Determinación de actividades y vulnerabilidades SSL/TLS y de encriptación

Determining SSL/TLS and encryption activities and vulnerabilities

Carlos Ramón Polanía Tovar[1]

Resumen

Nunca la historia en la evolución de la humanidad había registrado el uso masivo a escala global de un conjunto de redes de comunicación interconectadas que permiten el intercambio libre de información, internet. Esta alcanza vital importancia para la economía, la educación, los negocios, la recreación y casi todos los demás aspectos de la sociedad, y se convierte en una herramienta insustituible para el trabajo la obtención y generación de la información necesaria. El protocolo Secure Sockets Layer (SSL) tiene como finalidad en la capa de aplicación mantener la integridad, confidencialidad y autenticidad de la información que se maneja en un ambiente amplio de computación de forma que los usuarios tengan disponibles todos los componentes del sistema cuando así lo deseen y de forma segura. Por su parte, el protocolo Transport Layer Security (TLS), es el estándar para cifrar las comunicaciones entre el cliente y el servidor. Hasta ahora se han descubierto muchas vulnerabilidades y debilidades en estos protocolos de internet.

Este artículo aporta en la divulgación de algunos de los elementos principales asociados a los graves riesgos que enfrenta nuestra sociedad, dependiente de sistemas informáticos interconectados, expuestos a constantes ataques y que no siempre se encuentran protegidos de manera adecuada.

Se describen vulnerabilidades en los protocolos SSL/TLS, que son esenciales para la seguridad en las comunicaciones en línea. También se establecen hipótesis y variables y se concluye con hallazgos y recomendaciones clave para mitigar estas

1 Estudiante de la Fundación Universitaria del Área Andina.

vulnerabilidades. Se analizarán los recientes ataques contra la implementación del protocolo Handshake, así como los ataques contra los demás elementos necesarios al protocolo SSL/TLS, para descubrir las fallas de seguridad explotadas, los modos de ataque y las posibles consecuencias, pero también se estudiarán los métodos de defensa.

Por medio de una investigación documental se llevará a cabo el análisis por medio de las siguientes fases: análisis, diseño, implementación, recolección de datos, análisis de datos y reporte de resultados.

Palabras clave: criptografía, redes, seguridad, SSL/TLS, protocolo.

Abstract

Never before in the history of human evolution has the massive global use of a set of interconnected communication networks, enabling the free exchange of information, been recorded: the Internet. This network is vitally important for the economy, education, business, recreation, and almost every other aspect of society, becoming and continuing to become an irreplaceable tool for obtaining and generating necessary information. The Secure Sockets Layer (SSL) protocol, at the application layer, is intended to maintain the integrity, confidentiality, and authenticity of the information handled in a broad computing environment so that users have access to all system components securely and whenever they want. The Transport Layer Security (TLS) protocol, for its part, is the standard for encrypting communications between the client and the server. To date, many vulnerabilities and weaknesses have been discovered in these Internet protocols.

This article contributes to the dissemination of some of the main elements associated with the serious risks facing our society, which depends on interconnected computer systems, exposed to constant attacks, and not always adequately protected.

Vulnerabilities in the SSL/TLS protocols, which are essential for online communication security, are described. Hypotheses and variables are also established, and the article concludes with key findings and recommendations for mitigating these vulnerabilities. Recent attacks against the implementation of the Handshake protocol, as well as attacks against other elements necessary for the SSL/TLS protocol, will be analyzed to uncover exploited security flaws, attack modes, and potential consequences. Defense methods will also be studied.

Through desk-based research, the analysis will be conducted in the following phases: analysis, design, implementation, data collection, data analysis, and reporting of results.

Keywords: cryptography, networks, security, SSL/TLS, protocol.

Introducción

Todos los dispositivos conectados a internet dependen de los protocolos SSL y TLS para proteger la información en tránsito (Ristic, 2013). Cuando se diseñó internet se prestó poca atención a la seguridad, y como resultado, los protocolos de comunicación centrales son intrínsecamente inseguros y dependen del comportamiento honesto de todas las partes involucradas. La comunicación a través de diversos servicios que se ejecutan en internet tiene gran importancia en la vida habitual de las personas en todo el mundo. A finales de 2017, el Departamento Administrativo Nacional de Estadística (DANE) de Colombia reportó que el 50% de los hogares en el país contaban con conexión a internet, con un aumento respecto al año anterior (DANE, 2018).

El protocolo Secure Sockets Layer (SSL) y su sucesor, el protocolo Transport Layer Security (TLS), se utilizan para proteger la mayoría de las conexiones web actuales, desde HTTPS para sitios web a SMTP para correos electrónicos. Infortunadamente, la confianza en la seguridad de las implementaciones SSL/TLS predeterminadas está en entredicho (Krawczyk, Paterson y Wee, 2013).

Un componente clave de la seguridad de SSL/TLS es la fuerza criptográfica de los algoritmos subyacentes utilizado por el protocolo (Lee et al, 2007) . Es clave garantizar que los servidores que utilizan el protocolo SSL lo han empleado correctamente. El mal empleo de la criptografía puede ser un indicio de una seguridad mal administrada.

SSL 1.0 fue introducido por primera vez por Netscape Communications en 1994, seguido de las versiones 2.0 en 1995 y 3.0 en 1996 (Thomas, 2000). El protocolo se desarrolló aún más dentro del IETF, en el que su protocolo sucesor pasó a llamarse TLS (Dierks y Allen, 1999). Desde entonces, aparecen periódicamente nuevas versiones, con TLS 1.1 en 2006 (Dierks y Rescorla, 2006), TLS 1.2 en 2008 (Dierks y Rescorla, 2008) y TLS 1.3 en 2018, aún en desarrollo (Rescorla,

2018). El desarrollo de nuevas versiones estuvo inicialmente promovido por la introducción de nuevas características; sin embargo, la versión TLS 1.3 ahora introduce nuevos mecanismos de seguridad y protección contra ataques descubiertos recientemente (Sheffer, Holz y Saint-Andre, 2015a) y la discontinuación de algoritmos y tamaños de claves inseguros (Sheffer, Holz y Saint-Andre, 2015b).

