XYl55DvO6NWfpLBJPIoR0Y3n78SifQGZlkMhGGhTbgtUoV2V2NCv72OGEfwC
07lD9cJPIgq18zyBxKhz62UTNEfxZ106wV1apc0D02mU-3D

https://u17063761.ct.sendgrid.net/ls/click?upn=3DWBL=

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wfIPfroMA3eZgTs5SYzqPpTsASAawq87UYMTRsidU9HotlpstMlSHa5cuJkroFKxoc-3Dwy-l_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4WaiSnGa2gllM9ytY-2FUj7HyWu8umGJTD-2BrcG9-2FzcHU2p528rNFv3-2F-2BFp2tZwvl8yB4VH31TWhJ4BKKqPml0qL-2BgLc0Xw3sLsqBrtrrvCBdu92aD0Ao
AmIFgMtLLtkMUS4EWQyqHfbPSvk18hbkmx-2BDs-3D

https://www.ub.edu/nuevos_terminos/login

https://www.upc.edu/login

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i58ru3JCkEFT-2BF-2F9pfY2WgcDC5OIT1-2FMgwcNg30QD2kChQ-3DnFrl_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9
HO-2F-2B5z-2FfgRT6molGBJ4UIU0CBcuaqknv0i-2FnlmBg3JpEV77cFnyIoQW34gzqalTkU4EjMjNIs-2B6yKpU31lg8W-2B-2Bz0jQY4Ml-2F1JEP1tnSSdFC-2BIvZdQtmsXFfaE1PWzVl2VY2KjLqMNq-2FThQJZXFSfO8xrjZYonejYq34Ak1g8-3D

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wcu8QBYyyzkg1HcWZRO2i581YS5as4SZkl-2BSxlb4oaeH6kBmaNWA4tBV2KWNazDnfw-3DS8DZ_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4T4Je5tCvLHKjIcu7Qjpn5UXZlyoFxV2Qkry6-2FR0bl9IGt6Qqngn-2BQ6xKMa6iWadNGgIv-2FHEQ1mwWq-2B9bDKkThG66x-2BIv9UqMIapcWHTot33rQKMv1HuwFx0kTnD2vrqUOA6wVUX94CfeP-2Fg1qaMAus-3D

https://u17063761.ct.sendgrid.net/ls/click?upn=WBLqlOEcPJ4MT-2F1w9AVnTnRtUo0Ef0nx3MYAC-2BPj0wfIPfroMA3eZgTs5SYzqPpTsASAawq87UYMTRsidU9HotlpstMlSHa5cuJkroFKxoc-3D4sol_MQ-2BdZzAEi-2B85suYdHipVjQUhSVM9KUh87N6vp-2FntwVpLzxzCsYBmwxXUb3Zir1Vhv9HO-2F-2B5z-2FfgRT6molGBJ4S25hX-2BTstfpWdAFhH3nHRRuY5bIb-2B1DU3yLz2z-2Bhb-2B675-2F6aqrcMO8yzEU0u1G1SsekkRg4PLsLeJ4Ho2CbsvLuQaMi1Z483uap-2BvBSIjFyv6MB-2BaGIRpOqlgYSQ9I3rPvHVmn8mWmLKQrmLuoh5pl-3D

https://sso.ub.edu/CAS/index.php/login?service=https%3A%2F%2Fcampusvirtual.ub.edu%2Flogin%2Findex.php&gateway=true

Original Message RAW

Content-Type: text/plain; charset="UTF-8"Content-Transfer-Encoding: quoted-printableEl vie, 10 jun 2022 a las 12:29, Arnau Gris Garcia ()escribi=C3=B3:>>> El vie, 10 jun 2022 a las 12:20, Arnau Gris Garcia ()> escribi=C3=B3:>>>>>>> El vie, 10 jun 2022 a las 12:18, Arnau Gris Garcia ()>> escribi=C3=B3:>>>>>>>>>>> El vie, 10 jun 2022 a las 12:15, Arnau Gris Garcia (<>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 12:08, Arnau Gris Garcia (<>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 11:52, Arnau Gris Garcia (<>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 11:48, Arnau Gris Garcia (<>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 11:43, Arnau Gris Garcia (<>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 11:38, Arnau Gris Garcia (<>>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 11:35, Arnau Gris Garcia (<>>>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>> >>>>>>>>>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 11:33, Arnau Gris Garcia (<>>>>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>> El vie, 10 jun 2022 a las 11:28, Arnau Gris Garcia (<>>>>>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>> El vie, 10 jun 2022 a las 11:20, Arnau Gris Garcia (<>>>>>>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> El vie, 10 jun 2022 a las 11:16, Arnau Gris Garcia (<>>>>>>>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> El jue, 9 jun 2022 a las 15:46, Arnau Gris Garcia (<>>>>>>>>>>>>>> arnau.gris@gmail.com>) escribi=C3=B3:>>>>>>>>>>>>>>>> >>> El jue, 9 jun 2022 a las 15:17, Arnau Gris Garcia

3 de 4 10/06/2022, 12:46

Figura 43: Reporte del mail malicioso

8. Conclusiones y trabajo futuro

Finalmente podemos decir que el proyecto cumple con todos los requisitos y funcionalidades acordadas previamente. También se han cumplido todos los objetivos propuestos para este trabajo. El servidor finalmente puede ser útil para usuarios que quieran tener una barrera más contra el phishing gracias al reporte generado de cada correo y la información que este contiene.

