Detección y limitaciones de ataques clásicos con Honeynets virtuales

Hugo Fernández, Jorge Sznek, Eduardo Grosclaude
hugofernandeznqn@gmail.com, jsznek@uncoma.edu.ar,
oso@uncoma.edu.ar

Universidad Nacional del Comahue
Neuquén, Argentina

Resumen Las honeynets surgen como una herramienta de seguridad diseñada para ser sondeada, atacada y comprometida por hipotéticos intrusos. Se componen de entornos de redes, conjuntos de aplicaciones de análisis y monitoreo, y dispositivos de almacenamiento de eventos. Luego de realizada la instalación y configuración de todos estos componentes, la honeynet queda dispuesta para recibir ataques, con la intención de mantener un ambiente controlado para el estudio de los eventos ocurridos. Luego, mediante el análisis de esos eventos, es posible comprender los objetivos, tácticas e intereses de los atacantes sobre el entorno propuesto.
En el presente trabajo se han implementado dos honeynets virtuales con sus correspondientes herramientas de seguridad sobre diferentes topologías de red, a los efectos de estudiar en cada caso un conjunto de ataques previamente seleccionado. Como resultado de la experimentación, se ha logrado estudiar el impacto de cada uno de los ataques en cada una de las honeynets desplegadas, analizando lo ocurrido para finalmente proponer metodologías de prevención o mitigación de las vulnerabilidades encontradas.

Palabras clave: honeypots, honeynets, virtualización, NIDS, HIDS.

1. Introducción

Las redes y sus aplicaciones, y en particular Internet, han introducido nuevas posibilidades que también implican riesgos. Éstos surgen a partir de las *vulnerabilidades* que poseen los sistemas. La existencia de vulnerabilidades implica *amenazas*, cuya concreción son los *ataques*.

Las vulnerabilidades pueden ser aprovechadas, con diversos fines, por muchas clases de *atacantes*: expertos o legos; interesados en el recurso de información que piensan comprometer, o motivados por intenciones en contra de la organización que atacan. En los últimos años, la frecuencia de aparición de ataques ha crecido considerablemente [13]. Este hecho, unido a las vulnerabilidades, descubiertas o latentes, en todo tipo de sistemas operativos y aplicaciones, convierte a cualquier

organización en una víctima potencial. Este panorama plantea la necesidad de disponer de instrumentos que permitan descubrir y analizar los *agujeros* de seguridad que pueda presentar un sistema, así como las técnicas y herramientas utilizadas por los posibles atacantes.

Para prevenir o mitigar cualquier tipo de amenaza es necesario conocer y comprender las vulnerabilidades constitutivas del entorno. Una de las metodologías para esto es crear un ambiente de red controlado pero a la vez lo suficientemente atractivo para los atacantes, que permita detectar comportamientos maliciosos, para estudiarlos, entenderlos y actuar en consecuencia, ya sea de una manera proactiva o reactiva, sin perjudicar el ambiente de producción de la organización. Este es el fundamento de las *honeynets* [5].

Estos ambientes pueden construirse en base a sistemas total o parcialmente compuestos por elementos virtuales. La idea de virtualización ha sido aplicada, desde hace bastante tiempo, a diferentes aspectos y ámbitos de la informática, desde sistemas computacionales completos hasta capacidades o componentes individuales. El tema en común de todas las tecnologías de virtualización es la de ocultar los detalles técnicos a través de la encapsulación. La virtualización crea una interfaz externa que esconde una implementación subyacente mediante la combinación de recursos en locaciones físicas diferentes, o mediante la simplificación del sistema de control. Un reciente desarrollo de nuevas plataformas y tecnologías de virtualización han hecho que se vuelva a prestar atención a este maduro concepto.

El presente trabajo comprende el diseño, implementación y despliegue de dos honeynets virtuales, la selección de cada una de las aplicaciones que la conforman y el conjunto de ataques a las cuales serán sometidas. El objetivo consiste en contrastar la información suministrada por cada uno de los despliegues, analizar dicha información, brindar una explicación relativa a lo ocurrido a partir de la información recolectada y proponer las distintas maneras de mitigar cada uno de estos ataques.

En la sección siguiente nos referiremos al modelo de honeynet empleado en nuestro trabajo; en la sección 3 describiremos las topologías experimentales desplegadas; en la sección 4 se enumerará el software utilizado en la implementación. En la sección 5 describiremos los ataques realizados, detallando para cada uno:

- Su objetivo
- Las precondiciones que lo hacen posible
- El software utilizado en el ataque
- El desarrollo del ataque
- Resultados obtenidos y análisis
- Contramedidas para detectar o neutralizar el ataque

Finalmente, en la sección 6 daremos las conclusiones globales del trabajo y algunas propuestas de trabajos futuros relacionados.

2. Honeynets

La implementación de honeynets debe tener en cuenta dos aspectos importantes:

- Control de datos: Define el grado de vulnerabilidad de los sistemas expuestos para que sea apetecible para el atacante y también el grado de restricción para no permitir la pérdida total del control.
- Captura de datos: Define qué herramientas y métodos se van a utilizar para el monitoreo y registro de toda la actividad efectuada dentro de la honeynet.

El control de datos tiene dos objetivos: el primero, es presentar un escenario lo suficientemente concreto para dar a los intrusos la sensación de que poseen el control del ataque, y, fundamentalmente, para que actúen como suelen hacerlo al momento de comprometer otras máquinas víctimas. El segundo, es lograr que el ambiente generado garantice la confinación del daño que pueda ser ocasionado al ambiente de la honeynet.

La captura de datos es una funcionalidad que permite recolectar y almacenar la evidencia tanto de la actividad de la red como de las máquinas intervinientes en los ataques a los que fueron expuestas. No se trata solamente de almacenar el log o la traza del tráfico de red, sino que es una combinación de monitoreo de la actividad, observación del *modus operandi* del atacante y capacidad de reconocer los distintos perfiles de ataques, para interpretarlos mejor.

2.1. Arquitecturas.

Podemos diferenciar dos clases de honeynets: las de *primera generación* y las de *segunda generación*.

Honeynets de primera generación

Su arquitectura está compuesta de un firewall, responsable del control de datos, un sistema de detección de intrusos (Intrusion Detection System, o IDS), responsable de la captura de datos, un servidor centralizado de logs y un conjunto de máquinas víctimas o sistemas señuelos (*honeypots*). El IDS puede especializarse en eventos locales al host donde está instalado (HIDS) o eventos propios de la red (NIDS) [6].

Honeynets de segunda generación

En las honeynets de primera generación, no se contaba con la posibilidad de monitorear a los atacantes por un período de tiempo lo suficientemente largo sin que éstos detectaran la presencia de la propia honeynet. Esto se debió a limitaciones inherentes a la propia implementación: primero, las restricciones en el número de conexiones salientes desde la honeynet, y segundo, el excesivo uso de comunicaciones sobre la capa de red (capa 3 del modelo OSI [15]) en

dispositivos críticos como firewall, IDS y servidores de logs. Al revelarse su existencia, se incrementaba notablemente el riesgo de convertirse en objetivos de ataque.

La arquitectura de una honeynet de segunda generación está compuesta por un equipo denominado *honeywall*, que cumple el papel de firewall e IDS simultáneamente, y un conjunto de honeypots.

2.2. Honeynets virtuales

Las honeynets virtuales son un concepto relativamente nuevo. La idea consiste en combinar todos los elementos físicos de una honeynet dentro de una única computadora, utilizando para ello software de virtualización.

Las principales ventajas de las honeynets virtuales son la gran reducción en los costos, la portabilidad y la facilidad tanto en su despliegue como en el mantenimiento de toda la infraestructura, ya que todo se encuentra integrado en un único sistema. Por otro lado, presentan ciertos problemas.

- Limitaciones inherentes: Limitados tipos de sistemas operativos que se pueden desplegar debido al hardware subyacente y al software de virtualización utilizado.
- Incremento en los riesgos: Los atacantes podrían comprometer la seguridad de la máquina física, obteniendo todo el control de la honeynet, y corromper todos los mecanismos tanto de control como de captura de información.
- Riesgo de *fingerprinting*: fingerprinting es la habilidad para identificar la honeynet en forma remota o local. Las honeynets virtuales poseen firmas que las hacen únicas (como resultado de la virtualización). Los atacantes podrían indentificar dichas firmas y detectar de esta manera el propósito de nuestra honeynet.

3. Topologías experimentales

Debido a la necesidad de generar diferentes escenarios para llevar a cabo el desarrollo de la experimentación, seleccionamos dos tipos de topologías distintas, tomando como modelos aquellas de uso más común en ámbitos hogareños y de la pequeña y mediana organización. En ambos casos la base física será una única computadora, y el resto de los dispositivos y conexiones serán emulados por virtualización.

Topología virtual I

Es una red donde todos los equipos involucrados, ya sean virtuales o físicos, e incluyendo estaciones honeypot y atacante, están sobre el mismo dominio de broadcast. Comparten un mismo espacio IP y todas están sobre el dominio de alcance de protocolos como ARP o DHCP. La topología experimental refleja el caso real en el cual la red sometida a ataques es una LAN y el atacante es un miembro de la misma, posiblemente una máquina comprometida por un troyano (Figura 1).

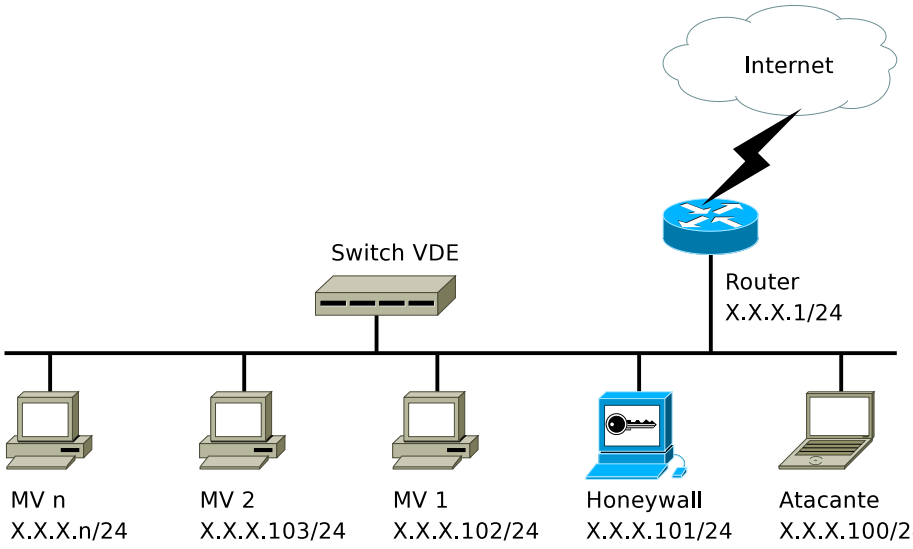

Figura 1. Topología I, dominio de broadcast único

Topología virtual II

Existe un elemento de conmutación que hace traducción de direcciones (NAT) y el espacio IP de los honeypots está separado del espacio IP del atacante. El router intermedio contiene los broadcasts, de modo que la difusión de protocolos como ARP y DHCP está restringida al espacio local de los honeypots. La topología refleja el caso en que el atacante se sitúa fuera de la red que intenta atacar (Figura 2).

4. Software utilizado en la experimentación

Para lograr la captura de la información acerca de los ataques o anomalías que se producen en la red trampa se utilizó un conjunto de programas, cada uno de éstos instalado estratégicamente en los nodos de interés a lo largo de la red.

- QEMU: VMM o software para administrar las máquinas virtuales [4].
- VDE-Switch: para efectuar la intercomunicación entre las máquinas virtuales y la máquina física [14].
- Snort: para ser usado como IDS de red (NIDS) [3].
- OSSEC: para ser usado como IDS de host (HIDS)[10].

Además se han utilizado las siguientes herramientas:

- MySQL: motor de base de datos utilizado para almacenar toda la actividad registrada por SNORT [16].
- Apache: servidor web para presentar los resultados [2].

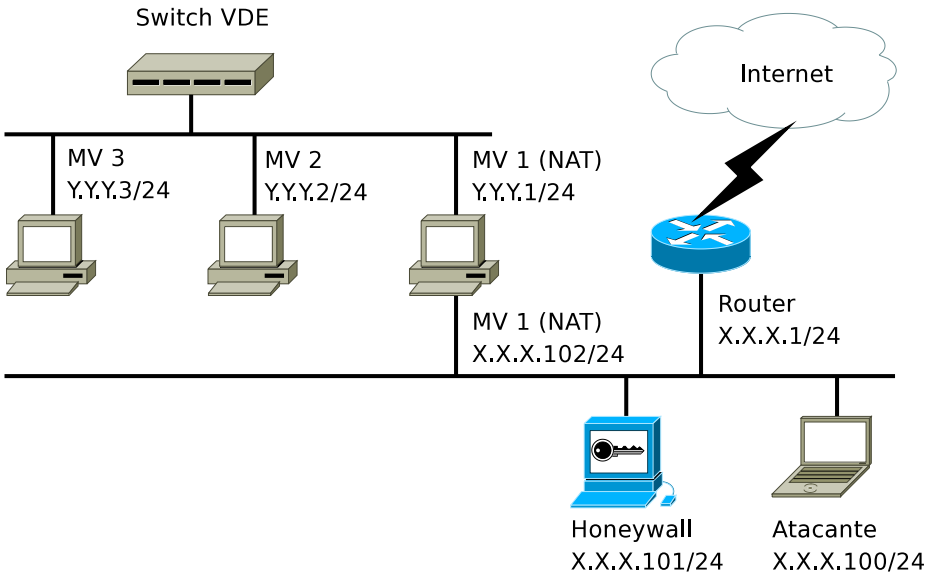

Figura 2. Topología II, traducción de direcciones

- Snort Report: aplicación web PHP para poder visualizar en tiempo real todo lo registrado por Snort en el motor de base de datos, mediante consultas programadas, mostrando gráficos acerca de lo sucedido [17].

Las motivaciones que llevaron a la elección de QEMU sobre otras herramientas de virtualización similares fueron:

- Software libre, licenciado bajo la General Public License (GPL).
- Proyecto en desarrollo y con gran cantidad de documentación.
- Se puede implementar en distintos sistemas operativos y sobre distintas plataformas de hardware.
- Emula completamente el hardware subyacente, logrando una transparencia que no es alcanzada por otros productos de virtualización.
- Presenta performance aceptable con el módulo kqemu (módulo del Kernel Linux que acelera la emulación de arquitecturas i386)
- Manejo conveniente de imágenes de disco, en varios formatos alternativos.

5. Ataques, experimentación y resultados

En esta sección se explicarán los cuatro ataques desarrollados: ArpSpoofing, Fuerza Bruta, Flooding y Keylogging.

5.1. ArpSpoofing

Descripción El protocolo ARP (Protocolo de Resolución de Direcciones) se encarga de la resolución de la dirección de hardware que corresponde a una

determinada dirección IP [1]. Ciertas características del protocolo ARP facilitan su uso de manera ilegítima para recibir tráfico ajeno. En particular, nos resultan relevantes las siguientes:

- Ausencia de autenticación en el protocolo. Es decir, una máquina modificará su comportamiento de acuerdo a los paquetes ARP recibidos, sin poder determinar de ningún modo la autenticidad de los mismos.
- Es posible -aunque evitable- modificar los contenidos de una caché ARP tan sólo con construir y enviar una petición o respuesta adecuada.
- Una máquina puede actualizar las caches ARP del resto de los sistemas de la red en cualquier momento.
- Como el protocolo ARP trabaja a nivel de capa de enlace de datos, esta técnica sólo puede ser utilizada en el ámbito de un dominio de broadcast.

El atacante aprovechará estas características en la técnica del *envenenamiento ARP*, o *ARP spoofing* [18], para denegar el servicio de la máquina remota, recibiendo tráfico ajeno en una red construida con switches. Mediante respuestas ARP, el atacante modificará el contenido de la caché de la víctima de forma que para la dirección IP de su interlocutor se corresponda la dirección MAC real del atacante. La idea consiste en "envenenar" la caché ARP de la máquina víctima con la dirección MAC del atacante, logrando de esta manera un ataque DoS ya que las peticiones de la víctima nunca serán respondidas.

Precondiciones Las precondiciones del ataque son las siguientes:

- La máquina atacante, conociendo las direcciones IP de los dos nodos cuyas comunicaciones quiere intervenir, resuelve mediante ARP las direcciones MAC que les corresponden.
- La máquina víctima está encendida y sin problemas de conectividad.
- Tanto el host víctima como el host atacante están dentro del mismo dominio de broadcast.

Debido a esta última precondición, el ataque estudiado no puede aplicarse en la topología 2 pues existe un router intermedio que separa la red donde se encuentra el host víctima de la del atacante.

Software utilizado La distribución Linux usada para realizar el ataque es BACKTRACK 2 versión Live, con software ARPSPOOF.

Ataque Para producir el "envenenamiento" de la caché ARP de la máquina víctima (IP 192.168.2.104) con el número MAC del atacante para todas las peticiones al gateway local (IP 192.168.2.1), el atacante inicia el ataque ejecutando el comando *arpspoof -t 192.168.2.104 192.168.2.1*.

Como consecuencia se provoca una denegación de servicio pues la máquina víctima no puede desempeñar su trabajo con normalidad. Por ejemplo, las aplicaciones que corren en ella no pueden navegar por internet porque al realizar peticiones al gateway, éste dirige el tráfico hacia la máquina atacante debido a la modificación producida en la caché ARP del host víctima.

Resultados y análisis El reporte de SNORT muestra el mensaje "*ICMP destination Unreachable - Communication with Destination Host is Administratively prohibited*", identificando completamente al atacante por su número de IP y su FQDN (Fully Qualified Domain Name). Un análisis más exhaustivo permite ver los detalles del suceso ocurrido a nivel de paquetes de red. A través del NIDS se pudo obtener información relevante sobre el ataque:

- Fecha y hora del evento de seguridad
- IP y FQDN del atacante.
- Reporte de detalles acerca del ataque a nivel de paquetes de red.

Por otro lado, el HIDS instalado en la máquina víctima no detectó nada que pudiera brindar indicios sobre la ocurrencia del ataque, ya que se trata de un ataque de red.

Contramedidas y elementos para la detección La defensa más completa contra el ARP Spoofing es activar la vinculación por MAC en el switch, que consiste en no permitir el cambio de la dirección MAC asociada a un puerto una vez que ha sido establecida. Los cambios legítimos podrán ser realizados por el administrador caso por caso.

Otra defensa es utilizar tablas estáticas. La cache ARP puede tener entradas estáticas de modo que los paquetes falsificados sean ignorados. Sin embargo, esta técnica no resulta muy práctica para redes corporativas pues las mismas están integradas por una gran cantidad de equipos.

En pruebas realizadas sobre sistemas operativos Windows se encontró que se siguen aceptando entradas ARP falsas y se siguen utilizando entradas dinámicas en lugar de las estáticas. Estos resultados vuelven nula la prevención mediante el uso de rutas estáticas.

Además, es recomendable utilizar métodos por software para detectar estos ataques, tales como:

- Arpwatch: programa que monitorea una red de computadoras desde la actividad ARP, generando un archivo de logs de los pares de direcciones (IP, MAC) a lo largo de un período de tiempo. La principal y más importante razón de monitorear la actividad ARP es detectar ARP spoofing [8].
- Ettercap: es un programa que posee la capacidad de interceptar segmentos de tráfico de red para luego efectuar un análisis de dicha captura [11].

5.2. Fuerza Bruta

Descripción El objetivo de este ataque es ingresar al sistema víctima con credenciales de administrador (*root*) a través de SSH v2.0. Para ello se apelará a la metodología de fuerza bruta, basada en diccionario, para obtener la contraseña del administrador. Una vez ingresado, el atacante desarrolla alguna acción maliciosa tal como instalar algún troyano con el cual garantizar el acceso en caso de que se cambie la contraseña, modificar la contraseña para denegarle el acceso al

usuario real, abrir puertos para que un segundo atacante pueda ejecutar alguna otra acción maliciosa, etc.

El mecanismo de autenticación basado en usuario y contraseña es vulnerable a la impostura. Un atacante que conozca, o conjeture, la combinación (usuario, password) correcta, obtendrá acceso al sistema sin otra condición. Un ataque muy común que hace uso de las vulnerabilidades de autenticación son los ataques basados en fuerza bruta (brute force attack), que consiste en bombardear al servidor con nombres de usuarios y contraseñas aleatoriamente generados. Una manera más "inteligente" de realizar ataques de fuerza bruta es mediante el uso de diccionarios de contraseñas, pues en muchas ocasiones las contraseñas utilizadas para los usuarios administrador son palabras comunes, secuencias de números, secuencias de teclas del teclado, etc. El país, cultura, idioma, etc., del sistema víctima cuentan en la elección de diferentes diccionarios.

Los ataques por fuerza bruta están muy extendidos debido a que con un simple script se puede testear cientos o miles de servidores de forma automática sin necesidad de intervenir manualmente en el proceso.

Precondiciones Las precondiciones del ataque son las siguientes:

- El archivo de configuración de SSH permitirá una cantidad de 29 reintentos de inicio de sesión.
- Se ubicó la clave correcta en la posición 20 del extracto del archivo de contraseñas de JOHN THE RI-PPER para poder efectuar un inicio de sesión exitoso luego de varios intentos fallidos.
- Ya que se ataca al usuario administrador de la máquina víctima, se supone que dicho usuario tiene acceso mediante SSH a ese sistema.

Software utilizado

- La distribución Linux usada para realizar el ataque es BACKTRACK 2, versión Live.
- Para escanear la red y localizar posibles víctimas se usó el programa NMAP [7].
- Para el ataque se utilizó el programa MEDUSA [9].
- El diccionario usado pertenece a un extracto del programa JOHN THE RIPPER [12](Incluido por defecto en la distribución).
- El servidor SSH de la víctima está incluido por defecto en la versión de FEDORA CORE 5 instalada en la máquina víctima.

NMAP es una utilidad para explorar y auditar redes informáticas. Utiliza los paquetes IP para investigar qué direcciones de red están asignadas a los hosts, qué servicios (nombre y versión de aplicación) están ofreciendo, qué sistemas operativos (y cuáles versiones) están en funcionamiento, qué tipo de filtros de paquetes o cortafuegos están en uso, y muchas otras características. Este programa fue diseñado para explorar rápidamente grandes redes.

MEDUSA es un programa que tiene como finalidad el acceso por fuerza bruta a una máquina víctima. Está elaborado en forma modular y con funcionamiento en paralelo, resultando muy veloz. JOHN THE RIPPER es un veloz y efectivo detector de debilidades y desencriptador de contraseñas. Actualmente disponible para muchos tipos de Unix, Windows, DOS, BeOS, y OpenVMS.

Ataque Lo que se señala a continuación es válido para ambas topologías.

- El atacante inicia su ataque utilizando el programa MEDUSA y el archivo de contraseñas de JOHN THE RIPPER, mediante la ejecución del comando *medusa -h 192.168.2.104 -u root -r3 -P /root/Desktop/listapwds.txt -M ssh.*
- Durante el ataque, el programa recorre automáticamente los datos contenidos en el archivo de contraseñas y se conecta vía SSH para probar cada una de ellas, utilizando el usuario administrador (*root*). Finalmente, obtiene acceso al sistema en el reintento número 20 y, como consecuencia, el sistema víctima es vulnerado y se obtiene el control de la máquina con todos los privilegios de super usuario.

Resultados y análisis El NIDS utiliza un sniffer para realizar las capturas de los paquetes que viajan por la red y los analiza según reglas incorporadas. Debido a que las sesiones generadas a través de SSH están cifradas, el NIDS no logra encontrar ninguna anomalía ya que no puede efectuar ningún análisis posible para este caso.

El HIDS captura las secuencias de reintento para la validación de usuario y muestra los intentos del atacante para ingresar al sistema.

Los intentos de acceso quedaron registrados en los logs del sistema operativo (/var/log/secure).

Finalmente, se puede concluir que el ataque es detectado una vez que se comienza a interactuar con la máquina víctima, independientemente de la topología de red en la que ésta se encuentre.

Contramedidas Algunas medidas que se pueden tomar para disminuir el riesgo de penetración pueden ser:

- Nunca permitir el acceso remoto al usuario *root* (administrador). Es preferible el acceso con un usuario no privilegiado y una vez dentro del sistema, adquirir privilegios de administrador.
- Utilizar SSH sólo dentro de la red interna, bloqueando el acceso al puerto 22 mediante un firewall.
- Utilizar contraseñas difíciles de conjeturar. Por ejemplo: más de 8 caracteres, mezcla de letras (mayúsculas y minúsculas), números, símbolos, etc. