METODOLOGIA: GENERACION DE ATAQUES / DETECCIÓN CON NIDS (Nivel de Inmadurez de los NIDS {segunda parte})

Por: Alejandro Corletti (acorletti@wanadoo.es - acorletti@hotmail.com)
José Ignacio Bravo Vicente (jseigbv@yahoo.com)

1. Presentación:

El presente trabajo es la continuación natural del publicado anteriormente denominado ***"Nivel de Inmadurez de los NIDS",*** que se encuentra en varios sitios de Internet.
En el mismo se realizó la evaluación de algunos productos de detección de intrusiones y luego de una serie de mediciones y comparativas, se obtuvieron las siguientes conclusiones:

a. Disparidad en la detección de un mismo evento por distintos productos:
b. Ausencia de detección del no cumplimiento a lo establecido por las RFCs.
c. Faltas de desarrollos en el relevamiento del software y hardware de red.
d. Faltas de iniciativas sobre trabajo en reglas "Proactivas".
e. Estudiar muy en detalle qué IDS se ajusta mejor a la red de la empresa.

Luego de estos hechos se continuó avanzando sobre lo estudiado y se trató de idear una metodología de trabajo que permita mejorar estos aspectos, y poder optimizar la detección de intrusiones. Relacionado a cada conclusión se puede describir lo siguiente:

- Al punto a. (Disparidad en la detección de un mismo evento por distintos productos):

  Se optó por emplear dos tecnologías diferentes en cada zona y evaluar las respuestas de ambas.

- Al punto b. (Ausencia de detección del no cumplimiento a lo establecido por las RFCs):

  Se hizo evidente una notable mejoría en la respuesta a este punto a lo largo de las sucesivas actualizaciones, en particular con Snort y los protocolos más importantes de la familia TCP/IP, en concreto TCP, IP e ICMP.

- Al punto c. (Falta de desarrollos en el relevamiento del software y hardware de red.

  Este es uno de los puntos que más se analizó y dio como resultado gran parte de esta nueva metodología que se propone aquí.

- Al punto d. (Faltas de iniciativas sobre trabajo en reglas "Proactivas"):

  Se comenzó a trabajar en este punto, desarrollando reglas que se ajusten a la propia red donde se instalen los sensores, de forma tal que permita evaluar el tráfico cotidiano y lo anómalo.

- Al punto e. (Estudiar muy en detalle qué IDS se ajusta mejor a la red de la empresa):

  Este es un punto que se mantiene bastante claro y que continua respondiendo a la capacidad del personal de red que se posea, a su inclinación a ciertas líneas de productos, al nivel de desarrollo en entornos GNU, a los recursos materiales, a la magnitud de la red, el soporte técnico, el grado de exposición e impacto de sus recursos, etc.

Los ítems de las conclusiones que pueden realmente aportar novedades en este artículo, van muy relacionados al avance que se produjo sobre los puntos “c”, “d” y “e”, que es donde se centra esta propuesta.

2. Introducción:

Como continuación de la evaluación anterior, se propuso el desafío de ingeniar una metodología de trabajo, que permita “Madurar”, estas falencias detectadas, pues sí se creía importante implementar la tecnología NIDS, ya era evidente que se trataba de un eslabón fundamental en la cadena de la seguridad, realidad de la que hoy son conscientes la masa de los administradores serios de sistemas, y que sin duda es casi indispensable, pues se repite una vez más, que los NIDS no compiten con los Firewalls ni con los routers, sino que colaboran como un dispositivo más en el plan de seguridad.

Ante este desafío, surgieron una serie de ideas y trabajos, de los cuales algo quedó en el camino, y otros se fueron encadenando permitiendo llegar a la combinación de dos productos que se aprecia son de muy buena calidad (para no entrar en discusión si son los mejores o no en sus rubros), ambos GNU. Se trata de **Nessus** como detector de vulnerabilidades (o generador de ataques) y **Snort** como Netwok Intrusion detection System (NIDS), empleando este último con todo el conjunto de módulos adicionales que permiten desarrollar cualquier función o servicio igual o mejor que cualquier otro producto comercial.

