UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Herramienta de análisis automático de vulnerabilidades SSL

Iago Sánchez Prieto
Tutor: Óscar Delgado Mohatar
Ponente: Eloy Anguiano Rey

Julio 2016

Herramienta de análisis automático de vulnerabilidades SSL

AUTOR: Iago Sánchez Prieto
TUTOR: Óscar Delgado Mohatar

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2016

Abstract

The main goal of this TFG is the design and developoment of a SSL tester web application. It allows to perform a complete assesment, providing detailed information, including known vulnerabilities, digital certificates or supported protocol versions.

On the other hand, this tool performs a real-time test in a remote URLs, IPs and servers, in a similar way that other well-known comercial solutions. It also supports user registration, who can then recover past scans and perform dayly analysis. Registration is mandatory in any case for the use of the tool, in order to reduce its potential bad use.

For last, the solution has the capability of including new exploits, under previous validation of the administrator, to increase the number of checked vulnerabilities during the test. It also presents exploits' statistics upon request.

Keywords

SSL, TLS, Exploit, Attack, Vulnerability

Resumen

Este Trabajo Fin de Grado, tiene como objetivo la realización de una aplicación web para el análisis de la capa SSL de páginas web, enfocándolo a usuarios interesados en conocer las vulnerabilidades, información de certificados, tipos de cifrados que soporta y protocolos.

Esta herramienta permite el análisis en tiempo real y de forma remota URLs, IPs o servidores. Además de esto, permite un registro para los usuarios que realicen análisis diariamente, puesto que es necesario mantener el control de los usuarios que puedan o tengan la intención de hacer mala praxis de la aplicación.

Así mismo, permite la inclusión de nuevos exploits, bajo previa validación del administrador de la aplicación para aumentar el número de vulnerabilidades analizadas en el test. Cuenta también con la posibilidad de ver las estadísticas de los exploits (cada uno por separado) y también la de todos ellos juntos.

Palabras clave

SSL, TLS, Exploit, Ataque, vulnerabilidad

Agradecimientos

Gracias a mi familia, por haberme ayudado a crecer como persona, por animarme durante todos estos años de carrera y a no dejar que tirará la toalla. En especial a mi padre, que a pesar de haberme exigido mucho, nunca ha dejado de creer que lo conseguiría. A mi hermano, por interesarse siempre en todo lo que hacía y hacerme todo mucho más ameno.

A mi tutor Óscar, por haberme dado la oportunidad de realizar un TFG con el cual realmente he aprendido.

A mis compañeros de clase por haber hecho la estancia en la universidad mucho más entretenida y divertida; sobre todo a Tito por su ayuda con las prácticas, a Rober por sus charlas interesantes y a Gubio por haber estado siempre ahí.

Por último, y no por ello menos importante, a Jimena, por soportarme en los días malos, alegrarme en los buenos y sobre todo por el apoyo recibido.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

Debido al uso masivo de las redes, desde distintos dispositivos (smartphones, tablets, ordenadores, etc), tanto en el ámbito privado como empresarial, se hace necesario el uso de protocolos y acciones que garanticen la seguridad en la red.

Hay que añadir, la histeria colectiva de las personas que creen que grandes empresas o países puedan obtener de forma sencilla la información privada, tanto de usuarios individuales como de empresas.

Teniendo en cuenta los casos que han salido a la luz que vulneraban la privacidad de jefes de estado o personas influyentes, se ha propiciado que durante estos últimos años se haya elevado la complejidad de los algoritmos criptográficos en los protocolos de seguridad.

La capa SSL permite configurar parámetros de los protocolos con el objetivo de mejorar a los requerimientos de seguridad de los distintos sistemas.

1.1 Motivación

Cabe destacar la creciente importancia que ha adquirido durante los últimos años la seguridad en la red, debido a que se maneja gran cantidad de información sensible respecto a personas, empresas, países, organizaciones o ideas. Esto, ha suscitado un enorme interés en salvaguardar dicha seguridad mediante la creación de protocolos. Pese a que actualmente existe una gran cantidad de aplicaciones que intentan satisfacer las necesidades generadas de dicho uso, todos los días se crean nuevas tecnologías para hacer que las comunicaciones y acciones sean más rápidas, simples y seguras para el usuario.

La continua mejora de los protocolos existentes para subsanar fallos, así como la creación de nuevos, hace interesante como Trabajo de Fin de Grado el análisis de la capa SSL. Dicho análisis se ha llevado a cabo mediante la creación de una aplicación web que permite realizar el análisis introduciendo una dirección IP, URL o un dominio. Así mismo, se incluirá en dicho análisis los protocolos soportados y cifradores, resaltando cuáles de ellos son seguros o inseguros, y se informará sobre las vulnerabilidades.

1.2 Objetivos

A lo largo de este Trabajo de Fin de Grado se tratarán diferentes aspectos de la "capa" de seguridad SSL (conjunto de protocolos criptográficos que proporcionan seguridad entre comunicaciones a través de la red). Una vez presentada la evolución que ha sufrido SSL a lo largo del tiempo para suplir algunas carencias, se dedicará el resto del documento a explicar la capa SSL. En detalle, la arquitectura SSL, los protocolos que soporta, los algoritmos que puede usar, los certificados digitales y las posibles vulnerabilidades que pueden aparecer en dicha capa.

Se explicará detalladamente la aplicación realizada para el análisis de la capa SSL, así como también su diseño, tecnologías empleadas y funcionalidades. El objetivo de este proyecto es la creación de una aplicación multiplataforma (página web), que, de forma remota, se le pueda pasar una IP, URL o dominio, con el fin de ser analizada. Este análisis será en primera instancia de información básica de la página analizada, como también de los certificados que tiene, los cifradores que soporta, los protocolos y por último una serie de vulnerabilidades e información específica. De esta forma, se mostrarán algunos diseños de la página, funcionalidades y ejemplos con el fin de mostrar y hacer más sencillo la comprensión de la aplicación. También se hará un árbol de los ficheros que componen la aplicación y un breve resumen de las funciones que proporcionan cada fichero de forma breve y precisa.

Por último, una vez explicada totalmente la capa SSL y el funcionamiento y las funciones de la página, se harán una serie de pruebas a modo de ejemplo y se mostrarán una serie de datos y gráficos para ver el estado de los servidores probados y el estado de la red.

1.3 Organización de la memoria

Inicialmente, se incluye una breve introducción histórica de la capa SSL con sus distintas versiones. A continuación, se analizan las funciones que realizan los cuatro protocolos usados en esta capa.

Se explicarán las versiones del protocolo SSL y TLS, haciendo hincapié en las diferencias respecto a la versión anterior. También se incluye el detalle de cada uno de los ataques y vulnerabilidades que han sido añadidos a la aplicación.

Una vez realizada esta introducción conceptual en la capa SSL, su funcionamiento y los ataques/vulnerabilidades usadas, se incluye la descripción de los requisitos de la aplicación, el diseño detallado de la misma, los programas usados para realizar el desarrollo y la metodología seguida durante el mismo. Se incluye también la integración, las pruebas realizadas y los resultados obtenidos. Por último, se incluye las conclusiones y trabajo futuro.

2 Estado del arte

La capa SSL, es un conjunto de protocolos criptográficos que son usados para poder proporcionar seguridad en las comunicaciones a través de la red. A lo largo de los años ha ido evolucionando para dar más seguridad, corregir errores y problemas de implementación y diseño. En principio, la capa SSL se creó por la necesidad de incorporar seguridad a Netscape Navigator (el navegador de la empresa Netscape Communications), en el cual se añadió el protocolo HTTPS en 1992(capa de aplicación). Esta misma empresa es la que desarrolló la capa SSL.

El protocolo SSL se inició en 1993 junto con el protocolo HTTPS, y su versión 1.0 se desarrolló antes de 1995. Esta versión nunca vio la luz debido a la cantidad de fallos graves que contenía, por lo que en Febrero de 1995 se presentó la versión 2.0, pero al igual que su predecesora, esta contenía bastantes fallos de seguridad. Fue prohibido el uso de la versión 2.0 en el año 2011, en el RFC 6176 a causa de su inseguridad. La nueva versión fue la 3.0, que se presentó en 1996. Debido a las dos versiones anteriores y sus fallos, en esta ocasión se decidió hacer un rediseño completo, el cual fue producido por Paul Kolcher, junto con Phil Karlton y Alan Freier (ambos trabajadores de Netscape Communications). Finalmente esta versión se declaró obsoleta en junio de 2015 por el RFC 7568, por los agujeros de seguridad que existían y por la aparición de la vulnerabilidad POODLE(CVE-2014-3566) en el año 2014.

Después de la versión 3.0, la nomenclatura del protocolo, pasa de SSL (*Secure Sockets Layer*) a TLS (*Transport Layer Security*). De este cambio, nace una actualización de SSL 3.0 en enero de 1999, recogido en el RFC 2246. Esta nueva actualización pasa a llamarse TLS 1.0. Posteriormente, en abril de 2006 se define la versión 1.1 del protocolo TLS en el RFC 4346. En agosto de 2008 se define la versión 1.2 del protocolo TLS; en Marzo de 2011 se redefinió el protocolo para establecer su retro compatibilidad, es decir, se establece que esta versión no tuviese retro compatibilidad con la versión SSL 2 del protocolo. Finalmente, en estos últimos años se ha estado trabajando en la nueva versión, llamada TLS 1.3, más concretamente en enero de este año se ha publicado un borrador, aunque por el momento está incompleto.

2.1 Arquitectura capa SSL

En cuanto a los protocolos que intervienen en la capa SSL, decir que SSL fue diseñado para que junto a TCP (protocolo de transporte), pudiese proporcionar comunicaciones sin pérdida y fiable extremo a extremo y que diese servicios de seguridad al nivel de aplicación, en este caso a HTTP. Este servicio es proporcionado por el protocolo Record SSL. También se implementan otros tres protocolos implementados en el nivel de aplicación, son *Handshake Protocol*, *Change Cipher Spec Protocol* y *Alert Protocol*. A continuación, se explican unos conceptos para la mejor comprensión del protocolo, como también la pila de protocolos SSL (estos serán explicados en el anexo Protocolos de la capa SSL, C)[1].

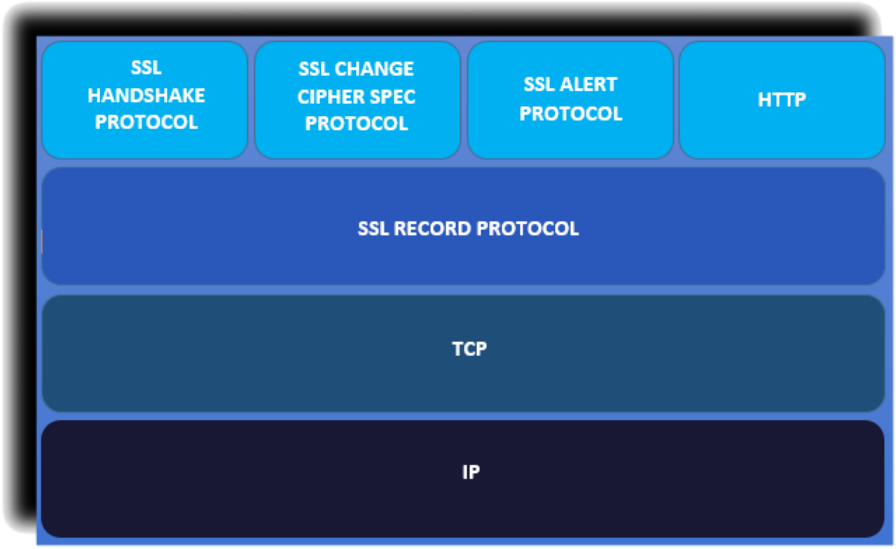


Figura 1: Protocolo de la capa SSL

2.2 Protocolos SSL

En este apartado se definirán exhaustivamente los protocolos descritos anteriormente como también las diferencias entre los mismos, qué incluyen y qué corrigen con respecto a la versión anterior.

Empecemos con SSL 1.0. Este nunca vio la luz, así que sobre este protocolo no se tiene información ni características.

A continuación, se define SSL 2.0. Esta es la primera versión de este protocolo que se usa en la red. Aunque esté en desuso ya que contenía una gran cantidad de fallos. Como es la primera versión en uso, se listarán los fallos para comparar con las posteriores versiones.

- Las claves criptográficas usadas en SSL 2.0, se usaban tanto para la autenticación de mensajes como para el cifrado. Si negocian un algoritmo de cifrado débil, es bastante fácil obtener las claves.
- El MAC de SSL 2.0 es bastante débil, pues utiliza un hash MD5 junto a una clave secreta, por ello es vulnerable a los ataques de extensión de longitud.
- SSL 2.0 no otorga ningún tipo de seguridad a la hora del *handshake*, es decir, es posible hacer un ataque "*man in the middle*" sin que los extremos de la comunicación sospechen.
- Debido a que SSL 2.0 hace uso del cierre de la conexión TCP para indicar el fin de los datos de un mensaje, entonces es posible hacer ataques de truncamiento. Esto es, puesto que es posible que los datos sean divididos en varios paquetes, y un atacante puede generar un mensaje de TCP FIN. Si esto ocurriese, el receptor de estos mensajes tiene un fin de mensaje erróneo.

Esta versión de SSL está desactivada por defecto en las últimas versiones de los navegadores: Internet Explorer (versión 7 o superior), Mozilla Firefox (versión 2 o superior), Opera (versión 9.5 o superior) y Safari (versión 5 o superior)[9][5].

La versión SSL 3.0 corrige alguno de los errores provenientes de su anterior versión. Aunque SSL 3.0 no está desactivado por defecto como su versión anterior, se considera inseguro, puesto que sigue teniendo bastantes fallas de seguridad. Se listan los cambios y/o correcciones incluidos en esta versión:

- Se hace la separación entre la capa de mensajes y el transporte de los datos. Mientras que en SSL 2.0 cada paquete contenía un único mensaje de negociación, en SSL 3.0 un registro puede contener una parte de un mensaje, un mensaje o varios mensajes.
- En SSL 2.0 solo se podía usar como mecanismo de intercambio de claves RSA, mientras que para SSL 3.0 se añadieron los métodos *Diffie-Hellman*. Estos métodos pueden garantizar la confidencialidad directa (*forward secrecy*).
- En SSL 2.0, un ataque "*man in the middle*" podía cambiar la suite de cifrado para hacer que este sea inseguro (cifrado de 40 bits) de forma invisible, en SSL 3.0 se incluye en el *Handshake* un hash con los últimos mensajes enviados para que esto no se produzca.
- En SSL 2.0 se crean MACs que son débiles, esto se cambia en SSL 3.0.
- En SSL 3.0 se modifica para que no existan ataques de truncamiento con el campo de relleno de longitud.
- En SSL 2.0 no se puede realizar una renegociación una vez iniciada la conexión, en SSL 3.0 se puede cambiar los algoritmos criptográficos usados y las claves siempre que se desee.
- SSL 3.0 permite el uso de cadenas de certificados, de esta forma se establece la jerarquía de certificados.
- SSL 3.0 permite la compresión y descompresión de los registros.
- Permite hacer uso de SSL 2.0 bajo SSL 3.0 (*downgrade*).

Aún con los cambios realizados, SSL 3.0[5] tiene los siguientes errores:

- Debido a que sigue haciendo uso de MD5 en la mitad de la función de generación de claves, se sigue considerando inseguro.
- Existe un problema a la hora de firmar certificados, puesto que las entidades certificadoras que no ponían la variable *basicConstraintsCA* a false, hacían posible poder firmar certificados en los nodos hoja.
- Otro problema reside en la posibilidad de hacer un *downgrade* a SSL 2.0, lo que muy inseguro.
- Por último, hay varias vulnerabilidades que se han ido descubriendo a lo largo del tiempo de uso de este protocolo. Estas, serán expuestas en el siguiente punto.

Después de SSL 3.0, se decide cambiar levemente la estructura del protocolo. Esta versión es desarrollada por la IETF (*Internet Engineering Task Force*), con el objetivo de establecer un estándar para las comunicaciones con seguridad, llamada TLS 1.0. A continuación, se listan las correcciones y/o cambios con respecto a SSL 3.0:

- En el caso de TLS 1.0 se añaden nuevos algoritmos de cifrado, mucho más robustos. Mientras que en SSL 3.0 siguen teniendo algunos algoritmos bastante débiles.

- Se incluye la posibilidad de usar un puerto que no sea el establecido por defecto (para las conexiones con SSL es el 443).
- En TLS 1.0 es eliminada la interoperabilidad con protocolos de versiones anteriores (SSL 2.0 y SSL 3.0).
- En vez de usar la función MAC para la autenticación de los mensajes, en TLS 1.0 se usa la función HMAC.
- TLS 1.0 incorpora más mensajes de alerta.
- En el mensaje *finished* se envía un hash de todos los mensajes anteriores durante el handshake de las dos partes.

En TLS 1.0[6] todas las grietas de seguridad encontradas, son causadas por vulnerabilidades descubiertas, estas serán definidas y expuestas posteriormente.

En 2006 se actualiza el protocolo TLS en su versión 1.1, implementando los siguientes cambios con respecto a TLS 1.0:

- En TLS 1.1 el vector de inicialización para el método de cifrado por bloque se sustituye, se protege de ataques de encadenamiento de bloques de cifrado.
- Se cambia el manejo de errores de relleno, se usa el *bad_record_mac* en vez del *decryption_failed* para protegerse de los ataques de encadenamiento de bloques de cifrado.
- Inclusión de registros IANA (*Internet Assigned Numbers Authority*) para los parámetros del protocolo.
- Los cierres prematuros ya no provocan que la sesión no se pueda reanudar. Mientras que en TLS 1.0 sí ocurre.

En cuanto a las fallas en TLS 1.1 son causadas por vulnerabilidades descubiertas[7].

TLS 1.2 es una revisión del protocolo TLS 1.1, por lo tanto, como tal, no hay correcciones que hacer, sin embargo, esta nueva actualización aporta o incorpora los siguientes cambios:

- En TLS 1.1 en la función pseudoaleatoria (PRF), se usaba junto con las suites de cifrados la combinación de MD5 y SHA-1, mientras que en TLS 1.2, todas las suites de cifrado usan SHA-256.
- La combinación de MD5 y SHA-1 en el campo firmado digitalmente ha sido modificado, y en TLS 1.2 solo se usa un único hash. Estos campos firmados ahora incluyen un campo en el cual se especifica el hash utilizado.
- Se ha endurecido la comprobación de números de versión en *EncryptedPreMasterSecret*.
- Las definiciones de extensiones de TLS y las suites de cifrado AES se fusionan.
- Las suites de cifrado IDEA han sido eliminadas en esta versión al igual que DES, ya que se consideran inseguras.

En cuanto a los fallos, sólo puede ser víctima de vulnerabilidades[8].

Por último, queda mencionar la última versión de TLS, esta es la 1.3[11]. Aunque aún es un borrador (provisional, incompleto y no en uso) desde Enero de 2016, incorpora los siguientes cambios:

- Se eliminan curvas elípticas (EC) débiles y las menos usadas. También se retira el soporte a las funciones hash MD5 y SHA-224.
- Se ha añadido el requisito de pedir las firmas digitales incluso cuando se utiliza una configuración previamente establecida.
- Se ha prohibido el uso de SSL o RC4 en la negociación para la retro compatibilidad.
- Añade una lista de las suites de cifrado PSK (*Pre-Shared Key*).
- Actualiza la lista de suites de cifrado AEAD (*Authenticated Encryption with Associated Data*) disponibles y las alertas de error.
- Reduce la longitud máxima del registro AEAD de 2048 bytes a 256 bytes.

A continuación, se muestra una tabla con las versiones de los protocolos de SSL y TLS con las fechas de inicio y fin de los mismos:

Protocolo/Versión	Incluido	Prohibido/Desuso
SSL 1.0	- (No llegó al público)	- (No se usó)
SSL 2.0	Febrero de 1995	Marzo de 2011 (Prohibido)
SSL 3.0	Octubre de 1996	Junio de 2015 (Desuso)
TLS 1.0	Enero de 1999	- (Aún en uso)
TLS 1.1	Abril de 2006	- (Aún en uso)
TLS 1.2	Agosto de 2008	- (Aún en uso)
TLS 1.3	Mayo de 2015 (Borrador)	- (Es un borrador)

Tabla 1: Versiones de la capa SSL

2.3 Ataques de la capa SSL

En este apartado se mostrarán los ataques más importantes y conocidos de la capa SSL. A fin de catalogarlos, estos ataques serán definidos y se dará información de las versiones de SSL y TLS afectadas. Además de reportar la severidad que presentan en el protocolo cuando son ejecutados.

2.3.1 Renegociación

Esta vulnerabilidad, que fue descubierta en agosto de 2009. Permite al atacante apropiarse de una conexión https de un tercero, para poder hacer sus propias peticiones y enviarlas al servidor que esté conectado el cliente. Es bastante similar a un ataque *man in the middle*, con la diferencia de que, en este caso, el atacante no puede obtener en texto plano de los mensajes enviados por el cliente.

Esta vulnerabilidad funciona de la siguiente forma. Es muy común que primero sea el atacante el que se conecte al servidor, este puede enviar cualquier número de mensajes al servidor al igual que recibir respuestas.

A continuación, cuando el cliente se disponga a conectarse- El atacante roba la conexión. (también es posible que el cliente se conecte). Una vez conseguido esto, el atacante podrá enviar mensajes al servidor haciéndose pasar por el cliente y el cliente no sabrá que ha realizado una renegociación[12] con el servidor. De esta forma, por ejemplo y puesto que SSL funciona como un protocolo de seguridad, este ataque puede usar la cookie de la víctima para loguearse o realizar acciones en el servidor a través de su perfil.

Un ejemplo sería que el atacante se conectara a un servidor que albergue una página de venta por internet, y a través de la víctima consigue cargar en su cuenta algún tipo de artículo. Hay que recordar que, aunque el atacante envíe mensajes haciéndose pasar por el cliente, esté nunca recibirá las respuestas. Porque serán enviadas al cliente. De tal forma que:

El atacante envía:

GET /articles?buyidarticle=1234;address=attackersaddress HTTP/1.1
X-Ignore-This:

Observar que la última línea se deja vacía y sin retorno de carro, para que cuando el cliente envíe su petición ésta sea omitida. Entonces el cliente envía:

GET /articles?showidarticle=5934;address=victimssaddress HTTP/1.1
Cookie: victimscookie

De tal forma que la unión de los mensajes queda como resultado:

GET /articles?buyidarticle=1234;address=attackersaddress HTTP/1.1
X-Ignore-This:GET /articles?showidarticle=5934;address=victimssaddress HTTP/1.1
Cookie: victimscookie

Como se puede observar, al final, el mensaje que llega al servidor es el de comprar el artículo con id 1234 y no el de mostrar el artículo con id 5934.