En el proceso de desarrollo de una aplicación web, elegir una óptima configuración y probarla continuamente para detectar vulnerabilidades está fuera del alcance y experiencia de la mayoría de los implementadores. Cuando se trata de protocolos SSL/TLS, esto se incrementa por los diferentes niveles de soporte para versiones de protocolos y conjuntos de cifrado entre dispositivos cliente y servidores (Rescorla, 2018). Algunos navegadores modernos también han comenzado a agregar soporte para la versión de TLS v1.3 (Eldewahi, 2015).

El protocolo SSL/TLS

SSL es un protocolo estándar para la comunicación segura entre un servidor web y un navegador web, protegiendo toda la información que se transfiere para que conserve su integridad y sea confidencial.

El protocolo SSL/TLS se puede utilizar como capa de seguridad implícita o explícita. Su mayor implementación en la industria se utiliza para preservar la comunicación por medio de HTTPS, correos electrónicos e incluso para redes privadas virtuales (VPN) y autenticación WiFi (Ristic, 2013). El cliente que se conecta a un servidor mediante HTTPS abrirá una conexión TCP y establecerá una sesión TLS directamente.

Además, el protocolo SSL/TLS se emplea explícitamente en el servicio de correo electrónico SMTP (Rescorla, 2018; Sheffer Holz y Saint-Andre, 2015b). Un cliente envía mensajes de aplicación SMTP, y si es necesario emite un comando STARTTLS para proponer el uso de TLS para intercambios posteriores en la capa de aplicación.

La capa SSL/TLS es flexible porque se puede configurar para utilizar diferentes algoritmos criptográficos y admitir subconjuntos de versiones y extensiones de protocolo.

El protocolo SSL/TLS consta de cuatro subprotocolos y también tiene extensiones a partir de TLSv1 (Sheffer Holz y Saint-Andre, 2015b; Hopwood, 2018). Las dos extensiones principales son *tickets* de sesión y algoritmos de firma. Los subprotocolos se describen a continuación:

1. *Handshake protocol* (Protocolo de enlace o apretón de manos)

La capa de protocolo de enlace realiza el establecimiento de sesión y la negociación de opciones determinando las claves simétricas por sesión que la capa de registro utiliza de forma masiva (Thomas, 2000).

Implica la negociación de sesiones, versiones del protocolo SSL/TLS, conjuntos de cifrado, intercambios de cadenas de certificados y cualquier autenticación necesaria. Durante la fase de protocolo de enlace (*handshake*) se establecen todas las primitivas criptográficas (básicas) responsables de la protección de la conexión. La comunicación entre el cliente y el servidor durante la fase de *handshake* se realiza con el mensaje con formas predefinidas:

1. El cliente SSL/TLS envía el mensaje "Cliente Hola", que incluye información sobre la versión de TLS/SSL y los conjuntos de cifrado admitidos por el cliente.
2. El servidor SSL/TLS responde con un "Hola del servidor", que tiene el conjunto de cifrado seleccionado por el servidor. También contiene ID de sesión, además de que el servidor envía su certificado digital al cliente.
3. El cliente SSL/TLS verifica el certificado digital de los servidores mediante la clave pública del servidor.
4. Luego, el cliente SSL/TLS enviará un *byte* generado aleatoriamente que se utilizará para encontrar la clave secreta que se utilizará para cifrar las próximas conversaciones. Este *byte* generado aleatoriamente se cifrará con la clave pública del servidor y este podrá descifrarlo utilizando su clave privada.
5. Si el servidor SSL/TLS solicitó al cliente una solicitud de certificado. El paso 4 se realizará además del certificado digital de los clientes que se enviará al servidor. Y en esta etapa el servidor puede verificar la identidad del cliente.
6. El servidor SSL/TLS verificará el certificado del cliente.
7. El cliente SSL/TLS envía "cliente finalizado", que está cifrado con la clave secreta generada antes. Esto será una indicación para el servidor de que el cliente terminó su parte en el apretón de manos.

8. El servidor SSL/TLS envía el mensaje "servidor finalizado" para indicar que el protocolo de enlace del servidor también ha finalizado.

A lo largo de la sesión que se inició una vez iniciado el protocolo de enlace, tanto el servidor como el cliente pudieron enviar y recibir mensajes que utilizan un mecanismo de cifrado simétrico.

2. *Record*

En esta capa, los mensajes tienen un encabezado y un valor *hash*. La carga útil se puede comprimir y los mensajes pueden encapsular la verificación de la integridad y longitud de la carga útil, así como los parámetros del protocolo.

3. *Cipher Spec*

Este es el comienzo de la comunicación segura, ya que el conjunto criptográfico acordado por la pareja servidor/cliente se utiliza para cifrar todos los mensajes.

4. *Alert*

Cualquier error o advertencia que se produzca se informa aquí en forma de descripción de alerta y nivel de gravedad.

Intercambio de claves y autenticación

SSL utiliza dos protocolos comunes para el intercambio de claves y dos protocolos comunes para la autenticación, utilizados en las siguientes tres combinaciones. Una es utilizar el algoritmo de intercambio de claves Ephemeral Diffie Hellman (EDH) con Digital Estándar de firma (DSS) para autenticación; otra es utilizar el algoritmo de intercambio de claves EDH con RSA para la autenticación y la que se emplea con más frecuencia es RSA para ambos intercambios de claves y autenticación (Thomas, 2000). Estos algoritmos se explican en la siguiente sección.