3. Metodología:

Al hacerse evidente que ningún NIDS detectaba la totalidad de los ataques que se realizan hacia una red, se propuso la idea de determinar cuáles eran detectados y cuáles no. Para esta tarea se evaluaron las distintas herramientas que permiten detectar vulnerabilidades en sistemas. Se trabajó con varias, y nuevamente otra propuesta GNU (Nessus) quedó excelentemente posicionada, una vez más no se entrará en la discusión si es la mejor o no, pero sí se afirma que es muy buena y sus reglas son las que se actualizan con más frecuencia en el mercado.

Al empezar a detectar vulnerabilidades con Nessus, en los sistemas propios desde Internet hacia las zonas desmilitarizadas (DMZ), se comenzó a tabular las vulnerabilidades con los eventos detectados por Snort. Aquí aparece el primer problema (ya detectado en el trabajo anterior), pues no es fácil relacionar un ataque de Nessus con un evento detectado por Snort, esto se debe a que ambos tienen diferente codificación y denominación del evento. Esta relación sólo se puede realizar cuando el mismo posee referencia hacia CVE o Bugtraq, hecho que aparece en un muy bajo porcentaje de reglas, tanto de Nessus como de Snort, y aún así existen ataques que responden a una misma CVE y eventos detectados por Snort en los que sucede lo mismo, por lo tanto, ni siquiera en estos casos, se trata de una relación unívoca entre un ataque y un evento detectado por el sensor.

Para comenzar a relacionar estos dos hechos se trabajó de la siguiente forma:

a. Se aisló en laboratorio una pequeña red.
b. Se realizó un scan con Nessus.
c. Se estudiaron las vulnerabilidades detectadas.
d. Se analizaron las reglas de Nessus, (las cuáles se realizan mediante el lenguaje NASL, propuesto por este software, con el que existe una regla por cada ataque).
e. Se llegaba hasta la regla que generaba cada uno de los ataques detectados en el Laboratorio.

f. Se generaba únicamente este ataque, teniendo en cuenta aquí que Nessus, la mayoría de los ataques, solo los lanza si encuentra ese puerto abierto, es decir, si se trata por ejemplo de un ataque para detectar una vulnerabilidad en un servidor de correo, primero intentará establecer una conexión con el puerto TCP 25, y si esta no se establece, entonces no lanza el resto del ataque (que es lo que interesa capturar en este análisis), sólo lanzará la totalidad del ataque programado por su regla correspondiente (xxxx.nasl) si se cumple esta primera condición. Esta táctica la emplea para mejorar su rendimiento, pues no tendría sentido seguir generando tráfico sobre un puerto inexistente y por lo tanto hacia un servicio que no se está prestando. Por lo expresado entonces, se tuvo que instalar cada uno de los servicios que se deseaba evaluar.
g. Se capturaba la totalidad del ataque con un analizador de protocolos.
h. Se evaluaba si Snort lo detectaba o no.

Hasta aquí fue la tarea de tabulación e individualización de cada una de las vulnerabilidades presentes en este laboratorio, dejando en una hoja de cálculo qué ID de Nessus se correspondía con cuál ID de Snort (junto con otros datos adicionales). Al finalizar la misma, se puso de manifiesto la posibilidad de seguir adelante comenzando a crear las reglas de Snort que permitan detectar aquellas que este NIDS "no veía", y así siguió este trabajo.

Un comentario adicional se debe realizar aquí, pues cualquiera puede plantearse el tema de las vulnerabilidades existentes en una red, bajo la idea que si existe una vulnerabilidad, entonces hay que solucionarla, dejando ese servicio, host, puerto o sistema asegurado respecto a este evento. En realidad esto no es tan fácil de realizar pues en muchos casos simplemente NO SE PUEDE, pues hay muchas causas que no permiten hacerlo, por ejemplo:

- Parches no existentes.
- Sistemas que no lo permiten.
- Aplicaciones propietarias que al modificar puertos o protocolos dejan de funcionar.
- Software enlatado que no se puede tocar.
- Servicios que fueron siendo modificados a lo largo de los años, y se hace muy peligroso de parchear.
- Sistemas que al ser bastionados dejan de funcionar, y no se está muy seguro de por qué.
- Servicios que no pueden dejar de prestarse.
- Accesos que deben ser abiertos sí o sí.
- Políticas empresariales, que no permiten asegurar esa vulnerabilidad.
- Dependencias de sistemas ajenos al organismo de seguridad.
- Etc, etc, etc, ...

