ESCUELA SUPERIOR POLITÉCNICA AGROPECUARIA DE MANABÍ MANUEL FÉLIX LÓPEZ

II EVENTO INTERNACIONAL

"LA UNIVERSIDAD EN EL SIGLO XXI"

PONENCIA: SIMPOSIO 3

Propuesta Integral de Seguridades contra Ataques a Aplicaciones Web basada en Software Libre.

AUTORES:

Walter Fuertes

Santiago Salvador

Ana Cristina Yépez

Sandro Soto

FECHA:

AGOSTO DE 2013

INTRODUCCIÓN

Las aplicaciones Web son el blanco preferente de los atacantes cibernéticos que buscan explotar alguna vulnerabilidad en el software, hardware o dispositivo de comunicación, causando impactos negativos en la seguridad del sistema, que luego podría repercutir en la imagen de la organización. De acuerdo con el Proyecto de Seguridad para Aplicaciones Web (Open Web Application Security Project**,** OWASP), entre los 10 tipos de ataques de seguridad más comúnmente explotados en aplicaciones Web se encuentran la Denegación de Servicio (DoS) con el 32% y la Inyección SQL con el 21% [1]. Estos ataques pueden provocar robo de información, modificación de datos, intrusión en el servidor Web y suplantación de identidad.

Ante este escenario, la comunidad científica viene desarrollando varias investigaciones que permitan disminuir estos ataques: En [2] se presenta un método para la construcción de esquemas de identificación resistentes contra suplantación de identidad y ataques man-in-the-middle. En relación a los ataques de inyección SQL en [3] se presenta una arquitectura multiagente para la clasificación de consultas con inyección SQL en aplicaciones Web. En [4] los autores presentan una técnica para detectar y prevenir ataques de inyección SQL, que utiliza un enfoque basado en modelos para detectar consultas ilegales antes de que se ejecuten en la base de datos. En relación a los ataques DoS, Chen et al., en [5] proponen un nuevo modelo de detección basados en campos aleatorios condicionales. El trabajo propuesto por [6], presenta un modelo del estado del protocolo y la resistencia de los ataques de DoS, a partir de dos aspectos: uno es el contexto adversario, el otro el proceso aplicando el cálculo de pi. En relación a esquemas de seguridad, trabajos anteriores de nuestro grupo de investigación en [7] y [8] repotenciaron software de ataques de escaneo de puertos y sistemas de firewall de código abierto.

El presente proyecto presenta un enfoque armónico tomando como referencia las mejores prácticas de OWASP, desde una perspectiva experimental de los ataques más comunes a las aplicaciones Web: hombre en el medio, DoS e Inyección SQL. Para llevarlo a cabo se diseñó e implementó una topología experimental basada en equipos reales y máquinas virtuales en las que se

instaló software libre, perpetuando ataques desde el lado del atacante (interno y externo). Para detectar, evaluar, controlar y mitigar los ataques desde el lado del servidor de Aplicaciones Web, se delineó una propuesta integral de mecanismos de seguridad. Posteriormente se evaluó el impacto de los ataques en el consumo de CPU, memoria y rendimiento de la red. Los resultados obtenidos muestran el nivel de precisión alcanzada de la propuesta integral de mitigación.

El resto del artículo ha sido organizado de la siguiente manera: La sección 2 describe el fundamento teórico que sustenta esta investigación. En la sección 3 se explica el diseño, implementación y ejecución de la topología experimental, las pruebas realizadas, los algoritmos desarrollados y la forma de mitigación de cada ataque. En la sección 4 se ilustran los resultados obtenidos. En la sección 5 se analizan los trabajos relacionados. Finalmente en la sección 6 se exponen las conclusiones y el trabajo futuro.

DESARROLLO

En esta sección se describe los ataques utilizados en esta investigación, que son los más comunes que usan los atacantes para burlar las seguridades. También se analizan las herramientas de ataque y las que permiten mitigarlos, todas de software libre. Además describe OWASP, que provee a la academia de mejores prácticas para precautelar la seguridad de redes:

Tipos de ataques

a) **Ataque “hombre en el medio”, (man-in-the-middle).-** Tiene como objetivo leer, insertar y/o modificar los mensajes entre dos extremos de la red (estación del cliente y Servidor Web, por ejemplo), sin que ninguno de ellos descubra de la presencia de este intruso, interceptando los mensajes y obteniendo información privilegiada. Entre sus principales variantes podemos citar: Intercepción de comunicaciones, ataque a partir de textos cifrados escogidos, ataque de sustitución de identidad, ataque de repetición, ataque de DoS. En este proyecto se eligió la Intercepción de comunicaciones, incluyendo análisis de tráfico y ataques de texto planos, atacando al protocolo de la capa de transporte (Secure Sockets Layer, SSL).

b) **Ataque de denegación de servicios, (Denail of Services).-** Tiene como objetivo imposibilitar el acceso a los servicios y recursos de una organización durante un período indefinido de tiempo. Por lo general, este tipo de ataques está dirigido a los servidores de una compañía, para que no se puedan realizar consultas sobre ellos.

c) **Ataque inyección SQL.-** Consiste en la aplicación de una consulta SQL o parte de ella, desde los datos de entrada que tiene una aplicación cliente hacia el servidor. Un atacante puede efectivizar cualquiera de las siguientes acciones: **i)** Suplantar identidad, alterar datos existentes, causar problemas de repudio, permitir la revelación de todos los datos en el sistema incluyendo datos sensibles, destruir los datos o si no volverlos inasequibles; **ii)** Tomar como victimas comunes las aplicaciones PHP y ASP; **iii)** Potenciar el ataque en virtud de la habilidad e imaginación del atacante; y **iv)** Convertirse en administrador del servidor de base de datos.

Herramientas de ataque.

a) **Ettercap**.- Es una herramienta diseñada para realizar ataques de hombre en el medio [9]. Es una de las más utilizadas con este propósito, se la puede considerar como un sniffer, utilizado para redes LAN con un switch de por medio. Soporta direcciones activas y pasivas de varios protocolos (incluso aquellos cifrados, como Secure Shell (SH) y HTTPS). También hace posible la inyección de datos en una conexión.

b) **SSLStrip**.- Es una herramienta que ataca directamente al protocolo Security Socket Layer (SSL) [10]. Funciona engañando a la víctima que está bajo una conexión segura. Generalmente este ataque abusa del desconocimiento de la víctima sobre su entorno tecnológico, ya que la víctima no se fija si está bajo tráfico cifrado o no. Para la víctima es trasparente su presencia dentro de la red.

c) **Low Orbit Ion Cannon.-** Conocida como LOIC, desarrollada para realizar ataques DoS basada en lenguaje de programación C# [11]. La aplicación realiza un ataque enviando una gran cantidad de paquetes TCP, paquetes UDP o peticiones HTTP con objeto de determinar cuál es la cantidad de peticiones por segundo que puede resolver la red objetivo antes de dejar de funcionar, lo

que permite identificar el puerto, el tipo de ataque, así como realizar el ataque por URL o por IP.

Herramientas para mitigar el ataque

a) **Suricata.-** Fue la herramienta seleccionada en este proyecto como IDS/IPS y como motor de monitoreo de seguridad de red. Es un software libre para distribuciones Linux y licencia GPL, que se puede descargar desde los repositorios de su portal [12] y que es compatible con Snort [13].

b) **Oinkmaster.-** Fue utilizada como complemento de Suricata. Es una herramienta de software libre, escrita en Perl, que ayudó a actualizar y cargar automáticamente las reglas de Snort. Permite personalizar la forma en que las reglas se actualizan posibilitando que el administrador las elija [14].

El Proyecto OWASP

"Es una comunidad abierta dedicada a permitir a las organizaciones realizar el desarrollo, adquisición y mantenimiento de aplicaciones fiables. Todas las herramientas, documentos, foros y delegaciones del OWASP son libres y abiertos a cualquiera interesado en mejorar la seguridad de las aplicaciones. Se enfoca a la seguridad en las aplicaciones como un problema tecnológico y un proceso que involucra soluciones integrales y complementarias para fortalecer la seguridad de información de las organizaciones. La fundación OWASP es una entidad sin ánimo de lucro que asegura el mantenimiento del proyecto a largo plazo" [15]. Entre los principales materiales de Educación de OWASP se pueden citar: OWASP Top 10 attacks; Guía de Desarrollo OWASP; Guía de Testing OWASP y Guía OWASP para aplicaciones Web Seguras, que ha sido la base que fundamenta este proyecto. III.

CONFIGURACIÓN DEL EXPERIMENTO

Diseño e implementación de la topología de prueba

La Fig. 1, muestra el diseño e implementación de la topología de experimentación. La red está conformada por los siguientes equipos y dispositivos: Enrutador inalámbrico físico CISCO Linksys para el enlace de

comunicaciones y acceso al Internet; un servidor de aplicaciones Web y Base de Datos, con VirtualBox Versión 4.2.12, instalado Debian 6.0.7, Apache: 2.2.16, PHP: 5.3.3.3-7 y MySql: 3.3.7; un Servidor Web con Debian 6.0.7, Apache: 2.2.16, PHP: 5.3.3.3-7' MySql: 3.3.7; una estación de trabajo con Ubuntu Desktop 12; y finalmente 3 Estaciones de Trabajo con Windows 7 (máquinas físicas). En esta topología se perpetuaron los tres tipos de ataques obteniéndose los resultados de la línea base, constantes en la Tabla 1, del apartado 3.3. Como se puede apreciar, la topología experimental fue configurada con un atacante externo y un atacante interno. El equipo víctima fue el Servidor de Aplicaciones Web. Todos estos elementos fueron pintados en rojo:

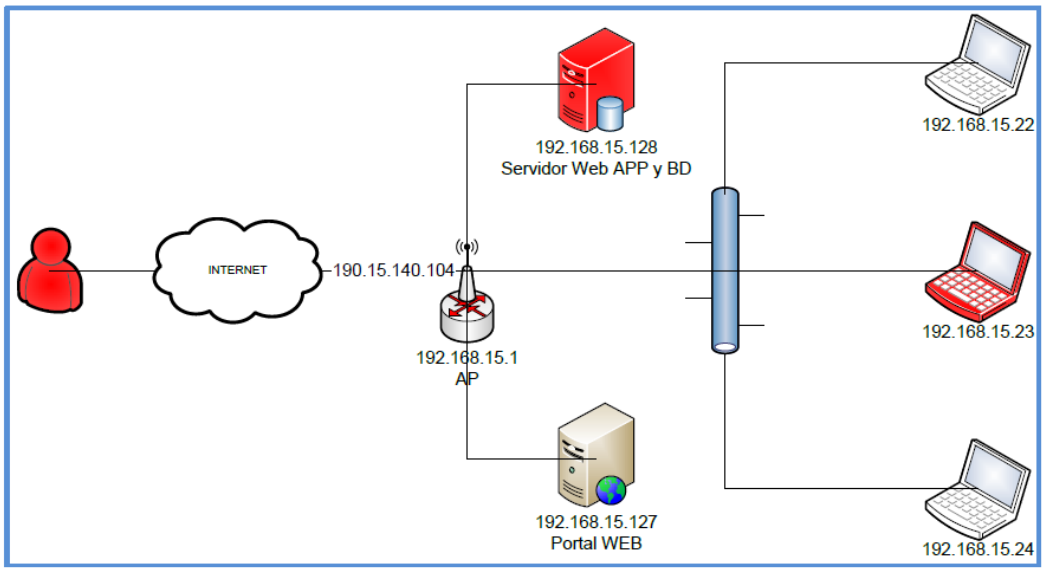

Fig. 1. Topología de Prueba

Perpetuación de ataques

Ataque Hombre en el medio: El objetivo de este ataque fue obtener las claves del sistema del aplicativo colocado en el denominado "Servidor Web APP y Base de Datos". Para ejecutar este ataque se utilizó dos herramientas: *Ettercap y SSLStrip* que se ejecutan en una estación cliente dentro de la red LAN, instaladas sobre Debian.

Ataque DoS: El objetivo fue deshabilitar el aplicativo institucional (sistema de información Web), desde una red externa. Para desarrollar este ataque se

utilizó *Low Orbit Ion Cannon***.** Para empezar este ataque se debió identificar los puertos activos e inactivos que tiene la aplicación, para lo cual utilizamos la herramienta Web *MX Lookup Tool*, que despliega los puertos del servidor de la víctima y su estado. Automáticamente se consiguió que el servidor de aplicaciones presente problemas como el alto consumo de memoria que genera una reducción de la mitad de la memoria disponible, inhabilitando el acceso interno o externo a la Aplicación.

Ataque de inyección SQL: El Objetivo fue penetrar a un sistema basado en Web, mediante la inyección de código SQL desde su página de ingreso. Para el desarrollo de este ataque, un primer método seleccionado fue identificar aquellos sitios que son vulnerables a este tipo de ataque. A través de una sencilla búsqueda en el Google con el argumento de búsqueda "id = cat index.php?", arrojó los resultados con los sitios que contengan en su URL algo similar, por lo tanto serán las más propensas a sufrir el ataque. Un segundo método consistió en enfocarse en un sitio específico, para hacer múltiples intentos de conseguir acceder al esquema de la base de datos o de plano ingresar a la aplicación Web. Este método fue seleccionado por ser medible y con posibilidades de mitigarlo, haciendo uso del mismo portal Web usado en el experimento.

Se procedió a realizar el ataque, ingresando en el campo de USUARIO el extracto de código SQL "**' or 1=1#**", con lo cual se realizó el envío de la información. Se procuró entonces hallar una deficiencia en la sentencia SQL que permita manipularla y poder concatenar con el código SQL inyectado, para que la sentencia genere un resultado exitoso y por lo tanto permita ingresar al sistema. En este caso el ataque fue efectivo y se pudo visualizar el contenido de la información del sistema:

Evaluación de Resultados Línea Base

La Tabla 1, muestra los resultados de los tres ataques perpetuados. Como se puede observar los ataques fueron ejecutados sin la implementación de mecanismos de seguridad y se demuestra que fueron efectivos:

Tabla. 1. Resultados de la perpetuación de ataques

PARAMETRO/TIPO DE ATAQUE	Hombre en el Medio	DoS	Inyección SQL
Número de intentos	1	1	15
Tiempo requerido para el ataque	2 h30 min.	15 minutos	25 minutos
% Consumo CPU	17 %	15%	No aplica
% Consumo Memoria	63%	66%	No aplica
% Rendimiento de la Red	6.10%	1,22%	No aplica

Propuesta Integral para mitigar los ataques

La propuesta de mitigación de estos tres ataques planteados tiene un enfoque integral. Para proteger el servidor de aplicaciones Web y base de datos, se precisaba implementar un firewall mediante software, un Sistema de Detección de Intrusos (IDS), un servidor DNS y separar la Aplicación de la base de datos empresarial. Así mismo se requería la creación de algoritmos que respondan a las recomendaciones OWASP, dentro de la aplicación orientada a la Web. En la Fig. 2 se presenta una topología de prueba mejorada con un mayor nivel de abstracción y con la implementación lógica y física de la propuesta:

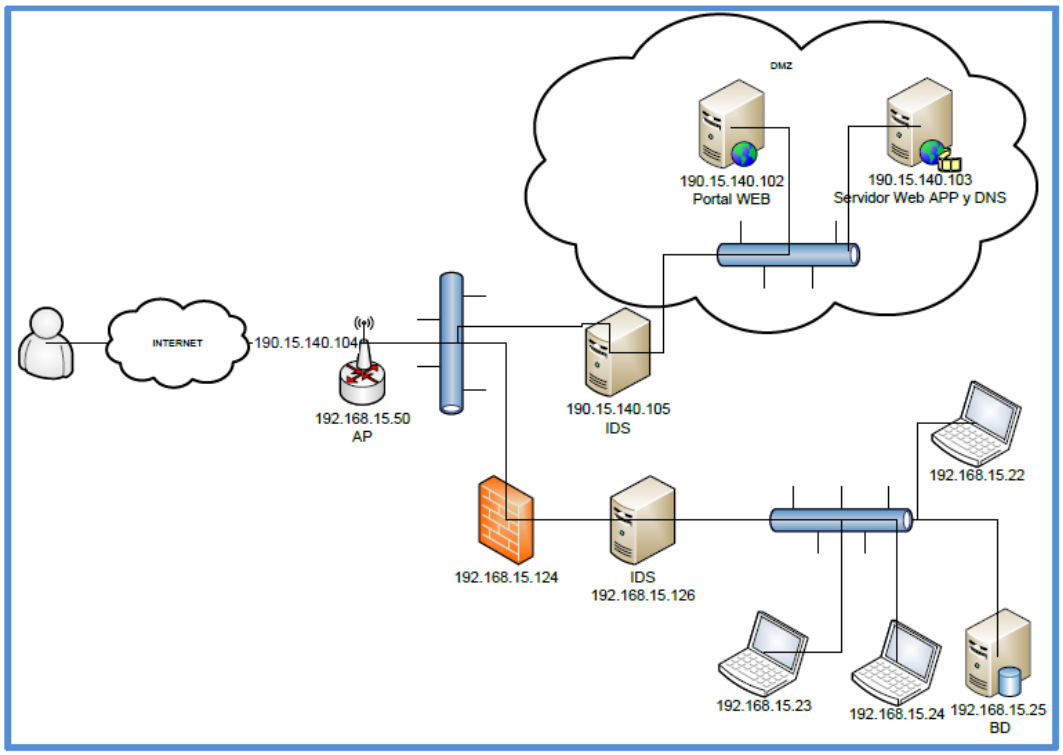

Fig. 2. Abstracción lógica y física de la Propuesta integral de mitigación

a) **Ataque hombre en el medio:** La propuesta de mitigación para este ataque se conformó por dos elementos: **i)** La implementación de un Sistema de Nombres de Dominio (DNS), para que la Aplicación orientada a la Web tenga acceso mediante un nombre distintivo, en lugar de una dirección IP. En esta configuración también se logra separar la base de datos del servidor de Aplicaciones, eliminando así estas dos vulnerabilidades inmediatamente; y, **ii)** la Implementación de protocolo HTTPS seguro, mediante la configuración de certificados digitales, en base a las configuraciones de la herramienta OPENSSL.

b) **Ataque DoS:** Para este ataque sse configuró un IDS de software libre llamado *Suricata*, con el fin de analizar y detectar la actividad del sistema. Para ello se tomó como referencia las mejores prácticas para sistemas de IDS/IPS indicadas por OWASP, que recomienda instalar dentro del esquema de red de mitigación, dos IDS, uno para la Zona Desmilitarizada (DMZ) y otro para la red interna (Intranet/LAN). Para su aplicación, se generó los directorios raíz donde se guardan las configuraciones, archivos de logs y reglas a ser utilizadas para realizar el filtrado de información. Otra de las configuraciones obligatorias fueron las destinadas a establecer los puertos en los cuales se analizaría el tráfico saliente y entrante. El último paso importante fue la configuración de las reglas que establecen como analizar las tramas y paquetes dentro de la red. Para ello se instaló una librería de reglas ya establecidas en base a otros IDS/IPS, llamado *OINKMASTER*, que establece un conjunto de reglas que se pueden actualizar ya que se renuevan diariamente según como se manifiesten las nuevas vulnerabilidades y según el grado de ataque realizado y mitigado. Posteriormente, se obtuvo una interacción eficiente entre el OINKMASTER y el Suricata, pues el primero copió desde el repositorio de reglas las configuraciones necesarias que se poblarían dentro del directorio de reglas de Suricata, interponiéndose a los intrusos, realizando la detección y rechazando las peticiones enviadas a la aplicación o al portal dentro de la estructura de red.

c) **Inyección SQL**: Para la mitigación de este ataque utilizó dos métodos que resultaron complementarios y exitosos conforme a las recomendaciones de OWASP.

i) Uso de cláusulas con parámetros vinculados: Este tipo de consultas conocida como sentencias preparadas permitieron mantener separadas la consulta en sí y los datos transferidos a ella, a través del uso de marcadores de posición denominados parámetros vinculados. A continuación, se muestra el algoritmo implementado:

```
String nomcliente = request.getParameter("nombreCliente"); //Esta entrada debe ser también validada
//Ejecuta una validación de entrada para detectar ataques
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, nomcliente);
ResultSet results = pstmt.executeQuery( );
```

ii) Cláusulas dinámicas en procedimientos almacenados: Esta fue una forma de "dinamizar" los procedimientos almacenados, lo cual resultó un mecanismo eficaz para evitar la mayoría de las formas de ataques de SQL Inyección. En combinación con las consultas con parámetros vinculados la probabilidad de que el ataque de SQL Inyección se lleve a cabo dentro de la aplicación fue menor:

```
create proc QueryDinamicoVulnerable(@NombreUsuario nvarchar (25)) as
declare @sqlcmd nvarchar (255)
set @sqlcmd = "select * from users where username =
+ @NombreUsuario  + '
exec sp_executesql @sqlcmd
```

En síntesis, se procedió a mitigar el ataque Inyección SQL combinando el uso de cláusulas dinámicas, el paso de parámetros vinculados y el evitar el uso de un súper usuario para la conexión a la base de datos. Una vez aplicada la mitigación a nivel del código, se procedió con la misma prueba de SQL inyección, demostrándose que el código inyectado se interpretó como el valor del login del usuario y no como parte de la sentencia SQL para que altere la consulta, por tanto el resultado fue exitoso. La Fig. 3 muestra la interface gráfica de usuario y sus resultados:

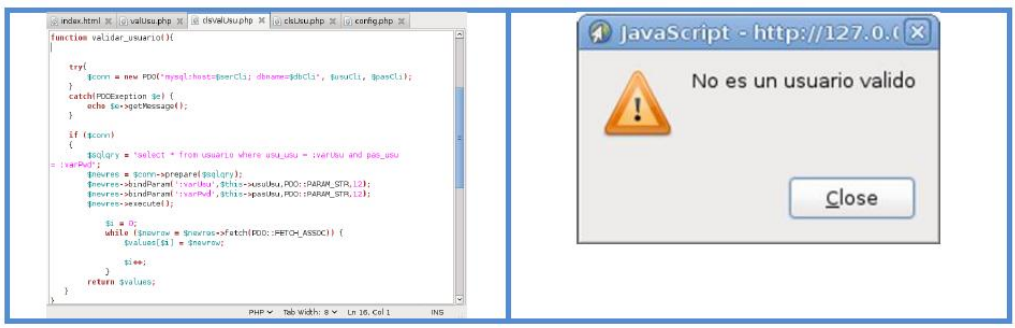

Fig. 3. a) Sentencias de código SQL modificadas y b) Comprobación de SQL inyección fallido

EVALUACIÓN DE RESULTADOS Y DISCUSIÓN

La Fig. 4 muestra los resultados obtenidos del consumo de CPU, memoria y rendimiento de la red, antes y durante el ataque del hombre en el medio, y después de la mitigación. Como se puede observar existe un aumento de consumo de recursos durante la ejecución del ataque.

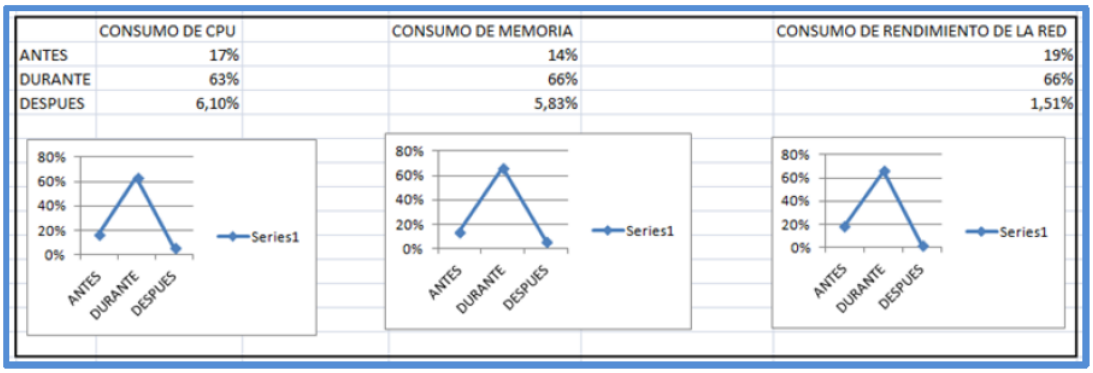

Fig. 4. Gráficos lineales del Consumo de recursos en el ataque hombre del medio.