Esta vulnerabilidad se da tanto en SSL 3.0 como también en las versiones de TLS. En cuanto a la severidad de la vulnerabilidad mencionar, que esta viene dada por el protocolo de nivel superior (nivel de aplicación), por consiguiente, puede ser bastante dispar. Esto, sumado a que el atacante no tiene forma (en principio, si hay otras vulnerabilidades sí) de ver la respuesta, se puede decir que el nivel de gravedad puede ser bajo.

Por último, se muestra una imagen que ilustra y aclara el funcionamiento de este ataque:

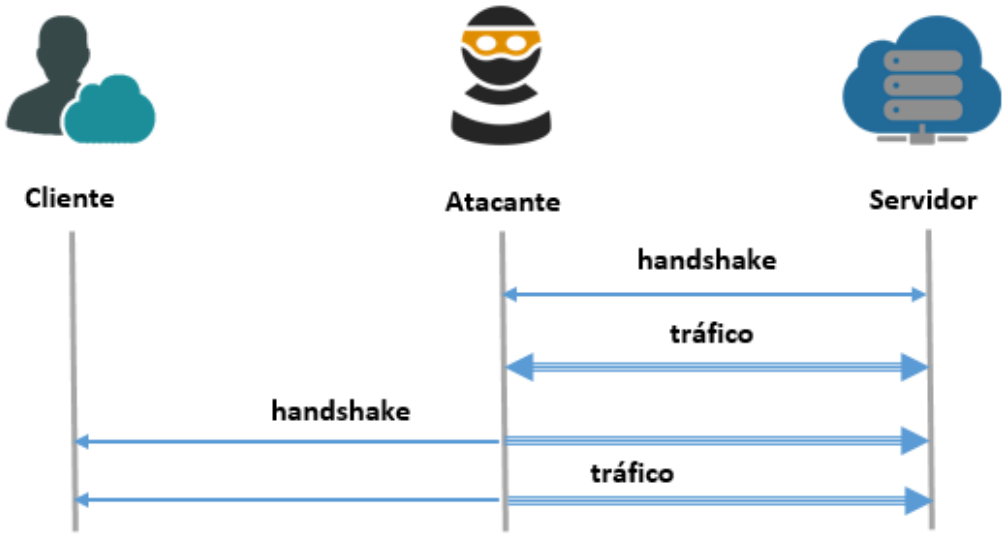


Figura 2: Ataque de Renegociación

Para poder omitir esta vulnerabilidad en la capa SSL, lo más rápido y eficaz sería eliminar en el lado del servidor la opción de la renegociación. Otra sería que el servidor a la hora de hacer una renegociación, pida al cliente una autenticación a través de un certificado. Por último, la última forma de eliminar esta vulnerabilidad está recogido en el RFC5746, en el cual se hace uso de una extensión del protocolo TLS. Esta extensión, hace que tanto el cliente como el servidor, tengan que incluir y verificar información con respecto a los *handshakes* anteriores.

2.3.2 Downgrade

Esta vulnerabilidad existe, debido a la necesidad de los servidores, puedan establecer conexiones con clientes "antiguos", es decir, con clientes que usen versiones inferiores a la que usa por defecto el servidor. Esta vulnerabilidad permite al atacante establecer una versión del protocolo de TLS menor, esto se hace para poder elegir un cifrado más débil, un intercambio de claves más débil o aprovechar alguna de las fallas o vulnerabilidades de estas versiones más antiguas. Este ataque se suele usar junto con un *man in the middle*, puesto que no tendría sentido que un cliente normal intentase hacer un *downgrade[13]* a sí mismo, mientras que, si un atacante consigue establecer un ataque *man in the middle*, este usa este downgrade para aprovecharse y así poder obtener cierta información privada.

Un ejemplo de este ataque sería una vez realizado con éxito el ataque *man in the middle*, mandas una petición al servidor para usar una versión más antigua para establecer por ejemplo que la suite de cifrado use RC4 (es inseguro). De esta forma, el atacante tiene más probabilidades de obtener información.

Esta vulnerabilidad se da en todas las versiones del protocolo TLS, mientras que para SSL no ocurre. Esto se debe a que SSL 2.0 tiene prohibido su uso, y, por lo tanto, ningún servidor debería de soportar esta versión. No se puede hacer un *downgrade* de SSL 3.0, ya que no hay ninguna versión en uso menor que está. En cuanto a la gravedad de esta

vulnerabilidad, se puede establecer en medio, debido a que si un atacante puede llegar a establecer una conexión usando SSL 3.0, y asumiendo que de primeras esta versión de protocolo SSL es insegura, se puede obtener bastante información.

A continuación, se muestra una imagen de esta vulnerabilidad junto con un ataque *man in the middle*:

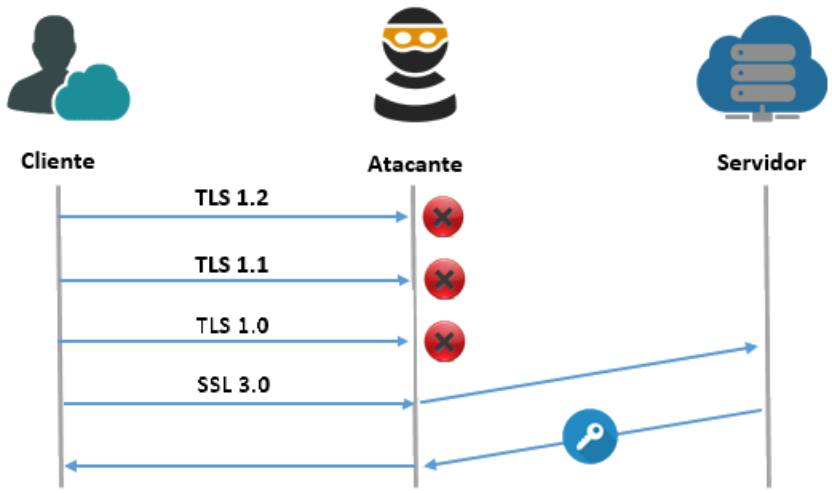


Figura 3: Ataque *Downgrade*

Para poder eliminar esta vulnerabilidad del protocolo TLS, se crea en TLS el SCSV (*Signaling Cipher Suite Value*). Esta es una extensión empleada al enviarse un *client_hello* para establecer la suite de cifrado, esta comprueba que, si el cliente soporta un protocolo mayor al que ha enviado y este sea soportado por el servidor, el servidor responderá con un mensaje de alerta (*inappropiate_fallback*). De esta forma, se elimina la posibilidad de hacer un downgrade. Otra forma de evitar el downgrade es imponiendo una única versión del protocolo SSL que soporte el servidor. Esto no es muy aceptable, puesto que muchos clientes no se encuentran actualizados y no usan la última versión.

2.3.3 BEAST

Esta vulnerabilidad fue descubierta en septiembre de 2011. Fue presentada por Thai Duong y Juliano Rizzo y se basa en un programa JavaScript que junto con un sniffer de red desencripta las cookies de los clientes con la información privada de estos. Esta vulnerabilidad se basa en un fallo encontrado en las suites de cifrado de TLS 1.0, concretamente en las suites de cifrado por bloque en modo CBC. BEAST[10] (*Browser Exploit Against SSL/TLS*). Obtiene esta información de la siguiente manera:

Al usar CBC se establece que, el texto plano se divide en bloques y estos son cifrados con una clave, este depende del texto en claro a cifrar (bloque correspondiente) y de la salida anterior del cifrado, es decir:

$$C_i = E\ (Key, C_{i-1} \oplus M_i)$$

Menos al cifrar el primer bloque, que este hace uso del vector de inicialización (IV), dando:

$$C_0 = E\ (Key, IV \oplus M_0)$$

El problema se encuentra, en que este tipo de cifrados estaba pensado para usar una vez para cifrar un archivo. Sin embargo, en la capa SSL se usa para cifrar con la misma clave varios registros, que se irán enviando. Por lo tanto, cada bloque siguiente tiene un residuo del anterior. De esta forma si el atacante sabe que en uno de los mensajes se está enviando la contraseña, este sacará la contraseña inyectando un registro a la hora de cifrar. De esta manera, si la contraseña fuese "C", se cifra como "IV ⊕ Ci-1 ⊕ C". Si C == Mi (Mensaje a cifrar), entonces quiere decir que la suposición del atacante era correcta. Esto sumado a que cada registro depende del anterior, el atacante lo único que debe de hacer es inyectar un parte en cada registro de cifrado, para obtener la información deseada. A continuación, se muestra una imagen a modo explicativo.

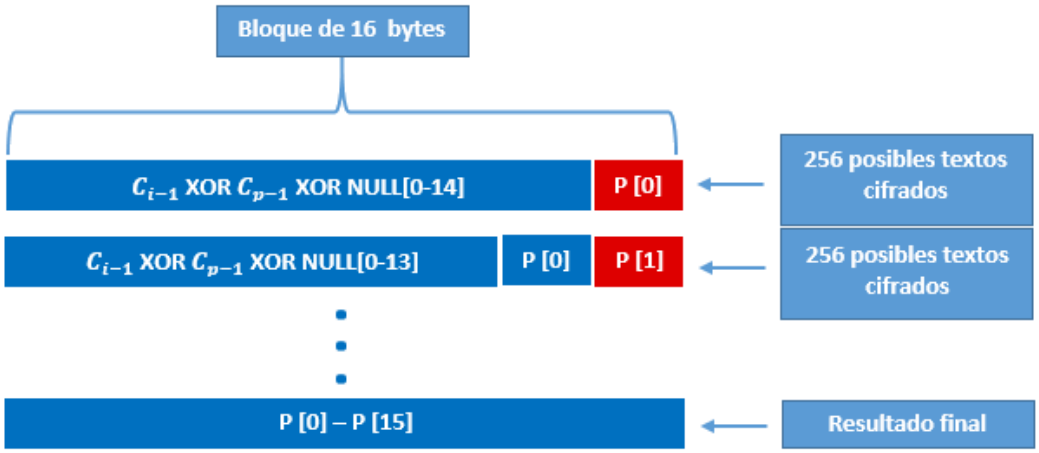


Figura 4: Cifrado CBC vulnerabilidad

Esta vulnerabilidad solo afecta a la versión 1.0 del protocolo TLS. En cuanto a la gravedad de la vulnerabilidad, mencionar que en principio parece difícil obtener una gran cantidad de información secreta de esta forma, pero es posible. Aun así, después del tiempo que ha pasado desde que la vulnerabilidad fue descubierta, se han realizado multitud de cambios para que no pueda ser explotado. Los navegadores de hoy día están parcheados para que no se pueda explotar, por todo ello, BEAST es poco grave.

La falla ha sido eliminada por parte de los navegadores web. Mientras que, por parte de los servidores, la única forma de mitigar este ataque, es que el propio servidor priorice por un cifrado RC4, más que por un cifrado como AES o DES. Aun así, se ha descubierto y verificado que el cifrado con RC4 es más débil de lo que se pensaba. Por eso, es preferible mantener el uso de cifradores como DES y AES, puesto que BEAST solo se da en TLS 1.0, mientras que el uso de RC4 de forma insegura se puede dar en cualquier versión de TLS.

2.3.4 CRIME

Esta vulnerabilidad se relaciona con la compresión de los datos antes de enviar el mensaje. CRIME[14] (*Compression Ratio Info-leak Made Easy*) fue creada por las mismas personas que el ataque BEAST. Dicho ataque se usa en conexiones donde se utilizan los protocolos HTTPS y SPDY que usan compresión y que permite apropiarse de una sesión autenticada de un tercero. Este ataque se basa en una combinación de inyección de texto plano y de fuga de información. Para ello, el atacante inyecta pequeños trozos en los bloques a comprimir y posteriormente cifrar. De esta forma, haciendo uso de varias conexiones y envío de mensajes (normalmente un múltiplo de los bytes secretos a

recuperar) es posible descifrar el mensaje original. Esto sucede, debido a que, a la hora de comprimir, si hay caracteres iguales juntos en el bloque, estos son comprimidos; un ejemplo sería:

AAAAABCDEFGH = 5ABCDEFGH

Como se puede comprobar, el texto pasa de tener longitud 12 a 9, esto quiere decir que se consigue un 25% de compresión. Es este motivo, el que propicia el ataque, puesto que al cambiar la longitud del bloque también cambia cuando se cifra. Dando como resultado:

Texto plano	Texto comprimido	Texto cifrado
ABCDEFGH	ABCDEFGH	Z@%fkT2r$#!B
AAAAABCDEFGH	5ABCDEFGH	jhG*4m,$A

Tabla 2: Ejemplo de compresión

Se puede observar que el texto con compresión cambia de tamaño al cifrarse, de esta manera, para un atacante el texto plano es desconocido, el solo conoce el texto que inyecta y el texto cifrado que se obtiene. De esta forma, el atacante hace una batería de pruebas, en la cual a cada bloque se le inyecta un carácter diferente para obtener alguna pista. Un ejemplo sería:

Texto plano	Texto comprimido	Texto cifrado
AAA[Texto_desconocido]	[Texto_desconocido]	Z@%fkT2r$#!BX(
BBB[Texto_desconocido]	[Texto_desconocido]	QvnQSHvQWB3*QR
CCC[Texto_desconocido]	[Texto_desconocido]	f*fB&M7sya*u7F
. . .		
ZZZ[Texto_desconocido]	[Texto_desconocido]	rAW^26uffH%8

Tabla 3: Ejemplo de ataque de compresión

Como muestra la tabla anterior el atacante encuentra una coincidencia con la última de las pruebas, donde ha inyectado “ZZZ”. Esto se debe a que el resto de textos cifrados tienen la misma longitud (14), mientras que para el último se reduce (12). De esta forma es como el atacante puede ir obteniendo la información secreta del cliente. A continuación, se muestra una imagen de cómo el atacante consigue la información.

El ataque CRIME afecta a cualquier versión del protocolo TLS y SSL que soporte la compresión de los datos. Debido a que los navegadores han eliminado esta vulnerabilidad, y muchos de los servidores han eliminado la opción de la compresión de los mensajes, este ataque se da en muy pocas ocasiones. Por ello la gravedad de este ataque es baja, puesto que es bastante difícil que se pueda utilizar.

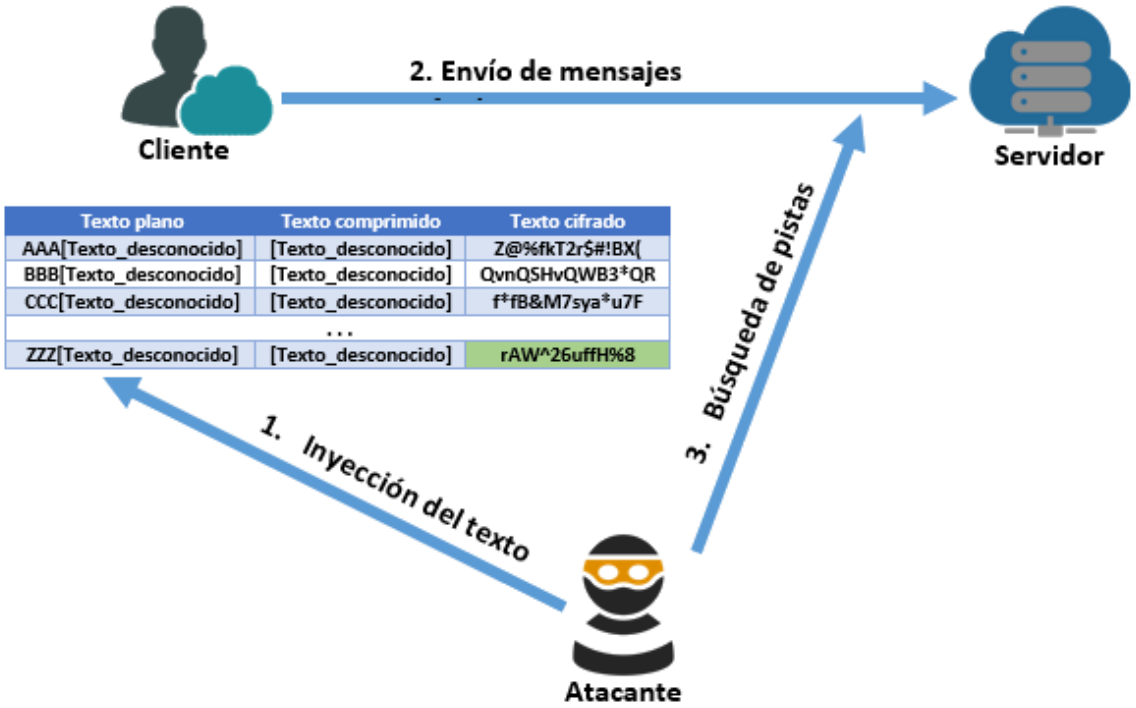


Figura 5: Ataque CRIME

Para la eliminación de este ataque, como se ha mencionado, los navegadores ya han conseguido mitigar la vulnerabilidad, y la mayoría de servidores igual. La forma de eliminar este ataque es la de evitar la compresión antes del cifrado, puesto que en el mensaje *client_hello* se envían los tipos de protocolos de compresión soportados por el cliente, y luego es el servidor quien decide qué tipo de cifrado usar, de este modo, si el cliente envía un mensaje *client_hello* con ningún método de compresión (*none*), el servidor puede dejar de usar la compresión.

2.3.5 POODLE

La vulnerabilidad POODLE[15] (*Padding Oracle On Downgraded Legacy Encryption*) se hizo pública en Octubre de 2014, esta primera publicación hacía referencia a la vulnerabilidad encontrada en SSL 3.0 recogido en el CVE-2014-3566, mientras que dos meses después de la publicación de esta vulnerabilidad, en Diciembre de 2014 se descubre otro ataque que aparecía en el protocolo TLS recogido en el CVE-2014-8730.

Esta vulnerabilidad se aprovecha del protocolo de renegociación, intenta hacer que el servidor falle al establecer una conexión con el protocolo TLS, de esta forma, el servidor aceptará una conexión de SSL 3.0. Una vez conseguido esto, funciona de forma parecida al ataque BEAST, puesto que hace uso de algoritmos de cifrado por bloque en modo CBC, de esta forma, se sabe que cada bloque debe de ser de un tamaño múltiplo de 8, y que una vez

cifrado el bloque, el último byte contiene el tamaño de bytes de relleno usado para el bloque. Así, un atacante puede directamente desechar los bytes de relleno y trabajar con el texto cifrado correspondiente al mensaje original. Al igual que en el ataque BEAST, este hace uso de la inyección de datos en el texto, para así poder obtener la información secreta que contiene. Gracias a este ataque se puede obtener las cookies de sesión, los datos de autenticación y más información secreta.

En cuanto al ataque POODLE en TLS, es igual que en SSL 3.0, salvo que no es necesario hacer un *downgrade* de protocolo, aunque en caso de explotar este ataque en TLS, es más complicado puesto que muy pocos servidores no hacen comprobación del texto de relleno.

Como se ha visto, este ataque afecta a la versión 3.0 del protocolo SSL, haciendo uso de un *downgrade*, además, puede afectar al protocolo TLS, mientras que el servidor que reciba los mensajes no compruebe el texto de relleno. En cuanto a la gravedad, tanto del ataque POODLE en SSL como en TLS se puede considerar media, ya que si se piensa en SSL 3.0, esta versión está cada vez más en desuso debido a su inseguridad, y también a su complejidad de realizar este ataque. Mientras que el ataque en el protocolo TLS hay pocos servidores que no comprueban el relleno de los bloques al cifrar, eso hace que sea bastante complicado llevar a cabo este ataque.

Para poder eliminar el ataque POODLE en SSL 3.0, la forma más sencilla es que el servidor deje de soportar dicho protocolo, pero no suele ser la mejor opción porque muchos de los clientes usan este protocolo. Otra opción es implementar el TLS_FALLBACK_SCSV, que no deja hacer un *downgrade* a SSL 3.0. Por último, se puede implementar la "división de registros anti-POODLE", la cual hace imposible realizar el ataque. Puesto que los registros se dividan en varias partes para que no se pueda usar la inyección de texto para sacar información.

2.3.6 RC4

Esta vulnerabilidad fue publicada en 2013, aunque ya desde 2001 se fueron publicando varios artículos que demostraban que el algoritmo de cifrado por flujo RC4[16], no era tan robusto como se pensaba. Una de las vulnerabilidades encontradas en este algoritmo de cifrado, consiste en la obtención de cookies de sesión, contraseñas y/o números de tarjeta de crédito, este ataque se conoce como "*Invariance Weakness*". También es posible la obtención de datos transmitidos por una conexión, esta vulnerabilidad hace uso de los sesgos estadísticos de las claves RC4. Todos los ataques realizados con este algoritmo de cifrado, se hacen a través de un *man in the middle*, para poder obtener los datos cifrados que envían tanto el servidor como el cliente, y de esta forma, poder generar ataques estadísticos y sacar datos. Uno de los más usados es la obtención de información a través de los mensajes *finished* entre cliente y servidor.