8.1. Problemas encontrados

A continuación se exponen los problemas encontrados durante el desarrollo de este proyecto.

Problemas de rendimiento

Debido a las prestaciones bajas del servicio de hosting contratado, el servidor no es capaz de gestionar muchas peticiones a la vez, ya que este solo consta de 2 hilos en la CPU. También el host al tener un bajo ancho de banda es lento al realizar las peticiones HTTP, lo que incrementa el tiempo de respuesta del servidor desarrollado.

8.2. Propuestas de ampliación

El servidor de correo desarrollado en este trabajo de campo es ampliable de varias formas:

Modelo de predicción de correos maliciosos

El servidor ya incluye algunas funcionalidades que nos permiten detectar si se trata de un correo phishing, pero estas funcionalidades se puede ampliar, por ejemplo añadiendo al código algún modelo de ML (Aprendizaje automático) que a través de un dataset de correos se pueda determinar según el contenido de uno si otro correo puede ser identificado como phishing, a través de parámetros como la forma en que está escrito, la colocación de los elementos, las referencias que incluye el correo...

Módulo de detección de malware

Una propuesta de ampliación es añadir a este servidor SMTP algún módulo que permita identificar si algún archivo adjunto puede ser considerado malware o archivo malicioso.

Edición del correo

Una idea futura para el proyecto consiste en realizar cambios en el correo recibido, en cambio de generar un reporte, las indicaciones se muestran en el mismo contenido del correo a analizar.

Servidor distribuido

En cambio, de utilizar un único servidor centralizado, se puede idear una versión de este proyecto donde el modelo de este servidor sea distribuido.

9. Manual del proyecto

9.1. Instalación

Antes de empezar con la instalación del proyecto debemos cumplir los siguientes requisitos:

1. Tener disponible un acceso con privilegios de administrador (sudo) en una máquina con un sistema operativo Linux, a poder ser la distribución Debian.
2. Descargar el código fuente del proyecto, en caso de no estar disponible clonar del repositorio de Github [23].
3. Tener una base de datos MySQL remota.
4. Tener un dominio que permita modificar sus registros MX y una cuenta de correo en ese mismo dominio.
5. Generar un mail encargado de enviar los reportes.
6. Tener instalado Java y Maven en el host remoto.
7. (Opcional) Tener un certificado (CA) propio.

Configuración del servidor

Para configurar los aspectos del servidor debemos abrir el archivo config.yml que se encuentra dentro de la raíz config y rellenar todos los aspectos necesarios, estos se detallan en la sección **6.10 Ficheros de configuracion YAML**. Previamente, necesitamos tener el código clonado en nuestro host remoto.

Configurar el firewall

Debemos configurar el firewall de nuestro host remoto para aceptar las peticiones SMTP en los puertos determinados en config.yml. Esto se encuentra detallado en la sección **6.1 Configuración del Firewall UFW**.

Configurar el dominio

Para configurar el registro MX en nuestro dominio necesitamos acceder a nuestro hosting correspondiente al dominio y modificar el registro MX para que apunte a la dirección de nuestro servidor.

Configurar la base de datos

Para generar el esquema de la base de datos se incluye el archivo db.sql que se encuentra en la carpeta SQL, simplemente se deben copiar las instrucciones que contiene el archivo en el terminal de la base de datos o con MySQL WorkBench.

Puesta en marcha

Para poner el servidor en marcha solo necesitamos ejecutar 2 comandas Maven en el terminal.

```
mvn compile
sudo mvn exec:java -Dexec.mainClass=es.arnaugris.main
```

Prueba de funcionamiento

Para verificar que el servicio funcione correctamente podemos enviar una petición mail a cualquier correo electrónico del dominio que hemos configurado para que apunte a nuestro servidor. Pasados unos minutos llegará el reporte del mail enviado a esta dirección de correo. También podemos correr algunos test unitarios para comprobar que el servidor funciona correctamente, estos los podemos encontrar en el repositorio [24].

10. Apéndice

A continuación se introducen conceptos teóricos relacionados con elementos del proyecto.

10.1. Protocolo

Conjunto de reglas que se establecen en el proceso de comunicación entre dos sistemas [25].

10.2. Modelo OSI

El modelo OSI [27] es un modelo conceptual, creado por la Organización Internacional de Normalización (ISO), que permite que diferentes sistemas se comuniquen utilizando protocolos estándar. Este se divide en 7 capas las cuales se introducirán a continuación:

Capa de aplicación

Es la capa que más protocolos contiene, estos protocolos son utilizados por el software para enviar y recibir información importante para los usuarios, ejemplos de protocolos de esta capa pueden ser HTTP, SMTP, IMAP, DNS, POP3...

Capa de presentación

Esta capa se encarga de 'presentar' los datos para la capa de aplicación, se define como los datos se deben codificar, encriptar y comprimir para que sean recibidos correctamente por el otro lado de la conexión. Prepara los datos para la capa de sesión.

Capa de sesión

Esta capa es la encargada de crear canales de comunicación, más conocidos como sesiones. También se encarga de mantener esta conexión abierta y funcional mientras se están transmitiendo los datos.

Capa de transporte

Esta capa obtiene los datos de la capa de sesión y los separa en fragmentos, también se encarga de volver a juntar estos fragmentos convirtiendo estos en los datos originales. Esta capa incluye control de errores, comprobaciones de sí los datos se han recibido y si han sido recibidos correctamente, también incluye control de flujo.

