# CAPÍTULO 5 PRUEBAS DEL SISTEMA DE DEFENSA



### PRUEBAS DEL SISTEMA DE DEFENSA

#### 5.1 PLAN DE PRUEBAS

En este capítulo se hace una serie de pruebas dirigidas a ciertas áreas del sistema de defensa con los siguientes objetivos en mente:

- Detectar un error.
- Tener un buen caso de prueba, es decir que se tenga más probabilidad de mostrar un error no descubierto antes.
- Descubrir un error no descubierto antes.

La pruebas no pueden asegurar que no existen errores en la configuración total del sistema sólo pueden mostrar que existen defectos en las áreas que se están analizando, dichos defectos pueden ser tanto de instalación como de configuración y uso del software analizado.

Los elementos del sistema de defensa que se someterán a prueba son: John the Ripper, el analizador de vulnerabilidades web Nikto, Wireshark, Nessus, Turtle *Firewall* y el Sistema de Detección de Intrusos.

Todos estos elementos se probarán por separado, ya que el funcionamiento de dos o más elementos al mismo tiempo podría alterar los resultados de las pruebas, por ejemplo, un análisis de vulnerabilidades con Nikto se registraría en el detector de intrusos y se podría perder de vista un ataque real al sistema.

Los elementos que no se probarán son: *OpenSSH* y Cracklib por ser elementos que ya se integran por defecto en la mayoría de los sistemas Linux y únicamente requieren una correcta configuración. La instalación, configuración y prueba de Bastille Linux se realiza en el capítulo 6. A continuación se presentan varios casos de uso de los programas instalados y configurados en el sistema de defensa, en cada uno de ellos se determina la estrategia a seguir para la prueba, los participantes en la prueba, y se analizan los resultados obtenidos a partir de las condiciones dadas.

# **5.2 JOHN THE RIPPER**

Nombre:	Prueba de John the Rippper.									
Descripción: se utiliza John pueden ser una vulnerabilidad	de Ripper para encontrar contraseñas débiles que en el sistema.									
Actores: usuario con privilegios.										
Precondiciones: El usuario con privilegios puede ejecutar la herramienta John the										
Ripper y leer los archivos /etc	/passwd y /etc/shadow. Se crea una contraseña débil									
para root y se espera que la her	ramienta la descubra.									
Flujo normal: El usuario con p	rivilegios ejecutará John the Ripper con un archivo de									
lista de palabras, la herramient	a obtendrá las contraseñas inseguras o débiles.									
Flujo alternativo: la herramien	ta no detecta contraseñas inseguras y continúa con su									
ejecución hasta que el usuario c	con privilegios decide detener la prueba.									
Poscondiciones: se detecta la co	ntraseña débil y se muestra el resultado.									

Los sistemas Linux recientes utilizan passwords encriptados con *SHA-256* o *SHA-512* (passwords que empiezan con \$5\$ o \$6\$ respectivamente, en el archivo shadow), estos algoritmos tienen ventajas ante los algoritmos utilizados en sistemas Linux viejos, como *MD5* (\$1\$), pero John the Ripper no tiene soporte para los nuevos algoritmos implementados.

Al ver la primera línea del archivo /etc/shadow podemos cerciorarnos de que este sistema en particular utiliza *SHA-512*,

# root**\$6**\$VdinK4PM\$ZZlLNW76hm0oWfkmU9W4I3xDpbw9Ou31s7D2FidFQpilvQUEF4 4HfuOzI47CykLWy5I2fuyG2k6.RXVCdk5Jg0:14756:0:99999:7:::

por lo que John the Ripper no puede crackear este tipo de contraseñas.

Sin embargo, se puede hacer uso de un script en *perl* que utiliza la función crypt y la lista de palabras de John the Ripper. Aunque de rendimiento muy lento, este script es una opción viable mientras no exista soporte para John the Ripper con el algoritmo SHA-512.

Primero se cambia la contraseña; en este caso y únicamente como prueba se usará el usuario root.

[kakaroto@localhost run]# passwd root

Cambiando la contraseña del usuario root.

Nueva UNIX contraseña: CONTRASEÑA INCORRECTA: Es demasiado corta. Vuelva a escribir la nueva UNIX contraseña: passwd: todos los tokens de autenticación se actualizaron exitosamente. You have new mail in /var/spool/mail/root

Ahora se utiliza el comando unshadow para combinar los archivos /etc/passwd y /etc/shadow, para que puedan ser usados por John the Ripper.

[kakaroto@localhost run]# ./unshadow /etc/passwd /etc/shadow > crack.db

Y finalmente se usa el script en *perl* y una lista de palabras para desencriptar las contraseñas.

[kakaroto@localhost run]# cat password.lst | perl cryptcrack.pl -f crack.db

Read 49 hashes from file Spawning 4 threads 11.071 keys per second. 12.201 keys per second. 11.001 keys per second. 12.001 keys per second. 12.001 keys per second. 12.201 keys per second.

.11.201 keys per second. 12.201 keys per second. 11.801 keys per second. 18.001 keys per second.

#### **RESULTADOS:**

FOUND: hola

(\$6\$VdinK4PM\$ZZlLNW76hm0oWfkmU9W4I3xDpbw9Ou31s7D2FidFQpilvQUEF44HfuOzI47 CykLWy5I2fuyG2k6.RXVCdk5Jg0)

Cracked passwords:

John the Ripper muestra la contraseña encontrada, y la línea del archivo crack.db que fue descifrada, con lo que se tiene un resultado satisfactorio ya que se pudo verificar que el programa utilizado es adecuado y funciona correctamente.