Para ello, el atacante tiene que introducirse en una sesión entre un cliente y servidor que usen como algoritmo de cifrado el RC4. Una vez realizado esto, el atacante hace un muestreo de los mensajes entre el cliente y el servidor, y espera hasta detectar una clave débil. De esta clave débil obtiene los LSB (*Less Significant Bits*), que son usados para extraer los bits menos significativos del texto plano a través del texto cifrado. A continuación, se muestra como se conservan los LSB al cifrar en RC4:

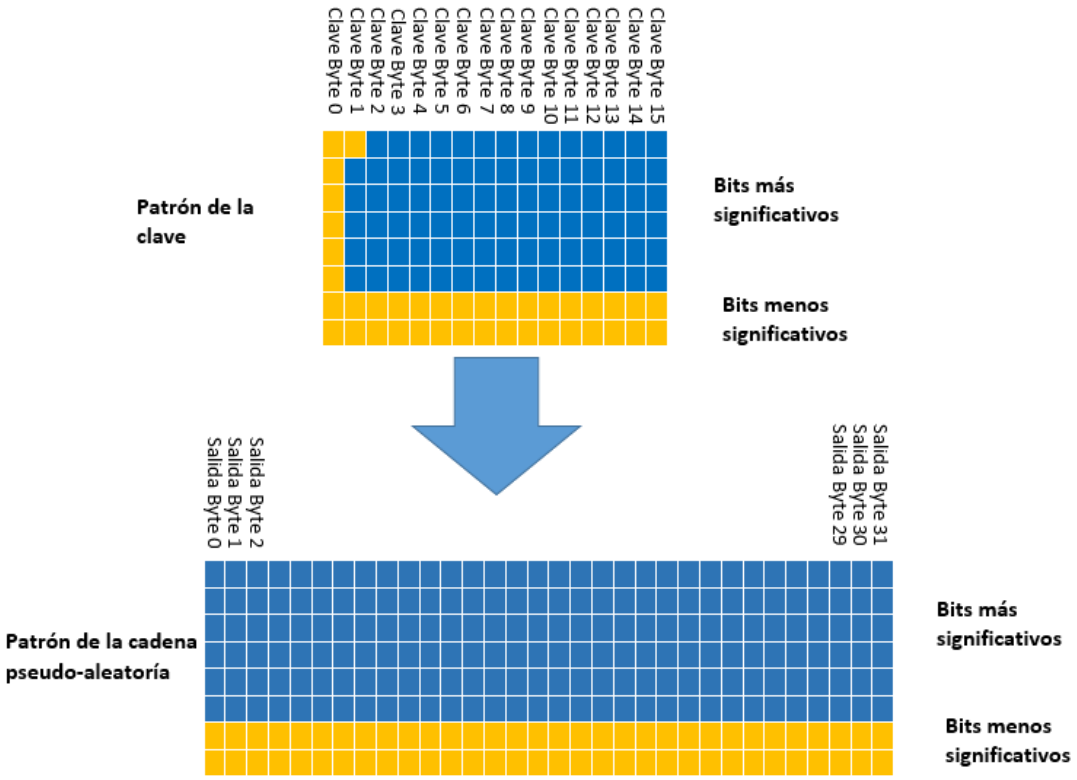


Figura 6: The invariance Flow

Como se observa en la imagen anterior, se puede observar como al cifrar los datos mediante RC4, los menos significativos, se mantienen inalterados. En la imagen siguiente se muestra como se realiza este ataque:

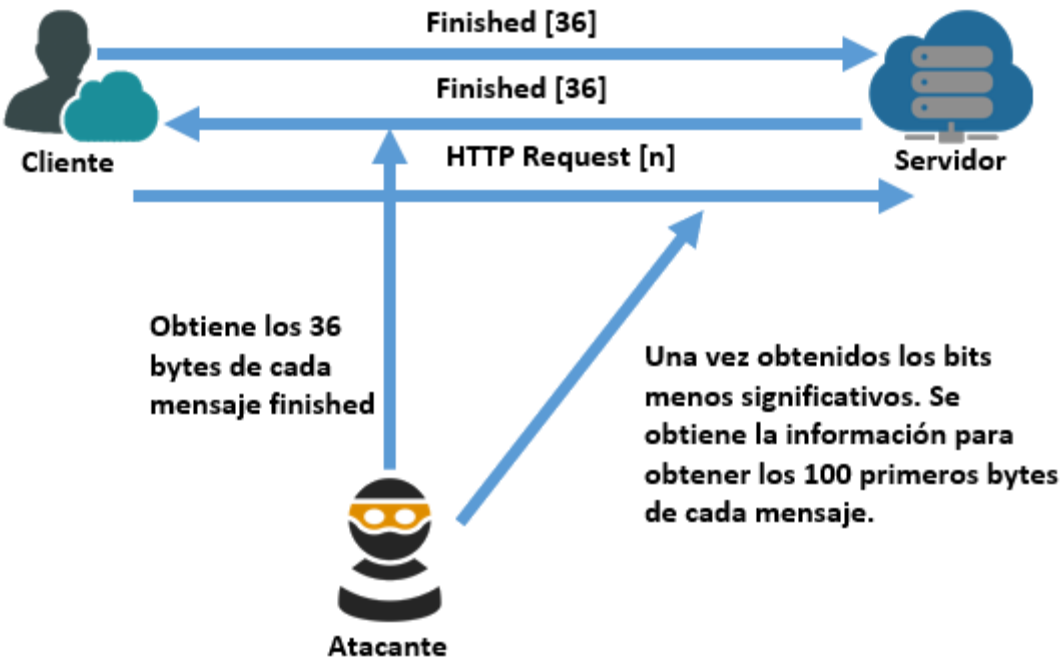


Figura 7: Ataque RC4

En cuanto a los protocolos (SSL y TLS) que pueden ser vulnerables a este ataque, son todos aquellos en los cuales se permita el uso de RC4, por lo tanto, son susceptibles con SSL como con TLS. Debido a que el ataque se puede dar tanto en SSL como TLS, y puesto que, a través de este, se puede sacar la cookie de sesión, la contraseña o los números de la tarjeta de crédito se puede considerar de una gravedad alta.

Para poder mitigar el ataque, la única posibilidad que hay, es eliminar la opción en los servidores de que un cliente pueda negociar dicho cifrado. Esta vulnerabilidad es bastante alta en cuanto a la gravedad, mientras que otras como BEAST es media, ya que suele existir un problema a la hora de elegir el tipo de cifrado. Esto se debe al uso de cifrados por bloque como DES o AES, tenemos el ataque BEAST, mientras que, si decidimos RC4, tendremos este ataque. De forma consensuada, se establece que es mejor optar por cifrados por bloque, a usar RC4, ya que al menos, por parte de los navegadores está solventada la vulnerabilidad BEAST.

2.3.7 Heartbleed

Esta vulnerabilidad es una de las más importantes y famosas de los últimos años. Este ataque fue descubierto en Abril de 2014, la cual permite ver la memoria de los sistemas protegidos. En sí, no es un problema criptográfico ni de implementación de los protocolos SSL y TLS. Esta vulnerabilidad afecta a la librería OpenSSL, que es ampliamente usada.

Este ataque se realiza de la siguiente manera. Primero, el atacante crea un paquete "*Heartbleed*"[17] y es enviado al servidor, el cual hace uso de OpenSSL. Segundo, el servidor procesa el paquete, y el código ejecutado en el servidor toma 64 KBytes extra de memoria, esto se realiza, con la intención de que, en esta información extra sacada del servidor, tenga información privada. Después, el servidor crea un paquete de respuesta con este extra de información y lo envía al atacante. Por último, el atacante analiza los datos para ver si ha devuelto algo interesante, sino, vuelve a realizar otra vez todo el proceso. A continuación, se muestra una imagen de cómo se realiza este ataque[18]:

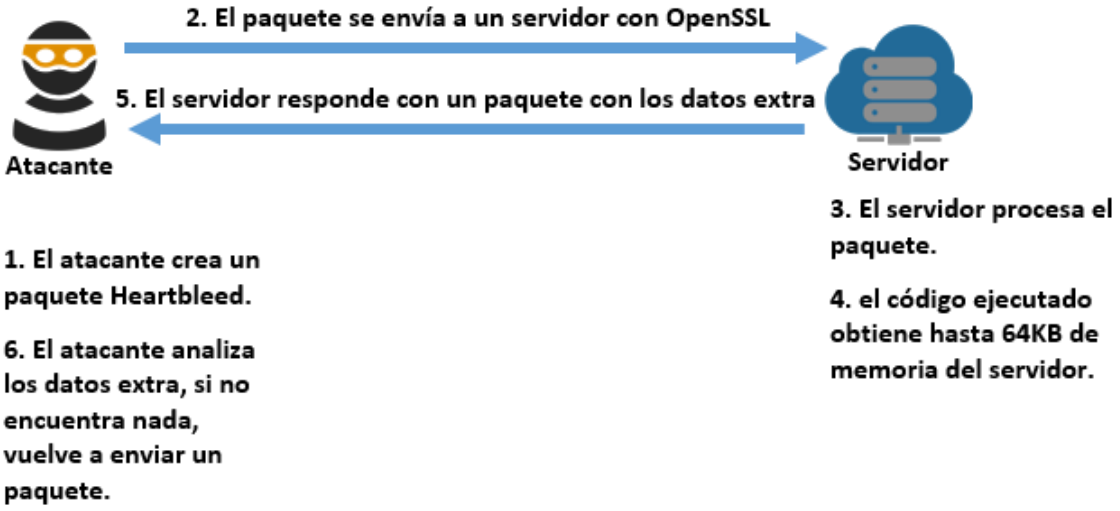


Figura 8: Ataque Heartbleed

Debido a que es un fallo de la librería OpenSSL, afecta a todos los servidores que usen las desde la versión 1.0.1 hasta la versión 1.0.1f, es decir, es indiferente el protocolo SSL o TLS que soporten. Debido al gran uso de esta librería y a la información sensible de obtener a través de este ataque, la gravedad de esta vulnerabilidad es muy alta.

Para poder mitigar esta vulnerabilidad es necesario actualizar la versión de OpenSSL a una versión posterior a las nombradas anteriormente, además, se recomienda encarecidamente realizar las siguientes acciones:

- Generar nuevas claves, obtener nuevos certificados y revocar los antiguos.
- Cambiar los tickets de sesión (si estos son usados).
- Cambiar otros datos sensibles, que hayan podido ser sustraídos de la memoria del servidor como por ejemplo las contraseñas de los usuarios.

2.3.8 DROWN

Este ataque se publicó en Marzo de 2016 y se recoge en el CVE-2016-0800. Esta vulnerabilidad afecta a todos los servidores que soporten la pila de protocolos TLS para el uso de SSL 2.0, es una vulnerabilidad de protocolos cruzados. La vulnerabilidad DROWN[19] (*Decrypting RSA with Obsolete and Weakened eNcryption*) no puede mitigarse en la parte de los clientes, puesto que es un fallo que aparece al usar varios protocolos, este permite romper el cifrado para obtener información sensible (números de tarjeta de crédito, contraseñas, datos privados, etc.).

Para realizar este tipo de ataque, el atacante primero debe de hacer un sondeo en el servidor web al que quiere atacar y comprobar que éste tiene el protocolo SSL 2.0 activo. Una vez comprobados los protocolos que usa, se obtiene a través de SSL 2.0 la clave privada del servidor, puesto que este protocolo es muy inseguro. Una vez obtenido, este se usa para descifrar los mensajes que llegan a los protocolos TLS (seguros), para así obtener la información privada. A parte de obtener la información de clientes, este ataque también es capaz de suplantar a un servidor, modificando el contenido que los clientes reciben. A continuación, se muestra una imagen de cómo se realiza el ataque:

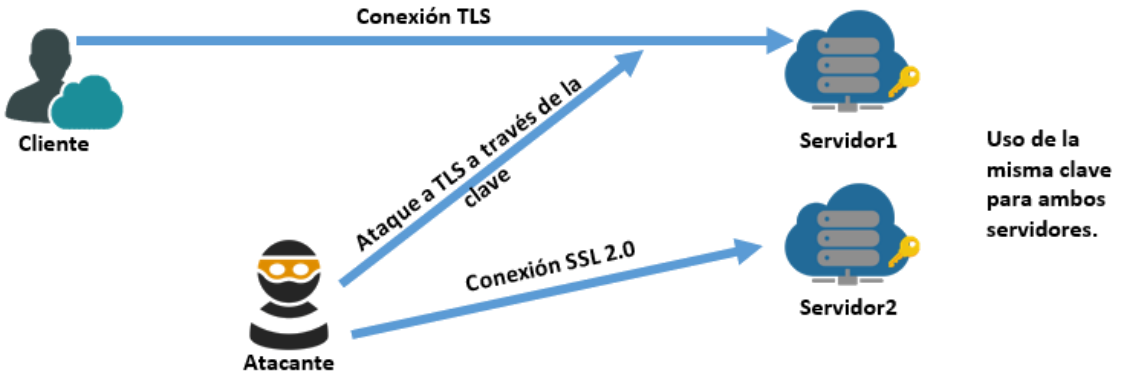


Figura 9: Ataque DROWN

Este ataque aparece en los servidores que usan el protocolo TLS y que además soportan SSL 2.0 para clientes desactualizados. Recordar que el uso de SSL 2.0 debería de estar desactivado puesto que es muy inseguro, pero este se deja debido a la gran cantidad de clientes que se conectan con este protocolo. Realizar este ataque es relativamente sencillo, y puesto que se obtiene un gran número de datos privados de los clientes, tales como, números de tarjetas de crédito, contraseñas, mensajería instantánea, correos, etc, y además en algunos casos el atacante se puede hacer pasar por el servidor modificando los datos que le llegan a los clientes. Por todo esto, se considera que este ataque es de gravedad alta.

Para poder eliminar esta vulnerabilidad en los servidores se pueden hacer dos acciones, pero antes de ello, se debe de comprobar que en el servidor esté activo SSL 2.0 y que se use la misma clave de cifrado en algún otro protocolo del servidor, como por ejemplo en SMTP, IMAP y POP. Una vez comprobado esto, la primera opción sería la de eliminar el soporte de SSL 2.0 en el servidor, aunque esto no es lo mejor, ya que muchos clientes se pueden quedar sin servicio debido a que no soporten protocolos superiores. La segunda opción es usar la librería criptográfica en una de las últimas versiones, en las cuales esta vulnerabilidad desaparece.

2.3.9 FREAK

El ataque FREAK[20] (*Factoring RSA Export Keys*) fue publicado en Marzo de 2015 bajo el código CVE-2015-0204. Este ataque se basa en un ataque *man in the middle*, en el cual se fuerza al servidor a usar un cifrado EXPORT con una clave RSA de 512 bits o inferior. Esto se debe a que Estados Unidos tenía miedo de exportar certificados cifrados tan seguros (claves robustas) que ni siquiera ellos mismos pudieran descifrar. De este

modo forzaron a que los cifrados EXPORT usen una clave RSA de longitud de 512 bits o menor.[21]

El ataque se realiza de la siguiente manera. Primero, el cliente intenta establecer conexión con el servidor, el atacante modifica la petición para usar el cifrado EXPORT. Entonces, el cliente establece conexión con el servidor y obliga a usar el cifrado EXPORT en la comunicación, de esta manera, el atacante consigue la clave, y es capaz de descifrar mensajes que son de terceros. Posteriormente, el atacante usando la técnica de *man in the middle*, se puede introducir en una comunicación entre el servidor y un cliente, a parte del cliente ya atacado, de esta manera el atacante puede descifrar y cifrar los mensajes que se envían sin que el cliente o el servidor lo sepan. A continuación, se muestra una imagen de su funcionamiento:

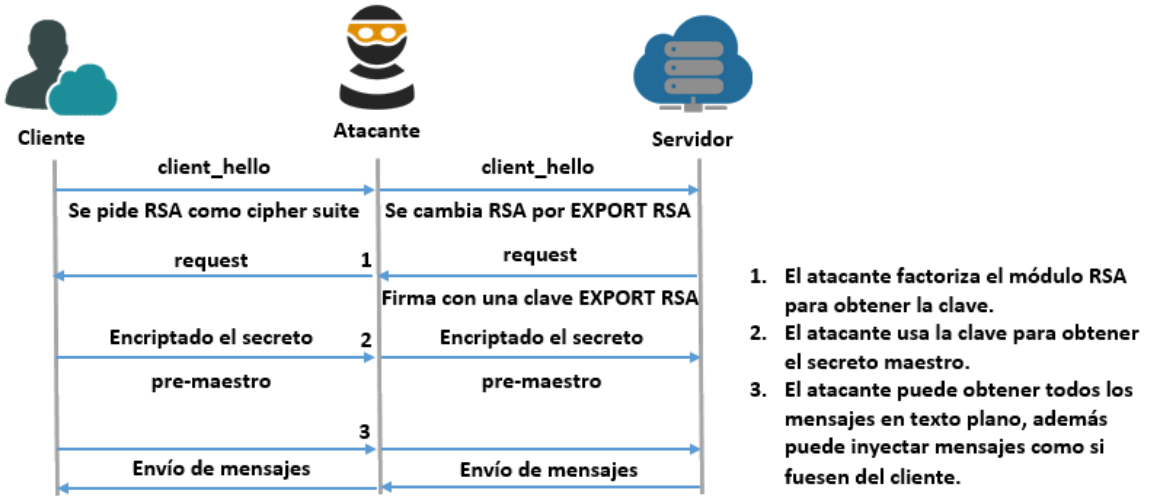


Figura 10: Ataque FREAK

Este ataque se puede producir en cualquiera de los protocolos, ya que no depende de un fallo de un protocolo, sino de la utilización de un cifrado. Debido a que solo se puede lograr este ataque a través de los cifrados EXPORT, y estos no están siendo muy utilizados. Además, el atacante sólo consigue los mensajes en claro de los clientes. Se puede considerar que este ataque tiene una gravedad media.

La forma de mitigar este ataque de la capa SSL es bastante sencillo, por parte de los clientes, es suficiente que actualicen sus navegadores, mientras que por parte de los servidores deberán dejar de dar soporte a los cifrados EXPORT.

2.3.10 OpenSSL CCS

El ataque OpenSSL CCS[22] (*OpenSSL Change Cipher Spec*) fue publicado en Junio de 2014 bajo el código CVE-2014-0224. Este ataque se basa en un ataque *man in the middle*, en el cual el atacante aprovechando el inicio del *handshake* entre cliente y servidor, justo después de los mensajes *client_hello* y *server_hello*, envía a ambas partes un mensaje de *change_cipher_spec* con una clave maestra de tamaño cero.

Esto hace que ambos usen una clave vacía, de esta forma, una vez terminado el *handshake*, el atacante podrá descifrar los mensajes, e incluso inyectar o cambiar los mensajes. Para que este ataque tenga éxito, primero debe de cumplirse que ambos extremos (cliente y servidor) hagan uso de la librería OpenSSL. A continuación, se muestra una imagen de la realización del ataque:

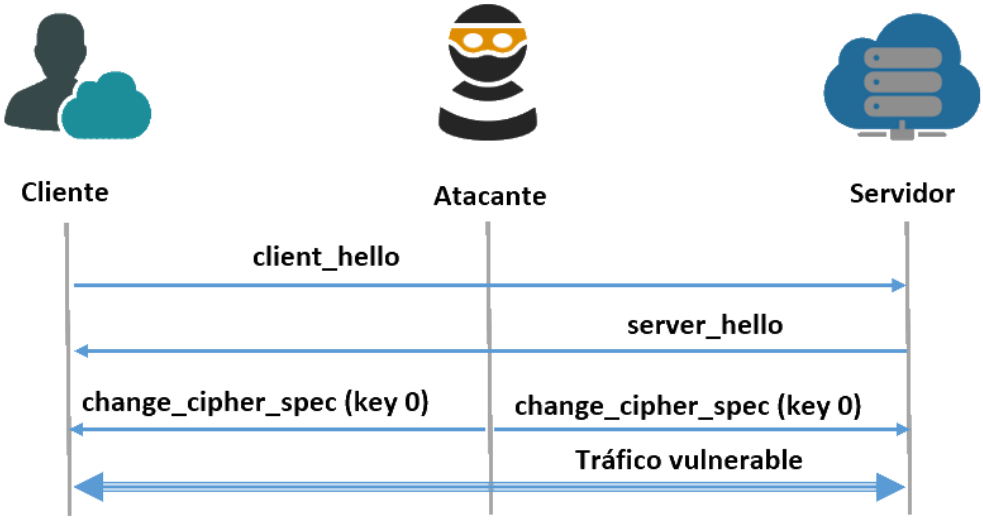


Figura 11: Ataque OpenSSL CCS

Este ataque se puede producir en cualquiera de los protocolos de la capa SSL, ya que la vulnerabilidad existente procede de la librería OpenSSL. Debido a que solo es necesario el envío de un mensaje, que no es preciso el uso de cálculos o estadísticas, que el atacante es capaz de descifrar los mensajes e incluso de modificarlos y dado que es bastante sencillo realizar este ataque, tiene una gravedad muy alta.

Para poder eliminar esta vulnerabilidad, solo es necesario actualizar la librería OpenSSL, las versiones afectadas por esta vulnerabilidad son: de la versión 1.0.1 hasta la 1.0.1g, de la versión 1.0.0 hasta la 1.0.0l y las versiones anteriores a la 0.908y

2.4 Resumen y comparación de los ataques

A continuación, se mostrará una tabla como resumen de los ataques analizados anteriormente. En dicha tabla se mostrarán los nombres de los exploits, el código y la gravedad que tienen (explicación en anexo Gravedad de los ataques en la capa SSL, D), así como los protocolos a los que afecta. La tabla es la siguiente:

Nombre	Código CVE	Gravedad	Protocolos/versiones afectados
Renegociación	CVE-2009-3555	Baja	Cualquiera
Downgrade	CVE-2014-3511	Media	Protocolo TLS
BEAST	CVE-2011-3389	Baja	TLS 1.0
CRIME	CVE-2012-4929	Baja	Cualquiera (Con compresión de datos)
POODLE	CVE-2014-3566(SSL) CVE-2015-8730(TLS)	Media(SSL) Media(TLS)	Protocolo SSL y TLS
RC4	CVE-2013-2566	Alta	Cualquiera (Con RC4)
Heartbleed	CVE-2014-0160	Muy alta	Cualquiera (versiones OpenSSL 1.0.1 hasta 1.0.1f)
DROWN	CVE-2016-0800	Alta	Protocolo TLS (si el servidor tiene SSL 2.0)
FREAK	CVE-2015-0204	Media	Cualquiera (cifrados EXPORT)
OpenSSL CCS	CVE-2014-0224	Muy alta	Cualquiera (versiones OpenSSL 1.0.1 hasta 1.0.1g, 1.0.0 hasta 1.0.0l y anteriores a 0.9.8y)

Tabla 4: Tabla de vulnerabilidades

3 Diseño

En esta sección se tratará de dar una descripción detallada del proyecto, los objetivos y funcionalidad de éste, como también la arquitectura y esquema general del proyecto y los requisitos necesarios para la aplicación.

3.1 Descripción del proyecto

Este proyecto se basa en un analizador de vulnerabilidades e información básica de la capa SSL. Para poder realizar el proyecto se ha creído conveniente hacer un estudio de las páginas y aplicaciones similares que se pueden encontrar por internet. Una vez hecho este estudio previo, se estableció, que el proyecto debería ser una página web, en la cual, se pudiese introducir una URL, dominio o IP a analizar. Siendo este un analizador de la capa SSL, la aplicación no realiza análisis de IPs, URLs o dominios que no tengan esta capa. Esta aplicación contiene un login para que solo los usuarios puedan hacer más de un escaneo al día, puesto que, sin autenticar, no se puede tener un control sobre los usuarios que usan la herramienta, y estos podrían hacer un uso malicioso de la aplicación. A parte del login se tiene la posibilidad de ver unas ciertas estadísticas de cada ataque (distribuidas en tiempo), como también de todos los ataques juntos. También se puede ver la información relacionada con los exploits usados en la aplicación y un cargador de exploits que permite subir a los usuarios nuevos exploits. Por último, y relacionado con el admin, se ha habilitado la opción de actualizar los datos de los exploits de forma automática desde la propia aplicación.