Capa de red

Esta capa se encarga de separar los fragmentos de la capa de transporte en paquetes de red, y de volver a juntar estos paquetes. Esta capa suele utilizar direcciones de red, comúnmente se utilizan direcciones IP.

Capa de enlace

La capa de enlace establece y finaliza una conexión entre dos dispositivos conectados físicamente, esta separa los paquetes en frames. Emplea las direcciones MAC (Media Access Control) para conectar los dispositivos y definir permisos para transmitir y recibir datos.

Capa física

Básicamente la capa física se puede entender como el cable o la conexión wireless entre dos nodos. Es el encargado de transmitir los datos en 'raw' es decir en series de 0's y 1's.

10.3. Modelo TCP/IP

El modelo TCP/IP, más usado en la actualidad, se comprende de 4 capas:

- Capa de aplicación, comprende las capas 7, 6 y 5 del modelo OSI.
- Capa de transporte, es la misma que en el modelo OSI.
- Capa de red, es igual a la capa 3 del modelo OSI.
- Capa de enlace/MAC, comprende las capas 2 y 1 del modelo OSI.

10.4. Protocolo DNS

Existen 2 maneras de identificar a un host (ordenador), mediante un nombre de host (dominio) o mediante una dirección IP. Las personas prefieren utilizar un nombre de host, ya que es mucho más fácil memorizar google.com que una IP como 142.250.200.78, mientras que los dispositivos de enrutacion, routers, prefieren utilizar las direcciones IP, puesto que tienen una longitud fija y siguen una estructura jerárquica. Entonces, la tarea principal del protocolo de la capa de aplicación DNS (Domain Name System), definido en el RFC 1034 [28], es traducir un nombre de host a una dirección IP.

DNS es una base de datos distribuida implementada en una jerarquía de servidores DNS y a la vez también es un protocolo de la capa de aplicación que permite a los hosts consultar esta base de datos. Este protocolo se ejecuta con UDP y emplea el puerto 53.

10.4.1. Otras funciones de DNS

DNS también ofrece algunas otras funciones aparte de la traducción de nombres de host a IP:

Alias de host

Un host puede tener uno o más alias, por ejemplo google.com puede tener algunos alias como www.google.com o search.google.com. Entonces google.com se conoce como nombre de host canónico. Una aplicación DNS se puede usar para obtener el nombre de host canónico de un determinado alias, asi como su respectiva dirección IP.

Alias del servidor de correo

Por defecto las direcciones de correo no son canónicas, una aplicación DNS puede utilizar el servicio DNS para obtener el nombre del host canónico de una dirección de correo gracias al registro MX.

Distribucion de carga

DNS también se emplea como distribuidor de carga, es decir, pongamos que google.com está replicado en una cantidad de servidores con diferente dirección IP, cuando un cliente decide consultar google.com el servidor responderá con la dirección IP que tenga menos carga en ese momento, es decir DNS se encargara de distribuir el tráfico del servicio google.com.

A continuación se presenta a alto nivel como funciona DNS **Figura 44**, para empezar el cliente especifica que nombre de host necesita traducir, entonces la aplicación DNS en el host del usuario envía la consulta a la red, todos los mensajes son intercambiados usando datagramas UDP en el puerto 53. A continuación, el servicio DNS del cliente recibe un mensaje de respuesta DNS que proporciona la traducción del nombre de host indicado.

Figura 44: Ejemplo simplificado de petición DNS

Cabe remarcar que DNS es un servicio implementado de forma muy compleja, ya que este servicio está distribuido por un grande número de servidores DNS alrededor del mundo. Un diseño simple de DNS seria uno en que un único servidor contuviera todas las traducciones de nombres de host, aunque esto parece una implementación viable, no lo es para nada, puesto que al tratarse de un sistema centralizado aparecen ciertos problemas, como pueden ser:

- Si el servidor falla, toda la red de internet se quedaría sin servicio DNS.
- El servidor tendría que tener una capacidad para administrar todas las consultas DNS de la red.
- Al tratarse de un servicio centralizado, los lugares alejados geográficamente de este servidor tendrian que realizar consultas a través de una distancia muy larga, por lo que se generarían retardos indesados.
- El servidor debería mantener registro de todos los hosts de Internet.

En resumen, el principal problema de esta idea de DNS centralizado es que este no podría escalarse.

10.4.2. DNS distribuido

Para solucionar el problema de la escalabilidad de una implementación centralizada, existen millones de servidores DNS organizados de forma jerárquica y repartidos por todo el planeta. Podemos decir que existen 3 tipos de servidores DNS:

- Servidores DNS raiz.
- Servidores DNS Top-Level Domain (TLD).
- Servidores DNS autoritativos.

Para entender como interactúan estos tipos de servidor vamos a emplear el siguiente ejemplo, suponemos que el cliente DNS quiere resolver el nombre de host www.google.com, primero el cliente DNS se comunica con un servidor DNS raíz, a continuación este servidor raíz busca el servidor TLD correspondiente con la terminación .com, por último el servidor TLD busca el servidor autoritativo que corresponde a google.com, este servidor autoritativo contiene la información de los subdominios de google.com, asi que devuelve la dirección IP de google.com. Entonces podríamos representar el esquema jerárquico como se muestra en la **Figura 45**.