Como conclusión a este comentario adicional entonces, se puede decir (y así nos ha sucedido en muchos casos), que si existe una vulnerabilidad detectada, la secuencia lógica es:

a. Ser consciente que se puede detectar o no.
b. Si es posible, solucionarla (No siempre se podrá).
c. Si no se puede solucionar, asegurar que **sí o sí** será detectada por los sensores.
d. Al ser detectada en la red, entonces **se trata de una alerta crítica** (Pues se sabe fehacientemente que se es vulnerable).
e. Generar una alarma en línea.

Basado en el manual de Snort se continuó realizando las reglas que permitieran detectar los ataques, tomando como referencia la regla de Nessus correspondiente (xxxx.nasl) y lo capturado con el analizador de protocolos. Cada nueva regla creada, se alojaba dentro de las local.rules de Snort, que están pensadas justamente para esta actividad y son tenidas en cuenta cada vez que se actualiza el conjunto de rules de Snort, pues las local.rules no son tocadas.

Se fueron generando nuevas reglas, teniendo como detalle adicional el incluir {nessus} dentro del mensaje de cada regla, para identificar posteriormente que se trataba de una detección procedente de un evento generado por este software.

El resultado final fue que se contaba con un NIDS que detectaba la totalidad de las vulnerabilidades presentes en este laboratorio. Este punto es de suma importancia, pues si bien con este aporte no se está totalmente seguro del 100 % de detección, pues siempre existen y existirán ataques que no son contemplados por Nessus, se pudo verificar que cubre un altísimo porcentaje de anomalías, y a su vez, también se hizo evidente la velocidad de actualización de plugins por parte de Nessus en cuanto aparece una nueva vulnerabilidad y/o exploit en Internet, cosa que no sucedía con otros productos. Este aspecto proporcionaba un avance significativo a lo estudiado con anterioridad y que dio origen al artículo *"Nivel de Inmadurez de los NIDS"*, pues ahora se puede trabajar con sensores que se ajusten a la red en particular y garanticen una alta confiabilidad en la detección de eventos.

Al llegar aquí en el laboratorio, apareció una nueva forma de trabajar con NIDS.

¿Por qué no realizar la misma tarea en la red en producción?, mezclando la tecnología Nessus – Snort, para incrementar el nivel de seguridad del sistema. Es decir, en este punto se contaba con una metodología de trabajo que permitía:

- Detectar vulnerabilidades reales en la propia red.
- Verificar si nuestros sensores las reconocían en los patrones de tráfico.
- Generar las reglas necesarias.
- De hacerse presente un ataque de este tipo en la red, se trata de un caso crítico, pues ya se sabe que se es vulnerable en ese punto.

4. Mediciones:

Se presenta a continuación ejemplos de este trabajo:

a. La tabla que se presenta a continuación, es un resumen de la que se emplea para codificar (Asociar) eventos entre Nessus y Snort. En la misma se contemplan una serie de columnas que son las que proporcionan la información suficiente para aplicar esta metodología, que en definitiva el resultado final es obtener los Id de Nessus y de Snort en una misma línea, que son los datos que después permitirán la detección de un ataque que se sabe que en la red existe como vulnerabilidad.

   La descripción de los campos es:

   - Id Plugin (Nessus): Número que identifica unívocamente a ese ataque.
   - Name (Nessus): Nombre que aparece en la pantalla del cliente de Nessus.
   - Risk (Nessus): Riesgo que le asigna Nessus (no necesariamente en la red que se está analizando tendrá el mismo valor o impacto).
   - Nasl (Nessus): Regla específica que lanza ese ataque.
   - Summary (Nessus): Descripción breve del ataque.