# **5.3 NESSUS**

Nombre:	Prueba de Nessus.									
Descripción: Se ejecutará Nessus para encontrar las vulnerabilidades del sistema operativo, de la red y de las aplicaciones en el sistema operativo con una instalación base, sin actualizar.										
Actores: usuario con privilegios.										
Precondiciones: el sistema operativo no se ha actualizado por lo que Nessus deberá detectar todos los paquetes y programas que requieren actualización.										
Flujo normal: El usuario con p	privilegios ejecutará Nessus y este programa detectará									
las vulnerabilidades por falta d	le actualización de software.									
Flujo alternativo: la herramie	nta no detecta ninguna de las vulnerabilidades o sólo									
detecta algunas de las posibles	vulnerabilidades.									
Poscondiciones: se detectan l correspondiente.	as vulnerabilidades del sistema y se crea el informe									

El primer paso consiste en ejecutar el cliente de Nessus, para ellos se abre una terminal y se ingresa el siguiente comando:

[root@kakaroto]#/opt/nessus/bin/NessusClient

Ahora en la interfaz de Nessus se ejecuta un escaneo al servidor local, con todos los plugins por default activados, figura 5.1 Plugins de Nessus.



Figura 5.1, Plugins de Nessus.

Nessus permite generar un archivo html con los resultados del escaneo, parte de dicho archivo se muestra a continuación, en él se detallan las vulnerabilidades del sistema y la manera de corregirlas, figura 5.2, Resultados de Nessus.



Figura 5.2, Resultados de Nessus.

## **RESULTADOS:**

En este caso Nessus detectó 101 vulnerabilidades de riesgo bajo, 39 de riesgo medio y 55 de alto riesgo, éstas últimas relacionadas en su totalidad con la falta de actualización del sistema operativo y programas de aplicación. Las vulnerabilidades de riesgo medio y bajo fueron en su mayoría vulnerabilidades del servidor web y de actualización de librerías.

La prueba de Nessus se considera exitosa al detectar más vulnerabilidades de las que se esperaban encontrar en un principio. Esto muestra que Nessus está funcionando correctamente de acuerdo con los plugins activados.

# **5.4 NIKTO**

Nombre:	Prueba de Nikto.											
Descripción: Se ejecutará el p verificar, en este caso el puerto	Descripción: Se ejecutará el programa Nikto en el servidor web con el puerto a verificar, en este caso el puerto 80.											
Actores: usuario con privilegios	S.											
Precondiciones: El servidor web ha sido instalado pero sin hacer una configuración segura, se ejecutará el programa Nikto y se espera que detecte las vulnerabilidades a que está expuesto.												
Flujo normal: El usuario con j las vulnerabilidades existentes o	privilegios ejecutará Nikto y la herramienta detectará en el servidor web.											
Flujo alternativo: la herramienta no detecta ninguna vulnerabilidad.												
Poscondiciones: se detectan vul	nerabilidades web y se muestran en pantalla.											

Se ejecuta Nikto en el servidor local y el puerto 80.

[root@localhost nikto-2.1.0]# perl nikto.pl -h localhost:80

\*\*\*\*\* SSL support not available (see docs for SSL install instructions) \*\*\*\*\*
 Nikto v2.1.0/2.1.0

- + Target IP: 127.0.0.1
- + Target H . 127.0.0.1 + Target Hostname: localhost
- + Target Port: 80
- + Start Time: 2010-05-28 17:56:50

-----

+ Server: Apache/2.2.9 (Fedora)

+ OSVDB-0: Apache/2.2.9 appears to be outdated (current is at least Apache/2.2.14). Apache 1.3.41 and

2.0.63 are also current.

+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST

+ OSVDB-682: /usage/: Webalizer may be installed. Versions lower than 2.01-09 vulnerable to Cross Site

Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.

+ OSVDB-3092: /manual/: Web server manual found.

+ OSVDB-3268: /icons/: Directory indexing is enabled: /icons

+ OSVDB-3268: /manual/images/: Directory indexing is enabled: /manual/images

+ OSVDB-3233: /icons/README: Apache default file found.

+ 3582 items checked: 8 item(s) reported on remote host

+ End Time: 2010-05-28 17:57:14 (24 seconds)

<sup>+</sup> OSVDB-0: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE

## • 1 host(s) tested

## **RESULTADOS:**

Se encontraron 8 vulnerabilidades en el servidor web, entre ellas, destacan por su severidad, la falta de actualización del servidor web y la vulnerabilidad conocida como Cross Site Tracing (XST), vulnerabilidad que explota controles ActiveX, Flash, Java y otros que permiten la ejecución de una llamada http trace para obtener el valor de las cookies de un navegador.

Esta prueba se considera exitosa ya que Nikto fue capaz de detectar las vulnerabilidades a que está expuesto el servidor web, lo que permite en su caso generar un informe con las alertas para que el área web corrija la configuración del servidor analizado.

#### **5.5 WIRESHARK**

Nombre:	Prueba de Wireshark.
Descripción: Se ejecutará Wir que pueda ser detectada, en e mensajería instantánea para h web.	eshark y después se creará cierta actividad en la red ste caso, se establecerá una comunicación a través de nacer el seguimiento TCP stream y se abrirá un sitio
Actores: usuario con privilegio	S.
Precondiciones: Se debe crear usuario privilegiado que ejecut	cierta actividad en la red que sea monitoreada por un e Wireshark.
Flujo normal: El usuario con detectará la actividad en la red	n privilegios ejecutará Wireshark y la herramienta
Flujo alternativo: la herramien	ta no detecta la actividad creada.
Poscondiciones: se capturan análisis.	los paquetes de la actividad creada para posterior
El usuario con privilegios debe log los siguientes comandos:	guearse y ejecutar Wireshark, para ello se deben teclear

[root@kakaroto]#su – estudiante [estudiante@kakaroto]#wireshark

Se abre la interfaz de Wireshark, en el menú Capture se hace click en la pestaña Start, para iniciar la captura de paquetes. Ahora se abre un navegador de Internet y para este ejemplo se ingresa la dirección http://www.openwall.com, en la interfaz de Wireshark se muestra la captura de la actividad creada al abrir una página web, figura 5.3, Captura de Wireshark.