Modos de codificación de mensajes

Una vez que tanto el cliente como el servidor tienen acceso a la clave compartida, los mensajes entre los dos se cifran con la clave en uno de varios modos de cifrado de bloques posibles. Estos modos permiten cifrado de datos de cualquier longitud mediante cifrados en bloque.

Ejemplos de modos de cifrado de bloques son Electronic Codebook (ECB), Cipher Block Chaining Mode (CBC) o modo de encadenamiento de

bloques de cifrado (CBC) y el modo Galois/Counter Mode (GCM) (Ristic, 2013). . GCM es el más seguro de los tres y está ganando una base de usuarios más grande. GCM también logra proporcionar confidencialidad y controles de integridad de los datos, pero solo es compatible con TLSv1.2 y TLSv1.3.

"Ciphersuites" (Conjuntos de cifrado) y configuraciones

En conjunto, los algoritmos utilizados para una conexión se denominan *ciphersuite* (conjunto de cifrado). El conjunto utilizado para la comunicación entre un par servidor/cliente se determina dinámicamente cuando la conexión se inicia. La elección de un conjunto de cifrado para su uso en el protocolo SSL/TLS especifica el método de autenticación, el intercambio de claves, la función *hash* y la MAC algoritmo, entre otros. Los protocolos SSL/TLS generalmente pueden utilizar RSA, EC, AES-CBC, AESGCM, RC4, 3DES, MD5, SHA1, MAC, *Signatures* (firmas) y PRF como *cryptographic primitives* (criptografías primarias).

Existen varios esquemas de nombres para conjuntos de cifrado, *ciphersuite*, la internet. Assigned Numbers Authority (IANA) es la entidad que supervisa la asignación global de direcciones IP, sistemas autónomos, servidores raíz de nombres de dominio DNS y otros recursos relativos a los protocolos de internet

Para Windows 11 v22H2, la *cipher suites* están habilitados y en este orden de prioridad de forma predeterminada utilizando el proveedor Microsoft Schannel.

En versiones anteriores de Windows, los conjuntos de cifrado TLS y las curvas elípticas se configuraban mediante una sola cadena.

Firmas digitales

Se utilizan para verificar el remitente de un mensaje. Un ejemplo de firma digital en el contexto de SSL/TLS son los códigos de autenticación de mensajes (MAC), que combinan *hash* con autenticación para verificar la integridad de los datos. Los datos cifrados se envían junto con el *hash*, de modo que si un atacante modifica el contenido del mensaje de datos, el *hash* revela la manipulación.

Criptografía

Es la ciencia y el arte de la comunicación segura. Aunque asociamos el cifrado con la era moderna, en realidad hemos utilizado la criptografía durante

miles de años (Fernández, 2004). Los primeros usos se centraron en el empleo de métodos secretos, ya sea en herramientas como *scytales*, o algoritmos como la sustitución de letras o patrones, para codificar mensajes importantes a través de fronteras o mientras son transmitidos por agentes no confiables. En contraste, la criptografía moderna se fundamenta en claves secretas y utilizan algoritmos de cifrados públicos o personalizados diseñados con el supuesto de que los atacantes podrían conocer los algoritmos.

Técnicas de criptografía

La criptografía se divide en tres técnicas principales:

Criptografía de clave secreta (SKC)

Solo utiliza una única clave para cifrar y descifrar datos, también conocida como "cifrado simétrico" (AlFardan *et al.*, 2013). Cuando el remitente de los datos envía la información, los cifra con la misma clave que el destinatario utilizará para descifrar la información.

Cuando se trata del protocolo SSL/TLS, el cifrado simétrico se utiliza en forma de cifrados de flujo y cifrados de bloque para codificar los mensajes después del protocolo de enlace entre el cliente y el servidor.

El inconveniente de esta técnica es que la distribución de una única clave puede caer en manos de un atacante, que puede descifrar la información fácilmente.

Criptografía de clave pública (PKC)

A diferencia de SKC, la criptografía de clave pública utiliza un par de claves digitales. El sistema de dos claves permite a los usuarios comunicarse de forma más segura en la red pública. En este, cada usuario tiene un par de claves; una de las claves es una "clave privada", mientras que la segunda es una "clave pública", la cual se puede compartir entre los usuarios (AlFardan *et al*., 2013). Al enviar información, la remitente cifra la información utilizando la clave pública y el destinatario descifra la información utilizando su clave privada en un formato legible.

La criptografía de clave pública (PKI) es bastante lenta e inadecuada para utilizar con grandes cantidades de datos. Por esta razón, casi siempre se implementa para autenticación y negociación de secretos compartidos, que luego se utilizan para un cifrado simétrico rápido.

Para efectos del protocolo SSL/TLS, el cifrado asimétrico se puede utilizar de dos maneras: para cifrar mensajes a una parte A utilizando la clave pública conocida de A y para firmar digitalmente un mensaje codificando la firma con su propia clave privada.

RSA (llamado así por las iniciales de Ron Rivest, Adi Shamir y Leonard Adleman) es el método preferido en el cifrado asimétrico en la actualidad. El método recomendado actual de RSA es de 2.048 bits, lo que equivale a unos 112 bits en clave simétrica.

Funciones "hash"

Esta técnica no requiere ninguna clave digital, ya que utiliza un valor *hash* de longitud fija cifrado en texto sin formato. Un *hash* es simple: un número que se crea ejecutando un algoritmo *hash* contra cualquier dato. Si los datos permanecen intactos, el valor del *hash* no cambia. Este es un cifrado unidireccional que se utiliza para la integridad del mensaje. MD51, SHA2-1 y SHA2-2 son algunos algoritmos hash (AlFardan *et al*., 2013).

