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 de Ripper para encontrar contraseñas débiles que pueden ser una vulnerabilidad 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 herramienta la descubra.	
Flujo normal: El usuario con privilegios ejecutará John the Ripper con un archivo de lista de palabras, la herramienta obtendrá las contraseñas inseguras o débiles.	
Flujo alternativo: la herramienta no detecta contraseñas inseguras y continúa con su ejecución hasta que el usuario con privilegios decide detener la prueba.	
Poscondiciones: se detecta la contraseña 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.
11.801 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
(6VdinK4PM$ZZlLNW76hm0oWfkmU9W4I3xDpbw9Ou31s7D2FidFQpilvQUEF44HfuOzI47CykLWy5I2fuyG2k6.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 privilegios ejecutará Nessus y este programa detectará las vulnerabilidades por falta de actualización de software.	
Flujo alternativo: la herramienta no detecta ninguna de las vulnerabilidades o sólo detecta algunas de las posibles vulnerabilidades.	
Poscondiciones: se detectan las vulnerabilidades del sistema y se crea el informe correspondiente.	

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.

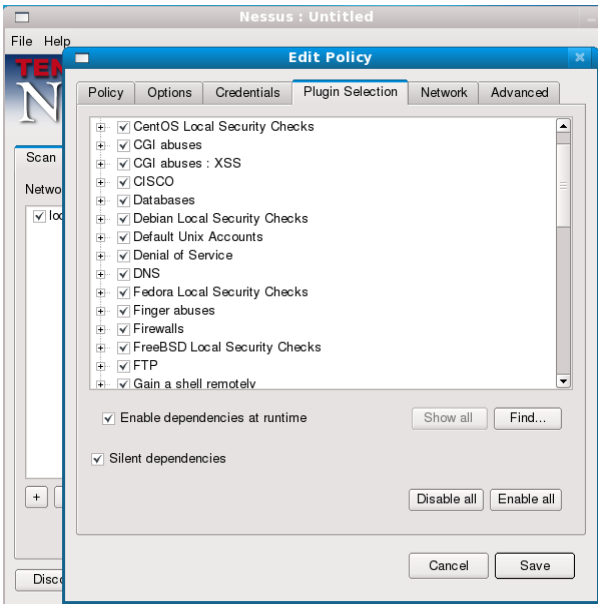


Figura 5.1, Plugins de Nessus.

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.

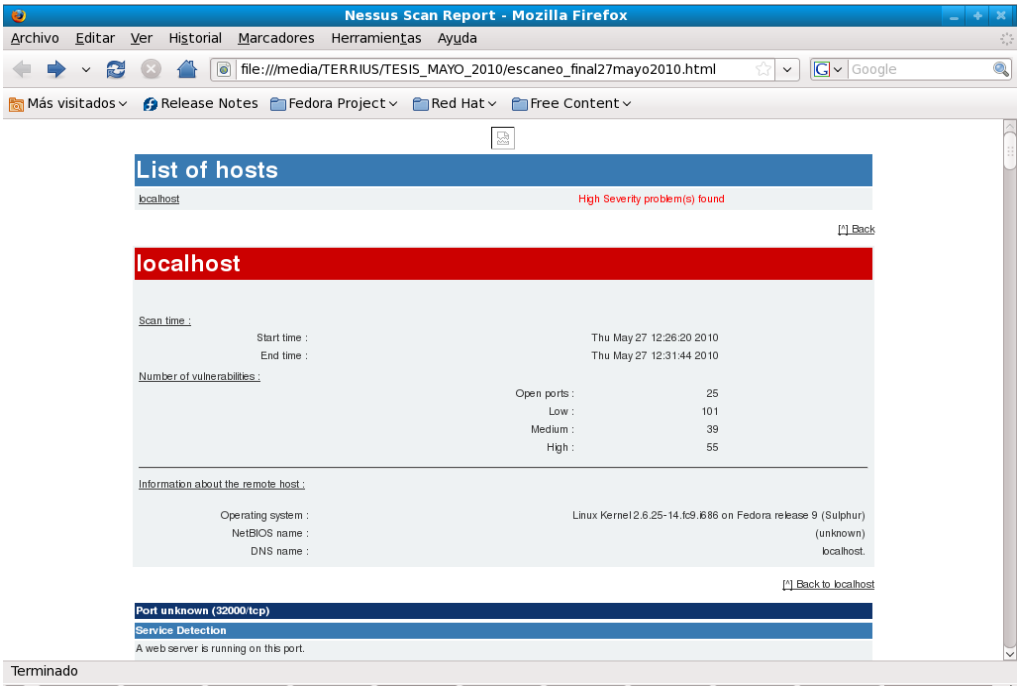


Figura 5.2, Resultados de Nessus.