Para no correr ningún riesgo a la hora de subir algún exploit, este no se sube automáticamente a la aplicación, primero deberá pasar un control para verificar que es correcto.

Se ha optado por usar HTML para la parte gráfica de la página, así como también CSS para el estilo. Mientras que para la lógica de la aplicación se hace un uso combinado entre PHP y JavaScript. Para mantener los datos almacenados, se ha usado PostgreSQL como tecnología para base de datos, y para la ejecución de los exploits se ha usado el lenguaje de programación Python.

A continuación, se listan los requisitos funcionales y no funcionales de la aplicación, de esta forma se establece el alcance.

3.1.1 Requisitos Funcionales

Los requisitos serán expuestos agrupados por las páginas en donde estén establecidos, es decir, serán definidos según aparezcan en la aplicación.

- Página principal (Index):
    - Requisito [id: 1]: El formulario para el análisis de la capa SSL sólo podrá aceptar como argumentos válidos: una IP, un dominio o una URL.

- Requisito [id: 2]: Si el usuario se ha logueado previamente, éste podrá hacer todos los escaneos que precise.
- Requisito [id: 3]: Si el usuario es anónimo, es decir, no se ha logueado en la aplicación, éste sólo podrá hacer un escaneo cada 24 horas.
- Requisito [id: 4]: Si la IP, URL o dominio no existe, la aplicación deberá de mostrar un mensaje de error.
- Requisito [id: 5]: Si la IP, URL o dominio introducida, no se ha podido establecer conexión con ella, la aplicación deberá mostrar un mensaje de error.
- Requisito [id: 6]: Si un usuario anónimo intenta realizar un segundo escaneo, antes de que se hayan pasado las 24 horas de espera, la aplicación deberá mostrar un error.
- Requisito [id: 7]: Si el usuario es anónimo, la aplicación deberá mostrar en la parte inferior izquierda de la página un botón de información, mientras que, si es un usuario logueado, deberá mostrar, en este mismo sitio, el botón de información del usuario como también el anterior.
- Requisito [id: 8]: Si un usuario intenta hacer un análisis de "localhost" o de la IP 127.0.0.1, la aplicación mostrará un error y no realizará el análisis.
- Requisito [id: 9]: Si un usuario intenta enviar el formulario de análisis vacío, este mostrará un mensaje de campo vacío.
- Requisito [id: 10]: Desde esta página se debe poder tener acceso al login si el usuario no está logueado, a la información de los exploits, a las estadísticas y a la opción de cerrar sesión (*sing out*) si el usuario está logueado.

- Página de información(info):

    - Requisito [id: 11]: Esta página deberá mostrar la información de uso de la aplicación.
    - Requisito [id: 7]: Si el usuario es anónimo, la aplicación deberá mostrar en la parte inferior izquierda de la página un botón de información, mientras que, si es un usuario logueado, deberá mostrar, en este mismo sitio, el botón de información del usuario como también el anterior.
    - Requisito [id: 12]: Desde esta página se debe poder tener acceso a la home, al login si el usuario no está logueado, a la información de los exploits, a las estadísticas y a la opción de cerrar sesión (sing out) si el usuario está logueado.

- Página de información de usuario(user):

    - Requisito [id: 13]: Está página debe mostrar los exploits que ha subido el usuario, si tiene alguno y un formulario para que el usuario pueda cambiar su contraseña.
    - Requisito [id: 14]: Desde esta página se debe poder tener acceso a la home, al login si el usuario no está logueado, a la información de los exploits, a las estadísticas y a la opción de cerrar sesión (sing out) si el usuario está logueado.
    - Requisito [id: 15]: Desde esta página se debe poder tener acceso a la home, al login si el usuario no está logueado, a la información de los exploits, a las

estadísticas y a la opción de cerrar sesión (sing out) si el usuario está logueado.
    - Requisito [id: 7]: Si el usuario es anónimo, la aplicación deberá mostrar en la parte inferior izquierda de la página un botón de información, mientras que, si es un usuario logueado, deberá mostrar, en este mismo sitio, el botón de información del usuario como también el anterior.

- Página de login(login):

    - Requisito [id: 16]: Esta página deberá mostrar tres botones para la selección de login, *register* o *forgotten password*.
    - Requisito [id: 17]: La página deberá mostrar por defecto el botón de login seleccionado, al igual que el formulario mostrado será el de login.
    - Requisito [id: 18]: Si la contraseña es incorrecta a la hora de hacer login, la aplicación mostrará un mensaje de error.
    - Requisito [id: 19]: Si el usuario no existe en la aplicación, el login deberá enviar un mensaje de error.
    - Requisito [id: 20]: Si el usuario no introduce uno de los campos, este mostrará un mensaje de error.
    - Requisito [id: 7]: Si el usuario es anónimo, la aplicación deberá mostrar en la parte inferior izquierda de la página un botón de información, mientras que, si es un usuario logueado, deberá mostrar, en este mismo sitio, el botón de información del usuario como también el anterior.
    - Requisito [id: 21]: Si se pulsa uno de los otros dos botones (los que no estén seleccionados) deberán mostrar el formulario correspondiente.
    - Requisito [id: 22]: Si el usuario está en la pestaña de registro, deberá registrarse con un nombre que tenga solo letras y números, además de una longitud máxima de 12 caracteres.
    - Requisito [id: 23]: Si el usuario está en la pestaña de registro e introduce un nombre ya en uso, la aplicación enviará un mensaje de error.
    - Requisito [id: 24]: Si el usuario está en la pestaña de registro e introduce la confirmación de contraseña de forma incorrecta, la aplicación mostrará un mensaje de error.
    - Requisito [id: 25]: Si el usuario está en la pestaña de registro e introduce todos los datos correctamente, este redirigirá a la home, con el usuario ya logueado.
    - Requisito [id: 26]: Desde esta página se debe poder tener acceso a la home, al login, a la información de los exploits y a las estadísticas.

- Página de exploits (exploits):

    - Requisito [id: 27]: Esta página deberá mostrar un listado con los exploits en uso de la aplicación, un botón de creación de exploits, y en caso de que el usuario fuese admin, otro botón para dar de alta los exploits.
    - Requisito [id: 28]: Si el usuario es anónimo, e intenta acceder a la información detallada de un exploit la aplicación mostrará un mensaje de error.
    - Requisito [id: 7]: Si el usuario es anónimo, la aplicación deberá mostrar en la parte inferior izquierda de la página un botón de información, mientras

que, si es un usuario logueado, deberá mostrar, en este mismo sitio, el botón de información del usuario como también el anterior.
    - Requisito [id: 29]: Desde esta página se debe poder tener acceso a la home, al login si el usuario no está logueado, a las estadísticas y a la opción de cerrar sesión (sing out) si el usuario está logueado.
    - Requisito [id: 30]: Si se pincha en el botón de cómo se crea un exploit, la aplicación muestra un mensaje emergente.

- Página de estadísticas(statistics):

    - Requisito [id: 31]: Esta página deberá mostrar un listado de los exploits y ataques de la aplicación, junto con un botón para mostrar todas las estadísticas.
    - Requisito [id: 32]: Si el usuario es anónimo e intenta pinchar en uno de los exploit o ataques de la lista, la aplicación mostrará un mensaje de error, al igual que si pincha en el botón de mostrar todas las estadísticas.
    - Requisito [id: 7]: Si el usuario es anónimo, la aplicación deberá mostrar en la parte inferior izquierda de la página un botón de información, mientras que, si es un usuario logueado, deberá mostrar, en este mismo sitio, el botón de información del usuario como también el anterior.
    - Requisito [id: 33]: Si el usuario ha sido logueado anteriormente, y pincha en un exploit, un ataque o el botón de mostrar todas las estadísticas, la aplicación redirigirá correctamente a la información.
    - Requisito [id: 34]: Desde esta página se debe poder tener acceso a la home, al login si el usuario no está logueado, a la información de los exploits y a la opción de cerrar sesión (sing out) si el usuario está logueado.

- Página de creación de exploits(CreateExploit):

    - Requisito [id: 35]: Esta página deberá mostrar el formulario para la creación o actualización de un exploit.
    - Requisito [id: 36]: Si el usuario introduce un nombre que ya está en uso y lo intenta crear, la aplicación mostrará un mensaje de error.
    - Requisito [id: 37]: Si el usuario realiza el formulario correctamente, la aplicación redirigirá a la home.
    - Requisito [id: 38]: Si el usuario introduce un nombre que no existe, y pulsa el *checkbox* para hacer actualizar un exploit, la aplicación mostrará un mensaje de error.
    - Requisito [id: 39]: Si el usuario introduce un nombre existente y está pulsado el *checkbox* para actualizar el exploit, la aplicación actualizará el exploit.
    - Requisito [id: 7]: Si el usuario es anónimo, la aplicación deberá mostrar en la parte inferior izquierda de la página un botón de información, mientras que, si es un usuario logueado, deberá mostrar, en este mismo sitio, el botón de información del usuario como también el anterior.
    - Requisito [id: 40]: Desde esta página se debe poder tener acceso a la home, al login si el usuario no está logueado, a la información de los exploits, a las estadísticas y a la opción de cerrar sesión (*sing out*) si el usuario está logueado.

- Página de selección de IP (adddress):

    - Requisito [id: 41]: Esta página deberá mostrar una tabla con las posibles IPs que se hayan encontrado.
    - Requisito [id: 42]: Si se pincha en una de las IPs, la aplicación procederá al análisis.

- Página de información SSL(ssl):

    - Requisito [id: 43]: Esta página deberá mostrar toda la información de la capa SSL relacionada con una IP, dominio o URL.
    - Requisito [id: 44]: Si un usuario pincha en uno de los apartados de clave pública o cadena de certificados, la aplicación abrirá una nueva pestaña con dicha información.

- Página de información de exploits (exploitInfo):

    - Requisito [id: 45]: Esta página deberá mostrar en primera instancia y si procede, una breve descripción del exploit a mostrar, seguido de un botón para poder ver el código del exploit.
    - Requisito [id: 46]: Si un usuario pincha en el botón para mostrar el código, la aplicación mostrará el código en una nueva pestaña.

- Página de gráficos de estadísticas (statisticInfo):

    - Requisito [id: 47]: Esta página deberá mostrar tres intervalos, con un gráfico cada uno de ellos. Si alguno no tuviese información, se ocultaría. Estos intervalos son: última semana, último mes y todos.
    - Requisito [id: 48]: Si las estadísticas a mostrar, son de todos los ataques, entonces a parte de los intervalos y los gráficos, también mostrará una tabla de los ataques con porcentajes.

Una vez vistos los requisitos funcionales del proyecto, se procederá a listar los no funcionales.

3.1.2 Requisitos no Funcionales

A continuación, se listan los requisitos no funcionales pertenecientes al proyecto. Estas irán divididas según el tipo:

- Seguridad:

    - Requisito [id: n1]: La aplicación deberá diferenciar entre los diferentes tipos de acceso de los usuarios, es decir, deberá diferenciar entre usuario anónimo, usuario logueado y administrador de la aplicación.
    - Requisito [id: n2]: La aplicación deberá tener un sistema de protección en la base de datos, para que las contraseñas, no puedan verse en texto plano, es decir, estarán cifradas.

    - Requisito [id: n3]: La aplicación deberá contener un mecanismo para que no se pueda sacar información a través de los formularios de la aplicación.

- Mantenibilidad y portabilidad:

    - Requisito [id: n4]: La aplicación deberá poder ser usada bajo cualquier sistema operativo, ya sea Windows, Mac o una distribución Linux.
    - Requisito [id: n4]: La aplicación deberá poder ser ejecutada en los navegadores más usados, tales como Chrome, Firefox, Safari y Edge.

3.1.3 Alcance

Una vez listados los requisitos y explicado a grandes rasgos de qué trata el proyecto, se puede concluir que ha sido desarrollado para dar de alguna forma ayuda y recursos para que personas de cualquier parte del mundo puedan analizar su "sitio" web, y gracias a la funcionalidad añadida de poder ir subiendo ataques (para mayor seguridad del cliente) según se vayan descubriendo nuevas vulnerabilidades. Además, también se pueden ver las estadísticas de todos los sitios web que son analizados. Por todo ello, en primera instancia, esta aplicación va dirigida a todas las personas que quieran analizar su sistema online, es decir, desde un ordenador a través de un navegador.

Una consideración para el futuro, sería la posibilidad de expandir esta aplicación a otras plataformas, como los móviles y las tablets, así como de soportar una mayor variedad de navegadores. Por último, incluir información más detallada de los parámetros que se presentan en la aplicación, como la adición de nueva información relevante.

3.2 Diseño detallado

En este apartado se tratará de explicar el diseño escogido, también las tecnologías usadas, dando razones de peso o explicaciones de porque se ha decidido el uso de ellas.

A continuación, se muestra una imagen, a modo explicativo, para ver estos 6 bloques y cómo se relacionan entre ellos para su posterior detalle:

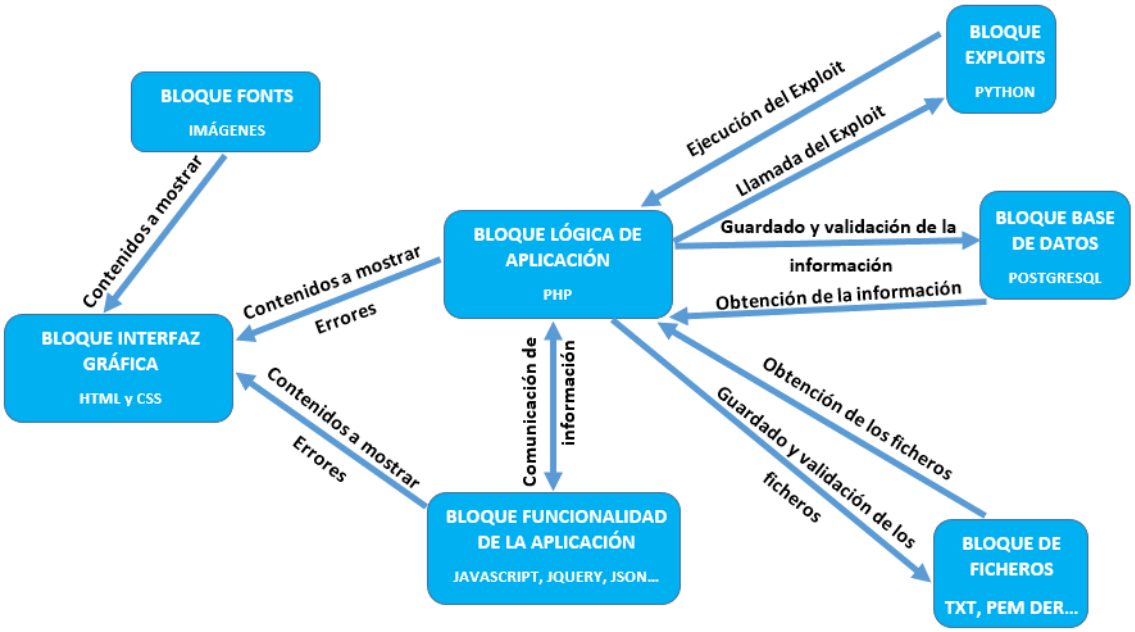


Figura 12: Bloques de la aplicación

Como se puede comprobar en la imagen anterior, la interfaz se comunica con los ficheros PHP y JavaScript para mostrar los datos necesarios en cada página de la aplicación. Se usa tanto PHP como JavaScript para que el contenido de la página sea dinámico, además, gracias al uso de JavaScript y AJAX se consigue hacer que una página de la aplicación cargue solo partes de la visualización sin necesidad de recargar toda la página entera. Los ficheros PHP son los que establecen comunicación con el "*backend*", es decir, PHP es el encargado de guardar, modificar y eliminar los distintos datos de los tres bloques restantes, y de que estos envíen la información correspondiente. PHP y JavaScript se encargan de comunicarse entre ellos para mostrar datos en la interfaz gráfica, esta comunicación se debe a que solo con el uso de PHP se consigue tener contenido dinámicamente, mientras que, con la inclusión de JavaScript, este contenido puede ser cargado asíncronamente y sin necesidad de recargar toda una página. Además, recordar que el código PHP se ejecuta en el servidor, mientras que el JavaScript en el cliente, de esta forma, se hace menos pesado el procesamiento en el servidor. Por parte del cliente, las funciones ejecutadas no tienen un coste muy alto.

Una vez explicado el diseño a rasgos generales de la aplicación, se procede a la explicación y definición de cada bloque por separado, en cada bloque se hará una descripción de los ficheros y las tecnologías que se usan, por último, se mostrará un árbol de directorios definiendo como es la configuración completa de la aplicación.

3.2.1 Interfaz gráfica

Como se ha visto antes, la interfaz gráfica hace uso de HTML y CSS para la estructura de la página y para el formato respectivamente. Primero se expone las tecnologías usadas en la aplicación así como la razón por las que han sido escogidas, y posteriormente de la organización y ficheros incluidos en el bloque.

- **HTML**:

HTML (*HyperText Markup Language*) es un lenguaje de marcado que se usa para la realización de las páginas web, en concreto para este proyecto se hace uso de HTML5, puesto que es la última versión de este lenguaje, y da una cierta facilidad a la hora de crear las páginas de la aplicación con respecto a versiones anteriores y añade nuevas etiquetas.

- **CSS**:

En cuanto al CSS (*Cascading Style Sheets*) es un lenguaje para definir un fichero HTML o XML, es decir, se encarga de dar formato al fichero HTML. Para este lenguaje se hace uso concretamente de CSS3, la última versión.

Se opta por el uso de estos lenguajes, debido a que el objetivo de este proyecto es que el usuario tuviese una herramienta para el análisis de la capa SSL, por ello, se eligió la realización de una página web, ya que da una gran facilidad en cuanto a portabilidad, al contrario que la realización de una app. Además, puede ser más accesible para todo el mundo, puesto que, una parte de la gente no usa mucho las apps (móviles y tablets) ni instalan aplicaciones (ordenador) mientras que si suelen usar la red.

- **Estructura**:

Para la estructura de la interfaz gráfica se ha decidido separar los ficheros CSS (formato) en un directorio en la raíz llamado "**css**", mientras que los ficheros HTML (estructura) están todos ubicados en la raíz de la aplicación. La estructura es la siguiente:

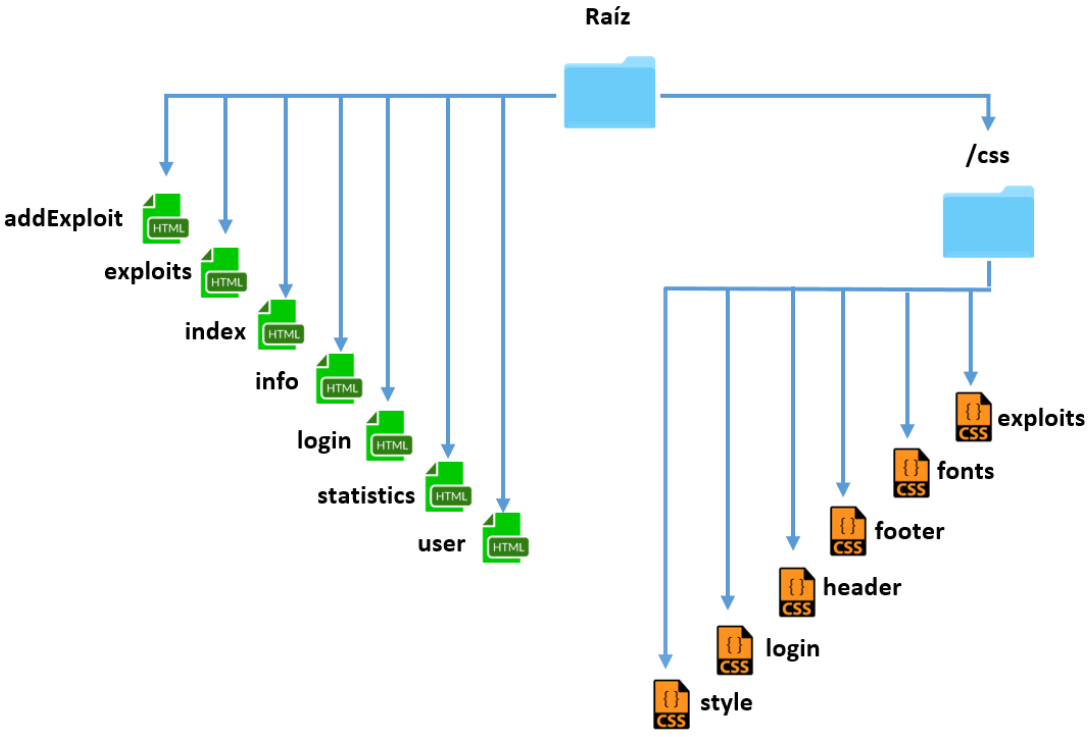


Figura 13: Interfaz gráfica

Como se puede observar en la imagen anterior, existen siete ficheros HTML y seis ficheros CSS, la funcionalidad de cada uno es:

- addExploit.html: Este fichero es el encargado de crear la estructura que presentará la página a la hora de añadir un exploit.
- exploit.html: Este fichero es el que se encarga de la estructura que presentará la página donde se muestran los exploits.
- index.html: Este fichero es la página principal de la aplicación, esta es la que se muestra cuando accedes a la aplicación, muestra el formulario para escanear una IP, URL o dominio.
- info.html: Este fichero es el que muestra la página de información de uso de la aplicación.
- login.html: Este fichero es el que crea la estructura de la página login, en la cual se establece el login, el registro y el "se me ha olvidado la contraseña".
- statistics.html: Este fichero es el que crea la estructura de la página de estadísticas.
- user.html: Este fichero es el que crea la estructura de la página de la información de usuario.
- exploits.css: Este fichero se encarga de dar formato a la página de exploits y también a la página de estadísticas.
- fonts.css: Este fichero se encarga de dar formato a la página home, a la de información, a la de login, a la de exploits, a la de añadir exploits a la de estadísticas y al de usuario.
- footer.css: Este fichero se encarga de dar formato al pie de la página a todas las páginas que se encuentran en la aplicación.
- header.css: Al igual que el footer, este fichero da el formato a la cabecera de la página, es decir, da el formato a todas las páginas.
- login.css: Este fichero se encarga de dar formato a la página de login, pero también da formato a la página de añadir exploit, a la home y a la de información.
- style.css: Este fichero se encarga de dar formato a la página de home y a la de información.