No.		Tim	ie		S	our	ce						)est	tina	tion	1			Pr	oto	col	Info																1	1
	23	13.	728	943	1	92.	168	. 1.	64				92.	168	. 1.	254			DI	IS		Star	ıda	rd	aue	rv A	WV	w. o	oenv	all	.con	n							
	24	13.	775	970	1	92.	168	3.1.	254			]	92.	168	. 1.	64			DI	IS		Star	ıda	rd	que	ry r	esp	oon s	eА	195	. 42.	179	9.20	92					
	25	13.	776	689	1	92.	168	3.1.	64			J	.95.	42.	179	9.202	2		TC	P		5751	L2	> h	ttp	[SY	N]	Seq	=0 L	_en=	0 MS	SS=1	L460	) T	SV=8	3677	066	TSE	
	26	14.	027	483	1	95.	42.	179	. 20	2		J	.92.	168	. 1.	64			т	P		http	) >	57	512	[SY	N,	ACK	] Se	eq=0	Acl	<=1	Win	1=5	840	Len	=0 M	SS=	1
	27	14.	027	574	1	92.	168	3.1.	64			]	.95.	42.	179	9.202	2		т	P		5751	L2	> h	ttp	[AC	ĸj	Seq	-1 A	\ck=	1 Wi	in=5	5888	3 L	en=(	9			1
	28	14.	030	942	1	92.	168	3.1.	64				95.	42.	179	9.202	2		HT	TΡ		GET	/	нтт	P/1	. 1													
	29	14.	311	348	1	95.	42.	179	. 20	2		]	92.	168	1.1.	64			т	P		http	>	57	512	[AC	K]	Seq	=1 A	۱c k=	408	Wir	1=69	912	Ler	n=0			
	30	14.	333	361	1	95.	42.	179	. 20	2		1	92.	168	. 1.	64			TC	P		[TCF	° s	egm	ent	of	аı	reas	semb	lec	PDU	1]							
	31	14.	333	446	1	92.	168	3.1.	64			1	.95.	42.	179	9.202	2		т	P		5751	L2	> h	ttp	[AC	K]	Seq	=408	B Ac	k=14	153	Win	1=8	768	Len	=0		
	32	14.	347	114	1	95.	42.	179	. 20	2		1	.92.	168	. 1.	64			т	P		[TCF	° 5	egm	ent	of	аı	reas	semb	lec	PDU	Л]							
	33	14.	347	207	1	92.	168	3.1.	64			1	.95.	42.	179	9.202	2		TC	P		5751	L2	> h	ttp	[AC	K]	Seq	=408	Ac	k=29	905	Win	1=1	171:	2 Le	n =0		
	34	14.	591	781	1	95.	42.	179	. 20	2		1	92.	168	. 1.	64			т	P		[TCF	° s	egm	ent	of	аı	reas	semb	lec	PDU	[ו							
	35	14.	591	863	1	92.	168	3.1.	64			1	.95.	42.	179	9.202	2		т	P		5751	L2	> h	ttp	[AC	К1	Seq	=408	Ac	k=43	357	Win	1=1-	4656	5 Le	n=0		-
< (																																							
ÞF	rame	1	(60	byt	es	on	wir	ъ,	60	oyte	es d	cap	tur	ed)																									٦
D E	ther	net	TT	Śr	· .	00.	24.	56:	9a :	16 · F	1	( 00	. 24	: 56	· 9a	: 16:	b1)	)	)st:	Br	oad	ast	(f	fif	fif	fiff	• f	fiff	)										1
	ddre		2050	1+	ion	Dr	oto	col	(n		ac+1	100				1 201	~ L)	· · ·			ouu	abe							<i>'</i>										
r •	aure	33 1	1030	, cu c	101		010			cque	.st,	,																											1
																																							4
0000	) f1	ff	ff	ff	ff	ff	00	24	56	9a	16	b1	. 08	06	00	01				\$ V	/																		
0010	00	3 00	06	04	00	01	00	24	56	9a	16	b1	. cG	a8	01	_ fe				\$ V	/																		
0020	) †1		††	††	††	††	C0	a8	01	40	00	00	00	00	00	00				• •	@																		
0030	0 00	00	00	00	00	00	00	00	00	00	00	00						• • •		• •																			
																																							- 1

Figura 5.3 Captura de Wireshark

Wireshark muestra los paquetes capturados, la dirección *IP* de origen, la dirección *IP* de destino, el protocolo utilizado e información adicional sobre el paquete, como el estado de la conexión.

Ahora, se crea una conversación con un cliente de mensajería instantánea, al hacer el seguimiento de *TCP* stream se puede observar toda la comunicación que ha habido entre los usuarios del servicio de mensajería, como se muestra en la figura 5.4, Follow *TCP* stream.

itream Content	
MSG kargoc@hotmail.com Karla 91 MIME-Version: 1.0	
Content-Type: text/x-msmsgscontrol TypingUser: kargoc@hotmail.com	
MSG 24 A 159 MTME-Version: 1 0	
Content-Type: text/plain; charset=UTF-8 User-Agent: pidgin/2.5.8	
<pre>X-MMS-IM-Format: FN=Segoe%20UI; EF=; CO=0; PF=0; RL=0 pop_ci_loc_dudocaCK_24</pre>	
MSG 25 U 95 MTME-Version: 1.0	
Content-Type: text/x-msmsgscontrol TypingUser: lsdpo_2005@hotmail.com	
MSG 26 A 145 MIME-Version: 1.0	
Content-Type: text/plain; charset=UTF-8 User-Agent: pidgin/2.5.8 X-MMS-TM-Format: FN=Segoe%20UT: FE=: CO=0: PE=0: RI=0	
:SACK 26	
MSG kargoc@hotmail.com Karla 91 MIME-Version: 1.0	
TypingUser: kargoc@hotmail.com	
MSG 27 U 95	
Content-Type: text/x-msmsgscontrol TypingUser: lsdpo_2005@hotmail.com	
MSG 28 A 149 MIME-Version: 1.0	
Content-Type: text/plain; charset=UTF-8 User-Agent: pidgin/2.5.8 V MME The Formati FM Second 2001; FE-1; CO-0; PE-0; PL-0	
<pre></pre>	
Eind Save As Print Entire conversation (43369 bytes)	💌 🔿 ASCII 🔘 EBCDIC 🔘 Hex Dump 🔘 C Arrays 🖲 Ra

Figura 5.4 Follow TCP stream

Se establece una conexión a la página de Snort, para ello se realiza un proceso denominado Three hand shake (figura 5.5 Three hand shake), en el cuál, la computadora cliente, dirección IP 192.168.2.10 envía un paquete SYN para comunicar que se desea establecer la comunicación, el servidor, en este caso el equipo con dirección IP 68.177.102.20 envía una respuesta SYN/ACK, indicando que recibió un paquete SYN. Finalmente el cliente envía un paquete ACK indicando que recibió el paquete SYN/ACK y de esta manera se establece la comunicación, después de esto se inicia la transmisión de la página web que se está solicitando (paquete número 32).