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 programa Nikto en el servidor web con el puerto a verificar, en este caso el puerto 80.	
Actores: usuario con privilegios.	
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 privilegios ejecutará Nikto y la herramienta detectará las vulnerabilidades existentes en el servidor web.	
Flujo alternativo: la herramienta no detecta ninguna vulnerabilidad.	
Poscondiciones: se detectan vulnerabilidades 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 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-0: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ 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)

---------------------------------------------------------------------------
```

- 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 Wireshark y después se creará cierta actividad en la red que pueda ser detectada, en este caso, se establecerá una comunicación a través de mensajería instantánea para hacer el seguimiento TCP stream y se abrirá un sitio web.	
Actores: usuario con privilegios.	
Precondiciones: Se debe crear cierta actividad en la red que sea monitoreada por un usuario privilegiado que ejecute Wireshark.	
Flujo normal: El usuario con privilegios ejecutará Wireshark y la herramienta detectará la actividad en la red.	
Flujo alternativo: la herramienta no detecta la actividad creada.	
Poscondiciones: se capturan los paquetes de la actividad creada para posterior análisis.	

El usuario con privilegios debe loguearse y ejecutar Wireshark, para ello se deben teclear los siguientes comandos:

[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.

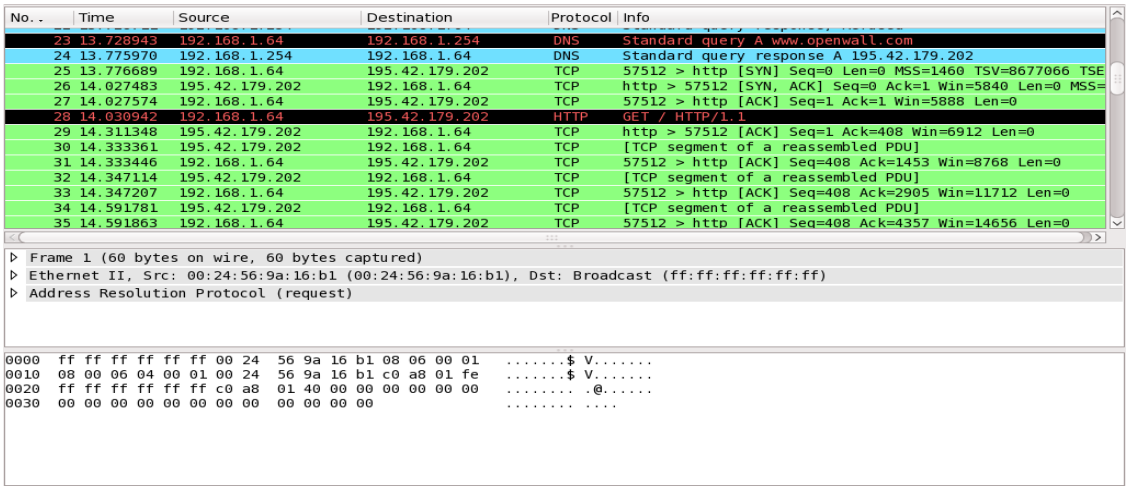


Figura 5.3 Captura de Wireshark

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.

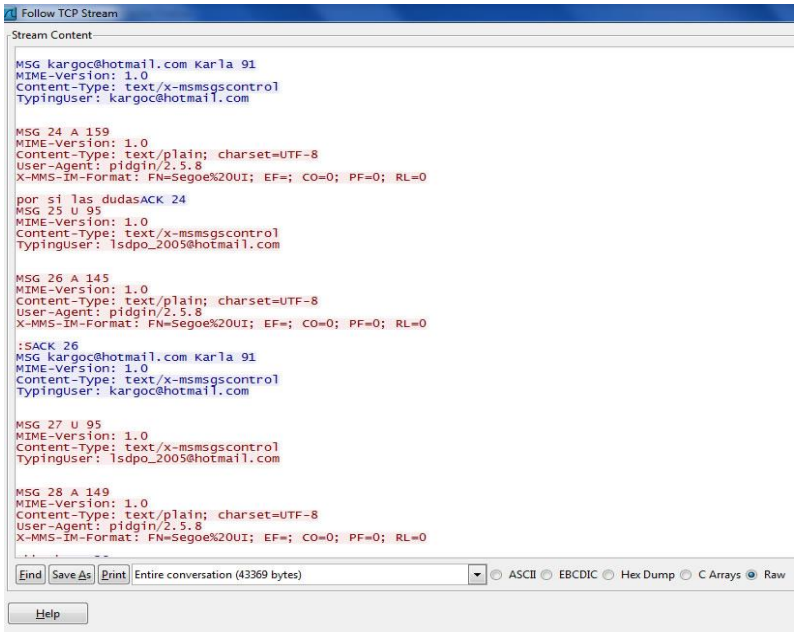


Figura 5.4 Follow TCP stream

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 *Firewall*.**
Descripción: Se ejecutará cierta actividad en la red interna para cerciorarnos de que el *Firewall* permite el paso del 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*.

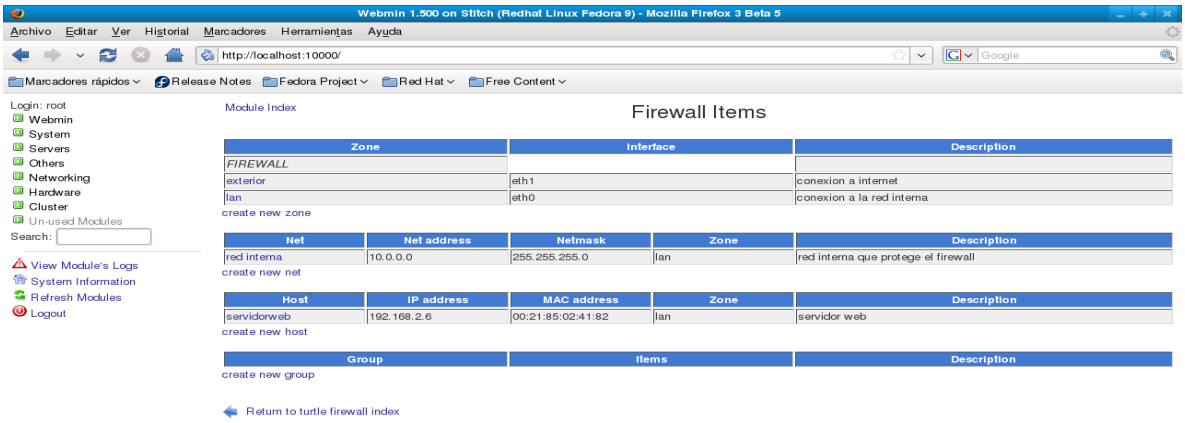


Figura 5.6, Elementos del *firewall*.

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.

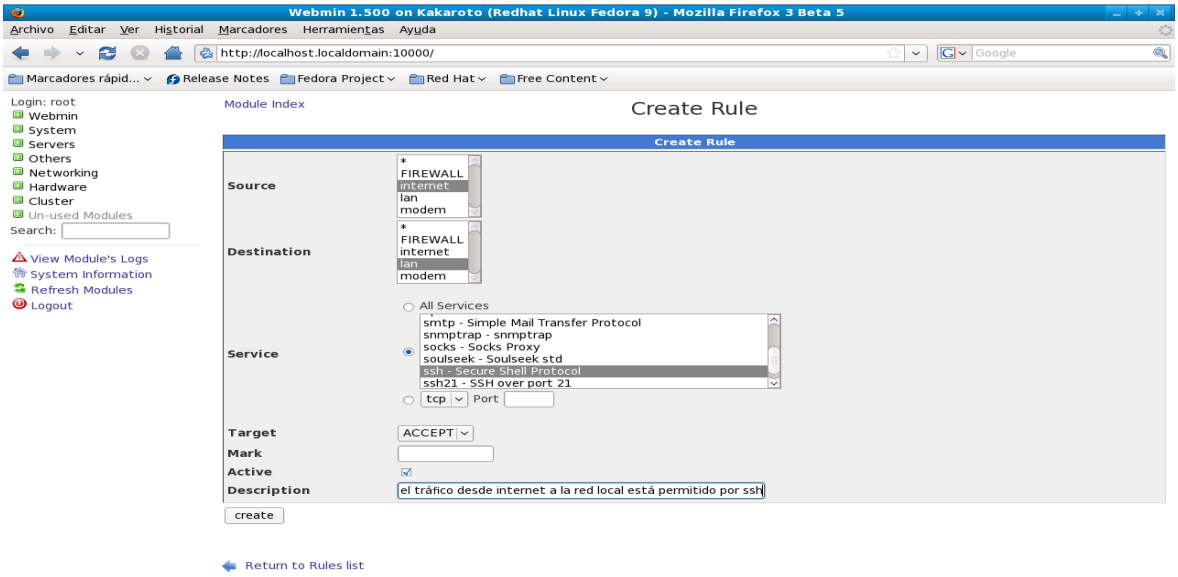


Figura 5.7 Creación de reglas.

