

Detección de sesiones interactivas en SSH

Autores

- Sebastián García (`sgarcia [at] {ufasta.edu.ar, citefa.gov.ar}`)
- Leandro Ferrari (`lferrari [at] ufasta.edu.ar`)
- Esteban Nicolás Cano (`sad_but_true [at] ufasta.edu.ar`)

Sección 0: Abstract

Hoy en día se encuentra ampliamente difundido el estudio de intrusos mediante la utilización de honeypots. Uno de los problemas que se presentan en el intento de análisis y clasificación de intrusos es el procesamiento y reconocimiento de las sesiones SSH [7] interactivas, ya que la mayoría del tráfico generado por los intrusos a la hora de ingresar comandos se realiza por este protocolo. Analizar el tráfico manualmente en busca de las sesiones SSH no es imposible, pero cuando se trabaja con gran cantidad de conexiones, almacenadas en archivos *pcap* de más de 1GB de tamaño, se ve la necesidad de un proceso automatizado. La tarea se dificulta por el hecho de que todas las conexiones SSH están cifradas.

Se clasifica al tráfico SSH en dos grandes tipos: las conexiones que lograron acceder al honeypot por poseer la contraseña correcta y las conexiones intentan adivinar la contraseña. Ambos tipos de conexiones se pueden producir manualmente o de forma automática. Si la contraseña es correcta los intrusos ingresan para ejecutar comandos de forma interactiva. Se tornó esencial el reconocimiento de las sesiones SSH que ejecutaban comandos de forma interactiva entre el resto de las demás sesiones.

El Laboratorio de Investigación en Seguridad Informática Si6 [1] de CITEFA [2] necesitaba solucionar el problema del reconocimiento de sesiones SSH interactivas. En el mes de Septiembre del 2006 se creó el Grupo de Estudio en Seguridad Informática de la Universidad FASTA [3] sede Mar del Plata, en convenio con el Si6 para abordar este problema.

Los objetivos del presente trabajo son:

1. Desarrollar alguna técnica para diferenciar las sesiones interactivas
2. Implementar la técnica mediante un software que clasifique automáticamente estas sesiones

Las técnicas que se lograron implementar son innovadoras en esta área específica: identificación de los distintos banners de los clientes SSH junto con un análisis de la cantidad de Bytes utilizados en cada sesión. Se realizaron experimentos con sesiones generadas en el laboratorio y con sesiones de ataques reales en tres años de capturas en honeypots, logrando identificar el 100% de las sesiones interactivas en honeypots y un gran porcentaje de las sesiones interactivas en laboratorio. Se encontraron algunas dificultades en la efectividad de las técnicas y se plantean sugerencias para trabajos futuros. Estas técnicas y esta herramienta permitieron resolver satisfactoriamente el problema planteado para el Si6.

Sección 1: Introducción

La utilización de honeypots para el estudio de intrusos es una técnica muy implementada actualmente en el ambiente de la seguridad informática [4]. Proveen una excelente posibilidad de entender cómo un intruso se adentra en un sistema de forma no autorizada, simulando sistemas en producción. El intruso trabaja como si realmente estuviera penetrando en una organización, lo que permite profundizar en su análisis sin riesgos de daños en los datos.

El proyecto nace de la necesidad del laboratorio Si6 de Citefa de obtener una herramienta capaz de identificar sesiones interactivas del protocolo SSH. El problema principal del Si6 era analizar los paquetes capturados en archivos de gran tamaño con formato *pcap*. Al contar con archivos que poseen un

tamaño que fluctúa entre 500MB y 2GB provoca que la búsqueda de sesiones SSH de forma manual sea virtualmente imposible.

A través del contacto con la Universidad FASTA sede Mar del Plata a raíz de esta necesidad, se creó un Grupo de Estudio que pudiese abordar la problemática. El objetivo principal del Grupo de Estudio era desarrollar una herramienta capaz de resolver el problema planteado, además de la realización de un paper que represente por escrito este trabajo.

Desde noviembre del 2006 hasta marzo del 2007 las reuniones quincenales se realizaron de forma virtual-remota, dada la distancia geográfica del director con el resto del grupo. De forma paralela a las reuniones quincenales se asignaron tareas para el hogar que mantenían al grupo en contacto con la problemática planteada. En las clases virtuales se realizaban discusiones de los diferentes temas y todos los integrantes exponían sus conclusiones y/o problemáticas que se les hubiesen presentado. Desde marzo de 2007 en adelante las reuniones se realizaron de forma presencial, semanalmente, en los laboratorios de la Universidad FASTA. En algunas ocasiones también se realizaron en la casa de algún integrante del grupo.