- Family (Nessus): Familia dentro de la cual esta asignado este ataque en la pantalla de cliente Nessus. Este dato se emplea para luego buscar ese ataque específico en la consola y poder filtrar para generar únicamente este y no otro.
- Snort name (Snort): Nombre con que se visualizará este evento en formato Log.
- Id_snort (Snort): Número que identifica unívocamente ese evento. El mismo, puede existir ya en las rules (Número menor a 1.000.000) o se trata de una regla generada para esta red, la cual está incluida en el conjunto de las local.rules (número mayor que 1.000.000).

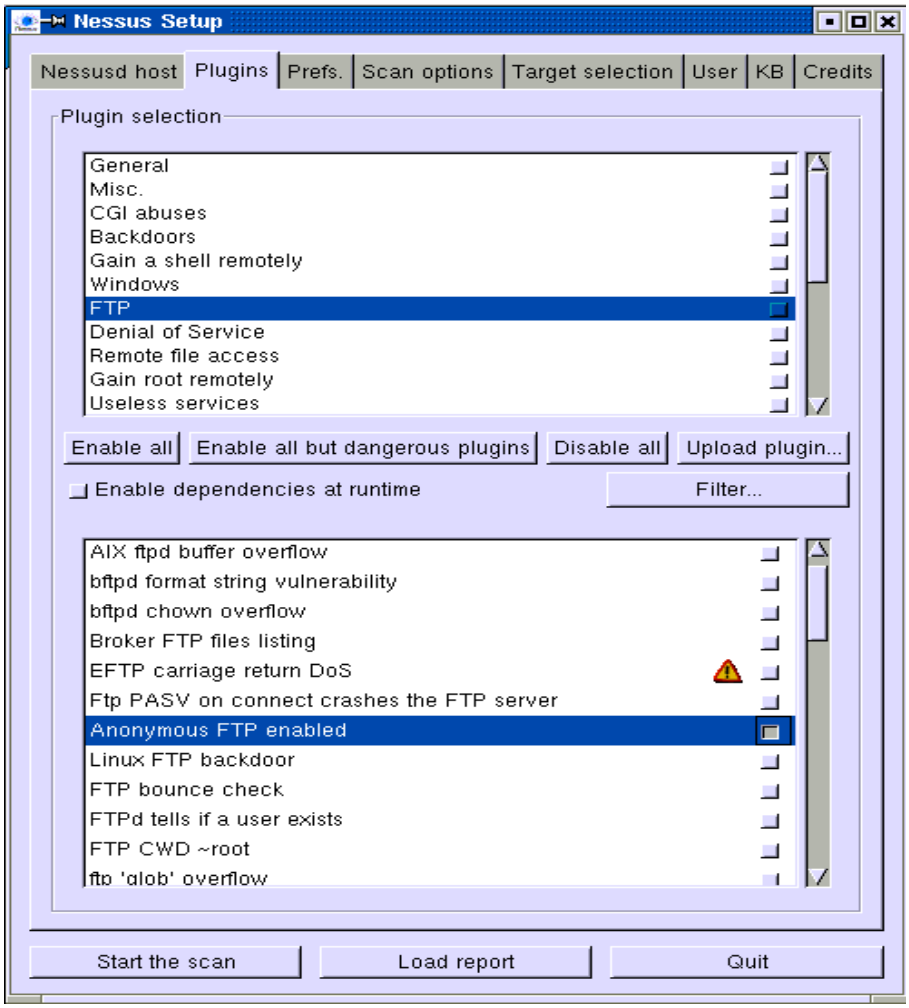
NESSUS						**SNORT**		**AMBOS**
id plugin	**Name**	**risk**	**nasl**	**summary**	**family**	**Snort name**	**Id_snort**	**referencias**
10012	Alibaba 2.0 buffer overflow	High	alibaba_overflow.nasl	Alibaba buffer overflow	Gain root remotely		1001019	CAN-2000-0626
10019	Ascend Kill	High	ascend_kill.nasl	Crashes an ascend router	Denial of Service	DOS Ascend Route	281	
10022	Axent Raptors DoS	High	axent_raptor_dos.nasl	Crashes an axent raptor	Denial of Service		1001021	CVE-1999-0905
10077	Microsoft Frontpage exploits	High	frontpage.nasl:	Checks for the presence of Microsoft Frontpage extensions	CGI abuses	WEB-FRONTPAGE _vti_rpc access	937	/www.securityfocus.com/bid/2144
10079	Anonymous FTP enabled	Low	ftp_anonymous.nasl	Checks if the remote ftp server accepts anonymous logins	FTP	Anonymous FTP enabled {nessus}	1001001	CAN-1999-0497
10080	Linux FTP backdoor	High	ftp_backdoor.nasl	Checks for the NULL ftpd backdoor	FTP		1001015	CAN-1999-0452
10088	Writeable FTP root	Serious	ftp_root.nasl	Attempts to write on the remote root dir	FTP		1001002 - 1001003	CAN-1999-0527
10096	rsh with null username	High	rsh_null.nasl	attempts to log in using rsh	Gain a shell remotely		1001022	CVE-1999-0180
10107	HTTP Server type and version	Low	http_version.nasl	HTTP Server type and version	General	WEB-IIS script access	1287	
10119	NT IIS Malformed HTTP Request Header DoS Vulnerability	High	iis_malformed_request.nasl	Performs a denial of service against IIS	Denial of Service		1001023	CVE-1999-0867
10150	Using NetBIOS to retrieve information from a Windows host	Medium	netbios_name_get.nasl	Using NetBIOS to retrieve information from a Windows host	Windows		1001016	
10160	Nortel Contivity DoS	Serious	nortel_cgiproc_dos.nasl	crashes the remote host	Denial of Service	WEB-CGI Nortel Contivity cgiporc DOS attempt	1763-1764	CAN-2000-0064
10161	rlogin -froot	High	rlogin_froot.nasl	Checks for rlogin -froot	Gain root remotely	RSERVICES rsh froot	604-609	CAN-1999-0113
10179	pimp	Serious	pimp.nasl	Crashes the remote host via IGMP overlap	Denial of Service		1001024	
10188	printenv	Medium	suse_cgi_bin_sdb.nasl:	Checks for the presence of /cgi-bin/printenv	CGI abuses	ATTACK RESPONSES 403 Forbidden - WEB-IIS scripts access	1201 - 1287	