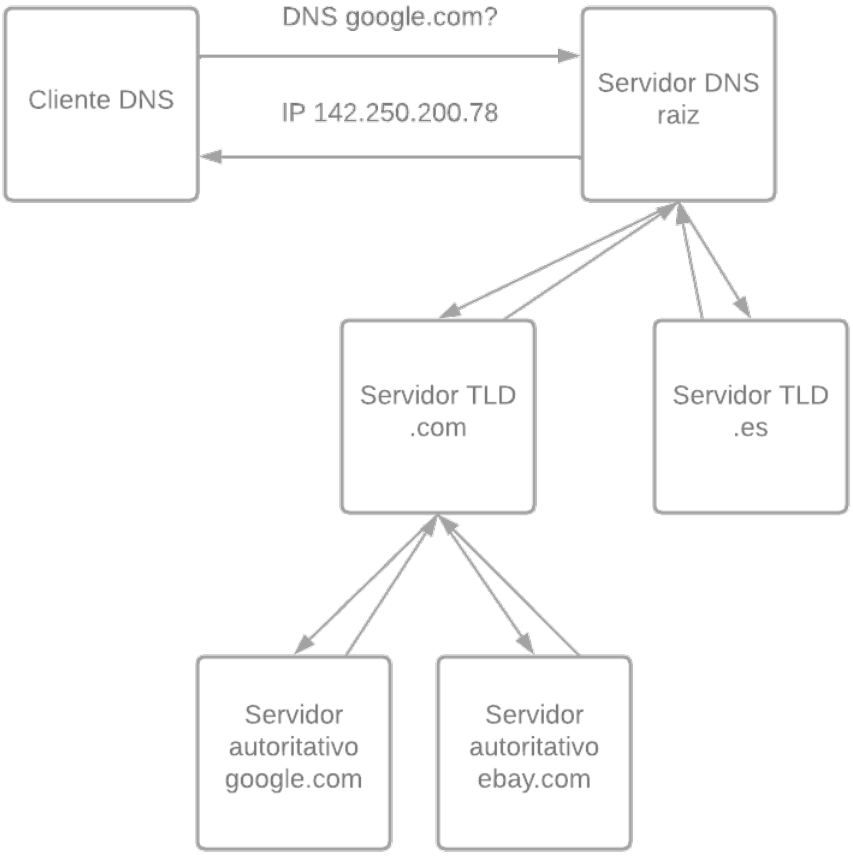

Figura 45: Ejemplo esquema jerárquico DNS

10.4.3. Paquete DNS

Todos los paquetes DNS siguen el mismo formato, este se muestra en la **Figura 46**.

Figura 46: Formato paquete DNS

A continuación entraremos en detalle en cada elemento del paquete DNS. Para empezar tenemos la cabecera que se compone de los campos mostrados en la **Figura 47**.

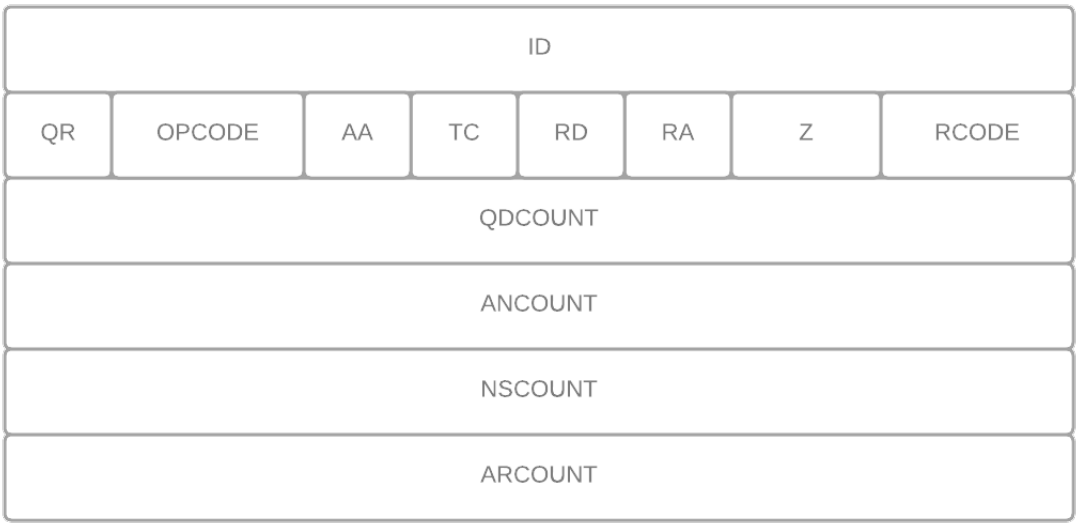

Figura 47: Formato Header DNS

Los campos en concreto son:

- ID - Es un identificador de 16 bits, este se incluye en la respuesta para identificar cuál es su pregunta correspondiente, este es generado por el servidor DNS.

- QR - Consta de 1 bit que identifica si se trata de una pregunta (0) o una respuesta (1).
- OPCODE - 4 bits que identifican de que tipo es el mensaje.
- AA - 1 bit, indica si el servidor que responde es autoritativo o no.
- TC - 1 bit, específica si el mensaje ha sido truncado.
- RD - 1 bit, determina si la petición se debe realizar de forma recursiva.
- RA - 1 bit, indica si la recursividad está disponible.
- Z - Reservado para un uso futuro.
- RCODE - 4 bits, incluye códigos de error: 0 No hay condición de error. 1 Error de formato. 2 No se ha procesado la petición por problemas del servidor. 3 El dominio iniciado no existe o no se ha encontrado. 4 El servidor no soporta la petición solicitada. 5 Petición denegada.
- QDCOUNT - 16 bits unsigned integer, específica el número de entradas en la sección de preguntas.
- ANCOUNT - 16 bits unsigned integer, índica el número de records que se incluye en la seccion de respuesta.
- NSCOUNT - 16 bits unsigned integer, define el número de registros autoritativos del servidor.
- ARCOUNT - 16 bits unsigned integer, define el número de parámetros adicionales.