A medida que los conocimientos se afianzaban, el grupo fue practicando y simulando diversas situaciones para hacerse pasar por un intruso real, para conocer las reacciones típicas al ingresar en un sistema remoto: que comandos se ejecutarían primero, cuales después, etc. De esta manera se logra entender el comportamiento de los intrusos que se quieren analizar.

Las prácticas de intrusiones controladas brindaron la información necesaria para determinar las dos técnicas principales para detectar las sesiones interactivas del protocolo SSH. La técnica de detección de la versión del cliente SSH y la técnica de la cantidad de Bytes utilizada por la conexión SSH.

Después de esa fase de aprendizaje introductorio, se comenzó a investigar sobre las librerías pcap y las librerías libnids [5] para decidir cual se utilizaría en el desarrollo del programa. Después de haberlas comprobado, se decidió trabajar con las librerías libnids ya que resultaron las que mejor desempeño tuvieron.

Se desarrolló en Python el sniffer de conexiones SSH interactivas, implementando las técnicas estudiadas. Este sniffer se denominó "ssh-sniffer".

Posteriormente se realizó el diseño de los experimentos en base a las pruebas que se consideraron representativas del comportamiento de un intruso. Se realizaron las pruebas con el "ssh-sniffer" en el laboratorio de la Facultad de Ingeniería de la Universidad FASTA. Cada experimento fue documentado y cada prueba fue almacenada en un archivo pcap a fin de poder verificarlas en caso de ser necesario.

Contando con una cantidad de pruebas suficientes, se organizó la información para comenzar la creación del informe final.

Se ha hecho público en SourceForge.net tanto el código del programa "ssh-sniffer", publicado bajo la licencia GNUv2, como el informe final de la investigación. Se los puede encontrar en <http://www.ssh-sniffer.com.ar>.

Sección 2: Solución

Durante el análisis de las alternativas de resolución, se probó el funcionamiento de la herramienta SSHOW, capaz de analizar profundamente el protocolo SSHv1. Esta herramienta permite identificar las sesiones que ingresaron correctamente la contraseña, la ejecución de comandos y su longitud dentro de otras posibilidades. Se aprendieron diversas técnicas estudiando su funcionamiento y dadas algunas limitaciones de la misma se decidió desarrollar una herramienta propia que cubriera eficientemente las necesidades del proyecto.

El proyecto necesitó distinguir entre:

- Las conexiones que ingresaban bien la contraseña de forma manual
- Las conexiones que ingresaban bien la contraseña de forma automática
- Las conexiones que ingresaban mal la contraseña de forma manual
- Las conexiones que ingresaban mal la contraseña de forma automática

El análisis del problema derivó en el desarrollo de dos técnicas innovadoras en esta área específica, que permitieron distinguir entre diferentes tipos de sesiones SSH en ambos protocolos: versión 1 y 2.

No era suficiente detectar si la conexión implementaba el protocolo SSH correctamente, dado todos los tipos de conexiones SSH lo realizan perfectamente, eran necesarias técnicas más profundas.

Para distinguir entre las conexiones manuales y las conexiones automáticas implementamos la técnica de detección de banner del cliente.

Para distinguir entre las conexiones con contraseña correcta y las conexiones con contraseña incorrecta implementamos la técnica de cantidad de Bytes.

Cantidad de Bytes

Esta técnica consiste en comparar la cantidad de Bytes totales que una sesión utilizó contra una cantidad umbral límite. En los diversos experimentos controlados siguientes se definió el valor concreto de esa cantidad límite.

La cantidad de Bytes de una sesión depende de:

- La implementación del protocolo SSH por el cliente y su configuración
- La implementación del protocolo SSH por el servidor y su configuración
- Las tareas realizadas por el intruso

Definiremos una sesión interactiva mínima, como la sesión donde el intruso logra acceder al equipo de forma manual y sin ejecutar ningún comando corta la conexión con el mismo.

Definiremos un intento de acceso de tamaño máximo, como la sesión donde se intenta acceder (de forma manual o automáticamente) utilizando la mayor cantidad de Bytes posibles respetando el protocolo SSH, pero sin lograr acceder.

Veremos más adelante que el caso de acceso al equipo encontrando la contraseña de forma automática no es contemplado por diversas razones.

La cantidad de Bytes utilizada por la sesión interactiva mínima se comparó con la cantidad de Bytes del intento de acceso de tamaño máximo para determinar el valor límite del umbral de decisión.