b. Para ejemplificar, se toma una línea en concreto, en este caso la quinta:

10079	Anonymous FTP enabled	Low	ftp_anonymous.nasl	Checks if the remote ftp server accepts anonymous logins	FTP		1001001	

En principio se puede apreciar aquí que este evento no es detectado por Snort, pues se trata de una regla propia de esta red, por eso tiene asignado el ID 1001001.

c. Para llegar a esta codificación, primero se aisló desde Nessus, únicamente ese ataque, como se aprecia a continuación:



d. Se lanzó el ataque, el cual en esta caso no es detectado por Snort (Hasta que no se cree la regla correspondiente). Se presenta a continuación la captura del mismo realizado con un analizador de protocolos:

```
1 11.566632 BILLIO5A59F1 0030050830C6 TCP ....S., len:    0, seq:4282991099-4282991099,
ack 10.64.130.195 10.64.130.14

2 11.566632 0030050830C6 BILLIO5A59F1 TCP .A..S., len:    0, seq:1559011737-1559011737,
ack 10.64.130.14 10.64.130.195

3 11.566632 BILLIO5A59F1 0030050830C6 TCP .A...., len:    0, seq:4282991100-4282991100,
ack 10.64.130.195 10.64.130.14
```

En estas 3 primeras tramas se está estableciendo la sesión TCP hacia el puerto 21 (FTP)

```
4 11.566632 0030050830C6 BILLIO5A59F1 FTP Resp. to Port 2582, '220 w2000rst Microsoft FTP
S 10.64.130.14 10.64.130.195
```