29 29.734433	192.168.2.10	68.177.102.20	TCP	51582 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSV=173
30 29.852735	68.177.102.20	192.168.2.10	TCP	http > 51582 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1360
31 29.852843	192.168.2.10	68.177.102.20	TCP	51582 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0
32 29.853156	192.168.2.10	68.177.102.20	HTTP	GET / HTTP/1.1

#### Figura 5.5 Three hand shake

#### **RESULTADOS:**

Wireshark ha detectado la actividad en la red correctamente, tanto la captura de paquetes como el seguimiento de *TCP* stream por lo que podemos afirmar que la herramienta tiene un funcionamiento adecuado y nos permite hacer un seguimiento adecuado de todas las actividades en la red y los protocolos utilizados. Además el analizador de protocolos también detectó la actividad de otros sitios que se abrieron posteriormente, por lo que se puede concluir como una prueba exitosa.

# **5.6 TURTLE** *FIREWALL*

Nombre:	Prueba de Turtle <i>Firewall</i> .
Descripción: Se ejecutará cierta	a actividad en la red interna para cerciorarnos de que
el Firewall permite el paso de	el tráfico estrictamente permitido de acuerdo con la

configuración realizada.

Actores: usuario con privilegios, usuario sin privilegios.

Precondiciones: El usuario con privilegios realizará la configuración del *Firewall* en un equipo con dos interfaces, en este caso eth0 será la conexión a nuestra red interna y eth1 la conexión al exterior. Además se tiene un servidor web que será protegido por el *firewall* y el mismo *firewall* que se considera como un equipo independiente. Se crearán reglas que permitan el tráfico http y ssh de la red interna hacia el exterior (Internet) y se bloqueará todo el tráfico restante.

Flujo normal: El usuario sin privilegios tratará de establecer conexiones con el exterior mediante protocolos o servicios permitidos y el *firewall* debe permitir dichas conexiones pero después de haber sido analizadas por las reglas establecidas.

Flujo alternativo: el *firewall* permite el tráfico que debería estar bloqueado.

Poscondiciones: se permite el paso del tráfico permitido y se realiza el registro de dicho tráfico.

Lo primero que se debe hacer es ejecutar un navegador web y mediante la herramienta Webmin hacer la configuración del *Firewall*, para ello se debe ingresar la siguiente dirección en el navegador: http://localhost:10000.

Ahora en el menú Networking, se hace click en Turtle *Firewall*, finalmente click en el vínculo "Firewall Items".

En este apartado se definen los elementos que estará protegiendo el *firewall*, el propio *firewall* se considera como un elemento independiente en la configuración.

En nuestro caso se define una zona llamada exterior asignada a la interfaz eth1 de la tarjeta de red, que comprende el tráfico que llega desde el exterior hacia el *firewall*, una zona denominada lan, asignada a la interfaz eth0 que representa el tráfico que sale de la red

interna hacia el *firewall* y una zona llamada servidor web, que corresponde a una máquina que brinda dicho servicio y que estará protegida por el *firewall(DMZ)*, figura 5.6, Elementos del *firewall*.

0		Webmin 1.500 on Stitch (F	Redhat Linux Fedora 9) - I	Mozilla Firefox 3 Beta 5		_ + ×
<u>A</u> rchivo <u>E</u> ditar <u>V</u> er Hig	<u>s</u> torial <u>M</u> arcadores Herramien <u>t</u> as	Ay <u>u</u> da				ं
🔹 🔹 👻 😂	ttp://localhost:10000/				😭 🗸 🚺 Google	٩
🛅 Marcadores rápidos 🗸 💧	🤧 Release Notes  💼 Fedora Projec	t 🗸 💼 Red Hat 🗸 💼 Free	e Content ∽			
Login:root Webmin System	Module Index		Fi	rewall Items		
Servers		lone	Inte	rface	Description	
Others	FIREWALL					
Networking	exterior		eth 1		conexion a internet	
Hardware	lan		eth0		conexion a la red interna	
Cluster	create new zone					
Search:	Net	Net address	Netmask	Zone	Description	
A	red interna	10.0.0.0	255.255.255.0	lan	red interna que protege el firewall	
System Information	create new net					
Sefresh Modules	Host	IP address	MAC address	Zone	Description	
Use Logout	servidorweb	192.168.2.6	00:21:85:02:41:82	lan	servidor web	
	create new host					
	G	roup	Ite	ms	Description	
	create new group					
	🖕 Return to turtle firew	all index				

Figura 5.6, Elementos del firewall.

Una vez definidos los elementos que serán protegidos por el *firewall*, se tienen que crear las reglas que definen el tipo de tráfico que puede atravesar por el *firewall* y las acciones que el mismo llevará a cabo con dicho tráfico, figura 5.7, Creación de reglas.