La Fig. 5 muestra los resultados obtenidos del consumo de recursos durante el ataque de DoS. Como se puede apreciar existe una utilización similar al primer ataque, sobre todo durante la ejecución del ataque.

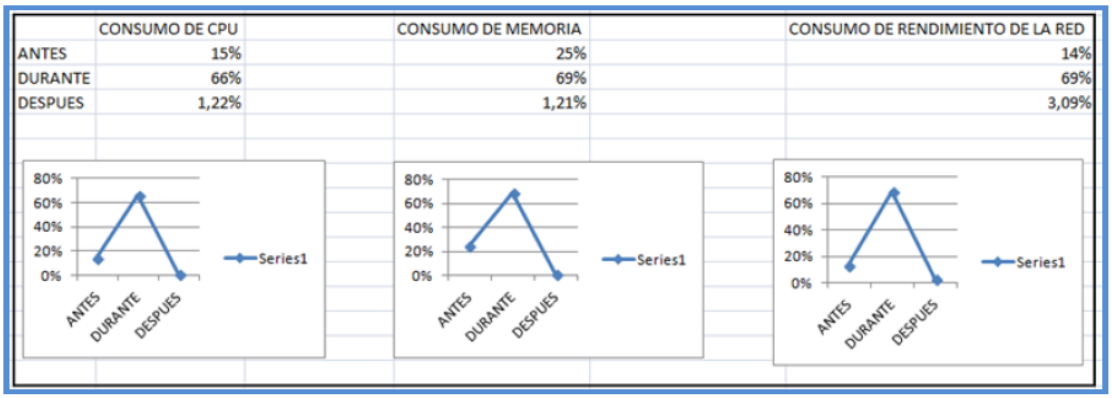

Fig. 5. Gráficos lineales del Consumo de recursos en el ataque de denegación de servicios.

La Fig. 6 muestra los resultados obtenidos antes, durante y después de un ataque Inyección SQL. Por el tipo de caso de estudio seleccionado lo único que se pudo medir fue el tiempo (eje vertical) que tomó cada ataque (eje horizontal) antes de la mitigación y posterior a la mitigación, donde se refleja una clara variación del tiempo, complicando al atacante ya que debe controlar más variables para vulnerar el sistema ejemplo.

Como se puede apreciar, el tiempo de intromisión dentro de un sistema o vulneración de seguridades es relativo al tipo de ataque que se esté ejecutando. Así mismo, las actividades de escucha dentro de la red intranet, son casos reales que perjudican a las instituciones de manera incuantificable.

Dado los resultados de las Fig. 4, 5 y 6, se demuestra que gran parte de la mitigación de este tipo de ataques se soluciona con una cultura tecnológica y de seguridad adecuada. Por tanto, responsablemente se debe implementar normativas, manuales, políticas y procedimientos que deben ser difundidos a toda la organización, con la finalidad de que cada funcionario tome las medidas de prevención correspondientes.

Por otro lado, es indiscutible que es una responsabilidad de todos implementar mecanismos que protejan a la organización y que el uso del software libre es una probada solución. Finalmente, se recomienda que cuando se desarrollan aplicaciones Web dentro de las organizaciones es responsabilidad de cada programador/desarrollador se consideren las recomendaciones de OWASP al momento de desarrollar las aplicaciones.

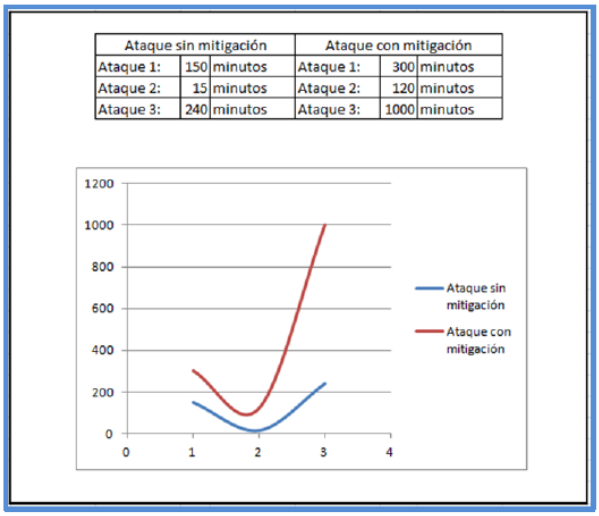

Fig. 6. Gráfico de dispersión durante el ataque de inyección SQL

TRABAJOS RELACIONADOS

En relación a ataques del hombre en el medio, en [2] se presenta un método para la construcción de esquemas de identificación resistentes contra suplantación de identidad y ataques man-in-the-middle. En relación a los ataques de inyección SQL [3] se presenta una arquitectura multiagente para la clasificación de consultas con inyección SQL en aplicaciones Web. La

arquitectura integra la clasificación de consultas SQL, la monitorización de las acciones de los usuarios y la extracción de conocimiento a partir de una bitácora de consultas. En este mismo contexto, en [4] los autores presentan una técnica para detectar y prevenir ataques de inyección SQL. En su parte estática, la técnica utiliza el análisis del programa que automáticamente construirá un modelo de las preguntas legítimas que podrían ser generados por la aplicación. En su parte dinámica, la técnica utiliza el monitoreo en tiempo de ejecución para inspeccionar las consultas generadas dinámicamente y los compararán con el modelo estático construido. En comparación con nuestro trabajo, nosotros hemos puesto en práctica los pasos sugeridos por OWASP implementando una solución con sentencias SQL con parámetros vinculados, combinados con sentencias SQL de carácter dinámico y la restricción de privilegios a los administradores o equivalentes