3.2.2 Base de datos

Para el desarrollo de la aplicación se ha decidido usar una base de datos para poder almacenar la información. De esta manera se ha optado por utilizar PostgreSQL para la creación de la base de datos y esta elección está fundamentada en que el sistema de gestión de base de datos ha sido una herramienta utilizada a lo largo de la carrera universitaria. Además hay que destacar otro punto muy importante a la hora de haber elegido PostgreSQL: es libre.

Para poder mantener toda la información coherente y bien definida se han creado cuatro tablas, una para la información del usuario, otra para los exploits, otra para las IPs y la última para las estadísticas. A continuación, se muestra el esquema de la base de datos y las relaciones entre tablas:

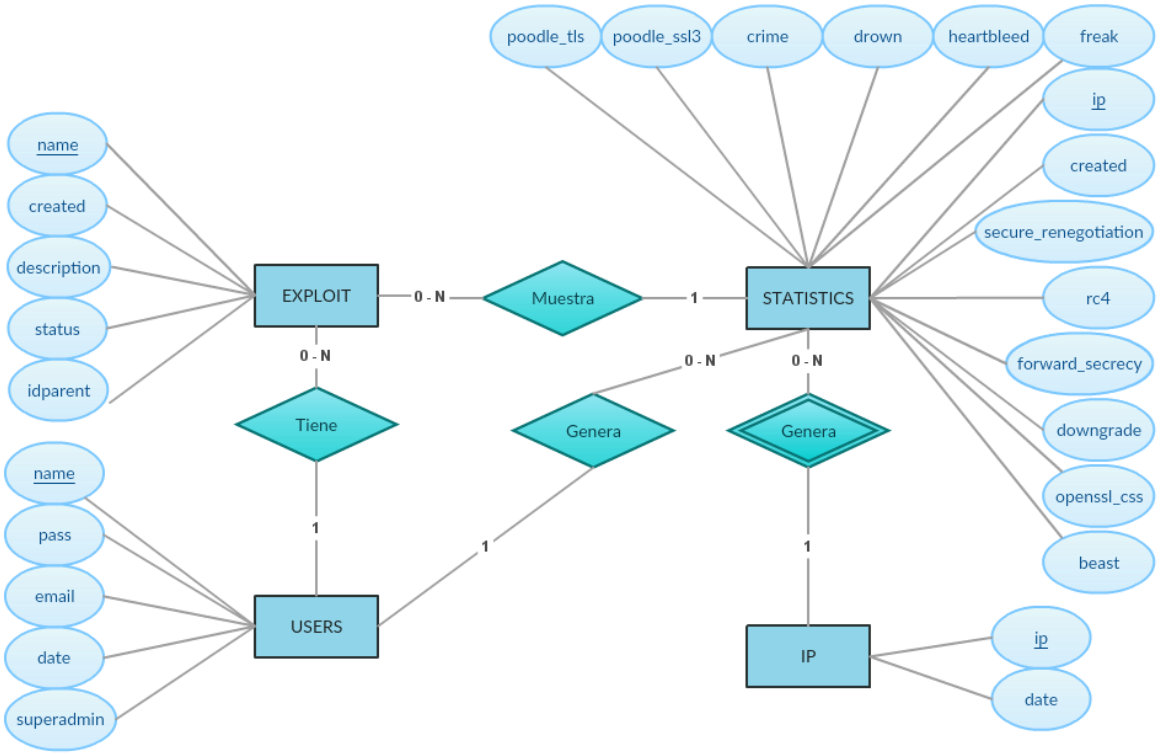


Figura 14: Base de datos

Como se puede ver en la imagen anterior la tabla USERS se relaciona tanto con la tabla EXPLOIT, ya que un usuario es el que crea un exploit o lo actualiza, como también con la tabla estadísticas, esto se debe a que es el usuario el que, al analizar un servidor, se generan las estadísticas. Para esta tabla encontramos los siguientes atributos:

- name: Este atributo es la clave primaria, debe de ser único y no nulo. Define el nombre del usuario.
- pass: Este atributo define la contraseña del usuario. Se usa la encriptación SHA-1 para mayor seguridad.
- email: Este atributo define el email del usuario.
- date: Este atributo define la fecha de alta del usuario.
- superadmin: Este atributo determina los privilegios del usuario, si tiene valor ‘0’, es un usuario normal, mientras que, si toma valor ‘1’, es superadmin.

La siguiente tabla es EXPLOIT, esta se relaciona como se ha visto antes con la tabla USERS y con STATISTICS, ya que son las estadísticas las que tienen implícito una lista de exploits. En esta tabla podemos encontrar los siguientes atributos:

- name: Este atributo define el nombre del exploit, debe de ser único y no nulo debido a que es la clave primaria de la tabla.
- created: Este atributo define la fecha de creación del exploit.
- description: Este atributo puede tener o no contenido, define la información sobre el exploit que el usuario puede subir a la aplicación.

- status: Este atributo define si el exploit está validado por el administrador para que se pueda hacer uso en el análisis de la capa SSL, toma el valor ‘0’ si no está validado y ‘1’ si lo está.
- idparent: Este atributo actúa como clave foránea con la tabla USERS, se usa el nombre del usuario para la relación.

La tabla IP se relaciona únicamente con la tabla STATISTICS, puesto que al igual que el usuario, esta actúa como uno, debido a que cuando un usuario anónimo quiere realizar un análisis, este se almacena a través de la IP. Y tiene los siguientes atributos:

- ip: Este atributo actúa como clave primaria de esta tabla y define la IP del usuario anónimo.
- date: Este atributo define la fecha en la cual el usuario anónimo ha realizado su último análisis, de esta manera se le obliga a esperar 24 horas.

Por último, queda la tabla STATISTICS, la cual se relaciona con el resto de las tablas tal como se ha nombrado anteriormente. Esta tabla tiene los siguientes atributos (se omitirán todos los atributos que sean exploits):

- ip: Este atributo actúa como clave primaria de la tabla, define la IP analizada.
- created: Es la fecha en la cual se ha hecho el análisis de la IP.
- [exploit]: Se refiere a todos los atributos que son exploits. Estos tienen valor ‘0’ en el caso de que no esté activa la vulnerabilidad y a ‘1’ si lo está.

3.2.3 Lógica de la aplicación

En este bloque se hace uso del lenguaje PHP, el cual se usa para dotar de contenido dinámico a una página web, además de dar la posibilidad de modificar su estructura. A continuación, se definirá el lenguaje usado y posteriormente se mostrará la organización de estos ficheros.

- PHP:

PHP (*PHP Pre Hypertext*) es un lenguaje de programación que se ejecuta en la parte del servidor y se usa para proveer a una página web de contenido dinámico. Para el desarrollo de este proyecto se ha hecho uso de la versión 5.3.10 de PHP.

Se opta por el uso de este lenguaje, debido a que es un lenguaje bastante flexible y potente, como también es uno de los más usados para el desarrollo web y es el lenguaje utilizado a lo largo de la carrera para la creación de páginas web.

- Estructura:

Para la estructura de la lógica de la aplicación se ha decidido mantener los ficheros PHP al igual que los HTML en el directorio raíz de la aplicación. La estructura es la siguiente:

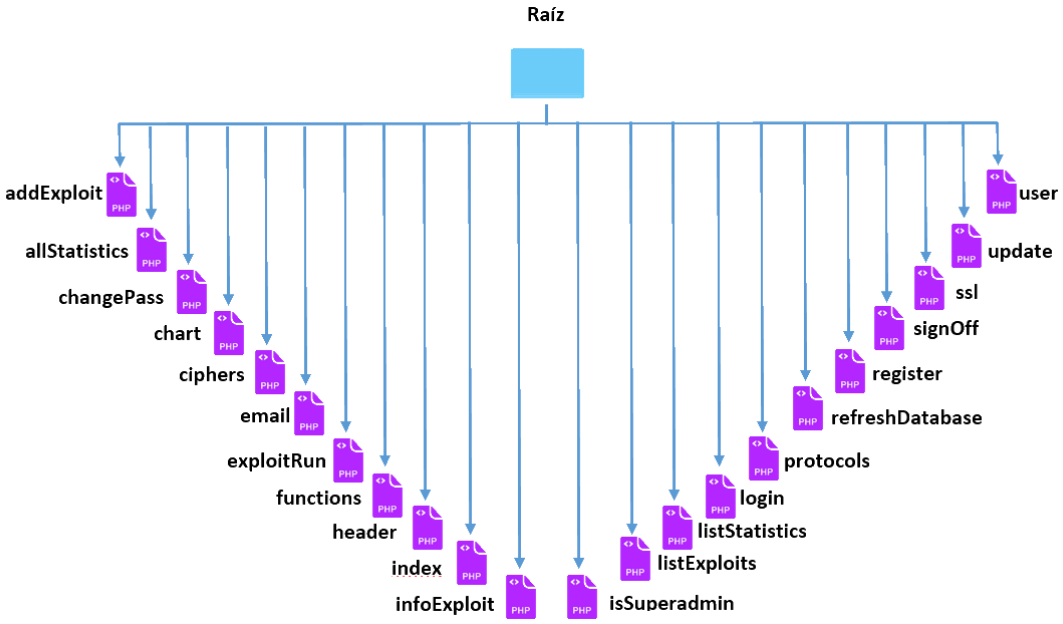


Figura 15: Lógica de la aplicación

Para la lógica de la aplicación se ha decidido crear 22 ficheros, a continuación, se define la funcionalidad de cada uno:

- addExploit.php: Este fichero realiza las funciones necesarias para añadir un nuevo exploit a la aplicación.
- allStatistics.php: Este fichero realiza las funciones necesarias para obtener los datos de todas las estadísticas para mostrarlas.
- changePass.php: Este fichero realiza las funciones del cambio de contraseña del usuario.
- chart.php: Este fichero realiza las funciones de obtener las estadísticas de un único exploit.
- ciphers.php: Este fichero realiza las funciones de testear todos los cifradores para saber cuáles soporta el servidor a analizar.
- email.php: Este fichero debería realizar el envío de mensajes cuando un usuario olvida la contraseña. Esta funcionalidad se deja para temas futuros.
- exploitRun.php: Este fichero se encarga de lanzar todos los exploits y guardar las estadísticas.
- functions.php: Este fichero, recoge todas las subfunciones utilizadas en el resto de la aplicación.
- header.php: Este fichero realiza las funciones que modifican el menú de navegación de la cabecera, dependiendo si está o no logueado el usuario.
- index.php: Este fichero realiza las funciones para comprobar si el usuario está o no logueado en la home para realizar el análisis, y en caso de que el usuario sea anónimo comprueba su último análisis.

- infoExploit.php: Este fichero realiza las funciones para obtener toda la información de un exploit.
- isSuperadmin.php: Este fichero comprueba si el usuario en la aplicación es superadmin.
- listExploits.php: Este fichero obtiene la lista de exploits para mostrarlos en la página de exploits.
- listStatistics.php: Este fichero obtiene la lista de estadísticas (exploits y otros datos) para mostrarlos en la página de estadísticas.
- login.php: Este fichero se encarga de realizar las funciones de login.
- protocols.php: Este fichero se encarga de realizar la verificación de protocolos soportados por el servidor a analizar.
- refreshDatabase.php: Este fichero realiza las funciones de actualizar la base de datos para añadir un nuevo exploit verificado al análisis de una página.
- register.php: Este fichero se encarga de las funciones de registro de usuarios de la aplicación.
- signOff.php: Este fichero se encarga de cerrar la sesión del usuario en la aplicación.
- ssl.php: Este fichero se encarga de todas las funciones del analizador de la capa SSL.
- update.php: Este fichero se usa para mostrar todos los exploits que no son usados en la aplicación. Esta función sólo es accesible si eres super admin.
- user.php: Este fichero realiza las funciones de mostrar la información del usuario.

3.2.4 Funcionalidad de la aplicación

Este bloque hace uso del lenguaje JavaScript, el cual se usa para dotar de contenido dinámico a una página web como con PHP, además de dar la posibilidad de modificar la estructura de la página, la diferencia entre estos dos lenguajes es el lugar donde se ejecutan. A continuación, se definirá el lenguaje usado, como bibliotecas y APIs usadas en JavaScript, posteriormente se mostrará la organización de estos ficheros.

- **JavaScript**:

JavaScript es un lenguaje de programación interpretado que se usa para el desarrollo de páginas web. Este lenguaje esta débilmente tipado y muy dinámico, por ello se hace uso de él, además, debido a que se ejecuta en la parte del cliente, es decir, en el navegador web. Junto con este lenguaje se hace uso de AJAX, una técnica usada en JavaScript para dotar a la página de interactividad gracias a las comunicaciones asíncronas. También se usa la librería jQuery, que permite una mayor facilidad para modificar los ficheros HTML. Por último, se usa una API llamada Amcharts, la cual es usado para la generación de los gráficos para las estadísticas.

- **Estructura**:

Para la estructura de la funcionalidad de la aplicación se ha decidido crear un directorio llamado "**js"** en el directorio raíz para tener todos estos archivos almacenados en un mismo directorio. Recordar que estos ficheros se encargan de algunas funciones en los ficheros HTML y se comunican con los ficheros PHP para obtener información. La estructura es la siguiente:

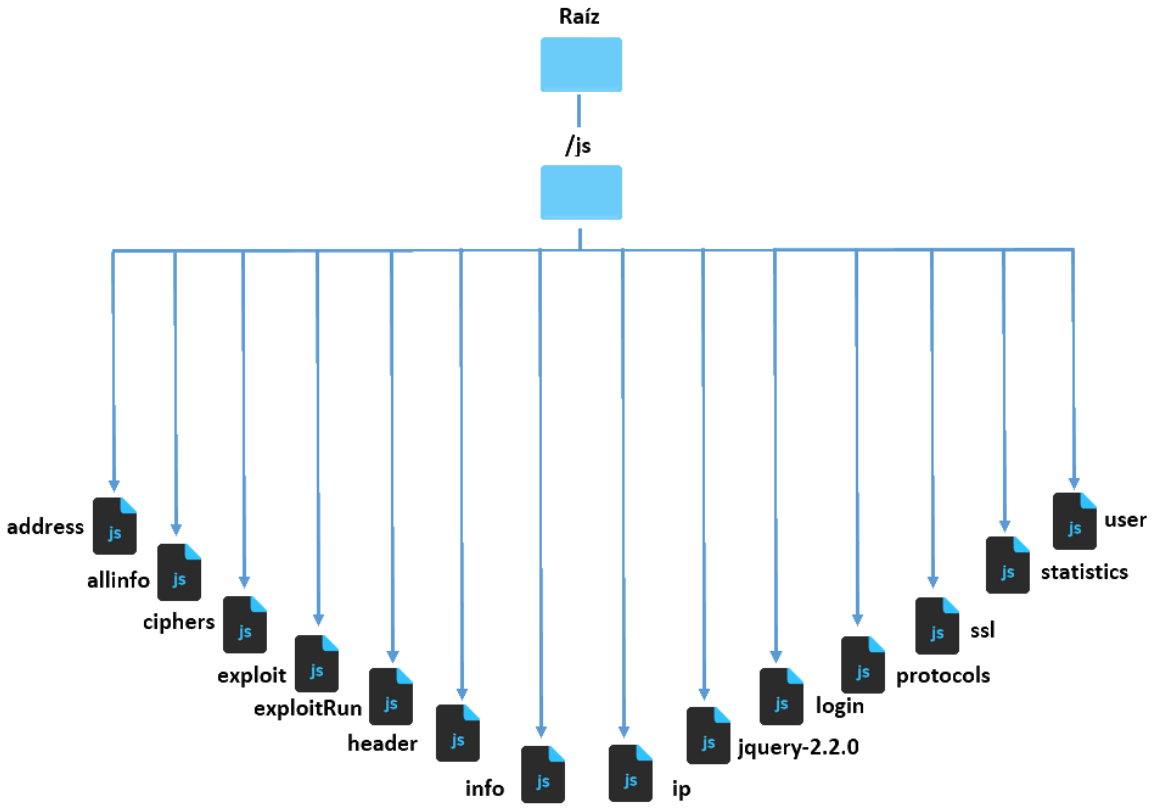


Figura 16: Funcionalidad de la aplicación

Como se puede ver, se han creado 14 ficheros JavaScript para realizar funciones de comunicación entre PHP y HTML, además de verificar los parámetros de los formularios. Se muestran a continuación las funciones que realicen cada fichero:

- address.js: Este fichero se encarga del control de errores del formulario de la home.
- allinfo.js: Este fichero es el encargado de preparar la página para el análisis de la capa SSL, como también de los errores.
- ciphers.js: Este fichero se encarga de obtener los cifradores obtenidos mediante PHP, para mostrarlos en la página.
- exploit.js: Este fichero es el encargado de crear los exploits o modificarlos.

- exploitRun.js: Este fichero se encarga del control de errores de la página de exploits y de la creación de ellos. Además de subir los exploits validados para su uso.
- header.js: Este fichero se encarga de actualizar el menú de cabecera de la aplicación a través de la llamada a los PHPs que modifican esta cabecera.
- info.js: Este fichero obtiene la información de la IP analizada.
- ip.js: Este fichero se comunica con PHP para obtener la lista de IPs posibles para el análisis de la página.
- jquery-2.2.0.js: Este fichero se encarga de dar soporte a la librería jQuery para el uso de esté en la aplicación.
- login.js: Este fichero se encarga del control de errores y de la gestión de la página de login.
- protocols.js: Este fichero se encarga de obtener los datos del PHP, para poder mostrar los protocolos soportados al analizar una página.
- ssl.js: Este fichero se encarga de obtener la información SSL recogida del PHP.
- statistics.js: Este fichero se encarga de obtener los datos de las estadísticas para crear los gráficos.
- user.js: Este fichero se encarga de obtener los datos del usuario para mostrarlos.

3.2.5 Exploits

Este bloque recoge tanto todos los ficheros que han sido creados por los usuarios para la inclusión de nuevos exploits en los análisis de las páginas, como los ficheros con los resultados. Para ello, se ha decidido conveniente el uso de Python como lenguaje usado para estos exploits. A continuación, se dará la explicación del lenguaje usado para los exploits y se definirá la estructura.

- **Python**:

Python es un lenguaje de programación interpretado, multiplataforma y bastante flexible, ya que permite orientación a objetos, programación funcional y programación imperativa. También porque muchos de los exploits se pueden encontrar en este lenguaje. En principio, el exploit aportado por el usuario debe de ser autosuficiente, es decir, debe de poder ejecutarse directamente el código aportado y dar el resultado de la vulnerabilidad respecto a una IP. Si fuese necesaria la introducción de una librería Python para la ejecución de un exploit, esta deberá de ser enviada al administrador de la aplicación, y así poder usar el exploit.

- **Estructura**:

En cuanto a la estructura, se ha decidido crear un directorio en la raíz llamado exploit, en el cual, albergará un directorio llamado txt que contendrá algunos de los resultados de estos exploits. Además, por cada usuario que haya subido un exploit y este haya sido verificado, se creará un directorio con el nombre del usuario y dentro de él, otro con el nombre del exploit, el cual contendrá el fichero. A continuación, se muestra la estructura:

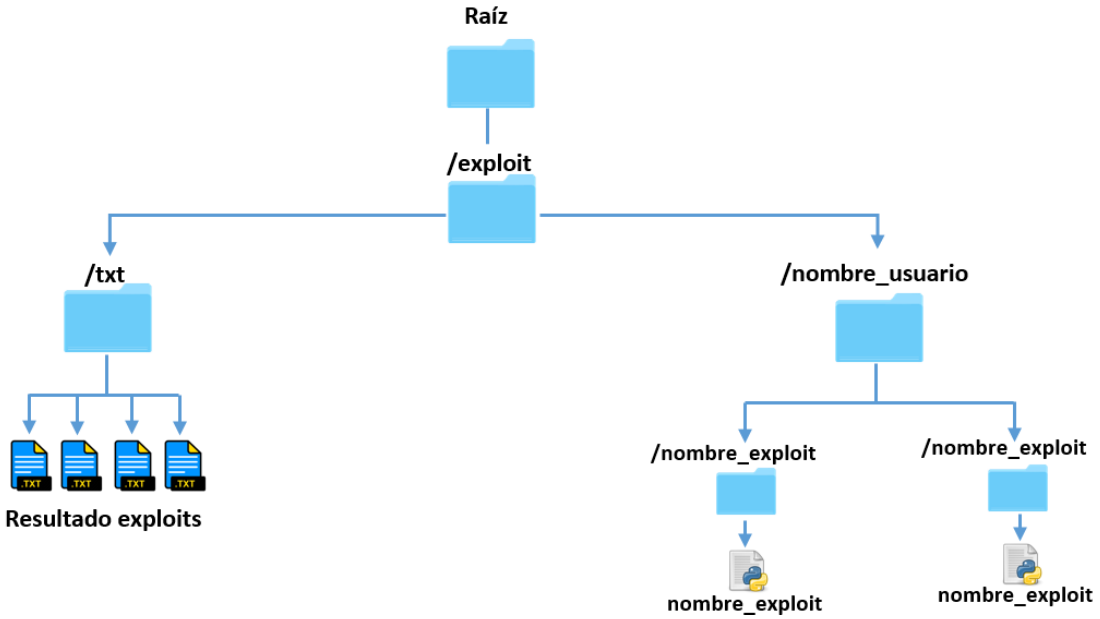


Figura 17: Exploits

3.2.6 Ficheros

En este bloque se recogen el resto de los ficheros que intervienen en el análisis de las vulnerabilidades y certificados. Estos ficheros se almacenan en un directorio contenido en la raíz llamado ssl, en el cual, a su vez se establecen tres nuevos directorios. Estos se llaman der, pem y key.

En el directorio der, se puede encontrar los certificados de las páginas analizadas con las respectivas listas de certificados revocados para verificar su estado.
En el directorio pem, se pueden encontrar todos los certificados de cada página además de una cadena de certificados donde se almacenan todos.
En el directorio key, se pueden encontrar las claves públicas de las páginas previamente analizadas. Por último, en el directorio raíz de ssl se encuentran los ficheros con las respuestas de las conexiones de los distintos protocolos de la capa SSL.

Este bloque recoge todos los ficheros que han sido creados por los usuarios para la inclusión de nuevos exploits en los análisis de las páginas, como los ficheros con los resultados. A continuación, se definirá la estructura.