8	Webmin 1.	500 on Kakaroto (Redhat Linux Fedora 9) - Mozilla Fire	fox 3 Beta 5 _ +	×				
<u>A</u> rchivo <u>E</u> ditar <u>V</u> er Hi <u>s</u>	torial <u>M</u> arcadores Herramien	<u>t</u> as Ay <u>u</u> da		1.				
🔙 🔶 👻 😂 🙆	🚡 🚷 http://localhost.localdor	nain:10000/	😭 🗸 🕞 Google	2				
🚞 Marcadores rápid 🗸 👩	🕽 Release Notes  🗎 Fedora Proj	ect 🗸 💼 Red Hat 🗸 💼 Free Content 🗸						
Login: root Webmin System	Module Index	Module Index Create Rule						
Servers		Create Rule						
Others Networking Hardware Cluster Upused Modules	Source	FIREWALL Internet Ian modem						
Search: A View Module's Logs	Destination	FIREWALL internet lan modem						
Arfresh Modules	Service	<ul> <li>All Services</li> <li>sintp - Simple Mail Transfer Protocol simptrap - simptrap</li> <li>socks - Socks Proxy</li> <li>socksek std</li> <li>ssh- Secure Shell Protocol</li> <li>ssh2 - SSH over port 21</li> <li>tcp v</li> <li>Port</li> </ul>						
	Target	ACCEPT ~						
	Mark							
	Active	<b>⊠</b>	_					
	Description	el tráfico desde internet a la red local está permitido por se						
	create							
	📥 Return to Rules list							

Figura 5.7 Creación de reglas.

Para cuestiones de prueba se crean 7 reglas para el funcionamiento de este *firewall*, se aceptan los protocolos http y https tanto para el tráfico entrante como para el saliente, se

rechaza cualquier actividad FTP que salga de la red y se aceptan las conexiones hechas por medio de *Secure Shell*, figura 5.8, Reglas del *firewall*.

٢	Firewali Rules - Mozilia Firefox 3 Beta 5													
Arc	chivo <u>E</u> ditar <u>V</u>	er Hi <u>s</u> torial <u>M</u> arca	adores Herra	mien <u>t</u> as	Ay <u>u</u> da				0					
🔹 🗸 🔮 🔕 🖀 🚳 http://localhost:10000/turtlefirewall/list_rules.cgi														
	]] Marcadores rápidos ∨ 🚱 Release Notes 👚 Fedora Project ∨ 💼 Red Hat ∨ 💼 Free Content ∨													
Mo	Module Index Firewall Rules													
#	Source	Destination	Service	Port	Target	Mark	Active	Description						
1	exterior	lan	http		ACCEPT		YES	se acepta el trafico web desde el exterior hacia la red interna						
2	exterior	lan	https		ACCEPT		YES	se acepta el trafico web ssi desde el exterior hacia la red interna	* † †					
3	lan	exterior	http		ACCEPT		YES	se acepta el trafico web desde la red interna hacia el exterior	****					
4	lan	exterior	https		ACCEPT		YES	se acepta el trafico web ssl desde la red interna hacia el exterior	¥++±					
5	lan	exterior	ftp		REJECT		YES	se rechaza cualquier intento de salida con FTP	<b>₹</b> ††±					
6	lan	FIREWALL	ssh		ACCEPT		YES	se aceptan conexiones SSH de la red interna hacia el Firewall	*†‡					
7	FIREWALL	lan	ssh		ACCEPT		YES	se aceptan conexiones SSH del firewall hacia la red interna	<b>†</b> ±					
cre	ate new rule													

< Return to turtle firewall index

Figura 5.8, Reglas del firewall.

Para probar el funcionamiento del *firewall* se abre el navegador web y se hace el intento de salir a Internet desde la red interna.

Desde la red interna se tiene salida a Internet por lo que únicamente falta verificar los registros del *firewall* para determinar que dicho tráfico haya sido analizado por el *firewall* antes de permitir su paso, Figura 5.9, Registro del *firewall*, puerto 80.

Module Inde Module Con	fig			Turtle Firewall					
Using logfile	/var/log/messages			(1) > >>					
Betresh	DROP	• •	TUO	MAC V	5AC	Dist •	PROTO	SPORT	
Jun 7 12:56:41	142-FIRDALL	ethO		ff:ff:ff:ff:ff:ff:ff:00;13:20:15:7a:ea:08:00	192.168.2.9	192.168.2.255	UDP	138	138
Jun 7 12:56:41	IAA-FIREMALL	eth0		ff:ff:ff:ff:ff:ff:ff:00:13:20:10:7a:ea:08:00	192.148.2.9	192.168.2.255	UDP	137	137
Jun 7 12:56:42	lan-FIREMALL	ethO		ff:ff:ff:ff:ff:ff:ff:00:13:20:15:7a:ea:08:00	192.168.2.9	192.168.2.255	COP	137	137
Jun 7 12156143	100-FIRDMALL	ethD		ff:ff:ff:ff:ff:ff:ff:00:13:20:1b:7a:ea:08:00	192.168.2.9	192,168.2.255	UDP	137	137
Jub 7 12:56:46	Iss-FIREMALL	eth0		ffiffiffiffiffiffiffi0011312011b174104108100	192.168.2.9	192.168.2.255	UDP	13.8	13.8
Jun 7 12:156:153	FIREMALL-Inn		ethO		192.168.2.7	132.248.204.1	UDP	54193	53
Jun 7 12156153	FIRSMALL-lan		eth0		192.168.2.7	132.248.10.2	UEP	55377	5.3
Jun 7 12:56:53	FIRBMALL-lan	1	ethO		192.168.2.7	132.245.204.1	UDP	45432	5.3
Jun 7 12:56:53	FIRENALL-lan	-	etho		192.168.2.7	132.240.10.1	LEOP	41741	53
Jub 7	141-FIREMALL	ethO		00:21:85:02:41:76:00:13:72:3e:78:42:08:00	74.125.45.101	192.165.2.7	TCP	80	51
Jun 7 12:56:57	Ian-FIRBALL	ethO		00:21:85:02:41:76:00:13:72:3+:78:42:08:00	74,125,45,157	192.168.2.7	TCP	8.0	36256
12:50 T	lan-FIREMALL	eth0		00:21:85:02:41:76:00:13:72:3e:78:42:08:00	74,125,45.157	192,168.2.7	TCP	80	3621
Jun 7 12:56:58	FIRENALL-180	-	etho		192.168.2.7	132.245.204.3	out	43155	53

Figura 5.9, Registro del firewall, puerto 80.

En los registros se muestran cuatro conexiones en particular establecidas el 7 de junio desde varias direcciones *IP* entre ellas la 74.125.45.101 y la 74.125.45.157, a través de la interfaz eth0 hacia un host de nuestra red interna (192.168.2.7), con puerto de destino 80.