En relación a los ataques de DoS, Chen et al., en [5] propone un nuevo modelo de detección basados en campos aleatorios condicionales, incorporando la firma y los métodos de detección basados en anomalías. Los resultados del experimento muestran que el método tiene una mayor precisión de detección y la sensibilidad, así como ha disminuido falsas alarmas positivas. El trabajo propuesto por [6], presenta un modelo del estado del protocolo y la resistencia de los ataques de DoS, a partir de dos aspectos: uno es el contexto adversario, el otro el proceso aplicando el cálculo de pi. Nosotros por nuestra parte decidimos configurar software IDS/IPS de libre distribución para definir reglas de filtrado y el aislamiento del Servidor de aplicaciones Web y la base de datos.

CONCLUSIONES Y TRABAJO FUTURO

La presente investigación tuvo como propósito evaluar tres ataques comunes a las Aplicaciones Web: hombre en el medio, denegación de servicios (DoS) e Inyección SQL. Para llevarlo a cabo se diseñó e implementó una topología experimental basada en equipos reales y máquinas virtuales cuyos resultados preliminares demostraron la necesidad de implementar una propuesta integral utilizando mecanismos de seguridad mediante software libre, para detectar, evaluar, controlar y mitigar todos los ataques desde el lado de la víctima. Luego se evaluó el impacto de los ataques en el consumo de CPU, memoria y

rendimiento antes, durante la ejecución de ataque y después de la mitigación. Los resultados obtenidos muestran el éxito alcanzado de la propuesta integral. Revelan además que el tiempo de intromisión dentro de un sistema o vulneración de seguridades es relativo al tipo de ataque que se esté ejecutando, que gran parte de la mitigación se soluciona con una cultura tecnológica y de seguridad adecuada, debiéndose implementar normativas, manuales, políticas y procedimientos que deben ser difundidos a toda la organización, y que la aplicación de OWASP y el uso del software libre son soluciones probadas y sin costo.

Como trabajo futuro se planea utilizar algoritmos de entrenamiento de redes neuronales como mecanismos de mitigación.

REFERENCIAS BIBLIOGRÁFICAS

[1] OWASP Top ten 2013, disponible en http://www.elladodelmal.com/2013/03/owasp-top-ten-2013-en-release-candidate.html. Última revisión, julio de 2013.

[2] Cramer, R., & Damgård, I. (1997, January). Fast and secure immunization against adaptive man-in-the-middle impersonation. In Advances in Cryptology—EUROCRYPT'97 (pp. 75-87). Springer Berlin Heidelberg.

[3] Cristian I. and Corchado J.M., Arquitectura de un Sistema Multiagente para la Clasificación de Consultas con Inyección SQL, Universidad de Salamanca, España, págs. 41-50, ISBN 978-84-612-1283-5, diciembre 2007.

[4] Halfond, W. G., & Orso, A. (2005, November). AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering (pp. 174-183). ACM.

[5] Chen, S. W., Wu, J. X., Ye, X. L., & Guo, T. (2013). Distributed Denial of Service Attacks Detection Method Based on Conditional Random Fields. Journal of Networks, 8(4), 858-865.

[6] Meng, B., Huang, W., & Li, Z. (2013). Automated Proof of Resistance of Denial of Service Attacks Using Event with Theorem Prover. Journal of Computers, 8(7), 1728-1741.

[7] Walter Fuertes, Patricio Zambrano, Marco Sánchez, and Pablo Gamboa, “Alternative Engine to Detect and Block Port Scan Attacks using Virtual Network Environments”, in IJCSNS-International Journal of Computer Science and Network Security”, Special Issues: Communication Network & Security. Vol. 11, No. 11, pp. 14-23. ISSN: 1738-7906. Seul, Korea, Nov. 30, 2011.

[8] Walter Fuertes, Patricio Zambrano, Marco Sánchez y Mónica Santillán, “Repotenciación de un firewall de Código abierto basado en una Evaluación Cuantitativa”, Publicado en la Revista de la Facultad de Ingeniería de dicha universidad, con ISSN 0121-1129, Sogamoso, Colombia, 2012.

[9] Ettercap Home page: URL: http://ettercap.github.io/ettercap/. Última consulta agosto de 2013.

[10] SslStrip: URL. http://sectools.org/tool/sslstrip/ Última consulta agosto de 2013.