Cifrado simétrico

Utiliza el mismo par de claves para cifrar y descifrar datos. También se conoce como "cifrado de clave secreta" o "cifrado de clave de sesión".

Cifrados de flujo

Son algoritmos de cifrados rápidos, gracias a su funcionamiento sencillo. Encriptan mensajes aplicando XOR a cada *byte* del mensaje de texto plano con un *byte* correspondiente de la secuencia de claves. Para descifrar, la operación XOR se realiza nuevamente en el texto cifrado y el resultado es el mensaje en texto sin formato. El flujo de claves se genera a partir de la clave de cifrado y resultados XOR anteriores y en general parecen datos aleatorios.

Los cifrados de flujo con esta configuración se consideraban seguros; el cifrado de flujo RC4 fue popular para codificar mensajes de protocolo SSL/TLS (AlFardan *et al*., 2013). .

Cifrados de bloque

Actúan sobre bloques de varios *bytes*. Utilizando la clave secreta, cada bloque posible se asigna a un bloque de salida cifrado, y para el descifrado se utiliza la misma clave secreta. AES (Estándar de cifrado avanzado) es un ejemplo de cifrados en bloque.

Clasificación de criptografía simétrica

Existe una gran cantidad de algoritmos para criptografía de clave simétrica, como AES, DES, 3DES, RC4, Blowfish y Twofish (Chandra *et al*., 2014).

Advanced Encryption Standard (AES)

El estándar de cifrado avanzado es un cifrado de bloques simétrico potente que cifra datos en tamaños de bloques de 128 bits. AES utiliza tamaños de clave de 128 bits, 192 bits o 256 bits, también conocidos como AES-128, AES-192 u AES-256. El tamaño de la clave corresponde directamente a su seguridad. AES-128 proporcionó una protección sólida; sin embargo, AES-256 proporciona una protección aún mayor, por su mayor tamaño. AES utiliza menos recursos en comparación con otros algoritmos y puede realizar cifrado y descifrado con rapidez, incluso en dispositivos pequeños como unidades flash USB (Gibson, 2020).

Data Encryption Standard (DES)

El estándar de cifrado de datos es otro cifrado de bloques simétrico que cifra datos en bloques de 64 bits. El proceso de cifrado se divide en 16 etapas, cada una de las cuales consta de 8 cajas S (Gibson, 2020). Primero, los bits se mezclan, se procede con una sustitución no lineal y finalmente se emplea la operación XOR para obtener el resultado. Utiliza un tamaño de clave pequeño de 56 bits que puede romperse con un ataque de fuerza bruta. Actualmente no se recomienda el uso de DES.

Algoritmo RC4

Este algoritmo fue desarrollado por Ronald Rivest, y también se conoce como "Código de Ron" o "Cifrado Rivest", y la versión más utilizada es RC4. Es un cifrado de clave simétrica que se puede utilizar entre 40 y 2048 bits. Requiere un cambio de estado sucesivo de entradas que se basan en la secuencia de teclas (Gibson, 2020). En RC4, el tamaño de la clave puede variar de 1 a 256 bytes y es mucho más rápido que el algoritmo DES. Desde 2013 la Agencia de Seguridad Nacional de los Estados Unidos pudo descifrar RC4, por lo cual se desactivó este algoritmo.

Algoritmo "Blowfish"

Es otro cifrado de bloque simétrico cuyo tamaño de clave varía de 32 bits a 448 bits y cifra los datos en bloques de 64 bits . Este algoritmo fue diseñado para ser un reemplazo de propósito general para DES, pero es más rápido que AES

porque este cifra datos en bloques de 128 bits, mientras que *Blowfish* cifra datos en bloques de 64 bits (Gibson, 2020).

"Twofish"

Es un algoritmo de cifrado de bloques simétrico relacionado con *Blowfish*, pero cifra datos en bloques de 128 bits y tiene un tamaño de clave de 128 bits, 192 bits o 256 bits.

Cifrado asimétrico

Los cifrados asimétricos utilizan dos claves para cifrar y descifrar datos: una "clave pública" y una "clave privada". Para cifrar los datos se utiliza una clave pública y la clave privada correspondiente para descifrarlos (Asaithambi, 2015) . Si la clave pública cifra la información, solo la clave privada correspondiente puede descifrarla. La clave de cifrado no puede producir la clave de descifrado.

Las claves asimétricas se utilizan generalmente para la distribución de claves, siendo algoritmo de cifrado robusto que utilizan una cantidad significativa de potencia de procesamiento para cifrar y descifrar datos (Asaithambi, 2015). Dado que la criptografía asimétrica es difícil de descifrar, es mucho más segura que la criptografía simétrica (Gibson, 2020).

Existen varios algoritmos para criptografía asimétrica, como Diffie-Hellman, RSA, ECC y DSA.

Algoritmo RSA

Es uno de los algoritmos asimétricos más utilizados, y se emplea tanto para cifrado como para firmas digitales (Gibson, 2020). Este es un cifrado asimétrico que utiliza ambos pares de claves mediante internet para mayor seguridad. Para extrema seguridad, el tamaño de la clave RSA debe ser de 1024 bits y superiores; 2048 bits es el estándar moderno. Utiliza una ecuación matemática para calcular números grandes, y la dificultad de factorizar esos números en números primos hace que RSA sea difícil de romper.