Ahora se verificará que la conexión por medio de SSH sea posible desde un host dentro de la red hacia el *firewall*, esto para poder llevar a cabo la administración del *firewall*.

Se realizó la conexión por medio de SSH con éxito y únicamente revisaremos que la actividad se haya registrado, figura 5.10, Registro del *firewall*, puerto 22.

Module Index Turtle						Firewall						
Using logfile	/vanlog/messages				<< < (ð) > >>							
DATE	OROP		IN	061	MAC	SRC		DST	PROTO	SPORT	DPORT	
Refresh	•	×	•	• •	•	•	~	•	· ·	• •	· •	
Jun 7 13:08:50	lan-F1REWALL		eth0		ff:ff:ff:ff:ff:ff:ff:00:l3:20:lb:7area:08:0	0 192.168.2.9		192.168.2.255	UDP	138	13.8	
Jun 7 13:08:50	lan-FIREMALL		eth0		ff:ff:ff:ff:ff:ff:00:13:20:1b:7a;ea:08:0	0 192,168,2,9		192.168.2.255	UDP	137	137	
Jun 7 13:08:51	lan-FIREMALL		eth0		ff:ff:ff:ff:ff:ff:ff:00:13:20:1b:76:ea:08:0	0 192.168.2.9	į.	192.168.2.255	UDP	137	137	
Jun 7 13:08:51	lan-FIREWALL		eth0		ff:ff:ff:ff:ff:ff:00:13:20:1b:7a:ea:08:0	0 192.168.2.9		192.168.2.255	UDP	137	137	
Jun 7 13:08:54	lan-FIRSWALL		eth0		ff:ff:ff:ff:ff:ff:00:13:20:1b:7a:ea:08:0	0 192.168.2.9		192.168.2.255	UDP	138	138	
Jun 7 13:09:42	FIREWALL-lon			ethD		192.168.2.7	5	132.248.204.1	UDP	53906	53	
Jun 7 13:10:41	FIREWALL-lon			eth0		192.168.2.7		132.248.204.1	UDP	52565	53	
Jun 7 13:11:20	lor PisceALL		eth0		00:21:85:02:41:76:00:21:85:02:41:82:08:0	0 192.168.2.6		192.168.2.7	TCF	12012	22	
Jun 7 13:11:23	lan-FIREMALL	Ì	eth0		00:21:85:02:41:76:00:21:85:02:41:82:08:0	0 192.168.2.6	ł	192.168.2.7	TCP	45043	22	
13:11:29	lan-FIREWALL	Ì	eth0		00:21:85:02:41:76:00:21:85:02:41:82:08:0	0 192.168.2.6	į.	192.168.2.7	TCP	45043	-	
Jun 7	FIREMALL-lan	Ť	_	CLOD		192 164 2.2	_	122.248.204.1	UDP	54212	53	

Figura 5.10 Registro del *firewall*, puerto 22.

Se registró la actividad de conexión al *firewall* por medio de SSH el 7 de junio a las 13:22 hrs, a través de la interfaz eth0, desde un equipo que pertenece a la red interna (ip 192.168.2.6) hacia el *firewall* (ip 192.168.2.7), con el protocolo *TCP* y con puerto de destino 22.

Toda la actividad de otros servicios, excepto web, fue bloqueada por el *firewall*, por lo que se considera que para la configuración creada, el *firewall* funcionó correctamente y el registro de las actividades en el *firewall* también se realizó en forma adecuada, la prueba por lo tanto fue exitosa.

# 5.7 EL SISTEMA DE DETECCIÓN DE INTRUSOS

Nombre:	Prueba del Sistema de Detección de Intrusos.
	i i ueba del Sistema de Delección de inti usos.

Descripción: Se crearán reglas de detección para Snort, se activará Barnyard para que interprete la actividad registrada en las bitácoras y con la consola de BASE se examinarán los resultados obtenidos.

Actores: Usuario con privilegios que activará Snort y Barnyard y el usuario sin privilegios que creará cierta actividad en el sistema que debe ser detectada por el Sistema de Detección de Intrusos.

Precondiciones: Se debe tener la base de datos funcionando, después se crearán un par de reglas para detectar si un usuario está visitando determinados sitios web y una regla para alertar sobre un escaneo de puertos mediante el comando nmap.

Flujo normal: El usuario sin privilegios visita los sitios web y realiza un escaneo de puertos con nmap, el SDI registra los sucesos y muestra las alertas al administrador, además de registrarlas en sus bitácoras. Finalmente muestra los sucesos mediante la consola de BASE.

Flujo alternativo: El SDI no detecta la actividad definida en las reglas de Snort.

Poscondiciones: Si el SDI detectó las actividades definidas en las reglas de Snort, concluye la prueba, de otra manera se deberá verificar si están mal escritas las reglas o si aun estando bien escritas Snort no las detectó como se esperaba.

Para probar el Sistema de Detección de Intrusos, se crearon dos reglas que permiten verificar si Snort detecta la actividad descrita en las mismas. Dos reglas alertarán sobre la visita de los usuarios a los sitios web www.google.com y www.youtube.com y la tercera alertará sobre un escaneo de puertos con nmap, esta última regla ya está hecha pero se debe activar para que funcione. Para ello, se edita el archivo local.rules que se encuentra en el directorio /etc/snort/rules y se agregan las siguientes líneas:

alert tcp any any -> any any (content:"http://www.youtube.com";msg:"alguien esta visitando youtube";sid:123124;) alert tcp any any -> any any (content:"www.google.com";msg:"alguien esta visitando google";sid:123123;) Una vez creadas las reglas, el administrador del SDI debe levantar el servicio Snort y Barnyard, después el usuario sin privilegios debe abrir un navegador de Internet y visitar los sitios descritos en las reglas, una vez hecho lo anterior, se abre una consola de comandos y se ejecuta el comando nmap al servidor local.