En una sesión interactiva mínima la determinación de la cantidad de Bytes utilizados está dada por la implementación específica del cliente y del servidor y no por los comandos utilizados por el intruso. Ya que cada comando ingresado aumenta la cantidad de Bytes.

En la mayoría de los ataques, el intruso ejecuta algún comando al ingresar y esto nos asegura una sesión interactiva mínima de gran tamaño.

La clasificación de conexiones por cantidad de Bytes separa eficientemente las sesiones que utilizan una contraseña correcta en forma manual, de las que utilizan una contraseña incorrecta tanto automáticamente como de forma manual. Ya que estos dos últimos casos utilizan siempre una cantidad de Bytes pequeña.

Versión del cliente utilizado

Esta técnica de filtrado se implementa en combinación con el análisis de la cantidad de Bytes para reconocer rápidamente los clientes SSH utilizados en las pruebas automáticas. Este filtro nació como consecuencia de la gran cantidad de sesiones que se han encontrado con estos clientes en las capturas de datos del honeypot.

El filtro de clientes de la herramienta incorpora los clientes "libssh" y "SSH-2.0-MEDUSA_1.0", que son los banners por omisión de las herramientas hydra y medusa correspondientemente. También es posible agregar mediante parámetros nuevas versiones de clientes SSH.

Si bien es trivial evitar este filtro cuando el intruso se lo propone (el programa medusa tiene una opción para cambiar el banner de la versión SSH), en la práctica nos posibilita evitar el 90% de las sesiones no interactivas.

Sección 3: Desarrollo de la herramienta ssh-sniffer

Ssh-sniffer se desarrolló totalmente en python, los requerimientos para su ejecución son los siguientes:

- Python 2.2 o posterior, <http://www.python.org/download/>
- Librería Pynids, <http://palcrow.madison.wi.us/pynids/>
- Librerías Libnids, <http://www.packetfactory.net/projects/libnids/>
- Sistema Operativo Linux. (Podría llegar a ejecutarse bajo Microsoft Windows pero no ha sido probado)

Capacidades de la Herramienta

Ssh-sniffer implementa las dos técnicas discutidas anteriormente. Es posible configurar ciertas condiciones de filtrado, como la cantidad mínima del umbral de decisión para el filtro por cantidad de Bytes o los banners de los clientes ssh que se consideran herramientas automáticas. Es posible incluso ver todas las conexiones SSH sin aplicar ningún filtro.

Detalles del funcionamiento interno

La herramienta utiliza las librerías libnids (a través de las librerías pynids) para trabajar con los paquetes conociendo su estado en el protocolo TCP.

La metodología de funcionamiento es:

1. Filtrar las conexiones bidireccionales que pertenecen al protocolo SSH
2. Monitorear las conexiones en progreso y opcionalmente imprimir la totalidad de las conexiones SSH
3. Verificar las conexiones terminadas filtrando por banner SSH y luego por cantidad de Bytes. Se imprimen las conexiones SSH que superan los dos filtros

Los filtros por cantidad de Bytes solo los aplicamos cuando la conexión ha terminado ya que conocemos la cantidad total de Bytes transferidos. Es por esto que no nos es posible decidir si una conexión superó el umbral de Bytes hasta que no concluye la misma.

El filtro de las conexiones SSH se realiza buscando el string "SSH-" tanto en el comienzo de los datos enviados hacia el servidor como en el comienzo de los datos enviados hacia el cliente. Este string es el que oficialmente se envía al comienzo de cada conexión para significar que se trata del protocolo SSH, sin el mismo ni los clientes ni los servidores SSH funcionarán. El RFC 4253 "The Secure Shell (SSH) Transport Layer Protocol" dice en su sección "Protocol Version Exchange":

```
"When the connection has been established, both sides MUST
send an identification string. This identification string
MUST be SSH-protoversion-softwareversion SP comments CR LF"
```

Ejemplo de utilización (los archivos tcpdump se pueden obtener desde el sitio web de la herramienta). El parámetro -i identifica el archivo pcap a utilizar y el parámetro -t identifica el tipo de filtro a aplicar.

```
ssh-sniffer.py -i 2007-06-06_SG_4.tcpdump -t all
192.168.1.179:1055 SSH-2.0-PuTTY_Release_0.60 -> 192.168.1.248:22 SSH-1.99-
OpenSSH_4.4
(Bytes: 169913, Id:1)
(1 total SSH connections, 1 interactive)

ssh-sniffer.py -i 2007-05-23_SG_3.tcpdump -t all
192.168.1.10:1911 SSH-2.0-OpenSSH_4.4 -> 192.168.1.23:22 SSH-1.99-OpenSSH_4.4
(Bytes: 4353, Id:1)
(1 total SSH connections, 1 interactive)
```

El archivo 2007-06-06_SG_4.tcpdump es un experimento utilizando el cliente Putty en un sistema Windows, ingresando correctamente la contraseña en el primer intento y con un servidor SSH cuyo banner de bienvenida había sido modificado para contener un banner de aproximadamente 10KB de tamaño.