Se hace presente el servidor FTP `220 w2000rst Microsoft FTP`

```
5 11.566632 BILLIO5A59F1 0030050830C6 TCP .A...., len:    0, seq:4282991100-4282991100,
ack 10.64.130.195 10.64.130.14
```

Se envía el ACK correspondirente

```
6 11.576647 BILLIO5A59F1 0030050830C6 FTP Req. from Port 2582, 'USER anonymous'
10.64.130.195 10.64.130.14
+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0x458A; Proto = TCP; Len: 68
+ TCP: .AP..., len:   16, seq:4282991100-4282991116, ack:1559011789, win: 5840, src: 2582
dst:   21 (FTP)
+ FTP: Req. from Port 2582, 'USER anonymous'

00000:  00 30 05 08 30 C6 00 10 60 5A 59 F1 08 00 45 00   .0..0Æ..`ZYñ..E.
00010:  00 44 45 8A 40 00 40 06 DB D8 0A 40 82 C3 0A 40   .DEŠ@.@.ÛØ.@,Ã.@
00020:  82 0E 0A 16 00 15 FF 49 41 FC 5C EC A1 CD 80 18   ,.....ÿIAü\ì¡Í .
00030:  16 D0 70 FF 00 00 01 01 08 0A 03 53 04 7D 00 43   .Ðpÿ.......S.}.C
00040:  FB 69 55 53 45 52 20 61 6E 6F 6E 79 6D 6F 75 73   ûiUSER anonymous
```

Aquí Nessus prueba si permite la validación como USER anónimo: **`USER anonymous`**

```
7 11.576647 0030050830C6 BILLIO5A59F1 FTP Resp. to Port 2582, '331 Anonymous access
allowed 10.64.130.14 10.64.130.195
```

Aquí se hace evidente aue permite este acceso: **`331 Anonymous access allowed`**

```
8 11.576647 BILLIO5A59F1 0030050830C6 FTP Req. from Port 2582, 'PASS nessus@nessus.org'
10.64.130.195 10.64.130.14
+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0x458B; Proto = TCP; Len: 76
+ TCP: .AP..., len:   24, seq:4282991116-4282991140, ack:1559011861, win: 5840, src: 2582
dst:   21 (FTP)
+ FTP: Req. from Port 2582, 'PASS nessus@nessus.org'

00000:  00 30 05 08 30 C6 00 10 60 5A 59 F1 08 00 45 00   .0..0Æ..`ZYñ..E.
00010:  00 4C 45 8B 40 00 40 06 DB CF 0A 40 82 C3 0A 40   .LE‹@.@.ÛÏ.@,Ã.@
00020:  82 0E 0A 16 00 15 FF 49 42 0C 5C EC A2 15 80 18   ,.....ÿIB.\ì¢. .
00030:  16 D0 E3 22 00 00 01 01 08 0A 03 53 04 7D 00 43   .Ðã".......S.}.C
00040:  FB 69 50 41 53 53 20 6E 65 73 73 75 73 40 6E 65   ûiPASS nessus@ne
```

Aquí Nessus prueba si permite cualquier contraseña: **`PASS`** nessus@nessus.org

```
9 11.606690 0030050830C6 BILLIO5A59F1 FTP Resp. to Port 2582, '230 Anonymous user logged
in 10.64.130.14 10.64.130.195
```

Aquí se hace evidente aue permite este acceso: **`Anonymous user logged in`**

```
10 11.626719 BILLIO5A59F1 0030050830C6 TCP .A...F, len:    0, seq:4282991140-4282991140,
ack 10.64.130.195 10.64.130.14

11 11.626719 0030050830C6 BILLIO5A59F1 TCP .A...., len:    0, seq:1559011892-1559011892,
ack 10.64.130.14 10.64.130.195
```