El formato de la sección de pregunta se muestra en la **Figura 48**.

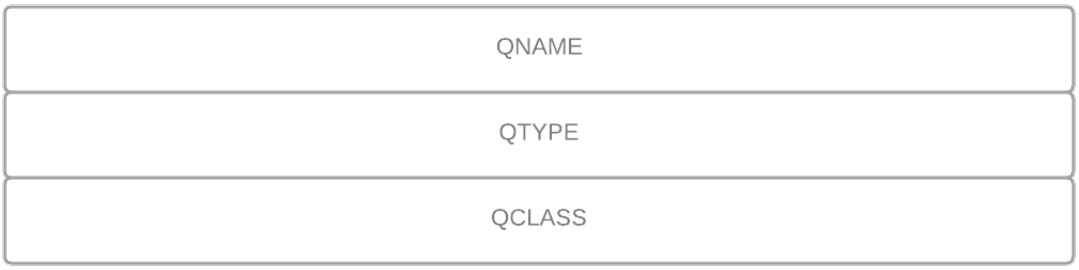

Figura 48: Formato pregunta DNS

Los campos de esta sección son:

- QNAME - Se indica el nombre del dominio separado por campos, los campos son octetos y se indica el fin de los campos con un octeto lleno de ceros.
- QTYPE - 2 octetos, se indica el tipo de registro por el que se está preguntando.
- QCLASS - 2 octetos, se indica la clase de consulta que se está realizando, por ejemplo 0x0001 se refiere a traducción de nombres de host.

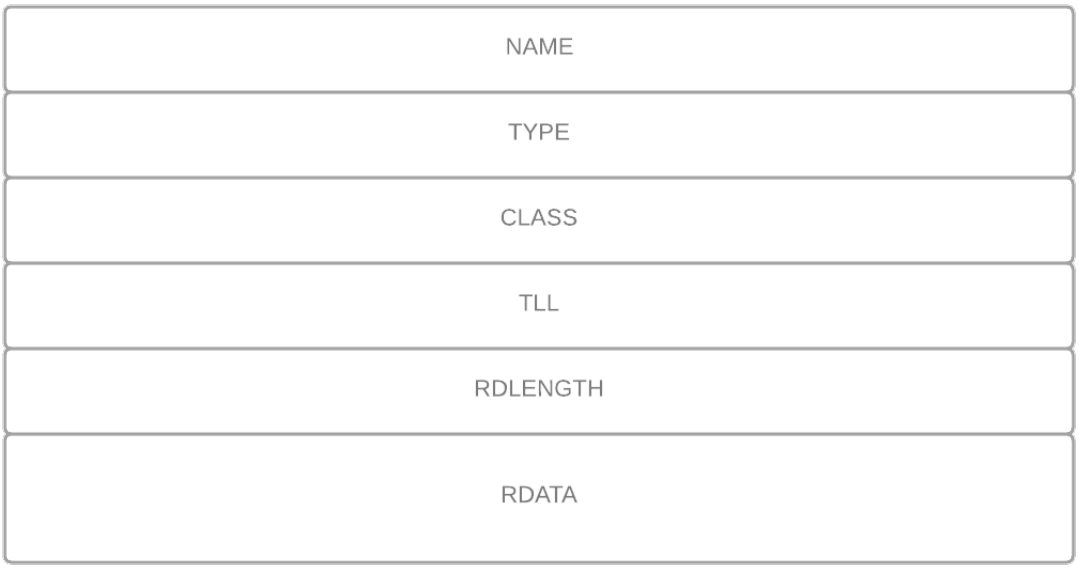

Figura 49: Formato respuesta DNS

Por último, el formato de la sección de respuesta se muestra en la **Figura 49**.

Los campos de esta sección son los siguientes:

- NAME - lo mismo que QNAME, es el dominio que se va a procesar.
- TYPE - 2 octetos, especifica el tipo de datos/registro del apartado RDATA.
- CLASS 2 octetos, especifica la clase de los datos del apartado RDATA.
- TTL - indica el tiempo que el resultado se guarda en caché.
- RDLENGTH - indica el tamaño del campo RDATA.
- RDATA - los datos de la respuesta, el formato varía según el TYPE.

Para reducir el tamaño de los datos, el servidor DNS se encarga de eliminar las repeticiones en los campos NAME, QNAME y RDATA.

10.4.4. Registros DNS

Un registro DNS debe cumplir 4 campos: Subdominio, Valor Destino, Tipo y TTL. Se conoce como TTL el tiempo que va a estar almacenado el registro en la caché. Existen varios tipos de registros, entre ellos diferenciamos:

- A - Resuelve una dirección IPv4.
- AAAA - Resuelve una dirección IPv6.
- CNAME - Contiene un nombre alternativo para el dominio.

- MX - Define los servidores de correo que corresponden al dominio.
- TXT - Contiene información en forma de texto.
- SRV - Contiene información sobre los servicios disponibles del dominio.
- LOC - Resuelve la localización geográfica del servidor.