El archivo 2007-05-23_SG_3.tcpdump es un experimento utilizando un cliente Open-SSH e ingresando a un Servidor Open-SSH con la primer contraseña correcta.

Sección 4: Experimentos realizados

Se diseñaron una serie de experimentos a fin de poder contar con una base estable de datos controlados. Se verificó la herramienta utilizando las capturas pcap de estas pruebas.

Listado de versiones de Servidores SSH

- OpenSSH_4.4p1, OpenSSL 0.9.8d
- OpenSSH_4.3p2, OpenSSL 0.9.8b
- SSH-1.99-OpenSSH_3.6.1p2
- SSH-2.0-OpenSSH_4.3p2 Debian-8ubuntu1

Listado de versiones de Clientes SSH (intentando obtener una gama de versiones representativa de un entorno real)

- SSH-2.0-PuTTY_Release_0.60
- OpenSSH_4.3p2, OpenSSL 0.9.8b
- Medusa: SSH-1.99-OpenSSH0.0
- Medusa : SSH-1.99-OpenSSH0.0
- OpenSSH_4.4p1, OpenSSL 0.9.8d
- OpenSSH_4.3p2, OpenSSL 0.9.8b
- Mandriva Linux 10 SSH-2.0-OpenSSH_3.6
- SSH-2.0-OpenSSH_3.6
- SSH-2.0-MEDUSA_1.0

Todos los experimentos se realizaron con el objetivo de obtener una entrada de datos controlada y permitir la verificación con las capturas del honeypot real. Para identificar los archivos de los datos obtenidos de la herramienta, se especificó un formato de nombre de archivo para las capturas realizadas, de la siguiente manera:

P1-YYYY-MM-DD_InicialNombre.tcpdump

Donde:

P1: es el número de la prueba.

YYYY: Año en que se realizó la prueba.

MM: Mes en que se realizó la prueba.

DD: Día en que se realizó la prueba.

InicialNombre: Es la letra inicial del nombre de la persona que realiza la prueba.

Ejemplo:

2007-05-23_SG_3.tcpdump

Metodología del experimento

Esta metodología fue fundamental en la realización de los experimentos de forma paralela entre los miembros del grupo, compartiendo los mismos equipos y la misma red.

Pasos de la metodología:

1. Seleccionar la prueba a realizar, el servidor y el cliente:

Lista de Pruebas:

- **Tres contraseñas incorrectas:** Se ingresan manualmente 3 contraseñas incorrectas
- **Primera contraseña correcta pero sin comandos:** Se ingresa manualmente una contraseña correcta y luego sin realizar ningún comando se desconecta del servidor (CTRL-D)
- **Segunda contraseña correcta:** Se ingresa manualmente la primera contraseña incorrecta, la segunda correcta y luego sin realizar ningún comando se desconecta del servidor (CTRL-D)
- **Tercer contraseña correcta:** Se ingresan manualmente la primer y segunda contraseñas incorrectas. Por último se ingresa la tercer contraseña correcta y luego sin realizar ningún comando se desconecta del servidor (CTRL-D)
- **Primera contraseña correcta con comandos:** Se ingresa manualmente una contraseña correcta y se ejecutan los siguiente comandos: w, ifconfig, ls, exit
- **Cracking con Hydra:** Se utiliza un cracking de contraseñas con el software Hydra, con diferentes combinaciones de pruebas:
 1. Primer Cracking: Se utiliza un diccionario de 50 contraseñas incorrectas
 2. Segundo Cracking: Se utiliza un diccionario de 50 contraseñas, con la contraseña correcta en la primer posición del diccionario
 3. Tercer Cracking: Se utiliza un diccionario de 50 contraseñas, con la contraseña correcta en la posición 25 del diccionario
 4. Cuarto Cracking: Se utiliza un diccionario de 50 contraseñas, con la contraseña correcta en la posición 50 del diccionario.
- **Cracking con medusa:** Cracking de contraseñas con el software Medusa, utilizando diferentes combinaciones de pruebas:
 1. Primer Cracking: Se utiliza un diccionario de 50 contraseñas incorrectas
 2. Segundo Cracking: Se utiliza un diccionario de 50 contraseñas, con la contraseña correcta en la primer posición del diccionario
 3. Tercer Cracking: Se utiliza un diccionario de 50 contraseñas, con la contraseña correcta en la posición 25 del diccionario
 4. Cuarto Cracking: Se utiliza un diccionario de 50 contraseñas, con la contraseña correcta en la posición 50 del diccionario
- **Cracking con retardo entre contraseñas:** Cracking de contraseñas automático con un retardo de un segundo entre intentos
- **Con Banner modificado de un tamaño de 10k:** Se modificó el banner de bienvenida de la terminal del servidor SSH, con una longitud de 10k, para que sin comandos obtengamos una gran cantidad de Bytes transferidos
- **Sin Banner:** Se deshabilitó el envío de banner del protocolo SSH
- **Con Banner Default:** Se dejó el banner que viene por defecto