Elliptic Curve Cryptography (ECC)

La criptografía de curva elíptica se utiliza a menudo con pequeños dispositivos inalámbricos, ya que no requiere mucha potencia de procesamiento para lograr el nivel de seguridad deseado. Utiliza ecuaciones matemáticas para formular una curva elíptica y luego grafica los puntos de la curva elíptica para crear las claves. Esto requiere menos potencia de procesamiento y es difícil de descifrar (Asaithambi, 2015).

Digital Signature Algorithm (DSA)

El algoritmo de firma digital se utiliza para generar y validar firmas digitales. DSA es una versión electrónica de una firma escrita que el destinatario puede utilizar para verificar la identidad del remitente y garantizar que los datos no hayan sido manipulados (Gibson, 2020). También se puede generar DSA para los datos almacenados para verificar más tarde que la integridad no ha sido cambiada.

Diffie-Hellman Algorithm (DH)

Diffie-Hellman es el primer algoritmo de cifrado asimétrico que permite a los usuarios intercambiar una clave secreta por medio de un canal inseguro sin conocer ningún secreto previo. Una vez que los usuarios conocen la clave simétrica, utilizan el cifrado simétrico para cifrar los datos. Diffie-Hellman se utiliza principalmente para el intercambio de claves en la red pública. Admite claves tanto estáticas como efímeras; RSA se basa en el concepto de intercambio de claves DH estáticas. Hay dos métodos Diffie-Hellman que utilizan claves efímeras (Gibson, 2020) :

- Diffie-Hellman Ephemeral (DHE) utiliza claves efímeras, generando claves diferentes para cada sesión.
- Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) usa claves efímeras generadas por ECC; otra versión, ECDH, usa claves estáticas.

Certificados X.509, PKI y autoridad de certificación

La infraestructura de clave pública se desarrolló para soportar la criptografía asimétrica. Es un grupo de tecnologías que se utilizan para solicitar, crear y gestionar certificados digitales.

Una parte esencial de TLS es la autenticación de al menos el servidor por parte del cliente. Para este propósito, TLS se basa en un sistema PKI en el cual las autoridades de certificación (CA) emiten certificados X.509 a las partes que operan un *host* con un nombre DNS específico (Mockapetris, 1987).

Los certificados suelen estar firmados por CA intermedias. Los clientes, por ejemplo los navegadores web, tienen un almacén de confianza proporcionado por el proveedor (para el sistema operativo) que contiene los certificados raíz de las CA utilizadas para firmar esos certificados intermedios. Por tanto, validar un certificado también se denomina validar la cadena de confianza.

Las autoridades de certificación (CA) pueden emitir certificados para

cualquier nombre de dominio en función de la confianza que se les brinda, dada su capacidad. Se supone que las CA se encargarán de validar las solicitudes de certificados que reciben antes de emitirlos a cualquier parte que afirme ser propietario de dominios. Las CA deben cumplir con las leyes locales y aprobar auditorías que prueben sus modelos de verificación de seguridad y propiedad (Hall, 2013; Ristic, 2013).

El proceso mediante el cual una CA garantiza que una entidad está autorizada para recibir un certificado para un nombre (tiene autoridad sobre el nombre) se llama validación. Tradicionalmente, existen validación de dominio (DV), validación de organización (OV) y validación extendida (EV) (Thomas, 2000).. Por lo general, la DV solo requiere que la parte solicitante del certificado demuestre autoridad sobre un proxy, por ejemplo, poder alojar contenido en el sitio web al que apunta un dominio o poder acceder a una dirección de correo asociada con un dominio (Durumeric *et al*., 2014). OV requiere que la CA emisora verifique una organización, mientras que EV requiere que la CA siga una guía extensa para verificar la entidad solicitante (Thomas, 2000). Por consiguiente, en teoría, los certificados EV deberían reducir de forma significativa los ataques de *phishing* (Mockapetris, 1987).

Ataques al protocolo SSL/TLS

En general, los ataques se pueden dividir en dos categorías: pasivos y activos. Los ataques pasivos implican espiar conexiones para conocer identidades y capacidades criptográficas de clientes y servidores. Los ataques activos, por otro lado, cambian los mensajes que se envían y potencialmente hacen que el atacante se haga pasar por el servidor o el cliente.

Existen limitaciones de compatibilidad y diferentes combinaciones posibles de conjuntos de cifrado utilizadas en implementaciones de protocolos SSL/TLS. La existencia de tales vulnerabilidades representa una amenaza significativa para la seguridad de las comunicaciones en línea, afectando tanto a usuarios individuales como a organizaciones. Dada la creciente dependencia de internet para actividades comerciales y personales, asegurar las comunicaciones en línea es crucial. El estudio de las vulnerabilidades de SSL/TLS es esencial para desarrollar estrategias efectivas que protejan contra ataques y aseguren la integridad y confidencialidad de los datos.

A continuación se analizarán brevemente algunos de los ataques al protocolo SSL/TLS *Handshake*, así como otros elementos necesarios en los protocolos de trabajo, para analizar las fallas

de seguridad que fueron aprovechados, los modos de ataque y las posibles consecuencias, pero también estudiarán los métodos de defensa. Los ataques que se discutirán son:

Ataque Man-In-The-Middle- (MITM)

Un ataque de hombre en el medio es un tipo de ciberataque en el que el atacante puede insertarse entre dos partes, imitar a ambas partes y acceder a la información que se envían entre sí. Un ataque de intermediario permite al atacante interceptar, enviar y recibir información destinada a otra persona o que no debe enviarse en absoluto (Alwazzeh *et al*., 2020).

El ataque Man in the Middle interrumpe secretamente el flujo de los datos entre el servidor y el cliente. Principalmente captura la clave pública del servidor y su propia clave pública al cliente, y este da por sentado que es la clave pública del servidor y envía más información al atacante, pero no al servidor (Radhika *et al*., 2017).