Un registro de tipo A para www.google.com en la dirección IPv4 142.250.200.78, con TTL=0 seria:

```
www.google.com. 0 IN A 142.250.200.78
```

10.4.5. DNSSEC

DNSSEC [26] es la versión de DNS que incluye seguridad mediante criptografía asimétrica. El funcionamiento de DNSSEC es exactamente igual a DNS, solo que en el intercambio de mensajes se incluyen las firmas de las claves necesarias. Su utilización añade una capa extra de autenticidad sin ningún tipo de extensión de seguridad. Su uso también nos puede proteger de ataques de suplantación de identidad (phishing) o ataques de observación e interceptacion de tráfico.

10.5. Protocolo POP3

POP3 es un protocolo de la capa de aplicación definido en el RFC 1939 [29], POP3 utiliza TCP para la comunicación cliente-servidor **Figura 50**, POP3 se encuentra alojado por defecto en el puerto 110 y en el puerto 995 en caso de usar cifrado SSL, este protocolo está implementado en 3 fases: autorización, transacción y actualización.

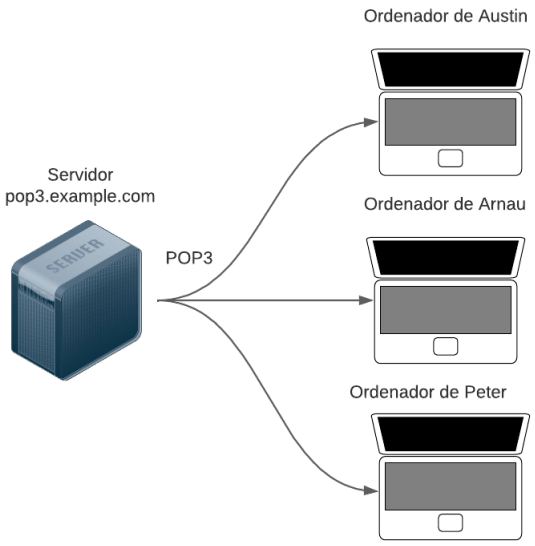

Figura 50: Ejemplo de comunicación POP3

Autorización

En esta fase el usuario se debe autenticar, para ello envía su nombre de usuario y la contraseña correspondiente. Esta fase se compone de 2 comandos de cliente principales:

- user <user>- el usuario indica su nombre de cuenta.
- pass <password>- el usuario indica la contraseña correspondiente a la cuenta.

El servidor responderá a estos comandos con:

- +OK - en caso de que todo fuera según lo esperado.
- -ERR - en caso de algún tipo de error o que se haya escrito un comando o secuencia invalida.

A continuación se muestra un ejemplo de comunicación de la fase de autorización:

```
telnet pop3.arnaugris.es 110
+OK POP3 conexión establecida
user arnau.gris@gmail.com
+OK
pass contrasenasegura123
+OK usuario logeado correctamente
```

Transacción

En esta fase el usuario obtiene los mensajes de correo, el usuario puede marcar los mensajes para borrado u obtener estadísticas del correo. Los comandos que puede utilizar el cliente en esta fase son:

- stat - se obtiene la cantidad de mensajes en el buzón.
- list - para obtener los mensajes no borrados y su longitud.
- top <num>- muestra la cabecera de un mensaje en específico.
- retr <num>- se obtiene el contenido de un mensaje en específico.
- dele <num>- se indica al servidor que se quiere eliminar un correo en específico.
- rset - se utiliza para restaurar los mensajes que se han solicitado eliminar en la sesión.
- quit - finaliza la sesión actual.

A continuación se muestra un ejemplo de comunicación de la fase de transacción:

```
C: list
S: 1 237
S: 2 480
S: 3 230
S: 4 579
S: .
C: retr 1
S: (...Hola arnau como estas, como te va todo...)
```

```
S: .
C: dele 1
C: retr 2
S: (...Arnau ayer no te vi por la universidad...)
S: .
C: dele 2
C: quit
S: +OK POP3 hasta la proxima
```

Actualizacion

Esta fase aparece al ejecutar el comando quit, y se borran los mensajes definidos en la fase de transacción. El problema de este sistema reside que al borrar el mensaje no va a poder ser accedido desde ningún dispositivo, por ejemplo si lo eliminas en el ordenador de tu universidad, luego no podrás acceder a él en tu ordenador personal.

Ventajas y desventajas

Ventajas:

- No requiere de una conexión a internet tras bajar los mensajes.
- Libera espacio en el servidor, ya que los mensajes son eliminados.

Desventajas:

- Combinación multidispositivo muy poco óptima.
- Se necesita espacio en tu dispositivo, ya que será donde se almacenen los correos.
- Si hay una interrupción en la descarga del correo, se perderá el archivo permanentemente.

10.6. Protocolo IMAP

Este protocolo de la capa de aplicación se encuentra definido en el RFC 3501 [30], el protocolo IMAP fue creado por Mark Crispin en 1986 con la idea de poder organizar los correos en carpetas y que fueran accesibles desde cualquier dispositivo y sistema operativo **Figura 51**, esto claro según hemos explicado anteriormente no era posible con el protocolo POP3.

Para el uso de IMAP existen 3 modos de conexión:

- Modo offline - El usuario se conecta de forma periódica al servidor para descargar los correos y sincronizar todos los cambios, sería el modo más similar al uso de POP3.
- Modo online - El usuario se conecta directamente y visualiza los datos cuando le hacen falta, todos los cambios se sincronizan casi inmediatamente.
- Modo desconectado - El usuario trabaja con una copia de los datos en local mientras no está enlazado a la red, la sincronización de los datos se realiza cuando este se vuelve a conectar a internet.

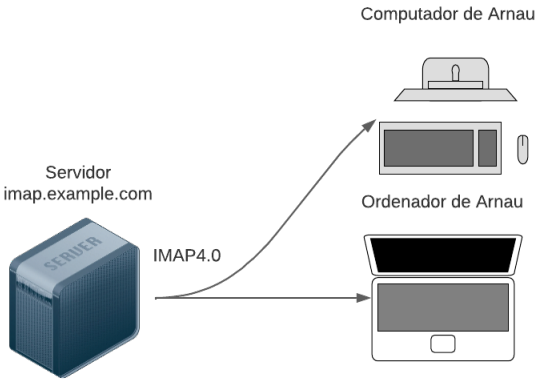

Figura 51: Ejemplo de comunicación IMAP 4.0 multidispositivo

IMAP nos ofrece, como hemos comentado anteriormente, la posibilidad de administrar de forma sencilla nuestro correo con carpetas que siguen una jerarquía similar a un árbol, por defecto la carpeta principal será la bandeja de entrada, y luego aparecerán las carpetas que hayamos creado, cada carpeta tiene ciertos atributos que pueden ser los siguientes:

- Noinferiors - se refiere a que la carpeta solo contiene correos.
- Noselect - la carpeta únicamente contiene carpetas.
- Marked - existen nuevos mensajes en la carpeta.
- Unmarked - no existen mensajes nuevos en la carpeta.

Para los correos que recibamos también existe un sistema de atributos, y son los siguientes:

- Seen - indica si el correo ha sido leído.
- Answered - indica si el correo ha sido contestado.
- Flagged - el correo ha sido marcado como destacado.
- Draft - indica que el correo es un borrador.

A bajo nivel IMAP funciona de forma similar a POP3, IMAP se encuentra en el puerto 143 o en el puerto 993 en el caso de usar algún tipo de cifrado SSL.

Comandos del protocolo

A continuación se muestran los diferentes comandos del protocolo IMAP 4.0:

- LIST - enumera las carpetas.
- SELECT - selecciona una carpeta.
- CREATE - crea una carpeta.

- RENAME - renombra una carpeta.
- DELETE - elimina una carpeta.
- SEARCH - obtiene los mensajes de una bandeja de correo.
- FETCH - obtiene el contenido de un mensaje.
- STORE - modifica los metadatos del mensaje.
- APPEND - guarda un mensaje nuevo en el servidor.

Ejemplo de ejecución

A continuación se muestra un ejemplo en el que se leen 2 mensajes de la bandeja de entrada con IMAP:

```
C: Arn100 LIST "" ""
S: * LIST (\Noselect) "/" ""
S: Arn100 OK LIST Completed
C: Arn100 FETCH 2:3
S: * 2 FETCH Hola Arnau como estas...
S: * 3 FETCH Hace mucho que no nos...
S: Arn100 OK FETCH completed
```

Ventajas y desventajas

Ventajas:

- Los correos no se descargan en local y se mantienen en el servidor.
- Al requerir conexión a internet, los cambios se verán en tiempo real
- Se pueden gestionar los archivos de forma sincronizada con varios dispositivos.
- Los correos son accesibles desde cualquier dispositivo
- Si se interrumpe una transacción no causará ningún problema porque los archivos se mantienen en el servidor.

Desventajas:

- Se necesita una conexión a internet.
- Al abrir un correo se realiza una descarga, aunque sea un archivo temporal
- Los correos ocupan espacio en el servidor.

10.7. Phishing

10.7.1. Metodos de phishing

Phishing dirigido (Spear)

En este tipo de phishing consiste en un ataque dirigido hacia una persona o entidad mediante el correo electrónico, para este se utiliza la investigación para obtener información sobre el objetivo y hacer un ataque personalizado hacia este, al ser un tipo de mail muy dirigido es mucho más fácil que la víctima se vea comprometida.

Session Hijacking

Según OWASP.org [32], en este tipo de ataque se usa el phishing para extraer información de sesión de la víctima, más comúnmente se emplea para robar el token de sesión de la víctima.

Email/Spam

Este es el tipo de phishing más común, consiste en enviar el mismo correo (genérico) al mayor número de usuarios posibles, normalmente se pide al usuario rellenar ciertos datos que estos son usados por los atacantes para diferentes fines.

Inyección de contenido

Consiste en que se le muestre al usuario una web muy parecida a la original, pero con algunos cambios en el contenido, esto se utiliza para que el usuario entre información personal como puede ser el usuario y contraseña que utilizaría para entrar en la web original.

Web Based Delivery

Más conocido como 'Man-In-the-Middle' el atacante se coloca entre la web original y el sistema de phishing, a la que el usuario envía datos a la web original, estos son interceptados por el atacante.

Phishing en motores de búsqueda

Se utilizan sitios web con sistema de búsqueda, donde los productos tienen un precio demasiado bueno, estos se aprovechan cuando un usuario inserta los datos de su tarjeta para realizar una compra y se quedan con estos datos.

Link manipulation

Se muestra un link que parece el de la web original, pero cuando le clicas a este te lleva a otra que no es la que aparece en la cadena de texto.

Vishing

Se emplean llamadas telefónicas a la víctima para intentar obtener algún tipo de información personal, como el número de banco, su nombre, donde vive...

Keyloggers

Se incluyen malware de keylogger en el mail, por lo que al instalar este malware el atacante recibe todo lo que la víctima está escribiendo por el teclado.