2. Iniciar el servidor SSH
3. Almacenar en un archivo pcap los paquetes que pertenecen a la prueba, filtrándolos de la siguiente manera:

```
tcpdump -n -s0 -v -w P1-2007-05-01_XX.tcpdump \
(src host IpLocal and dst IpRemota and dst port 22\
) or \
(dst host IpLocal and src host IpRemota and port 22\
)
```

4. Verificación del archivo generado de forma manual para asegurar que la captura fuera consistente:

```
tcpdump -n -s0 -r P1-2007-5-16_S.tcpdump
```

5. Prueba de la herramienta ssh-sniffer con los datos capturados pero sin aplicar los filtros:

```
python ssh-sniffer-0.7.9.py -i nombredelarchivo
```

Ejemplo de respuesta:

```
192.168.1.22:3442 SSH-1.99-OpenSSH0.0 -> 192.168.1.76:22 SSH-1.99-OpenSSH_4.3
(Bytes: 2878, Id:1)
(1 total SSH conections)
```

6. Prueba de la herramienta ssh-sniffer con los datos capturados aplicando todos los filtros:

```
python ssh-sniffer-0.7.9.py -i nombredelarchivo -t all
```

Ejemplo de respuesta:

```
192.168.1.22:2124 SSH-1.99-OpenSSH0.0 -> 192.168.1.76:22 SSH-1.99-OpenSSH_4.3
(Bytes: 18558, Id:1)
(1 total SSH conections, 1 interactive)
```

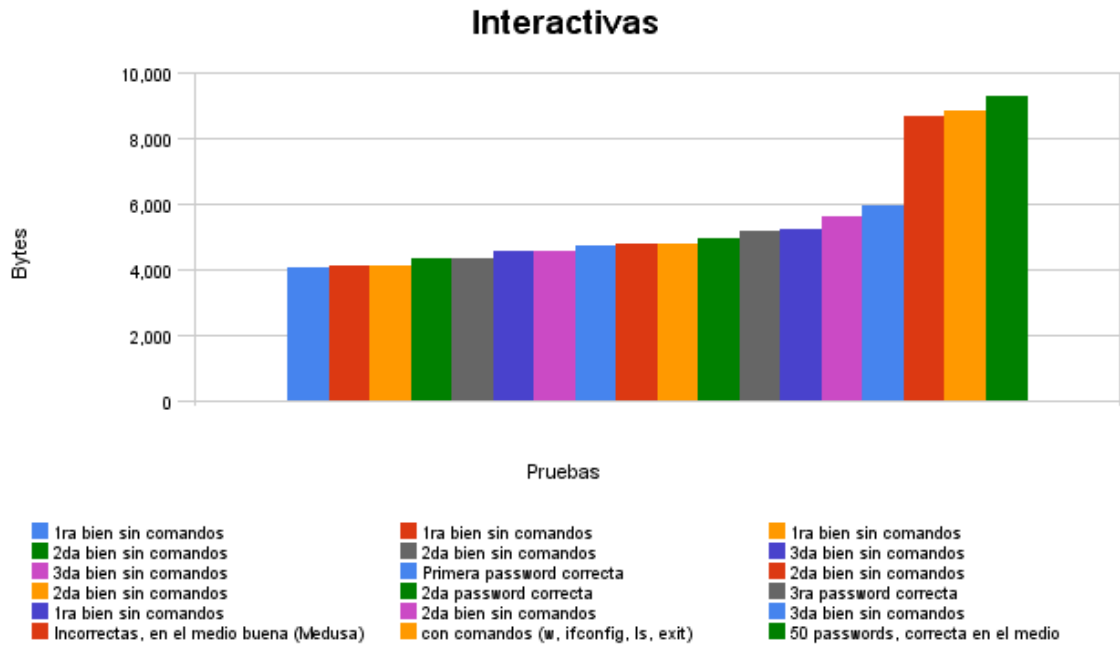
7. Se documenta los datos del experimento con los siguientes detalles:

1. Versión del Servidor
2. Versión del Cliente
3. Tipo de Prueba
4. Línea de ejecución de la herramienta
5. Resultado arrojado con la cantidad de Bytes

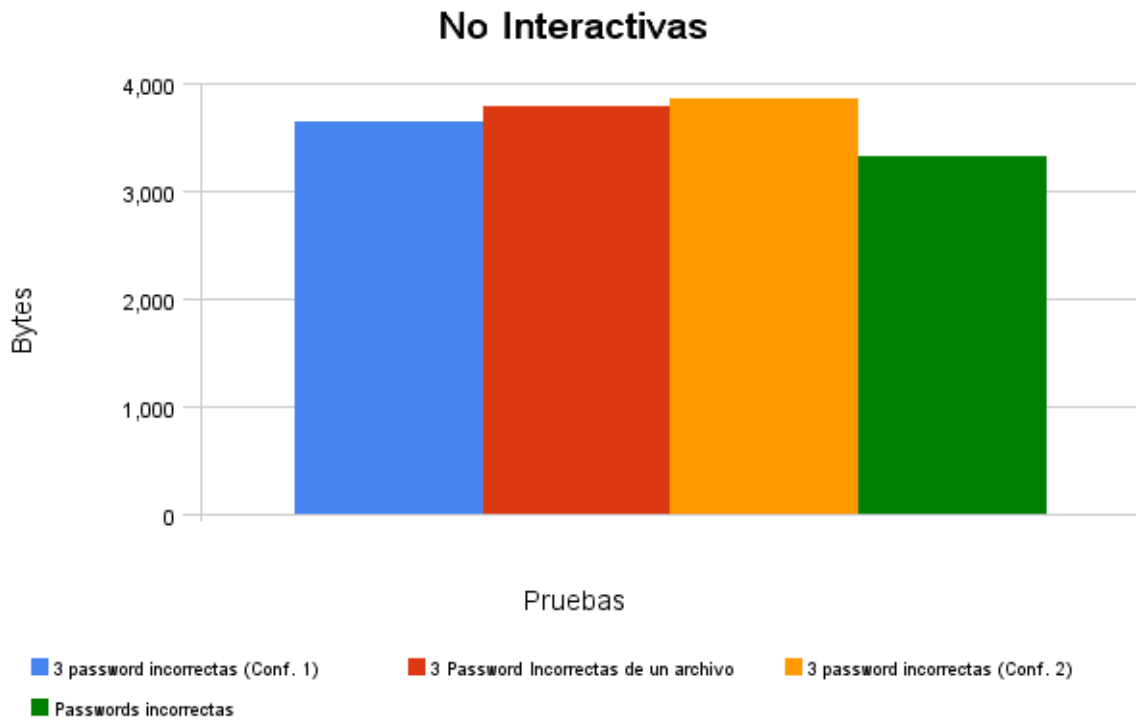
8. Se almacenan los resultados obtenidos al repositorio del grupo

Resultados de las pruebas realizadas

En el siguiente gráfico podemos observar las diferentes pruebas realizadas con sesiones interactivas y su respectivo tamaño en Bytes.



En el siguiente gráfico podemos observar las pruebas realizadas con sesiones no interactivas más significativas.



Búsqueda del Umbral

Para poder encontrar un umbral que nos permita configurar nuestra herramienta y poder separar las sesiones interactivas de las no interactivas por el método del tamaño de sesión, recurrimos a buscar los siguientes parámetros:

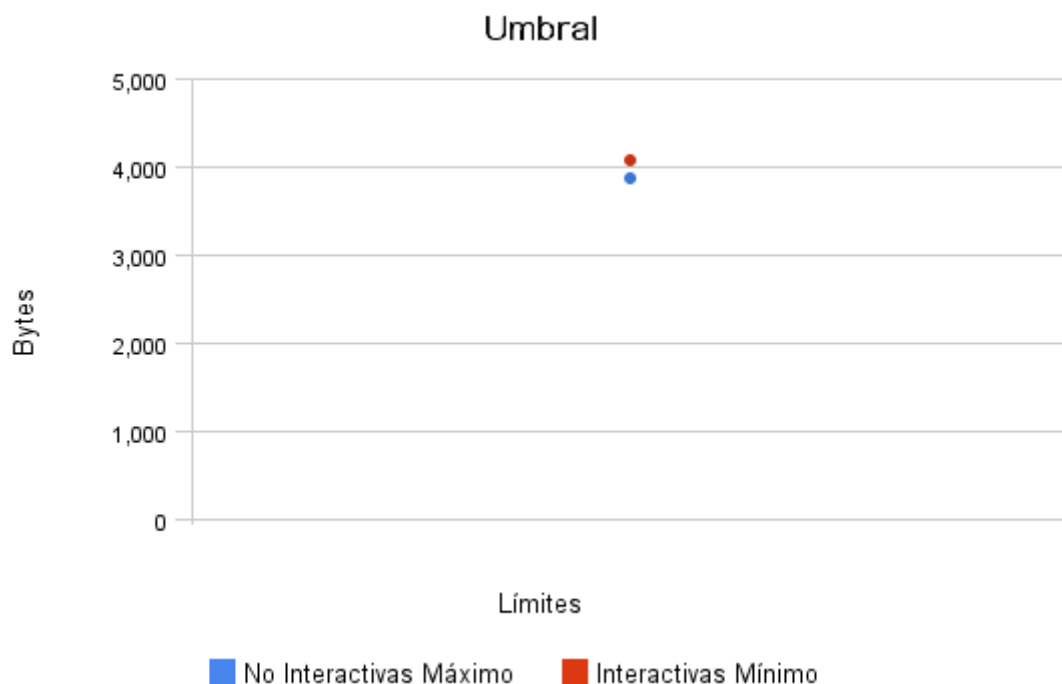
- Tamaño mínimo de sesión interactiva: de todas las sesiones interactivas, la de menor tamaño
- Tamaño máximo de sesión no interactiva: de todas las sesiones no-interactivas, la de mayor tamaño

Al encontrar estos valores podemos afirmar que, con la diferencia en Bytes entre ellos, tendremos la posibilidad de elegir un valor medio y configurar nuestra herramienta para detectar las sesiones interactivas.

Según las pruebas realizadas los resultados son los siguientes:

- Tamaño mínimo de sesión interactiva: 4073 Bytes
- Tamaño máximo de sesión no interactiva: 3861 Bytes

Con estos datos obtuvimos un umbral de 212 Bytes. En la herramienta ssh-sniffer se configuró como umbral el valor de 3970 Bytes para clasificar las sesiones (ver grafico).



Debemos aclarar dos situaciones en particular. La primera es que el software medusa utiliza una misma conexión TCP para probar las contraseñas de su diccionario. Los servidores SSH se encuentran configurados por omisión para permitir solamente tres intentos fallidos de acceso, cerrando la conexión TCP posteriormente, impidiendo al Medusa concluir las pruebas de las restantes contraseñas del diccionario.

En los experimentos con el software Medusa se modificó la configuración de los servidores SSH para permitir un número ilimitado de intentos de autenticación. De esta forma se logró obtener una captura de cincuenta intentos consecutivos de autenticación para analizar con el ssh-sniffer. Esta prueba utiliza una cantidad elevada de Bytes, debido a que la conexión TCP es ininterrumpida. Por este motivo, la herramienta detecta este caso en particular como una conexión interactiva cuando de hecho no lo es (falso positivo). Se descartará este resultado y no se tendrá en cuenta para los cálculos estadísticos por no encontrarse esta situación en un ambiente real.

La segunda situación a aclarar es que si el software hydra o medusa, encuentran la password correcta en su primer intento, esta conexión utiliza menos de 3861 Bytes, y por ser interactiva, dificultaría la detección por la técnica planteada en este trabajo ya que utilizaría menos Bytes que la conexión no interactiva de tamaño máximo. Esta fue la única situación que se detectó con estas características. Dado el carácter eminentemente práctico de la solución buscada con el software ssh-sniffer y dado que la situación planteada por las herramientas automáticas es inmensamente improbable de ocurrir, se descartará esta situación en particular.

Sección 5: Problemas y limitaciones de la técnica

Durante los trabajos realizados en esta investigación, se encontraron los siguientes problemas

- Existen una serie de *falsos positivos* cuando:
 - Se utiliza el software *hydra* o *medusa* con el banner de cliente SSH cambiado, y encuentran la contraseña correcta entre el primer y tercer intento, el *ssh-sniffer* detectará la sesión como no interactiva
 - Se utiliza el software *medusa* con su banner de cliente SSH cambiando y el servidor SSH está mal configurado para permitir el intento de más de tres contraseñas seguidas, el *ssh-sniffer* detectará la sesión como interactiva incluso si no se encuentra la contraseña correcta
- La herramienta *ssh-sniffer* no funciona correctamente detectando las sesiones interactivas en tiempo real, solo detecta las sesiones SSH una vez que concluyó la misma
- El intruso puede cambiarse el banner de su cliente SSH para que el *ssh-sniffer* detecte la sesión como no-interactiva y de este modo evite ser detectado
- Si bien se conoce que es posible analizar los tiempos de los paquetes SSH para determinar si verdaderamente corresponde a un intruso presionando teclas, ya que cada paquete corresponde a una letra y se vería la diferencia en tiempos, esta técnica no está implementada en el *ssh-sniffer*. Notar que esta técnica no detectaría las sesiones interactivas donde no se ingresan comandos.

Sección 6: Pasos futuros

Es posible mejorar la técnica y la herramienta en los siguientes puntos:

- Problemas de performance: al analizar capturas *pcap* de más de 300MB el tiempo que insume el `ssh-sniffer` para el análisis es demasiado largo. Este problema es común a todos los sniffers que necesitan guardar en memoria datos sobre los paquetes, por lo que se propone como solución mejorar la detección de sesiones en tiempo real.
- El `ssh-sniffer` no analiza si una sesión capturada es lógicamente correcta según la totalidad del estándar del protocolo SSH
- No se ofrece portabilidad a distintos sistemas operativos. Solo se corroboró su funcionamiento en Linux, pero no se descarta actual posibilidad de que el software funcione en otros sistemas operativos.
- Posiblemente se podría implementar un umbral para las conexiones SSH generadas por humanos y otro umbral para los programas automáticos, ya que su comportamiento en cantidad de Bytes es diferente.

Sección 7: Conclusiones

Al concluir el proyecto se ha llegado a diseñar una herramienta capaz de resolver el problema objetivo, aunque el software podría mejorarse ampliamente y aunque existan otras técnicas para la distinción entre sesiones interactivas.

Con respecto al grupo de estudio y a la forma de llevarlo a cabo el trabajo, se ha logrado funcionar como un equipo, aprendiendo nuevas experiencias y habilidades sobre los temas vistos, pasando por momentos de duda, satisfacción y aprendizaje, pero también y lo más importante para nosotros fue compartir juntos una investigación y llevarla a su terminación.

Sobre este tema aún quedan pendientes muchas mejoras e investigaciones, pero es un buen comienzo a partir de esta base que se ha desarrollado.

Sección 8: Agradecimientos

A nuestros compañeros de investigación y colaboradores en el proyecto: Federico Basualdo, Verónica Nisembaum, Sebastián Montini y Sebastián de la Fuente

A la Lic. Sandra Cirimelo por la gestión y el apoyo en la creación de este grupo y por la paciencia en las entregas.

A nuestro decano Roberto Giordano Lerena, por creer en la necesidad de la investigación y permitir que este grupo existiese.

Sección 9: Licencia

Copyright (c) 2008 Sebastián García, Leandro Ferrari, Esteban Cano.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2

or any later version published by the Free Software Foundation;

with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU

Free Documentation License".

Referencias

1. Si6, Laboratorio de investigación en Seguridad Informática de Citefa, <http://www.citefa.gov.ar/si6>
2. CITEFA, Centro de Investigaciones Científico-Técnicas de las Fuerzas Armadas, <http://www.citefa.gov.ar>
3. UFASTA, Universidad de la Fraternidad de Agrupaciones Santo Tomás de Aquino, <http://www.ufasta.edu.ar>
4. Honeynet, Organización dedicada a la investigación para mejorar la seguridad en Internet, sin costo alguno al público, utilizando honeypots, <http://www.honeynet.org>
5. Librería Libnids, Implementación de un E-component de un sistema de detección de intrusos, <http://libnids.sourceforge.net/>
6. Solar Designer, Dug Song, maintained as part of Dug Song's dsniff package, <http://monkey.org/~dugsong/dsniff/>. Originalmente publicado en <http://www.openwall.com/advisories/OW-003-ssh-traffic-analysis/>
7. SSH, Secure SHell, <http://www.ietf.org/rfc/rfc4251.txt>