En estas 2 últimas tramas se está cerrando la sesión TCP hacia el puerto 21 (FTP)

e. Si se desea se puede analizar la regla de Nessus que genera este ataque, la cual es: ftp_anonymous.nasl, que se presenta a contiinuación:

```
#
# This script was written by Renaud Deraison <deraison@cvs.nessus.org>
#
#
# See the Nessus Scripts License for details
#

if(description)
{
 script_id(10079);
 script_version ("$Revision: 1.23 $");
 script_cve_id("CAN-1999-0497");
 script_name(english:"Anonymous FTP enabled",
          francais:"FTP anonyme activé",
          portugues:"FTP anônimo habilitado");

 script_description(english:"The FTP service allows anonymous logins. If you do not
 want to share data with anyone you do not know, then you should deactivate
 the anonymous account, since it can only cause troubles.
 Under most Unix system, doing :
     echo ftp >> /etc/ftpusers
 will correct this.

 Risk factor : Low",
                    francais:"Le serveur FTP accepte les connections anonymes. Si vous
ne souhaitez pas partager des données avec des inconnus, alors vous devriez
désactiver le compte anonyme, car il ne peut que vous apporter des problèmes.
Sur la plupart des Unix, un simple :
     echo ftp >> /etc/ftpusers
corrigera ce problème.

Facteur de risque : Faible",
                              portugues:"O servidor FTP está permitindo login anônimo.
Se você não quer compartilhar dados com pessoas que você não conheça então você deve
desativar a conta anonymous (ftp), já que ela pode lhe trazer apenas problemas.
Na maioria dos sistemas UNIX, fazendo:
     echo ftp >> /etc/ftpusers
irá corrigir o problema.

Fator de risco : Baixo");

 script_summary(english:"Checks if the remote ftp server accepts anonymous logins",
                francais:"Détermine si le serveur ftp distant accepte les logins anonymes",
                portugues:"Verifica se o servidor FTP remoto aceita login como anonymous");

 script_category(ACT_GATHER_INFO);
 script_family(english:"FTP");
 script_family(francais:"FTP");
 script_family(portugues:"FTP");
 script_copyright(english:"This script is Copyright (C) 1999 Renaud Deraison",
                  francais:"Ce script est Copyright (C) 1999 Renaud Deraison",
                  portugues:"Este script é Copyright (C) 1999 Renaud Deraison");
 script_dependencie("find_service.nes", "logins.nasl", "smtp_settings.nasl");
 script_require_ports("Services/ftp", 21);
 exit(0);
}
```

```
#
# The script code starts here :
#

port = get_kb_item("Services/ftp");
if(!port)port = 21;

state = get_port_state(port);
if(!state)exit(0);
soc = open_sock_tcp(port);
if(soc)
{
 domain = get_kb_item("Settings/third_party_domain");
 r = ftp_log_in(socket:soc, user:"anonymous", pass:string("nessus@", domain));
 if(r)
 {
  security_warning(port);
  set_kb_item(name:"ftp/anonymous", value:TRUE);
  user_password = get_kb_item("ftp/password");
  if(!user_password)
  {
   set_kb_item(name:"ftp/login", value:"anonymous");
   set_kb_item(name:"ftp/password", value:string("nessus@", domain));
  }
 }
 close(soc);
}
```