Una vez que se creó esta actividad en el sistema, el SID detectó la actividad e inmediatamente mostró los siguientes resultados (figura 5.11, Alertas Snort):

#### Figura 5.11, Alertas Snort

Como resultado se muestra la fecha en que se efectuó la actividad que generó la alerta, su identificador, el tipo de actividad y la dirección de origen y destino y puerto de origen y destino.

Por otro lado, BASE muestra el tráfico *TCP*, *UDP* y de exploración de puertos, figura 5.12, BASE.

- Alertas de hoy:         - Alertas de los últimos 24 horas:         - Alertas de los últimos 72 horas:         - Mas recientel 54 Alertas:         - Utimos puertos de origen:         - Ultimos puertos de destino:         - Puertos de origen más frecuentes:         - Puertos de destino:         - Puertos de destino:         - Puertos de destino:         - Puertos de restino:         - Mas frecuentes 15 Directones:         - Más reciente 15 Directas únicas         - Más reciente s Alertas únicas	Único Único Único cualquier protocolo cualquier protocolo cualquier protocolo cualquier protocolo cualquier protocolo Origen	lista lista lista TCP TCP TCP TCP TCP Destino	IP Origen IP Origen IP Origen UDP UDP UDP UDP UDP	IP Destino IP Destino IP Destino ICMP	Consultado en: Tue July 13, 2000 162:13 Base de datos: snort@Jockhoct (Version de esquema. 107 Ventana de tiempo: [2010.07.10.04.11.36] - [2010.07.43.15.20.45 Buiscar Hacer gráfica de datos de alerta Hacer gráfica del tiempo de detectar alertas
Sensores/Total: 1 / 1 Alertas únicas: 7 Categorias: 3 Júmero de Alertas en Total: 30	Perfil de Ti TCP (50%)	ráfico por	Protocolo		
	001 (20/0)				
Orig. direcciones IP: 9     Dest. direcciones IP: 10     Enlaces IP Únicas 17	ICMP (0%)				
<ul> <li>Orig. direcciones IP: 9</li> <li>Dest. direcciones IP: 10</li> <li>Enlaces IP Unicas 17</li> <li>Puertos de Origen: 3</li> </ul>	ICMP (0%)	reloración	le Ruertos (2	27.961	

#### Figura 5.12, BASE.

BASE muestra las alertas generadas por Snort, en este caso, el identificador de la alerta, la firma, la fecha y hora en que se generó la alerta, dirección origen y destino, así como el puerto y el protocolo utilizado, figura 5.13, Alertas BASE.

۲	Basic Ar	alysis and Secur	ity Engine	(BASE) : I	Lista de Alertas -	Mozilla Firefox		_ + ×
<u>A</u> rcl	hivo <u>E</u> ditar <u>V</u> er Hi <u>s</u> torial <u>M</u>	arcadores Herram	nien <u>t</u> as Ay <u>u</u>	da				÷
-	🔶 👻 😂 🖀 💽 hi	tp://localhost/base	-1.4.5/base_	stat_alerts	s.php	☆ <b>∨</b>	Google	2
<b>1</b>	1ás visitados 🗸 👩 Release Note	s 💼 Fedora Projec	t 🗸 💼 Red	Hat 🗸 💼	Free Content ~			
<u>ک</u>	Basic Analysis 🔀 💁 bil : Insta	illing 🔀 🚺 htt	:p://lain.php	) 🔀 🛗 M	buTube - Bro 🔀	🖙 Snort ::	🛛 📶 Port 5	6453 (t 🛛 📉
					Perfil	de tiempo de alertas		
			Mostrando	alertas 1-7	de 7 en total			
	< Firma >	< Clasificación >	< Total # >	Sensor #	< Dirección Origen >	< Dirección Dest >	< First >	< Ultimo >
	[snort] portscan: TCP Portsweep	o desclasificado	<b>11</b> (37%)	1	1	9	2010-07-10 04:12:13	2010-07-13 15:20:03
	[snort] Snort Alert [1:1000002:0]	desclasificado	4(13%)	1	1	1	2010-07-10 04:11:36	2010-07-10 04:11:37
	[snort] Snort Alert [1:123124:0]	desclasificado	<b>2</b> (7%)	1	2	1	2010-07-13 13:08:37	2010-07-13 15:03:31
	[snort] Snort Alert [1:123123:0]	desclasificado	5(17%)	1	2	1	2010-07-13 13:56:55	2010-07-13 15:18:01
	[snort] DNS SPOOF query response with TTL of 1 min. and n authority	bad-unknown o	4(13%)	1	1	1	2010-07-13 14:44:44	2010-07-13 14:47:46
	[arachnids] [snort] DELETED SCAN nmap fingerprint attempt	desclasificado	1(3%)	1	1	1	2010-07-13 14:46:02	2010-07-13 14:46:02
	[snort] ATTACK-RESPONSES 40 Forbidden	3 attempted-recon	<b>3(</b> 10%)	1	1	1	2010-07-13 15:20:37	2010-07-13 15:20:45
				ACCIÓN				
	{ acc	ión }	[		Selected	ALL on Screen		
м	Mantenimiento de Grupos de Alertas   Cache & Estado   Administración							

Figura 5.13, Alertas BASE.

Uno de los logs de Barnyard mostró la siguiente actividad:

09/29-15:01:36.045066 [\*\*] [1:123123:0] Snort Alert [1:123123:0] [\*\*] [Priority: 0] {TCP} 66.102.7.104:80 -> 192.168.1.64:49336

09/29-15:02:16.596023 [\*\*] [122:3:0] portscan: TCP Portsweep [\*\*] [Priority: 3] {PROTO:255} 192.168.1.64 -> 87.98.229.71

09/29-15:03:11.480726 [\*\*] [122:3:0] portscan: TCP Portsweep [\*\*] [Priority: 3] {PROTO:255} 192.168.1.64 -> 74.125.127.191

09/29-15:03:36.679786 [\*\*] [122:3:0] portscan: TCP Portsweep [\*\*] [Priority: 3] {PROTO:255} 192.168.1.64 -> 69.63.189.31