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*.

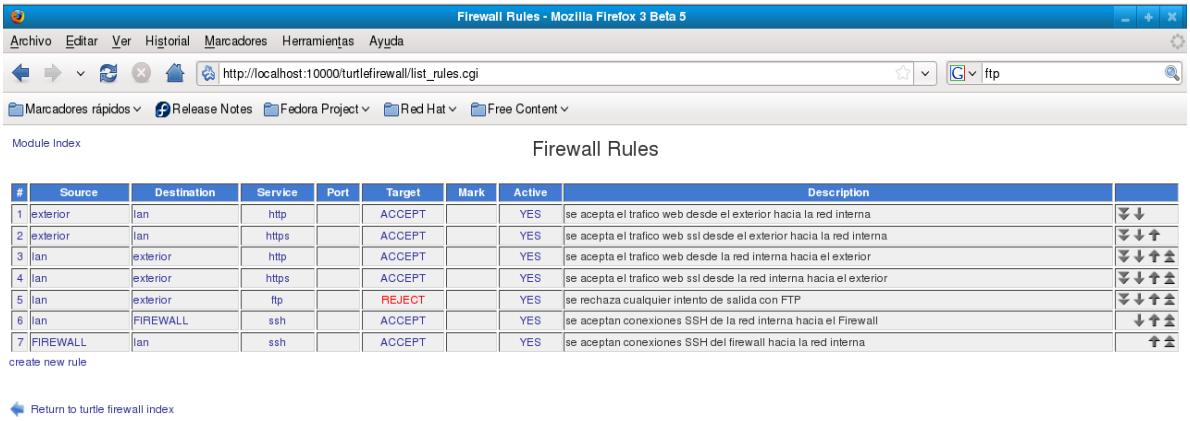


Figura 5.8, Reglas del *firewall*.

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.

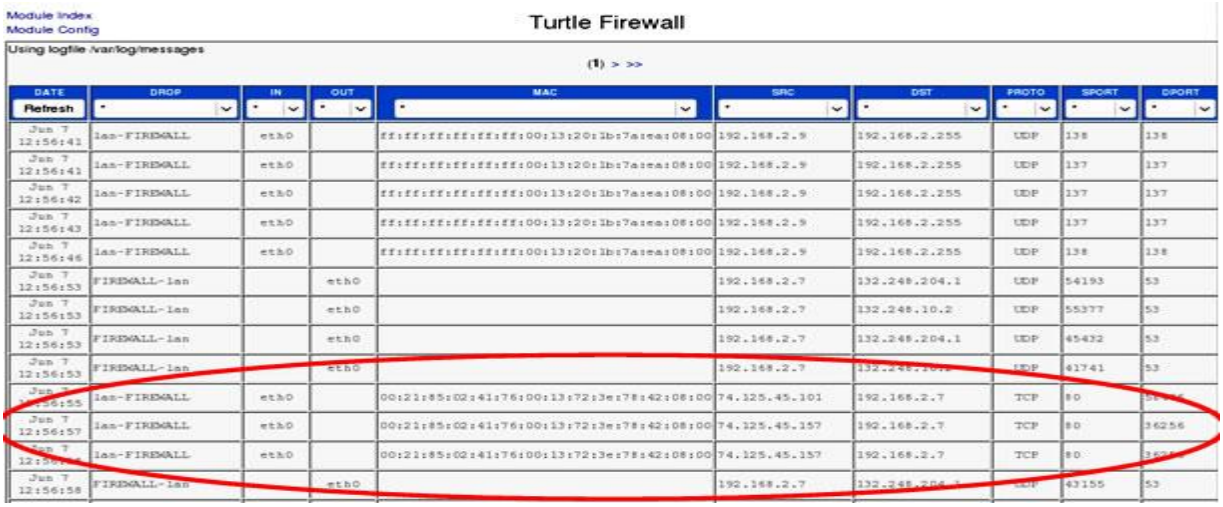


Figura 5.9, Registro del *firewall*, puerto 80.