Smishing

Similar al phishing por correo, pero en vez se utiliza un SMS de poca longitud en caracteres para redirigir al usuario a algún sitio en el que se le puedan extraer datos personales. Podemos observar un ejemplo en la **Figura 52**.

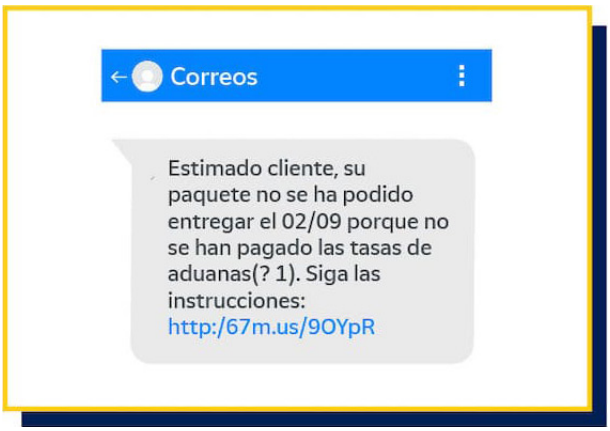

Figura 52: Ejemplo de SMS smishing

Malware

Se incluyen programas maliciosos en el correo, el usuario al ejecutarlo en su ordenador se ve infectado por estos, permiten al atacante varias posibilidades como: extraer información de la víctima, controlar el ordenador de forma remota, espiar al usuario (mediante la cámara)...

Trojan

Es un tipo de malware que se aprovecha del usuario engañándolo para hacer acciones que parecen legítimas, pero permite al atacante extraer información para la que no está autorizado.

Ransomware

Es un tipo de malware que hace que tus archivos sean inaccesibles hasta que realices un pago al atacante, hay varios tipos de ransomware entre ellos; los bloqueadores de pantalla que te impiden el uso de tu ordenador por completo, el scareware que incluye programas de seguridad falsos y ofertas de soporte técnico, el ransomware de cifrado (el más común) que te secuestra los archivos y los cifra. Podemos ver un ejemplo de como sería una pantalla de rescate cuando estás infectado de ransomware en la **Figura 53**.

Figura 53: Pantalla de rescate de ransomware

Malvertising

Consiste en publicidad web que contiene scripts diseñados para descargar malware o forzar a tu ordenador a obtener contenido no deseado.

11. Referencias

Referencias

[1] Java 8 Download
https://www.java.com/es/download/ie_manual.jsp

[2] IntellIJ webpage
https://www.jetbrains.com/idea/

[3] Sublime text webpage
https://www.sublimetext.com/

[4] Blacklistmaster
https://www.blacklistmaster.com/

[5] OneSimpleAPI
https://onesimpleapi.com/docs/url-unshorten

[6] Bitvise SSH
https://www.bitvise.com/ssh-client-download

[7] Overleaf
https://www.overleaf.com

[8] Git
https://git-scm.com/

[9] Instagantt
https://instagantt.com

[10] LucidChart webpage
https://www.lucidchart.com

[11] OpenSSL webpage
https://www.openssl.org/

[12] RFC 5321
https://datatracker.ietf.org/doc/html/rfc5321

[13] ASCII concepto
https://concepto.de/ascii/

[14] Phishing.org official website
https://www.phishing.org/what-is-phishing

[15] 5 ways to detect phishing
https://www.itgovernance.co.uk/blog/5-ways-to-detect-a-phishing-email

[16] Como evitar caer en Phishing
https://bit.ly/3zrj3LV

[17] Sophos.como
https://www.sophos.com/es-es/products/phish-threat?&cmp=115461&utm_campaign=GPD-2021-WE-ES-ES-PaidSearch-Google-SCH-NB-Generic-Phish-Threat-Anti-Phishing-DG-115461&utm_medium=cpc&utm_content=NB_Generic_Phish-Threat_Anti-Phishing&utm_term=antiphishing&utm_source=google-search&gclid=CjwKCAjwtIaVBhBkEiwAsr7-cySfb_Df7Oqzvu1eacEcb3AwHFWX4H6SkSjySOCYFygZLYkX7X7LthoCgQcQAvD_BwE&gclsrc=aw.ds

[18] sofistic.com
https://www.sofistic.com/productos/

[19] Tutorial Factory method
https://www.geeksforgeeks.org/factory-method-design-pattern-in-java/

[20] UTF_8 table
https://www.utf8-chartable.de/

[21] Redirección Apple Mail
https://support.apple.com/es-es/guide/mail/mlhl8b91a034/mac

[22] Redirección Outlook
https://bit.ly/3xlgQiv

[23] Repositorio GitHub del proyecto
https://owasp.org/www-community/attacks/Session_hijacking_attack

[24] Test unitarios proyecto
https://github.com/DarkDhz/Ubmail/blob/master/src/test/java/mailtest.java

[25] Definicio de protocolo
https://dle.rae.es/protocolo

[26] DNSSEC
https://bit.ly/3lJXMoF

[27] OSI Model
https://bit.ly/3PIgjzr

[28] RFC 1034
https://www.rfc-es.org/rfc/rfc1034-es.txt

[29] RFC 1939
https://www.ietf.org/rfc/rfc1939.txt

[30] RFC 3501
https://datatracker.ietf.org/doc/html/rfc3501)

[31] OVH Hosting Web
https://bit.ly/3wYbM3r

[32] Owasp sesion hijacking
https://owasp.org/www-community/attacks/Session_hijacking_attack