09/29-15:15:31.447242 [\*\*] [122:3:0] portscan: TCP Portsweep [\*\*] [Priority: 3] {PROTO:255} 192.168.1.64 -> 76.13.220.49

Las alertas TCP mostraron lo siguiente (figura 5.14, Alertas TCP):

Basic Analysis and Security Engine (BASE)	: Resultados de consulta	i - Mozilla Firefox	_ + ×				
<u>A</u> rchivo <u>E</u> ditar <u>V</u> er Historial <u>M</u> arcadores Herramien <u>t</u> as Ayuda			15 <sup>1</sup> 4 1614				
🔶 🗸 📓 🕒 💿 http://localhost/base-1.4.5/base_qry	main.php?new=1&layer4=TC	P#_res 🏫 🗸 💽 🗸 Google	٩				
🛅 Más visitados 🗸 🏼 🌈 Release Notes 👚 Fedora Project 🗸 👚 Red Hat 🗸 👚 Free Content 🗸							
🗑 Basic Analysis and Securit 💥 💿 Snort :: Error Code 500 🛛 🔀			~				
#30-(1-189430) [snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:40:26 74.12	5.19.190:80 192.168.1.64:54689	TCP				
#31-(1-189505) [snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:40:26 74.12	5.19.190:80 192.168.1.64:54689	TCP				
#32-(1-189429) [snort] Snort Alert [1:123123:0]	2010-07-13 15:40:01 74.125	.155.118:80 192.168.1.64:46526	TCP				
#33-(1-189504) [snort] Snort Alert [1:123123:0]	2010-07-13 15:40:01 74.125	.155.118:80 192.168.1.64:46526	TCP				
#34-(1-189501) [snort] Snort Alert [1:123123:0]	2010-07-13 15:38:45 74.12	5.19.147:80 192.168.1.64:43071	TCP				
#35-(1-189426) [snort] Snort Alert [1:123123:0]	2010-07-13 15:38:45 74.12	5.19.147:80 192.168.1.64:43071	TCP				
#36-(1-189425) [snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:20:45 69.16	.175.10:80 192.168.1.64:51976	TCP				
#37-(1-189500) [snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:20:45 69.16	.175.10:80 192.168.1.64:51976	TCP				
#38-(1-189424) [snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:20:41 69.16	.175.10:80 192.168.1.64:51975	TCP				
#39-(1-189499) [snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:20:41 69.16	.175.10:80 192.168.1.64:51975	TCP				
#40-(1-189423) [snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:20:37 69.16	.175.10:80 192.168.1.64:51974	TCP				
#41-(1-189498)[snort] ATTACK-RESPONSES 403 Forbidden	2010-07-13 15:20:37 69.16	.175.10:80 192.168.1.64:51974	TCP				
#42-(1-189496)[snort] Snort Alert [1:123123:0]	2010-07-13 15:18:01 74.125	5.19.147:80 192.168.1.64:38290	TCP				
#43-(1-189421) [snort] Snort Alert [1:123123:0]	2010-07-13 15:18:01 74.12	5.19.147:80 192.168.1.64:38290	TCP				
#44-(1-189420) [snort] Snort Alert [1:123124:0]	2010-07-13 15:03:31 74.125	.155.190:80 192.168.1.64:34105	TCP				
#45-(1-189495) [snort] Snort Alert [1:123124:0]	2010-07-13 15:03:31 74.125	.155.190:80 192.168.1.64:34105	TCP				
#46-(1-189488) [arachnids] [snort] DELETED SCAN nmap fingerprint attempt	2010-07-13 14:46:02 74.85.	65.214:1935 192.168.1.64:38093	TCP				
#47-(1-189413) [arachnids] [snort] DELETED SCAN nmap fingerprint attempt	2010-07-13 14:46:02 74.85.	65.214:1935 192.168.1.64:38093	TCP				

Figura 5.14, Alertas TCP

BASE muestra varias alertas "Atack response", un intento de acceso a un objeto no autorizado en el servidor, alertas creadas con el SID 123123 que son las generadas por una de las reglas creadas, y alertas generadas por un escaneo de puertos con nmap.

Snort brinda información sobre las alertas incluidas en la configuración por default, como se muestra en la figura 5.15 Snort, alertas en web:



5.15 Snort, alertas en web.

#### **RESULTADOS:**

En esta prueba Snort detectó la actividad creada por el usuario y alertó adecuadamente sobre dicho suceso, por una parte Barnyard en modo texto y por otra BASE en modo gráfico mostraron información sobre las páginas visitadas como su dirección ip, los puertos utilizados y la hora exacta en que se visitaron, también se detectó el escaneo de puertos con nmap y un barrido de puertos al servidor de fedora. Esta información es de vital importancia para un administrador de seguridad ya que le permite tomar decisiones sobre eventos casi de manera instantánea.

Como se observó en esta prueba, muchas reglas han sido creadas para trabajar con Snort desde su instalación, en este caso, el escaneo de puertos con nmap fue detectado por una regla prediseñada que se cargó en el momento de la instalación de Snort. Hay vulnerabilidades para las que aun no existen reglas, el administrador debe ser muy cuidadoso y crear reglas que protejan al sistema, como las dos reglas creadas anteriormente, para ello se debe estar al tanto de las últimas noticias en el plano de la seguridad.

Debido a lo anteriormente descrito, se considera que el SDI funciona adecuadamente y que puede detectar firmas de amenazas conocidas y generadas por el administrador de seguridad y alertar oportunamente sobre estos eventos con toda la información necesaria y suficiente para la toma de decisiones de seguridad, por lo tanto se considera que la prueba tuvo éxito.

En la siguiente tabla 5.1, se muestran las pruebas realizadas y los elementos del sistema de defensa que fueron probados:

Tabla 5.1, Pruebas realizadas					
Nombre de la prueba	Elementos del sistema analizado				
John the Ripper	Contraseñas				
Nessus	Vulnerabilidades del sistema operativo y la red				
Nikto	Vulnerabilidades web				
Wireshark	Tráfico de la red				
Turtle Firewall	Firewall				
Sistema de Detección de Intrusos	Firmas y reglas de Snort				