Clases de vulnerabilidades MITM

Hay cinco clasificaciones que identifican las vulnerabilidades que los atacantes aprovechan para efectuar ataques MITM (Stricot-Tarboton, Chaisiri y Ko, 2016).

Cipher Block Chaining

Los cifrados de bloques requieren bloques de longitud fija.

Si los datos del último bloque no son múltiplos del tamaño del bloque, el espacio adicional se llena con relleno. El servidor ignora el contenido del relleno. Solo verifica si la longitud del relleno es correcta y verifica el código de autenticación de mensaje (MAC) del texto sin formato. Eso significa que el servidor no puede verificar si alguien modificó el contenido del relleno. Los atacantes pueden utilizar vulnerabilidades inherentes en el modo de operación Cipher Block Chaining (CBC) para descifrar el contenido de un mensaje HTTPS.

"Compression"

Una parte clave de las comunicaciones HTTPS es la compresión del contenido del mensaje para reducir el uso de recursos. Los atacantes pueden aprovechar la compresión de mensajes comparando diferencias de tamaño, lo que permite inferir el contenido de los mensajes.

"Export Key"

Esta clasificación se aplica a los ataques que explotan claves de seguridad de grado de exportación. Estas claves se introdujeron originalmente para cumplir

con las regulaciones de exportación de criptografía de los Estados Unidos. Las regulaciones limitaron la solidez del *software* de criptografía, con la intención de que las agencias del Gobierno de los Estados Unidos pudieran romper las claves de exportación más débiles. Sin embargo, los atacantes también pueden aprovechar estas claves de seguridad de grado de exportación para atacar las comunicaciones HTTPS y descifrar el contenido de las comunicaciones.

"Implementation Error"

Estos errores suelen ser el resultado de una característica de seguridad mal aplicada o un error en el sistema. Los atacantes pueden aprovechar estos errores de implementación para lanzar ataques.

"Renegotiation"

Permite cambiar los parámetros de conexión HTTPS y las claves en conexiones existentes a pedido.

Mecanismos de protección

Estos son algunos de los mejores mecanismos que hacen más robusta la autenticación, con el propósito de proteger contra ataques MITM:

"Third-Party Solutions"

Son el enfoque más popular que proporciona protección de la primera conexión a un nuevo dominio y certificación escalable de certificados para todos los dominios públicos y requisitos mínimos para aplicaciones web (Dacosta, Ahamad y Traynor, 2012).

"Detection and mitigation using prior knowledge"

Si el cliente tiene información antes de conectarse al servidor, es posible que pueda detectar MITM. Existen varios enfoques para lograr esto:

- DNS (*domain name server*) *certification authority authorization* (DNS *record type* CAA) (Hallam-Baker y Stradling, 2013).
- Mitigar el ataque a las CA, al restringir qué CA pueden emitir certificados para un *host*.
- Mitigar ataques al servidor enviando el certificado del servidor al cliente en la respuesta DNS.
- Mitigar los ataques a servidores habilitando sitios web y declararse accesibles solo a través de HTTPS.

- *Public Key Pinning Extension for HTTP* (Evans, Palmer y Sleevi, 2015).
- Permitir que los sitios web declaren huellas dactilares de servidor autorizadas.
- CA para mitigar ataques al servidor.

Ataque a "ciphersuites"

El *ciphersuite* es una lista de algoritmos criptográficos que se proponen durante la fase de protocolo de enlace entre el cliente y el servidor. La lista de algoritmos propuestos viaja en formato de texto limpio como parte de los mensajes iniciales ClientHello, permitiendo al atacante MITM interceptar el mensaje y reemplazar el conjunto de cifrado del cliente con su conjunto de cifrado que admite versiones más débiles de algoritmos o lista de cifrado NULL, por lo que la comunicación continúa ocurriendo con algoritmos más débiles o no se utilizan algoritmos de protección en absoluto . Las consecuencias de un ataque de este tipo podrían ser desastrosas para el cliente: el atacante podría imitar a un usuario válido, podría acceder al servidor, obtener credenciales de usuario, etc.

En la versión SSL3.0, este error se resuelve con la autenticación de todos los mensajes de protocolo de enlace en el mensaje finalizado final, que contiene mensajes MAC en el protocolo de protocolo de enlace, codificados por el por el secreto maestro, por lo que el ataque al conjunto de cifrado podría notarse al final del proceso fase de apretón de manos y rechazar dicha sesión.

Ataque por longitudes cortas de claves criptográficas

Tener claves criptográficas de longitud corta reduce de manera significativa la seguridad del sistema. Una de las razones más comunes para tener claves criptográficas de longitud corta (normalmente limitadas a 512 bits) es el uso de conjuntos de cifrado de exportación que se construyeron para apaciguar las restricciones a la exportación de criptografía del Gobierno de los Estados Unidos que eran relevantes antes del año 2000 (Stricot-Tarboton, Chaisiri y Ko, 2016). En pospandemia, muchas implementaciones de protocolos SSL/TLS todavía tienen soporte integrado para claves criptográficas débiles en sus modos de exportación. Esto reduce la seguridad, sobre todo en combinación con ataques como *Freak* (Dacosta, Ahamad y Traynor, 2012). Los servidores que todavía tienen habilitados los conjuntos de cifrado de exportación pueden ser engañados para que se cambie a claves

débiles, aunque los modos de exportación nunca se necesitan en el contexto de los protocolos SSL/TLS en la actualidad (Hallam-Baker y Stradling, 2013).

En el momento del descubrimiento del ataque *freak*, alrededor del 36,7 % de los servicios en todo el mundo aceptaban conjuntos de cifrado de exportación RSA_EXPORT; este número ha caído en la actualidad, pero todavía es un porcentaje significativo (Evans, Palmer y Sleevi, 2015).