[11] Loic Home Page: URL. http://resources.infosecinstitute.com/loic-dos-attacking-tool/.

[12] Suricata Website: URL: http://suricata-ids.org/. Última consulta agosto de 2013.

[13] J. Zhai and Y. Xie,”Researh on Network Intrusion Prevention System Based on Snort”, In Proceedings of IEEE Strategic Technology (IFOST), 2011 6th International Forum on, vol. 2, pages:1133-1136, 2011.

[14] Oinkmaster Website: URL: http://oinkmaster.sourceforge.net. Última consulta agosto de 2013.

[15] OWASP, Guía de Pruebas, versión 3.0, Creative Commons Attribution-ShareAlike, https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf. Última revisión, agosto de 2013.

III. CONFIGURACIÓN DEL EXPERIMENTO

3.1 Diseño e implementación de la topología de prueba

La Fig. 1, muestra el diseño e implementación de la topología de experimentación. La red está conformada por los siguientes equipos y dispositivos: Enrutador inalámbrico físico CISCO Linksys para el enlace de comunicaciones y acceso al Internet; un servidor de aplicaciones Web y Base de Datos, con VirtualBox Versión 4.2.12, instalado Debian 6.0.7, Apache: 2.2.16, PHP: 5.3.3.3-7 y MySql: 3.3.7; un Servidor Web con Debian 6.0.7, Apache: 2.2.16, PHP: 5.3.3.3-7' MySql: 3.3.7; una estación de trabajo con Ubuntu Desktop 12; y finalmente 3 Estaciones de Trabajo con Windows 7 (máquinas físicas). En esta topología se perpetuaron los tres tipos de ataques obteniéndose los resultados de la línea base, constantes en la Tabla 1, del apartado 3.3. Como se puede apreciar, la topología experimental fue configurada con un atacante externo y un atacante interno. El equipo víctima fue el Servidor de Aplicaciones Web. Todos estos elementos fueron pintados en rojo:

Fig. 1. Topología de Prueba

3.2 Perpetuación de ataques

Ataque Hombre en el medio: El objetivo de este ataque fue obtener las claves del sistema del aplicativo colocado en el denominado "Servidor Web APP y Base de Datos". Para ejecutar este ataque se utilizó dos herramientas: *Ettercap y SSLStrip* que se ejecutan en una estación cliente dentro de la red LAN, instaladas sobre Debian.

Ataque DoS: El objetivo fue deshabilitar el aplicativo institucional (sistema de información Web), desde una red externa. Para desarrollar este ataque se utilizó *Low Orbit Ion Cannon***.** Para empezar este ataque se debió identificar los puertos activos e inactivos que tiene la aplicación, para lo cual utilizamos la herramienta Web *MX Lookup Tool*, que despliega los puertos del servidor de la víctima y su estado. Automáticamente se consiguió que el servidor de

aplicaciones presente problemas como el alto consumo de memoria que genera una reducción de la mitad de la memoria disponible, inhabilitando el acceso interno o externo a la Aplicación.

Ataque de inyección SQL: El Objetivo fue penetrar a un sistema basado en Web, mediante la inyección de código SQL desde su página de ingreso. Para el desarrollo de este ataque, un primer método seleccionado fue identificar aquellos sitios que son vulnerables a este tipo de ataque. A través de una sencilla búsqueda en el Google con el argumento de búsqueda "id = cat index.php?", arrojó los resultados con los sitios que contengan en su URL algo similar, por lo tanto serán las más propensas a sufrir el ataque. Un segundo método consistió en enfocarse en un sitio específico, para hacer múltiples intentos de conseguir acceder al esquema de la base de datos o de plano ingresar a la aplicación Web. Este método fue seleccionado por ser medible y con posibilidades de mitigarlo, haciendo uso del mismo portal Web usado en el experimento.

Se procedió a realizar el ataque, ingresando en el campo de USUARIO el extracto de código SQL "**' or 1=1#**", con lo cual se realizó el envío de la información. Se procuró entonces hallar una deficiencia en la sentencia SQL que permita manipularla y poder concatenar con el código SQL inyectado, para que la sentencia genere un resultado exitoso y por lo tanto permita ingresar al sistema. En este caso el ataque fue efectivo y se pudo visualizar el contenido de la información del sistema: 3.3 Evaluación de Resultados Línea Base

3.3 Evaluación de Resultados Línea Base

La Tabla 1, muestra los resultados de los tres ataques perpetuados. Como se puede observar los ataques fueron ejecutados sin la implementación de mecanismos de seguridad y se demuestra que fueron efectivos:

Tabla. 1. Resultados de la perpetuación de ataques

PARAMETRO/TIPO DE ATAQUE	**Hombre en el Medio**	**DoS**	**Inyección SQL**
Número de intentos	1	1	15
Tiempo requerido para el ataque	2 h30 min.	15 minutos	25 minutos
% Consumo CPU	17 %	15%	No aplica
% Consumo Memoria	63%	66%	No aplica
% Rendimiento de la Red	6.10%	1,22%	No aplica