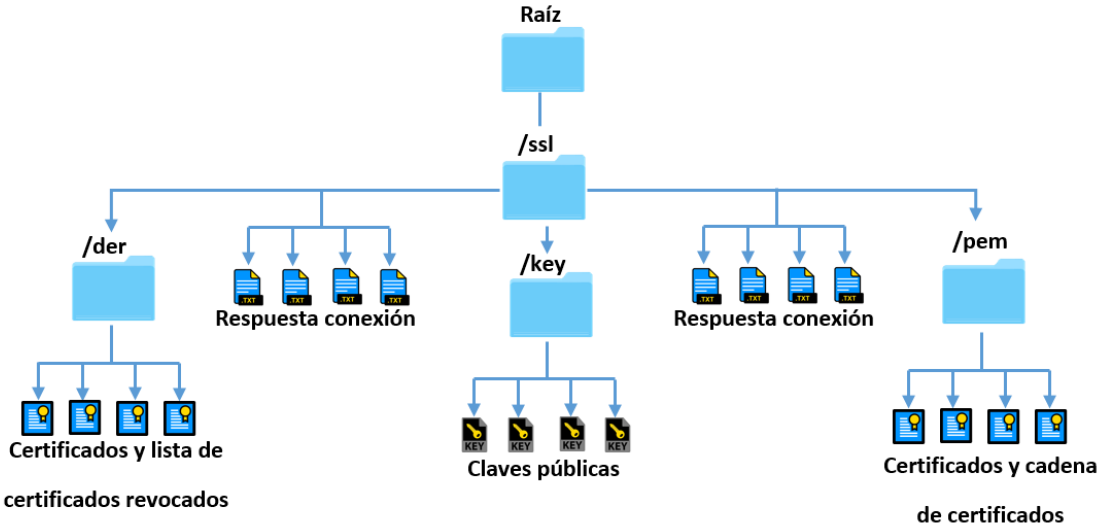


Figura 18: Ficheros

3.2.7 Fonts

Este es el último de los bloques, alberga todas las imágenes y archivos fuentes usados en la aplicación, tales como el logo o imágenes usadas como botones. Este bloque se encuentra en el directorio fonts, ubicado en la raíz de la aplicación. La estructura de este bloque es la siguiente:

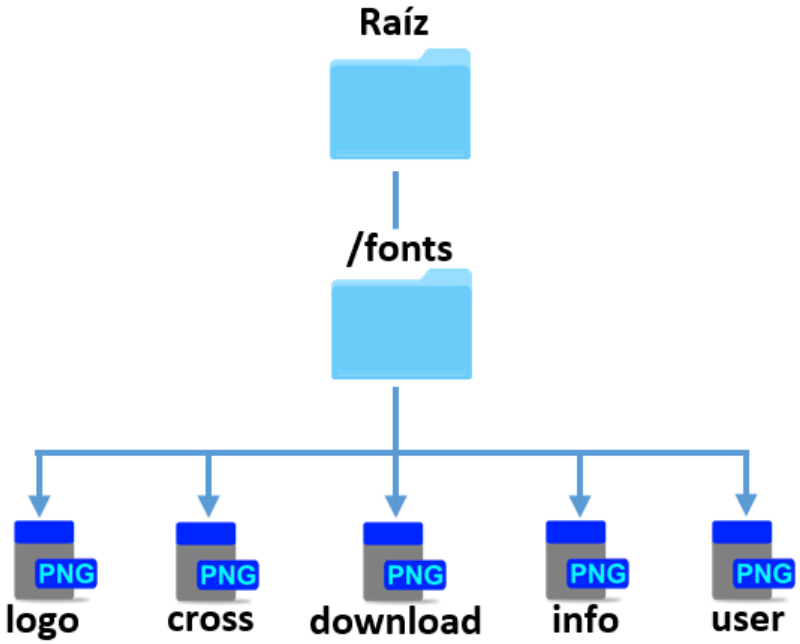


Figura 19: Fonts

4 Desarrollo

En este apartado se definirá cómo ha sido el desarrollo de esta aplicación, primero se verán los programas que se han usado para la realización de la aplicación, como también otros programas usados para diversas funciones. Posteriormente se explicará la metodología seguida para realizar el proyecto junto con su progresión.

4.1 Programas usados

Para la realización de este proyecto, lo primero es pensar en los programas usados para la edición de los ficheros de programación de la página web. También el programa usado para la gestión y creación de la base de datos, como también un programa o soporte para almacenar las versiones realizadas de la aplicación. Por otra parte, se ha decidido usar una aplicación para hacer un seguimiento diario y poder anotar errores o funciones a realizar. Por último, se hace uso de una máquina virtual para las pruebas de la capa SSL. A continuación, se listan los programas usados para el desarrollo de este proyecto como también cuál ha sido su función:

- NetBeans IDE 8.1: Este programa es un entorno de desarrollo libre, es usado para la creación de los ficheros HTML, CSS, JavaScript y PHP (realización de la página web). También se usa este programa para la creación de un servidor el local para las pruebas de la aplicación.
- pgAdmin III v1.22.0: Este programa es una interfaz gráfica para la administración de bases de datos PostgreSQL, el cual es libre. Se ha usado para la creación, modificación y verificación de las tablas.
- Google Drive: Este servicio de alojamiento se ha usado para la subida a la nube de las distintas versiones de la aplicación, para tenerlas accesibles en todo momento en cualquier lugar. Además, se ha usado una carpeta compartida con el tutor para mantener una guía a lo largo del proyecto.
- Paint.NET (v4.0.9): Es un editor de imágenes, que se ha usado para modificar o crear imágenes, tales como el logo de la aplicación o los botones.
- Wunderlist: Esta herramienta online se ha usado para mantener unos objetivos y tareas a lo largo del desarrollo del proyecto. Desde esta aplicación, se han creado varias tareas e hitos a realizar.
- Google API: Esta API aportada por Google ha sido usada solo para la visualización de los mapas de Google Maps a la hora de mostrar las estadísticas.
- Google Chrome y Mozilla Firefox: Estos dos navegadores web se han usado para la realización de las diferentes pruebas de la aplicación.
- VMware workstation 12: Esta herramienta permite la creación de máquinas virtuales, se ha usado debido a la necesidad de hacer pruebas de la capa SSL, cambiando configuraciones de la red.
- PyCharm (v2016.1.4): Este entorno de desarrollo se ha usado solo para la verificación de los ficheros Python.

4.2 Metodología seguida

En cuanto a la metodología seguida para este proyecto, se ha decidido en primer lugar, un análisis del proyecto y captura de requisitos, y una vez establecidos estos, desarrollar la aplicación. Se han seguido los siguientes pasos:

- Lo primero de todo, se ha realizado un estudio previo de aplicaciones similares. De esta forma se han establecido unas ciertas directrices y guías para hacer la aplicación. Este estudio previo incluye a la página ssllabs[2] y sslanalyzer.comodoca[3].
- Posteriormente, se ha realizado la captura de requisitos para la realización del proyecto. Estos se incluyen en la sección 3.1.1. y 3.1.2. de este documento.
- Una vez realizado el análisis y los requisitos se empieza por el desarrollo de la aplicación en sí. Este desarrollo comprende dos etapas, la primera de ellas es el desarrollo de la estructura de la aplicación, es decir, se empieza a desarrollar la interfaz de la aplicación (HTML, CSS y algún JavaScript). En la segunda etapa, se empieza a desarrollar la funcionalidad y lógica de la aplicación (PHP y resto de JavaScript). Para estas dos etapas de desarrollo se ha usado una metodología de desarrollo incremental, de esta forma, cada página (fichero de interfaz gráfica) se desarrolla de manera independiente y de forma exclusiva.

Para el desarrollo de cada página, primero se establecen los requisitos (recogidos anteriormente) y se realiza la página (interfaz gráfica), una vez realizada, se realizan pruebas y validaciones sobre la misma. Una vez pasadas las pruebas se procede con la siguiente página. Ya terminadas todas las páginas, se efectúan pruebas globales sobre todas ellas, estas pruebas tienen retroalimentación sobre cada uno de los incrementos para la modificación de los ficheros.

Después del desarrollo de las interfaces gráficas de todas las páginas, se procede al desarrollo de la funcionalidad y la lógica de las mismas. Para ello, se sigue la misma metodología, se codifica cada fichero por separado y una vez terminado se realizan las distintas pruebas sobre él. A continuación, se efectúa el siguiente fichero. Una vez realizados todos se hacen pruebas globales de todos los ficheros de la aplicación y se retroalimentan por si hay modificaciones.

- Por último, se realizan unas pruebas globales de todos los ficheros que conforman la aplicación, al igual que se hace una validación paralelamente de los requisitos y de las tareas anotadas en Wunderlist (herramienta para la anotación de tareas).

5 Integración, pruebas y resultados

A largo de esta sección se describirán las distintas pruebas y resultados arrojados por la aplicación, como también una breve mención de la integración de los ficheros citados con anterioridad.

5.1 Integración

Para la integración de toda la aplicación, debido a la metodología seguida, se ha realizado en tres etapas. Estas etapas vienen definidas por las pruebas realizadas en cada una de las fases de la aplicación, más las pruebas globales finales. De esta manera podemos encontrar:

- Primera integración: Esta integración se realiza con todos los ficheros codificados de la interfaz gráfica. A la par que se realizan las pruebas globales de la interfaz gráfica, estos ficheros se integran unos con otros para tener una interfaz gráfica completa, aunque sin todavía contener nada de funcionalidad.
- Segunda integración: Esta se realiza a la vez que las pruebas globales de la funcionalidad y lógica de la aplicación. Esta integración sólo compete a los ficheros que desempeñan la funcionalidad de la aplicación, ficheros PHP y JavaScript, no HTML y CSS. Además de estos ficheros, se integra la base de datos de la aplicación.
- Tercera integración: Esta compete a la última etapa de pruebas globales de toda la aplicación. Esta integración, realiza la función de comprobar y verificar que todo funciona correctamente, debido a los posibles cambios que han podido sufrir por separado los ficheros de interfaz gráfica con los de la lógica y funcionalidad.

De esta manera se integran de forma independiente la interfaz gráfica y la funcionalidad y la lógica. Esta separación se realiza para que, en la medida de lo posible, los módulos actúen como cajas negras y no tengan que depender de otros ficheros. Así, en la última etapa de la integración se une todo para verificar que todo es correcto, y si no, hacer los cambios necesarios.

5.2 Pruebas

Para las pruebas, según lo expuesto en la sección de metodología seguida, las pruebas se pueden definir en tres fases. La primera de ellas, empieza con una batería de pruebas unitarias de los ficheros de la interfaz gráfica, cada uno de estos será probado individualmente, una vez realizadas todas las pruebas unitarias se hace una prueba de integración de todos los ficheros que componen la interfaz gráfica.
Después de estas pruebas, se efectúan las pruebas unitarias relacionadas con la funcionalidad y la lógica. Después de pasar estas pruebas unitarias, se realizan unas pruebas globales para la parte funcional y lógica de la aplicación.

Por último, se hacen unas pruebas globales de todos los ficheros de la aplicación y de las funciones que se llevan a cabo. Estas pruebas vienen reflejadas en el anexo Pruebas de aplicación(E).

5.3 Resultados

Para los resultados, se ha optado por realizar una batería de pruebas con cien páginas, para que, de este modo, se pueda tener una visión un poco más extensa de los resultados arrojados por la aplicación de la capa SSL. Aun así, no son lo suficientemente grande, como para generalizar los resultados obtenidos, pero si son válidos, para poder hacer ver, cuáles de los ataques son los que más abundan.
En la siguiente tabla se muestran el número de IPs que tienen el ataque o la no vulnerabilidad:

Name of the exploits	Number of ips with	Total
beast	39	100
poodle tls	9	100
poodle ssl3	27	100
crime	0	100
freak	13	100
heartbleed	0	100
drown	4	100
openssl ccs	10	100
downgrade prevention	62	100
forward secrecy	90	100
rc4	45	100
secure renegotation	89	100

Tabla 5: Resultados de las pruebas

Como se puede observar, los exploits que dan seguridad (quitan la vulnerabilidad), aparecen en un 90% en el caso de la renegociación segura y de la seguridad perfecta y en un 60% para la prevención del ataque *downgrade*. Por otro lado, los ataques que más se han dado han sido RC4 y BEAST con un 45% y 39% respectivamente, mientras que *Heartbleed*, el ataque con mayor gravedad, ha conseguido un 0% de aparición.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

La seguridad en el ámbito de las redes y comunicaciones es vital. De forma continua, se establece nuevas formas de mantener y mejorar dicha seguridad. Debido al gran desconocimiento y/o desactualización de los servidores, se han creado en los últimos años herramientas y aplicaciones para el análisis de vulnerabilidades y otros parámetros de la seguridad.

A través de la aplicación realizada, se ha analizado la brecha de seguridad en la red mediante un análisis variado de un conjunto de cien páginas. Según los datos arrojados por la aplicación, se puede ver como algunos de los ataques se pueden producir en una gran cantidad de servidores, ya que en el caso de BEAST o RC4 tenemos un 40% de aparición. Además, muchos de estos servidores siguen usando cifrados que ya no son seguros e incluso llegan a soportar los protocolos SSL 3.0 y SSL 2.0, haciendo que estos servidores sean demasiado vulnerables ante ataques de cualquier índole.

La mejor forma de mantener la seguridad es usando los protocolos más nuevos, junto con los cifradores que no se consideren débiles para evitar así estas vulnerabilidades analizadas. A pesar de ello, nunca se está protegido totalmente, debido al descubrimiento de nuevas vulnerabilidades.

6.2 Trabajo futuro

Como trabajo futuro se querría aumentar la versatilidad de la aplicación, dando lugar a un soporte de exploits que acepte cualquier tipo de lenguaje, además, se desearía aumentar el rendimiento de las llamadas a las funciones de obtención de datos de la capa SSL, como también la inclusión de nuevos exploits y parámetros.

Otra posible mejora sería la de dotar de un servidor de correo a la aplicación, para la recuperación de contraseñas y facilitar la comunicación con el usuario.

Para un estudio más exhaustivo de la red, sería interesante hacer un análisis con un gran espacio muestral que superase las diez mil IPs.

Por último, se tiene en cuenta la posibilidad de crear un repositorio en GitHub, dando la opción de modificar o mejorar la aplicación a cualquier interesado. De esta forma, no solo la aplicación mejoraría en calidad, sino también en la inclusión de nuevos parámetros de seguridad y vulnerabilidades.

Referencias

[1] Stallings, William. “Network and internetwork security: principles and practice”. Vol. 1. Englewood Cliffs: Prentice Hall, 1995, pp 475.
[2] SSLlabs: https://www.ssllabs.com/
[3] COMODO SSL Analyzer: https://sslanalyzer.comodoca.com/
[4] Stephen A. Thomas. “SSL and TLS Essentials: Securing the Web with CD-ROM”, John Wiley & Sons, Inc., 2000, pp 224.
[5] “The Secure Sockets Layer (SSL) Protocol Version 3.0”, RFC-6101 August 2011: https://tools.ietf.org/html/rfc6101
[6] “The TLS Protocol Version 1.0”, RFC-2246 January 1999: https://tools.ietf.org/html/rfc2246
[7] “The Transport Layer Security (TLS) Protocol Version 1.1”, RFC-4346 April 2006: https://tools.ietf.org/html/rfc4346
[8] “The Transport Layer Security (TLS) Protocol Version 1.2”, RFC-5246 August 2008: https://tools.ietf.org/html/5246
[9] “Prohibiting Secure Sockets Layer (SSL) Version 2.0”, RFC-6176 March 2011: https://tools.ietf.org/html/6176
[10] Rizzo, J. and T. Duong, "Browser Exploit Against SSL/TLS", 2011: http://packetstormsecurity.com/files/105499/Browser-Exploit-Against-SSL-TLS.html
[11] Draft ietf tls 1.3: https://tools.ietf.org/html/draft-ietf-tls-tls13-13
[12] “Transport Layer Security (TLS) Renegotiation Indication Extension”, RFC-5746 February 2012: https://tools.ietf.org/html/rfc5746
[13] “TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks”, RFC-7507 April 2015: https://tools.ietf.org/html/rfc7507
[14] Rizzo, J. and T. Duong, "The CRIME Attack", EKOparty Security Conference, 2012: http://www.ekoparty.org/archive/2012/CRIME_ekoparty2012.pdf
[15] B. Möller, T. Duong, K. Kotowicz, “This POODLE Bites: Exlpoiting The SSL 3.0 Fallback”, September 2014: https://www.openssl.org/~bodo/ssl-poodle.pdf
[16] "Attacking SSL when using RC4: Breaking SSL with a 13-year-old RC4 Weakness", Imperva, 2015: http://www.imperva.com/docs/HII_Attacking_SSL_when_using_RC4.pdf
[17] Michael T. Raggo, Heartbleed and mobile, April 2014: https://www.mobileiron.com/en/smartwork-blog/heartbleed-and-mobile
[18] The Heartbleed Bug, 2014: http://heartbleed.com/
[19] The Drown Attack, 2016: https://drownattack.com/
[20] Tracking the FREAK Attack: https://freakattack.com/
[21] J. Albors, “Todo lo que debes de saber sobre FREAK Attack”, March 2016: http://www.welivesecurity.com/la-es/2015/03/06/debes-saber-freak-attack/
[22] CCS Injection Vulnerability, June 2014: http://ccsinjection.lepidum.co.jp/

Glosario

SSL	Secure Sockets Layer
TLS	Transport Sockets Layer
RFC	Request for Comments
CVE	Common Vulnerabilities and Exposures
BEAST	Browser Exploit Against SSL/TLS
POODLE	Padding Oracle On Downgraded Legacy Encryption
FREAK	Factoring RSA Export Keys
CRIME	Compression Ratio Info-leak Made Easy
DROWN	Decrypting RSA with Obsolete and Weakened eNcryption
PRF	Pseudorandom Function Family
JSON	JavaScript Object Notation
AJAX	Asynchronous JavaScript And XML
PHP	PHP Pre Hypertext
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
URL	Uniform Resource Locator
IP	Internet Protocol
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
TCP	Transmission Control Protocol
MAC	Message Authentication Code
IETF	Internet Engineering Task Force
IANA	Internet Assigned Numbers Authority
DES	Data Encryption Standard
AES	Advanced Encryption Standard
MD5	Message Digest Algorithm 5
SHA	Secure Hash Algorithm
IDEA	International Data Encryption Algorithm
EC	Eliptic Curve
PSK	Pre-Shared Key
AEAD	Authenticated Encryption with Associated Data
RC4	Rivest Cipher 4
SCSV	Signaling Cipher Suite Value
CBC	Cipher Block Chaining
IV	Initialization Vector
LSB	Least Significant Bit
SMTP	Simple Mail Transfer Protocol
IMAP	Internet Message Access Protocol
POP	Post Office Protocol
API	Application Programming Interface

Anexos

A Manual de instalación

Para poder hacer una instalación correcta en local de la aplicación, se han de tener los siguientes requisitos:

- Lo primero de todo, se recomienda el uso como sistema operativo de distribución Linux, ya que será más simple la instalación y manejo de las herramientas. Habrá que instalar apache(v2.2.22), PHP (v5.3.10), Python(v2.7.3) y alguna herramienta que permita levantar un servidor en local para hacer uso de la aplicación, como también una herramienta para la gestión de la base de datos.

- Además, se requiere de la instalación de la librería criptográfica OpenSSL Una vez realizado los procesos de instalación de los programas, se deberá de modificar el archivo de configuración de apache "ssl.conf" para que este permita todos los tipos de cifrado y todos los protocolos. También añadir los paquetes necesarios para ejecutar los exploits, estos son obtenidos viendo que librerías usan cada exploit.

- Para el correcto funcionamiento de la aplicación, habrá que crear la base de datos PostgreSQL con los siguientes parámetros:

  - Host: localhost
  - Dbname: SSL_ANALYZER
  - User: alumnodb
  - Password: alumnodb

Finalmente, la aplicación estará lista para ser usada en local.

B Navegación web

Para poder hacer más fácil el uso de la aplicación a los usuarios, se mostrará una navegación por la página. De esta manera, se irán mostrando capturas de pantalla mientras se van explicando las funcionalidades de la misma.