Ataques entre protocolos

Los servidores que intentan ser compatibles con protocolos antiguos pueden ser engañados para que utilicen esquemas más débiles. Los ataques *freak* se hacen pasar por servidores que utilizan claves RSA de 512 bits, como las que se utilizan en el modo de exportación (Mavrogiannopoulos *et al*., 2012)]. Dado que OpenSSL y otras implementaciones SSL/TLS intentan mantener la compatibilidad con protocolos débiles y obsoletos, siguen afectados por un subconjunto de ataques que de otro modo no se verían afectados (Brodkin, 2018).

Los ataques con frecuencia se realizan manipulando los mensajes que ven el cliente y el servidor, de modo que deciden usar protocolos de exportación u otros parámetros débiles en su modo de compatibilidad, aunque hayan comenzado con un protocolo fuerte de enlace RSA para autenticación. El ataque *freak* se considera crítico, dado que no requiere necesariamente la aceptación de las *suites* de exportación por parte del servidor o cliente y porque afectó a muchas bibliotecas de código abierto y a todos los principales navegadores distintos de Firefox. En este ataque, un atacante MITM envía el mensaje de intercambio de claves del servidor fuera de servicio, y las bibliotecas vulnerables manejan mal este mensaje, de modo que utilizan la clave débil recibida como si la hubiera enviado el cliente.

Ataque entre protocolos a partir del intercambio de claves ECC (criptografía de curva elíptica)

En 2012, varios investigadores descubrieron un nuevo ataque entre protocolos similar al de 2003. La idea principal es la misma que en el ataque anterior, pero ahora, en el lado del cliente, el atacante negociará un intercambio de claves *ephemeral Elliptic Curve Diffie-Hellman key exchangey* en el lado del servidor un intercambio de claves *ephemeral Elliptic Curve DiffieHellman key Exchange*. El atacante interceptará los mensajes de protocolo de enlace

TLS entre el servidor y el cliente y alterará algunos de ellos de modo que el cliente crea que se utilizan para el intercambio de claves *ephemeral Elliptic Curve Diffie Hellman key exchange* y el intercambio de claves *Diffie-Hellman* efímero de curva elíptica del servidor. El objetivo de un atacante es hacerse pasar por un servidor ante el cliente y ponerse entre ellos, para tener la oportunidad de leer, cambiar y reenviar todos los mensajes (Mavrogiannopoulos *et al*., 2012).

La solución propuesta para el ataque entre protocolos fue una nueva extensión de protocolo que indica el nuevo formato del mensaje *ServerKeyExchange* que incluye indicadores explícitos de la entidad (servidor), el tipo de algoritmo de intercambio de claves, los mensajes de protocolo de enlace intercambiados y los parámetros del intercambio de claves (Mavrogiannopoulos *et al*., 2012).

Conclusiones

SSL/TLS es el protocolo de seguridad utilizado para proporcionar integridad y privacidad de los datos entre aplicaciones que se comunican. Estos dos protocolos aislados se utilizan para proteger canales de comunicación entre dos extremos proporcionando dos capas de seguridad como autenticación y cifrado de datos de usuario. Un error lógico u operativo en estos protocolos permite que un atacante lo explote.

Los ataques iniciales al protocolo SSL aprovecharon el hecho de que no se produjo ninguna autenticación durante el proceso de protocolo de enlace (Mavrogiannopoulos *et al*., 2012). Entre los recientes ataques que generaron repercusión en todo el mundo se encuentran el "Fraudulent Microsoft Certificates" en enero de 2001 y el ataque prepandemia Zombie Poodle en febrero de 2019.

Un gran número de aplicaciones de comercio electrónico, como banca y adquisiciones, dependen en gran medida de la solidez del protocolo SSL/TLS, y se utilizan junto con otros protocolos como HTTP, SMTP, etc. Uno de los componentes clave proporcionados por SSL/TLS es el conjunto de algoritmos subyacentes que proporcionan la solidez criptográfica utilizada por el protocolo de seguridad (Radhika *et al*., 2017).

Sin embargo, en caso de que un atacante obtiene acceso a los recursos de los usuarios y comprometa cualquier servicio, de ninguna manera debe capturar fácilmente los datos confidenciales.

Las vulnerabilidades y efectos causados por un ataque de

"Man-In-The-Midlle" (MITM) se clasifican como un ciberataque extremadamente activo y muy difícil de detectar, mitigar y vencer. Para mitigar las posibilidades de éxito de los atacantes se deben implementar y utilizar técnicas robustas de autenticación mutua, algoritmos de cifrado y descifrado y una configuración adecuada del mecanismo de protocolo de enlace del cliente y del servidor.

La incidencia de este tipo de eventos cada día se torna más frecuente, lo cual exige que Gobiernos, grandes corporaciones, entidades públicas, academia y público en general avancen en la implementación de políticas de ciberseguridad para la operación de sistemas informáticos interconectados cada vez más seguros y menos expuestos.

Referencias

AlFardan, N., Bernstein, D., Paterson, K., Poettering, B., & Schuldt, J. (2013). *On the Security of RC4 in TLS. in Proceedings of the 22Nd USENIX Conference on Security, Berkeley, CA, USA*. https://www.usenix.org/system/files/conference/usenixsecurity13/sec13-paper_alfardan.pdf

Alwazzeh, M., Karaman, S., & Shamma, M. N. (2020). Man in the middle attacks against SSL/TLS: Mitigation and defeat. *Journal of Cyber Security and Mobility*, 449-468. https://journals.riverpublishers.com/index.php/JCSANDM/article/view/406

Asaithambi, N. (2015). A Study on Asymmetric Key Cryptography Algorithm, Trichy. *International Journal of Computer Science and Mobile Applications*, *3*(4). 8-13. https://www.ijcsma.com/articles/a-study-on-asymmetric-key-cryptography-algorithms.pdf

Brodkin, J. (2018). OpenSSL code beyond repair, claims creator of "LibreSSL" fork. *Ars Technica, 22*. https://arstechnica.com/information-technology/2014/04/opensslcode-beyond-repair-claims-creator-of-libressl-fork/.

Chandra, S., Bhattacharyya, S., Paira, Y., & Alam, S. (2014). A study and analysis on symmetric cryptography, 2014. *International Conference on Science Engineering and Management Research (ICSEMR),* Chennai, India. doi: 10.1109/ICSEMR.2014.704366

Dacosta, I., Ahamad, M., & Traynor, P. (2012). Trust No One Else: Detecting MITM Attacks against SSL/TLS without Third-Parties. In S. Foresti, M. Yung, F. Martinelli, (Eds.). *Computer Security-ESORICS 2012*. ESORICS 2012. Lecture Notes in Computer Science, vol 7459. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33167-1_12

DANE. (2018). *Boletín técnico. Indicadores básicos de tenencia y uso de Tecnologías de la Información y Comunicación en hogares y personas de 5 y más años de edad*. https://www.dane.gov.co/files/investigaciones/boletines/tic/bol_tic_hogares_2017.pdf

Dierks, T., & Allen, C. (1999). The TLS Protocol Version 1.0. *RFC, 2246*, IETF http://tools.ietf.org/rfc/rfc2246.txt

Dierks, T., & Rescorla, E. (2006). *The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, IETF.* http://tools.ietf.org/rfc/rfc4346.txt

Dierks, T., & Rescorla, E. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF.* http://tools.ietf.org/rfc/rfc5246.txt

Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaber, N., Adrian, D., Paxson, V., Bailey, M., & Halderman, J. A. (2014). The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference* (475-488). https://dl.acm.org/doi/10.1145/2663716.2663755

Eldewahi, A., Sharfi, T., Mansor, A., Mohamed, N., & Alwahbani, S. (2015). SSL/TLS attacks: Analysis and evaluation. *International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, Khartoum, Sudan. doi: 10.1109/ICCNEEE.2015.7381362.

Evans, C., Palmer, C. y Sleevi, R. (2015). Public Key Pinning Extension for HTTP, IETF. https://tools.ietf.org/html/rfc7469

Gibson, D. (2020). *CompTIA Security+ Get Certified Get Ahead SY0-501 Study Guide*. YCDA, LLC.

Hall, K. (2013). *Standards and Industry Regulations Applicable to Certification Authorities*. https://pkic.org/uploads/2013/04/Standards-and-Industry-Regulations-Applicable-to-Certification-Authorities.pdf

Hallam-Baker, P., & Stradling, R. (2013). *DNS Certification Authority Authorization (CAA) Resource Record, IETF*. http://tools.ietf.org/html/rfc6844.

Hopwood, D., Wright, T., Blake-Wilson, S., Mikkelsen, J., & Nystrom, M. (2018). *Transport Layer Security (TLS) Extensions*. https://tools.ietf.org/html/rfc4366.

Fernández, S. (2004). La criptografía clásica. *Sigma*, *24*(24), 119-141. https://dialnet.unirioja.es/servlet/articulo?codigo=950698

Krawczyk, H., Paterson, K. G., & Wee, H. (2013). On the Security of the TLS Protocol: A Systematic Analysis. In R. Canetti, J. A. Garay (Eds.), *Advances in Cryptology-CRYPTO 2013*. CRYPTO 2013. *Lecture Notes in Computer Science, 8042*. https://doi.org/10.1007/978-3-642-40041-4_24

Lee, H. K., Malkin, T., & Nahum, E. (2007, October). Cryptographic strength of SSL/TLS servers: Current and recent practices. In *Proceedings of the 7th ACM SIGCOMM conference on internet measurement* (pp. 83-92).

Mavrogiannopoulos, N., Vercauteren, F., Velichkov, V., & Preneel, B. (2012). A Cross-protocol Attack on the TLS Protocol. *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, New York, NY, USA, 2012. https://doi.org/10.1145/2382196.2382206

Mockapetris, P. (1987). Domain names-concepts and facilities. *RFC*, *1034*, IETF. http://tools.ietf.org/rfc/rfc1034.txt

Radhika, P., Ramya, G., Sadhana, K., & Salini, R. (2017). Defending Man In The Middle

Attacks. *International Research Journal of Engineering and Technology, 4,* 579-585. https://www.irjet.net/archives/V4/i3/IRJET-V4I3159.pdf

Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. *RFC 8446*, IETF http://tools.ietf.org/rfc/rfc8446.txt

Ristic, I. (2013). *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Feisty Duck.

Sheffer, Y., Holz, R., & Saint-Andre, P. (2015a). Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). *RFC, 7525*, IETF http://tools.ietf.org/rfc/rfc7525.txt

Sheffer, Y., Holz, R., & Saint-Andre, P. (2015b). Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS). *RFC, 7457*, IETF http://tools.ietf.org/rfc/rfc7457.txt

Stricot-Tarboton, S., Chaisiri, S., & Ko, R. (2016). *Taxonomy of man-in-themiddle attacks on HTTPS, TrustCom 2016.* IEEE Computer Society, Tianjin, China. https://www.computer.org/csdl/proceedings-article/trustcom-bigdatase-i-spa/2016/07846989/12OmNx0A7KI

Thomas, S. (2000). *SSL and TLS Essentials: Securing the Web*. John Wiley & Sons, Inc.