f. Y por último sólo quedaría crear la regla correspondiente en Snort, la cual podría ser la siguiente:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"Anonymous FTP enabled
{nessus}"; flags:AP; content:"USER anonymous"; nocase; depth: 16;
reference:CVE, CAN-1999-0497; classtype:attempted-user; sid:1001001; rev:1;)
```

g. Este es un caso interesante de análisis pues como se puede apreciar en la ftp_anonymous.nasl, se trata de un ataque estandarizado por CVE como candidata: script_cve_id("CAN-1999-0497");

5. Propuesta:

Luego de todos los avances realizados en estos años gracias al estudio de estas herramientas, el trabajo fue decantando en forma natural hacia esta arquitectura donde todos sus componentes son GNU. Al llegar a este estado de la cuestión y por tratarse justamente de una metodología abierta, se consideró la posibilidad de ponerlo a disposición de quien quisiera emplearlo y/o colaborar a su vez con el mismo. En este punto hubo un total acuerdo pues la evolución de estas arquitecturas de dominio público, demuestran que la suma de personas que desinteresadamente desarrollan estos proyectos aportan mucho mayor beneficio que el egoísmo personal o empresarial en cerrarse para guardar el secreto de una investigación y lucrar con ella.

Con esta postura, es que se publica el presente artículo en Internet, en el cual se invita a todos aquellos que estén interesados en implementar arquitecturas confiables de NIDS, bajo software gratuito, ajustando los mismos a su red en particular y colaborando en la confección de nuevas reglas y codificaciones entre Nessus y Snort. Para aquellos interesados, la propuesta es la siguiente:

SITUACION:

a. Se trata de una metodología de trabajo que puede realizar un aporte muy importante en la tecnología NIDS, si se logra sumar personas que lo mantengan vivo, sin egoísmos y transmitiendo todas sus experiencias.
b. Ya se tomó contacto con Snort.org y otros organismos que colaboran con el entorno Linux.
c. Aún no se ha decidido dónde residirá definitivamente, por lo cual, en los mismos sitios en los cuales se encuentra publicado el artículo, se va a informar más adelante en qué páginas Web se continuará con la investigación. Sobre este punto aún no se desea tomar una decisión hasta evaluar todas las opciones.
d. Se propone: Crear un proyecto de normalización de trabajo con las herramientas Nessus-Snort, bajo el cual, se pueda identificar cada ataque con la detección del mismo unívocamente. Sería muy interesante la participación de mitre.org, nessus.org y Snort.org en el mismo
e. **Desarrollar este trabajo principalmente en un entorno Hispano** (demostrando la capacidad que se posee en estos lugares del mundo, poco manifiesta en Internet, y en ningún sitio que se refiera a Nessus o Snort). Este apartado no excluye a otras lenguas ni traducciones de todo aquel que desee sumarse en otro idioma.
f. Se plantea:
    - Familiarizarse con estos productos.
    - Instalarlos en cada entorno participante (en laboratorio o producción).
    - Analizar las propias vulnerabilidades.
    - Comenzar a emplear analizadores de protocolos o sniffers.
    - Evaluar lo detectado por los NIDS.
    - Tratar de identificar y codificar, ID de ataques con ID de Snort.
    - Aprender a crear reglas para Nessus (Con el lenguaje NASL, documentado en Nessus.org)
    - Aprender a crear reglas con Snort (Acorde al manual, los documentos y How To de Snort.org).
    - Colaborar con la investigación y aportar nuevas ideas, códigos, reglas, actualizaciones, etc.
    - Tener siempre presente la idea de estandarizar procedimientos, reglas y definiciones.
g. Todo lo desarrollado estará a disposición una vez decidido el alojamiento.

MARCO DE TRABAJO:

a. Comenzar a realizar parte de lo planteado en el apartado anterior.
b. Una vez decidido el alojamiento de este trabajo, se insertarán documentos similares a los presentados en las mediciones. Los mismos estarán divididos en:
    - Plantilla de codificación Nessus- Snort.
    - Reglas aportadas para Snort.
    - Reglas aportadas para Nessus.
    - Cualquier otro módulo que amplíe el trabajo de estas herramientas.
c. Tratar de normalizar los eventos hacia un estándar, pareciera ser el más adecuado el propuesto por CVE (www.mitre.org), con quienes se tomará contacto a su debido tiempo.
d. Toda esta actividad la implementará un responsable para evitar alteraciones no debidas, deseadas o no.
e. Generar un foro de discusión sobre este proyecto, donde se pueda dialogar sobre el tema.

f. El ámbito queda abierto para todo aquel que desee participar y esté dispuesto a cumplir con lo establecido en el software GNU.
g. Los plazos y etapas se presentarán al estar disponible el proyecto.

NOTA: Hasta tanto se instale definitivamente el proyecto, cualquier contacto o idea se mantendrá a través de los correos electrónicos de este artículo, a través de los cuales se tratará de ir dando respuesta.