En primer lugar, la página principal (home). Esta página es la siguiente:

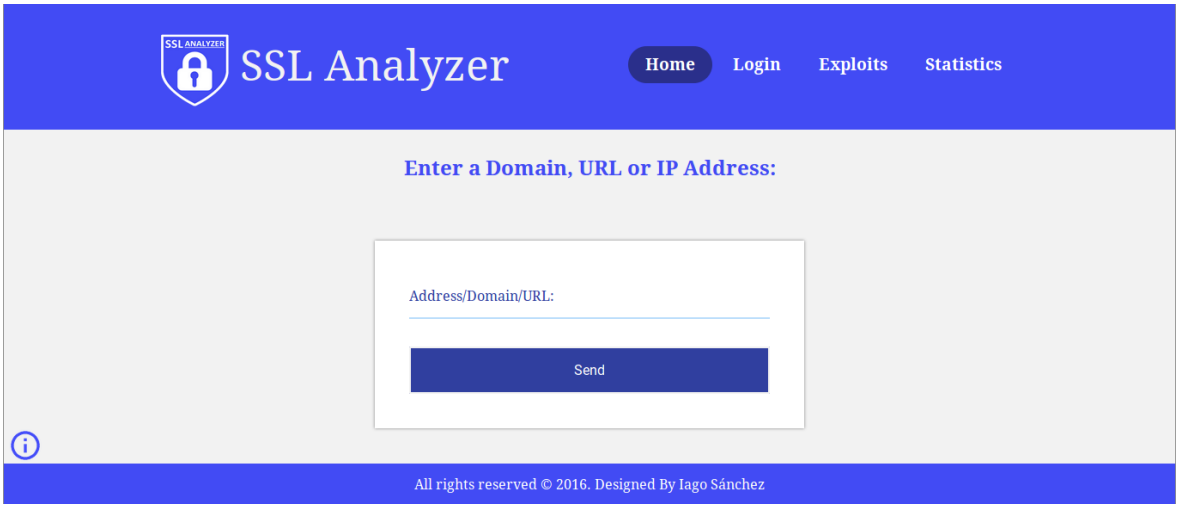


Figura 20: Página inicio

Al entrar en la aplicación, la primera página que se muestra es la home, desde esta, se puede navegar a los exploits, estadísticas y a la página de login. También se puede consultar la página de información sobre la aplicación. Como se puede comprobar, si aparece el botón login en la cabecera, significa, que no se está autenticado en la página. Entonces solo se podrá hacer un análisis cada 24 horas. Igualmente, no se podrá crear un exploit ni ver las estadísticas Para ello es necesario realizar el proceso de Log On. En la siguiente imagen se muestra la pantalla de correspondiente:

Figura 21: Pantalla Login

De esta forma, tras el log on, se podrá acceder a todas las funcionalidades que ofrece la página salvo las que están reservadas al administrador. Una vez autenticado, el usurario es redireccionado automáticamente a la home. Con el siguiente mensaje:

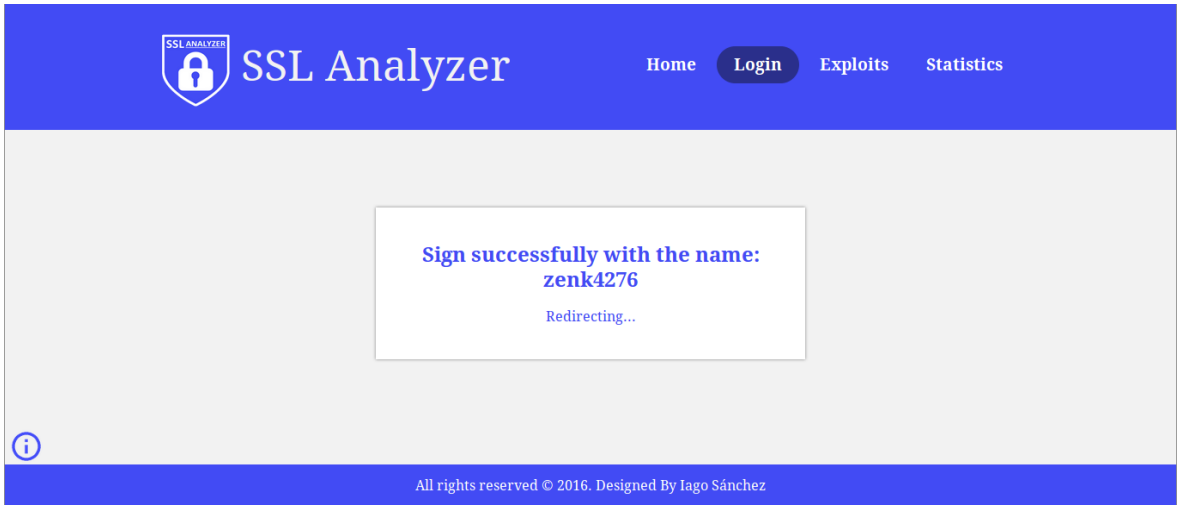


Figura 22: Redirección a la home

Una vez en la home, éste elimina el botón de login de la cabecera e incluye el botón de cerrar sesión junto con el nombre de usuario. Además, aparece un nuevo botón en la esquina inferior izquierda, el cual representa información relacionada con el usuario y la posibilidad de modificar la contraseña.

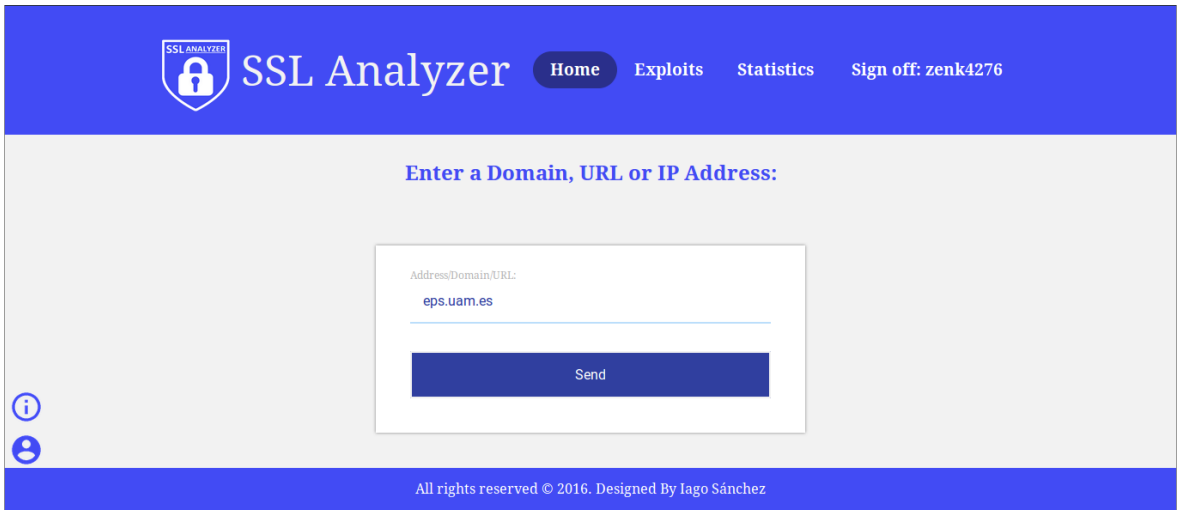


Figura 23: Página home (Usuario logueado)

Ahora se procederá a hacer un análisis de una página, para mostrar el funcionamiento. De este modo, se introduce en el formulario una IP, URL o dominio y se pulsa el botón "*Send*". En primer lugar, se verifica que el dato escrito en el formulario esté en línea y posteriormente se obtienen todas las IPs que apuntan a dicho dato. Mostrando

una tabla de IPs. En esta tabla se deberá seleccionar una dirección para ser analizada. La tabla es la siguiente: (Se usa como dato eps.uam.es)

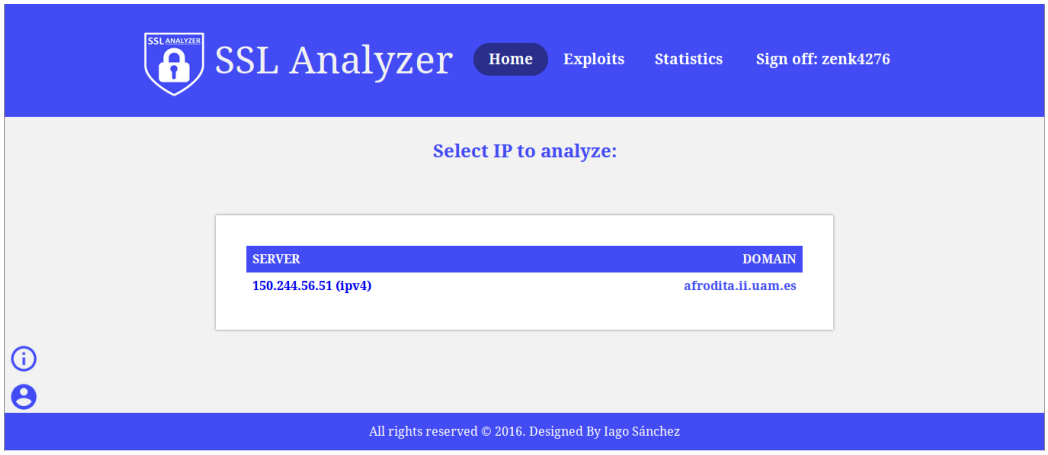
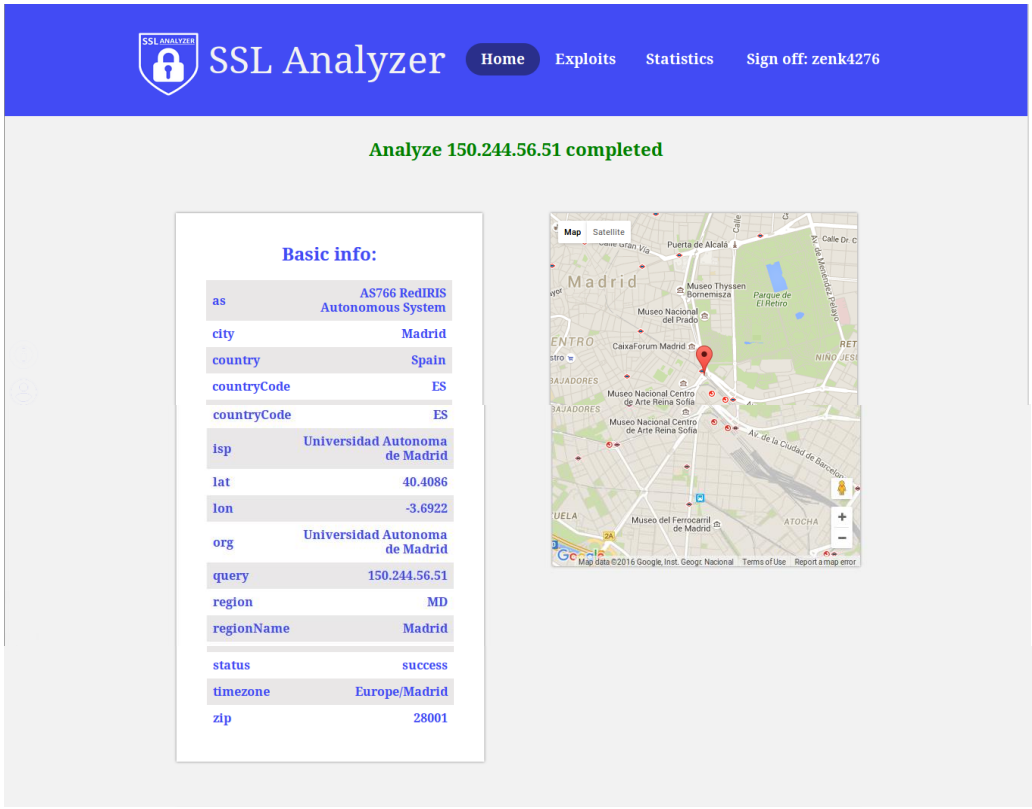


Figura 24: Tabla de IPs

En este caso, para la URL "eps.uam.es", solo se ha obtenido una IP.

Si se hace click sobre la IP, la aplicación comenzará con el análisis. Recordar que el análisis puede llevar tiempo, ya que tiene que ir probando uno por uno los cifradores, los protocolos y las vulnerabilidades. Por ello, y para hacer más sencillo la página, en la parte superior se muestra un título. Este se verá en azul y pondrá "Testing #IP" mientras se esté llevando a cabo el análisis. Cuando terminé el título se verá verde y pondrá "Analyze #IP completed". Se muestra el resultado del análisis:



SSL info:

Subject:	localhost.localdomain MISTMATCH Fingerprint: da04452b973f68d202729f51d244da7cfa35e998 PinSHA256: LwIjayVyQ5EWLgfTBqKP4c50IWQcJASVnCIXmRCzuXw=
Common Names:	localhost.localdomain
Alternative Names:	
Valid From:	Mon, 20 Nov 2000 12:59:21
Valid To:	Tue, 20 Nov 2001 12:59:21 (EXPIRED)
Key:	RSA 1024 (e 65537)
Public Key:	Click here
Issuer:	localhost.localdomain
Signature Algorithm:	md5WithRSA INSECURE
Extended Validation:	No
Revocation information:	
Revocation Status:	Revoked
Trusted:	No

Ciphers supported:

Name	Protocol	Key	Authentication	Encryption	Bits	MAC
DHE-RSA-AES256-SHA	SSLv3	DH	RSA	AES	256	SHA1
AES256-SHA	SSLv3	RSA	RSA	AES	256	SHA1
EDH-RSA-DES-CBC3-SHA	SSLv3	DH	RSA	3DES	168	SHA1
DES-CBC3-SHA	SSLv3	RSA	RSA	3DES	168	SHA1
DHE-RSA-AES128-SHA	SSLv3	DH	RSA	AES	128	SHA1
AES128-SHA	SSLv3	RSA	RSA	AES	128	SHA1
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES	40	SHA1
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES	40	SHA1
EXP-RC2-CBC-MD5	SSLv3	RSA(512)	RSA	RC2	40	MD5
EXP-RC4-MD5	SSLv3	RSA(512)	RSA	RC4	40	MD5
PSK-RC4-SHA	SSLv3	PSK	PSK	RC4	128	SHA1
EXP-RC4-MD5	SSLv3	RSA(512)	RSA	RC4	40	MD5
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES	40	SHA1
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES	40	SHA1
EXP-RC2-CBC-MD5	SSLv3	RSA(512)	RSA	RC2	40	MD5
EXP-RC4-MD5	SSLv3	RSA(512)	RSA	RC4	40	MD5
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES	40	SHA1
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES	40	SHA1
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES	40	SHA1
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES	40	SHA1
EXP-RC2-CBC-MD5	*SSLv3*	*RSA(512)*	*RSA*	*RC2*	*40*	*MD5*
EXP-RC4-MD5	SSLv3	RSA(512)	RSA	RC4	40	MD5

Protocols supported:

Name	Support
TLS 1.2	No
TLS 1.1	No
TLS 1.0	Yes
SSL 3	Yes
SSL 2	No

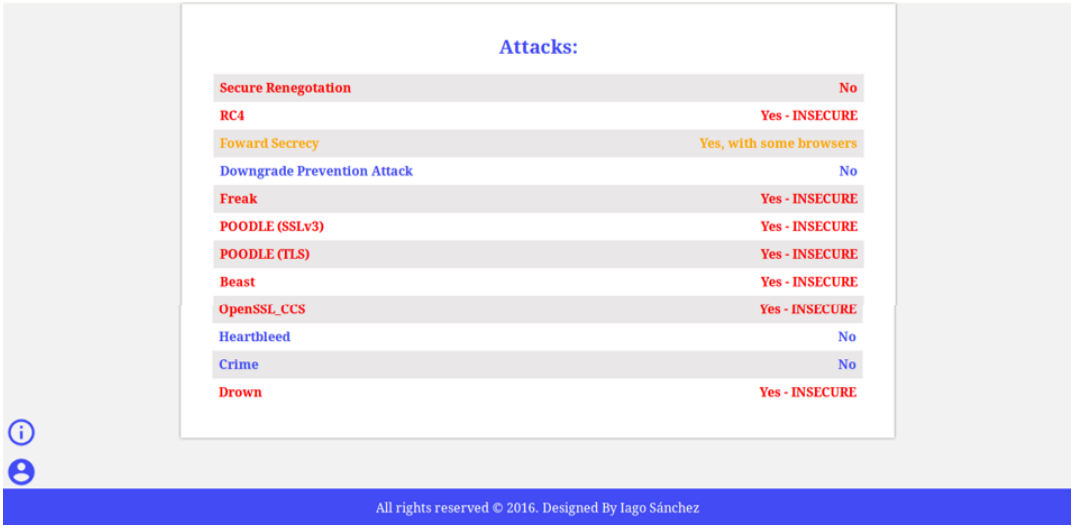


Figura 25: Análisis de eps.uam.es

Como se puede observar en la imagen anterior, una vez terminado el análisis, el título se muestra de color verde. Este análisis consiste en:

- Basic Info: Esta información muestra el lugar donde se aloja la página, en la parte izquierda muestra información relacionada con el país, ciudad, región, organización e ISP de la IP analizada. Y a la derecha muestra un mapa de Google Maps, señalando la localización del servidor.
- SSL Info: En esta tabla se muestra información relacionada con los certificados, nombre de la página y el estado de los certificados.
- Additional certificates: En este caso, esta IP solo tiene un certificado, si necesitase de más para validarse por parte del cliente, estos se mostrarían en este bloque.
- Ciphers Supported: Esta tabla muestra todos los cifradores soportados por la página, estos cifradores se muestran en verde si son seguros, rojos si son inseguros y naranjas si son un poco débiles.
- Protocols Supported: Se muestran los cinco protocolos de la capa SSL, estos se muestran en rojo si son inseguros, en verde si soporta TLS 1.2 y el resto en azul.
- Attacks: Por último, se hace la comprobación de que vulnerabilidades o prevenciones tiene. De este modo, se mostrarán en rojo las que no sean buenas para el servidor, mientras que en verde aparecerán las que sí. También pueden aparecer en naranja con el texto “Unknown”, esto informa de un error a la hora de analizar la vulnerabilidad.

Para la página de exploits se muestra una lista con los exploits subidos a la aplicación, como también la opción de subir uno. Al pinchar en uno de los exploits, la herramienta mostrará una descripción puesta por el creador del exploit (si existe) y un botón de descarga para el código del exploit.

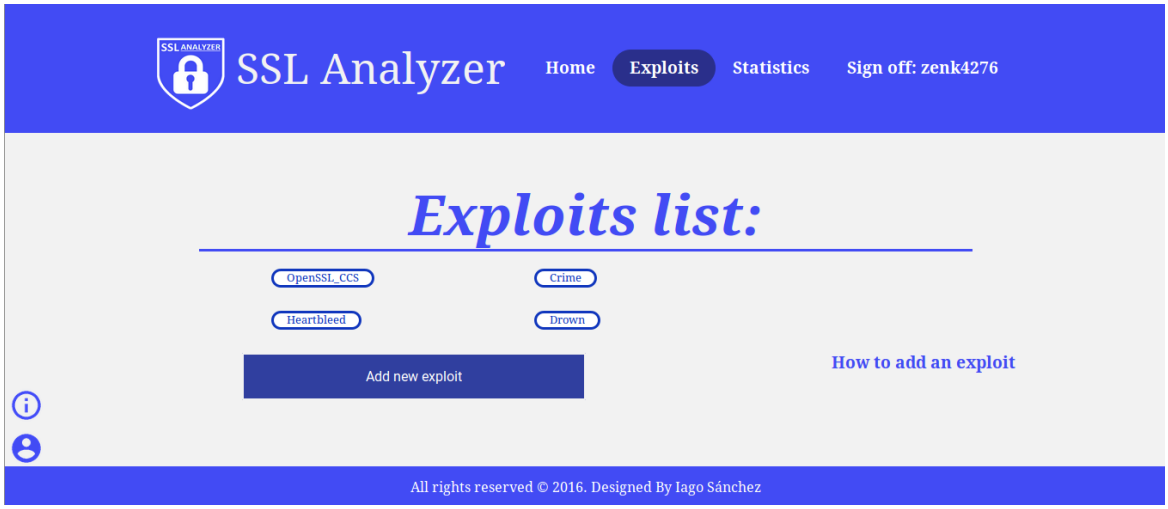


Figura 26: Página exploits

En esta prueba se ha pinchado en *Heartbleed*, la información cargada es:

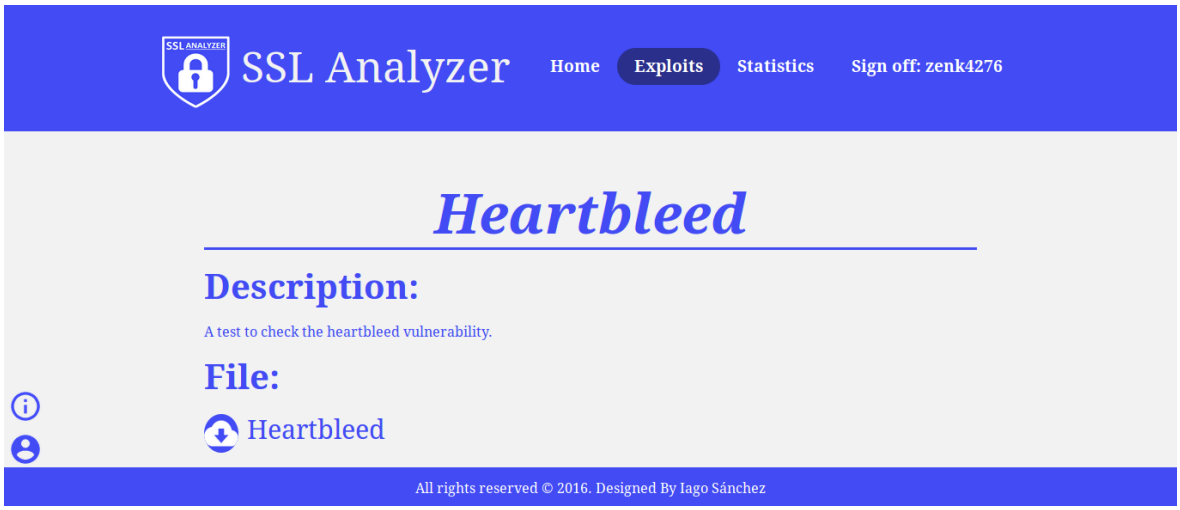


Figura 27: Información del exploit *Heartbleed*

Para la página de estadísticas, se muestran todos los exploits aportados por los usuarios como también aquellos propios de la página y un botón para mostrar todas las estadísticas de los parámetros listados justo encima. Esta página se muestra de la forma:

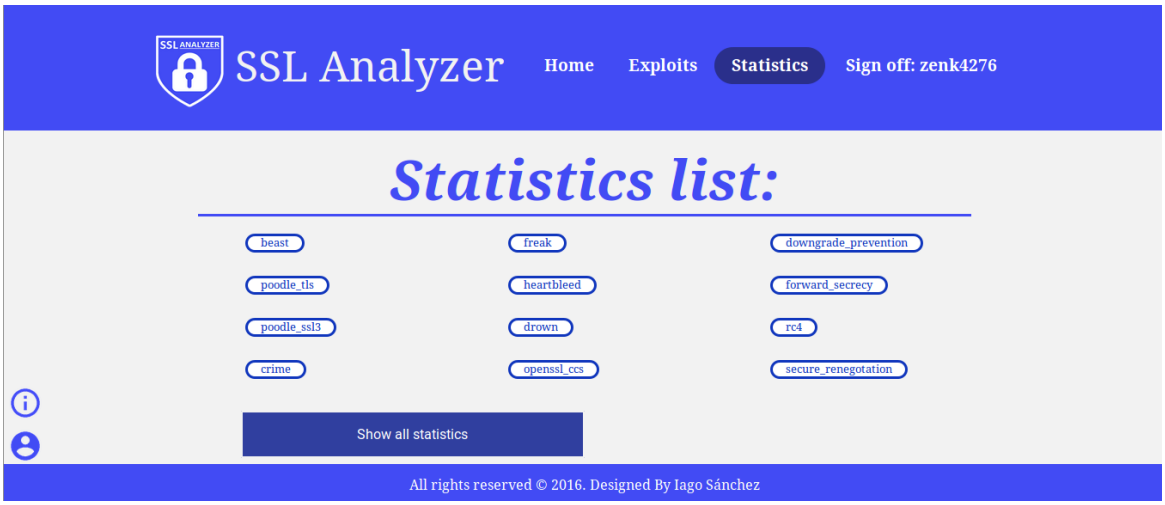


Figura 28: Página de estadísticas

Una vez pinchado en el botón de mostrar todas las estadísticas, la aplicación cargará tres secciones, una con los datos analizados la última semana, otra con los del último mes y por último las totales. En cada una de estas secciones se dividen por las vulnerabilidades que tienen y las que no. Quedando de esta forma:

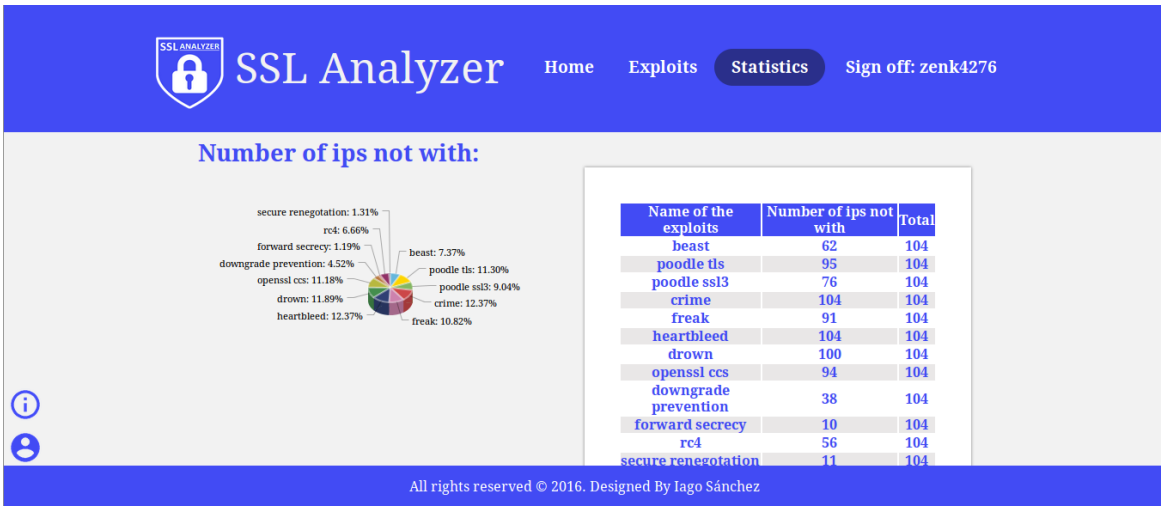


Figura 29: Todas las estadísticas

Para subir un exploit nuevo a la aplicación o modificar uno ya existente, el usuario debe de pinchar en el botón de añadir exploit, en el cual, se presentará un formulario a rellenar. Este, constará de un campo nombre, un campo descripción, un campo para el código del exploit y un checkbox que será usado para informar a la herramienta, si el exploit es para crear o para modificar. Este formulario es:

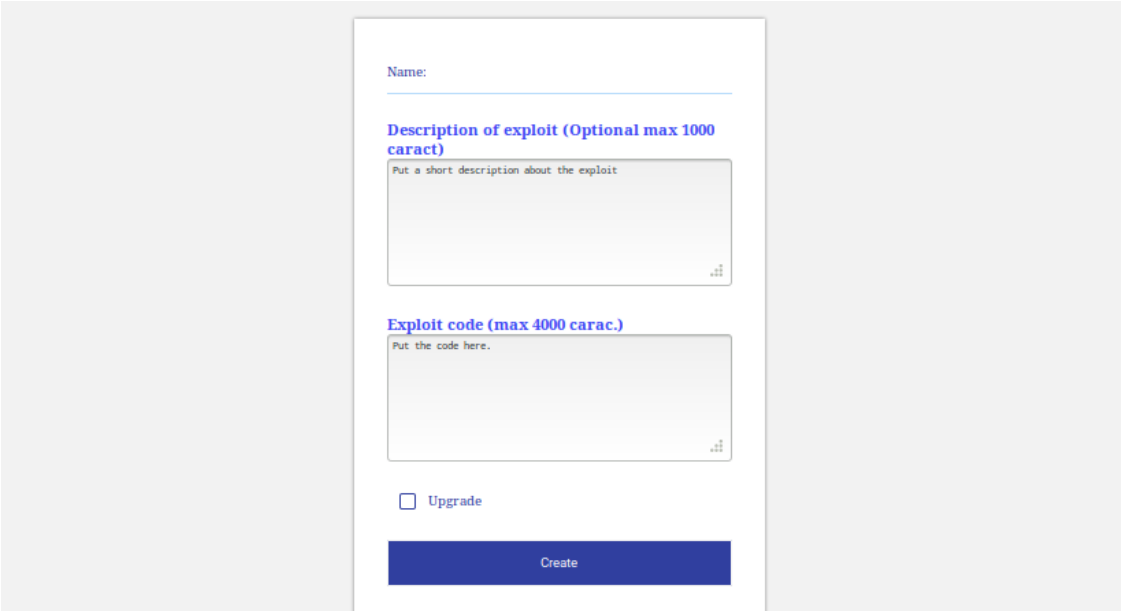


Figura 30: Formulario para añadir/modificar exploit

Por último, faltan las páginas de información de la herramienta y la de la información del usuario, esta última solo se mostrará si el usuario está logueado. Para la información de usuario se muestra los exploits que ha subido y la opción de modificar su contraseña, mientras que para la información se muestra un texto explicativo.

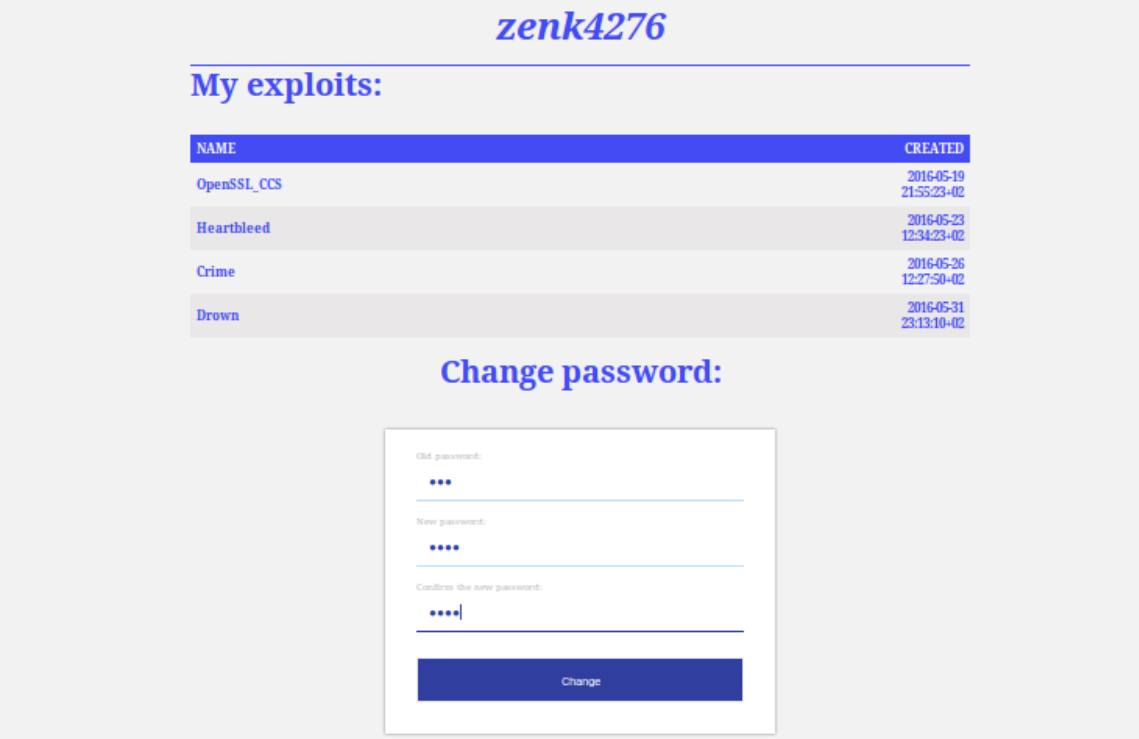


Figura 31: Página de información de usuario

En la imagen anterior se muestra los exploits creados junto con su fecha de creación, mientras que en la siguiente imagen se muestra el texto de información general de la aplicación.

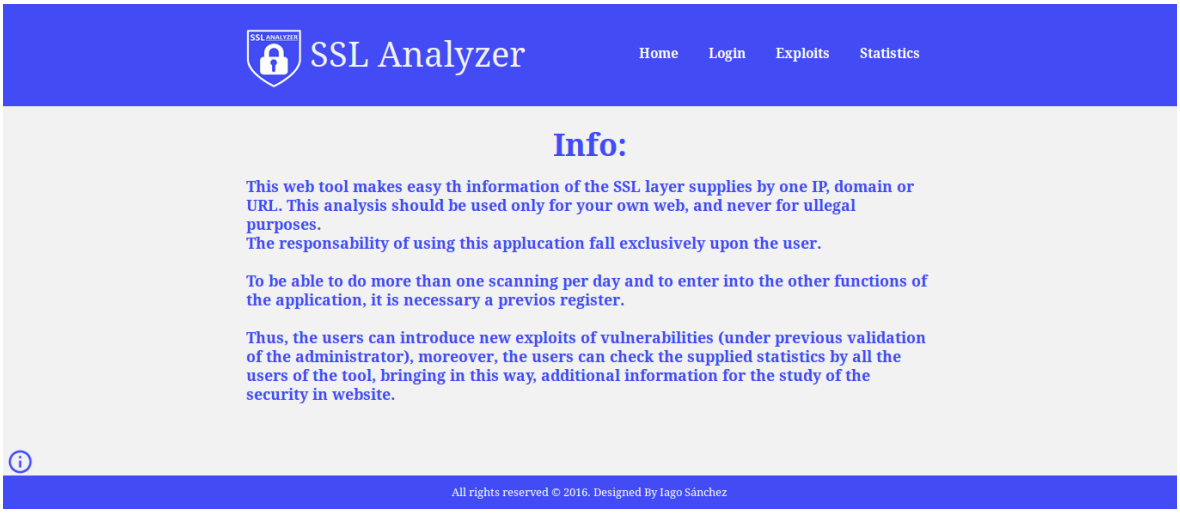


Figura 32: Pagina de información de la aplicación

C Protocolos de la capa SSL

- Record SSL

Se fragmentan los datos en bloques de 16384 bytes (2^{14} bytes) como máximo. Después se puede comprimir estos bloques por separado (opcional), aumentando un máximo de 1024 bytes por bloque. Posteriormente, se cifran los bloques con el código de autenticación (MAC) con la clave compartida de la conexión y posteriormente se cifra con una clave simétrica[1].

Por último, se muestra una figura que ilustra el protocolo de Record de la capa SSL, descrito anteriormente.

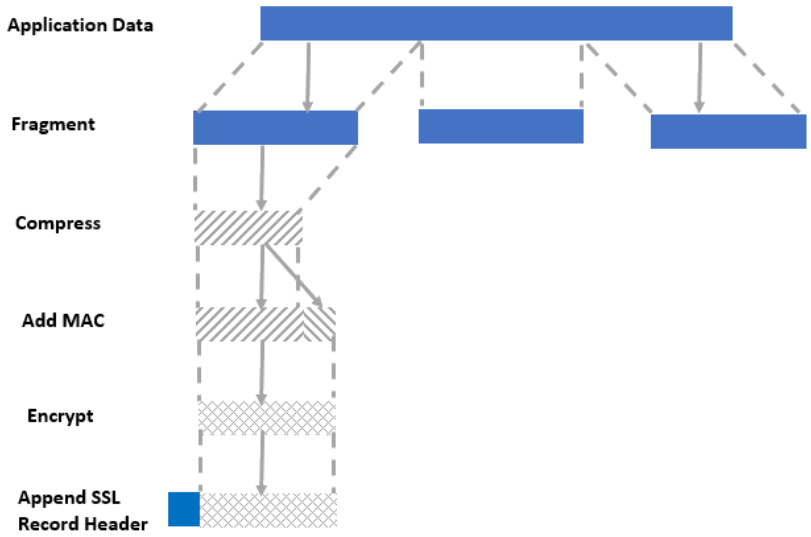


Figura 33: Protocolo Record

- Change Cipher Spec

Este es uno de los tres protocolos específicos de la capa. Es el más simple de los tres. Envía solo un byte para actualizar la suite de cifrado de la conexión.

- Alert

Este protocolo, se emplea para dar a conocer al otro extremo de la comunicación las alertas referentes a la capa SSL. Los mensajes de este protocolo se componen de tan solo 2 bytes de longitud. El primer byte únicamente toma dos valores. El valor 1 en el caso en el que la alerta sea un aviso, o el valor 2 en el caso en que la alerta sea fatal.

Si el aviso es fatal, la conexión por la cual se ha enviado el aviso se cierra. Y aunque el resto conexiones de la misma sesión se mantienen, ya no se pueden crear nuevas

conexiones en esa sesión. El segundo byte, contiene la alerta específica, es decir, informa al receptor del estado de la conexión.

- ## Handshake

Este es el protocolo más complejo de la capa SSL, es el encargado de permitir la autenticación tanto del cliente como del servidor, de establecer el algoritmo de cifrado, el cálculo del MAC y las claves criptográficas.

Handshake envía una serie de mensajes entre el cliente y el servidor. Todos ellos son enviados a través de este protocolo tienen la siguiente arquitectura, tienen los siguientes tres campos:

- Tipo (1 byte): Indica uno de los 10 mensajes mostrados en la siguiente tabla. Longitud (3 bytes): Indica la longitud del mensaje en bytes.

- Contenido (Mayor a un byte): Son los parámetros al mensaje.

Tipo de mensaje	Parámetros
hello_request	Null
client_hello	Versión, valor aleatorio, id de sesión, suite de cifrado método de compresión
server_hello	Versión, valor aleatorio, id de sesión, suite de cifrado método de compresión
certificate	Cadena de certificados X.509v3
server_key_exchange	Parámetros, firme
certificate_request	Tipo, autoridades
server_done	Null
certificate_verify	Firma
client_key_exchange	Parámetros, firma
finished	Valor hash

Tabla 6: Mensajes Handshake

El protocolo *Handshake* realiza una serie de intercambios de mensajes para establecer una conexión segura. De esta forma, se establece que el protocolo se puede dividir en cuatro fases.

- **Fase 1: Establecimiento de las capacidades de seguridad**

Esta fase es iniciada con una conexión, en la cual el cliente empieza enviando un mensaje *client_hello*. Éste especifica la versión SSL usada, un valor aleatorio para el intercambio de claves, un identificador de sesión. La suite de cifrado a usar y el método de compresión. Posteriormente al envío del mensaje *client_hello* por parte del cliente, es el servidor quien a continuación envía un mensaje *server_hello*. Para el envío de éste mensaje se usan los mismos parámetros que para *client_hello*, pero con los valores de los parámetros por defecto del servidor.

- **Fase 2: Autenticación del servidor e intercambio de la clave**

Una vez establecidos los parámetros de seguridad entre ambas partes, es el servidor el cual envía su certificado. De ser necesaria la autenticación, hará uso de los certificados de la entidad autentificadora, mandando así más de un certificado (*chain certs*). Además de esto, se puede enviar un mensaje del tipo *server_key_exchange*, este mensaje se usa para poder cambiar la clave del servidor.

El último de los mensajes de esta fase, es el único que de verdad es obligatorio, puesto que el mensaje de certificado (*certifícate*) a veces no es usado. Este mensaje es el *server_done*, que es enviado al final del mensaje *server_hello*. Después del envío de este mensaje, el servidor espera a recibir la respuesta de cliente.

- **Fase 3: Autenticación del cliente e intercambio de la clave**

Después de que el cliente reciba el mensaje *server_done* por parte del servidor, el cliente verifica los parámetros del *server_hello*, si son aptos y si también lo es el certificado aportado. Si en la fase 2, el servidor envió un mensaje de certificado (*certificate*), el cliente debe enviar un certificado apto para el servidor, si no es así, el cliente envía una alerta (*no_certificate*). A continuación, el cliente debe de responder al mensaje *server_key_exchange*, el cual establece los parámetros de la clave a usar.

Al final de la fase, el cliente puede enviar un mensaje *certificate_verify*, mediante el cual el cliente proporciona la verificación de un certificado.

- **Fase 4: Final**

Esta es la última fase del protocolo *Handshake*, que deja la conexión segura entre los dos pares. El cliente manda un mensaje *change_cipher_spec* para notificar que los mensajes posteriores serán protegidos por la suite elegida. A continuación de este mensaje, el cliente envía un mensaje *finished*. Este mensaje se envía para poder verificar que los intercambios de las claves y el resto de mensajes fueron correctos y no hubo alertas.

Por parte del servidor, una vez que ha recibido estos dos mensajes, él también envía un mensaje *change_cipher_spec*. También el servidor envía un mensaje *finished*. Después del envío de estos dos mensajes el protocolo *Handshake* ha terminado y ya pueden enviar mensajes del nivel de la capa aplicación[3].

D Gravedad de los ataques en la capa SSL

Para diferenciar los ataques vistos en la capa SSL, es necesario llevar un tipo de clasificación que permita conocer la gravedad de cada uno. Este criterio viene dado por un conjunto de parámetros que se tienen en cuenta, son los siguientes:

- Protocolos a los que afecta: Es uno de los parámetros más importantes a la hora de clasificar los ataques. No es lo mismo un ataque que afecte tan sólo a una versión del protocolo TLS o SSL, que si el ataque afecta a todas las versiones de ambos protocolos.

- Número máquinas afectadas: Este parámetro está bastante relacionado con el anterior, puesto que cuantas más versiones de los protocolos sean afectadas por el ataque, más máquinas estarán en riesgo. Pero también, es para hacer una diferenciación entre versiones de los protocolos, es decir, no es lo mismo que un ataque afecte al protocolo SSL 2.0, el cual está en desuso y no muchas máquinas lo soportan, que el que afecta a SSL 3.0 o TLS 1.0, los cuales son de los más usados hoy en día.

- Integridad de los datos: Este parámetro tiene en cuenta que tipo y cantidad de información se puede obtener con el ataque. Esto significa, que tiene en cuenta si el ataque es capaz de obtener el texto plano de los mensajes cifrados, o si solo es capaz de obtener la clave de la firma para poder suplantar a la persona. Igual pasa con el tamaño, tiene en cuenta si el ataque es capaz de obtener por ejemplo la información de un solo cliente, o de si es capaz de obtener todos los datos del servidor y todos los clientes que están conectados a ese servidor.

- Duración del ataque: Puesto que estamos hablando de comunicaciones por la red, los mensajes a través de ella suelen ser casi “instantáneos”. Debido a esto, este parámetro tiene en cuenta lo que tarda el ataque desde que se inicia, hasta que se consigue perpetrar dicho ataque. Como consecuencia de la velocidad de transmisión por internet, es favorable para un atacante que los ataques se realicen en cuestión de segundos, y no de horas o días.

- Dificultad de la realización del ataque: Este es el último de los parámetros para la clasificación, este parámetro tiene en cuenta si es necesario alguna acción previa o posterior al ataque, como también la complejidad para llevarlo a cabo. Es decir, diferencia entre los ataques que por ejemplo no necesitan ningún requisito previo para realizarse, o aquellos que es necesario usar físicamente el dispositivo de la víctima de los que tan sólo necesitan realizar el ataque.

E Pruebas de la aplicación

- Mensajes de error: Estos suelen aparecer en rojo en la aplicación, suelen originarse porque un campo es incorrecto o porque no se ha podido establecer conexión, estos son algunos ejemplos:

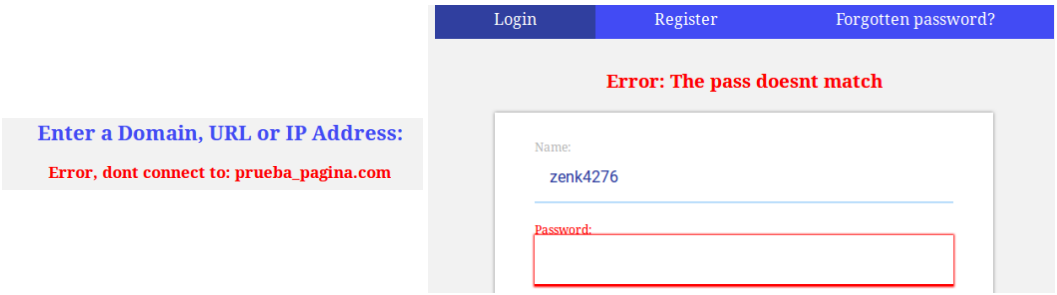


Figura 34: Mensajes de error

- Codificación BD: Se muestran en la propia página algunos campos de la base de datos.

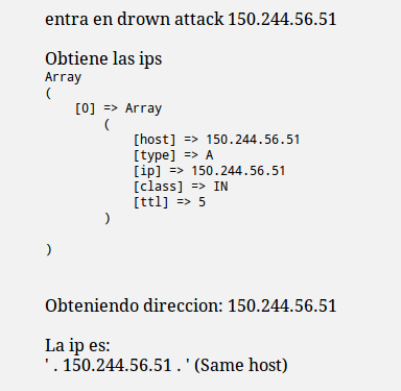
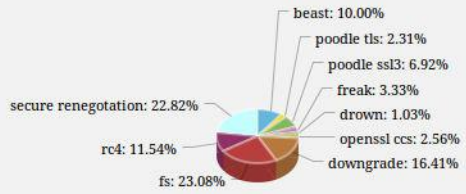


Figura 35: Traza

- Amchart: Se han creado ficheros de prueba para probar los gráficos.



Name of the exploits	Number of ips vulnerables	Total
beast	39	102
poodle tls	9	102
poodle ssl3	27	102
crime	0	102
freak	13	102
heartbleed	0	102
drown	4	102
openssl ccs	10	102
downgrade	64	102
fs	90	102
rc4	45	102
secure renegotation	89	102

Figura 36: Prueba amchart

- SSL: Por último, se han hecho pruebas para comprobar los diferentes parámetros mostrados al analizar una página.

SSL info:

Subject:	localhost.localdomain MISTMATCH Fingerprint: da04452b973f68d202729f51d244da7cfa35e998 PinSHA256: LwIjayVyQ5EWLgiTBqKP4c50lWQcJASVnClXmRCzuXw=
Common Names:	localhost.localdomain
Alternative Names:	
Valid From:	Mon, 20 Nov 2000 12:59:21
Valid To:	Tue, 20 Nov 2001 12:59:21 (EXPIRED)
Key:	RSA 1024 (e 65537)
Public Key:	Click here
Issuer:	localhost.localdomain
Signature Algorithm:	md5WithRSA INSECURE
Extended Validation:	No

Figura 37: Prueba parámetros SSL