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.

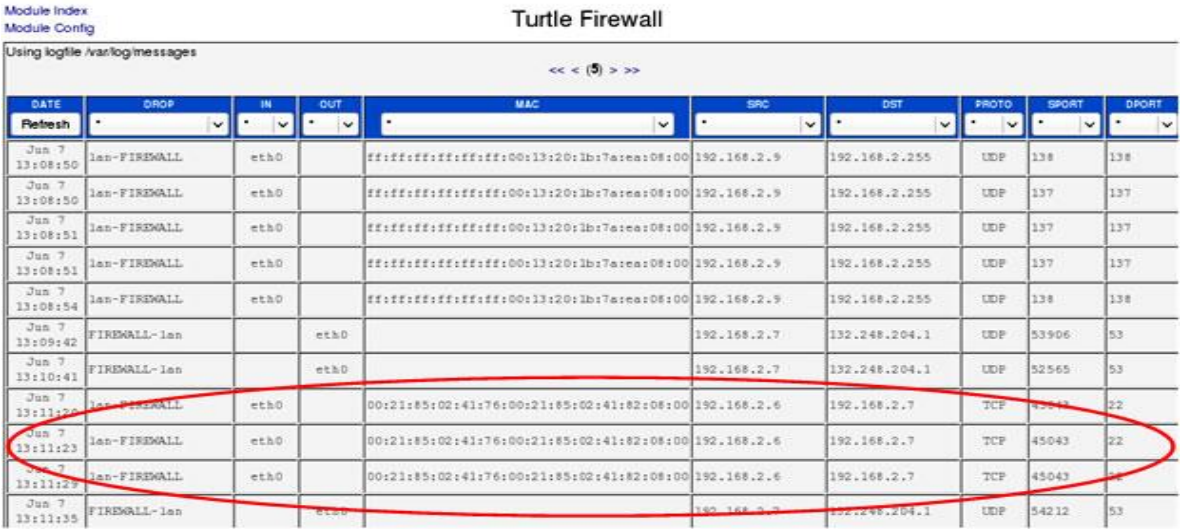


Figura 5.10 Registro del *firewall*, puerto 22.

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.**
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):

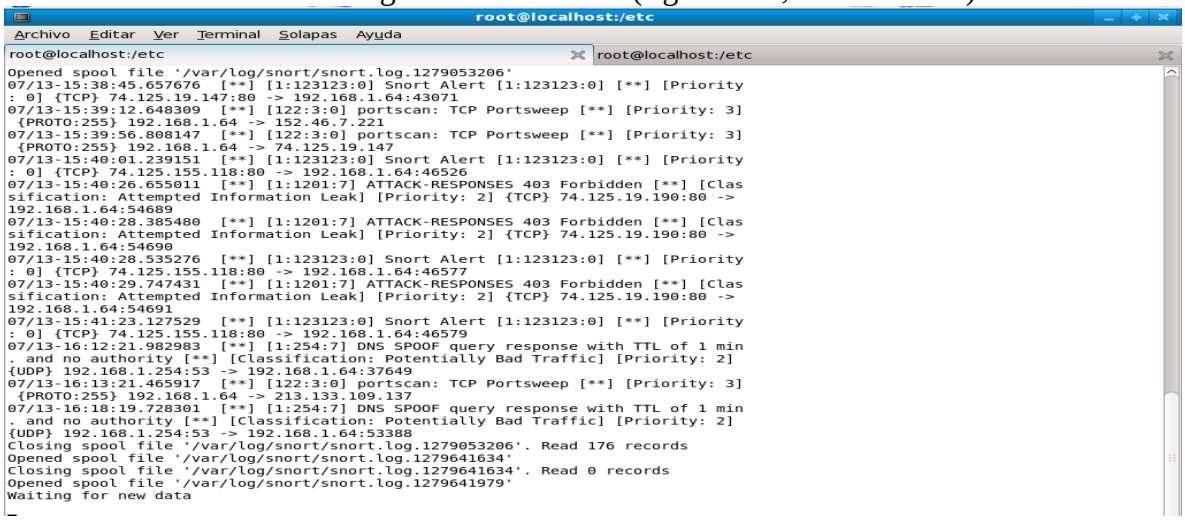


Figura 5.11, Alertas Snort

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.

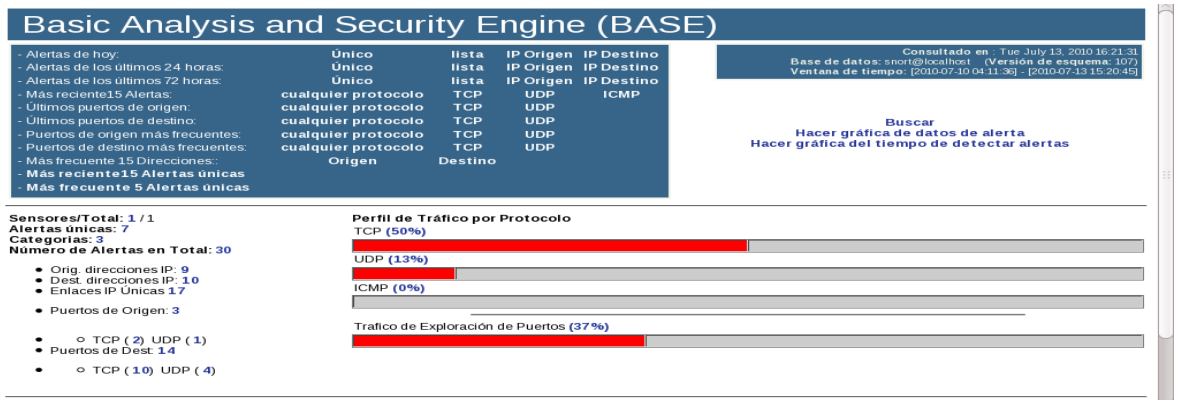


Figura 5.12, BASE.

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.

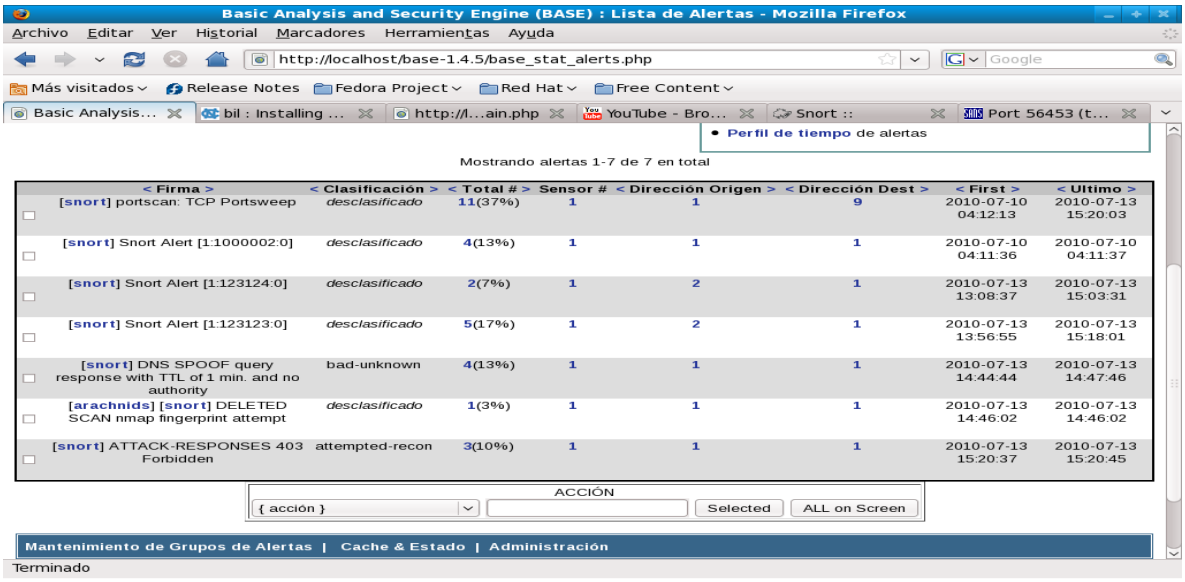


Figura 5.13, Alertas BASE.

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):

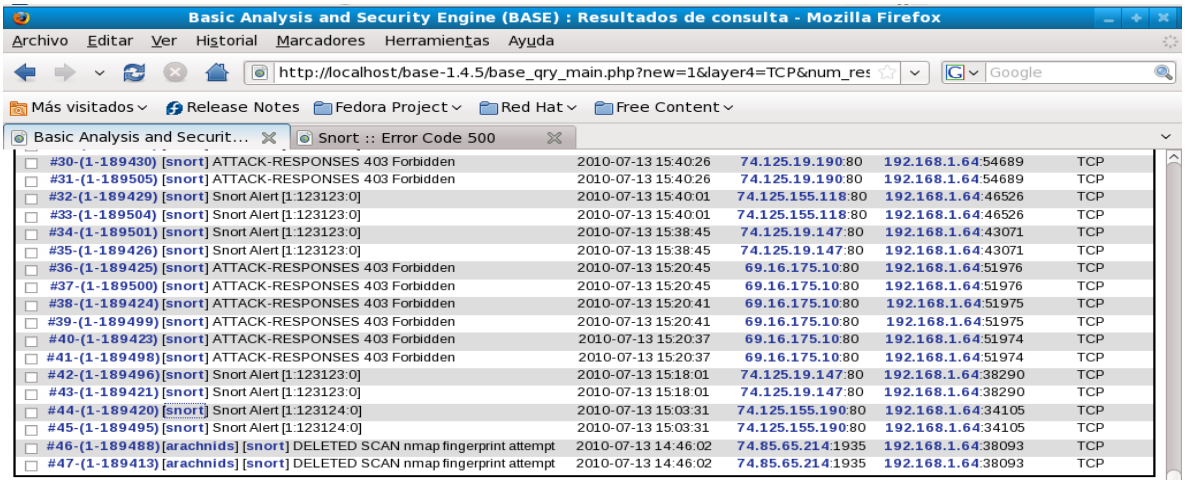
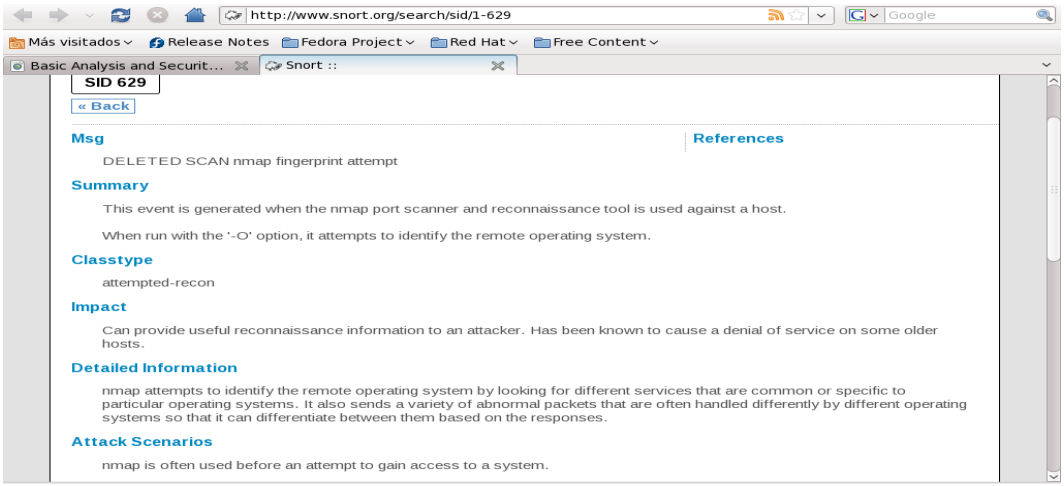


Figura 5.14, Alertas TCP

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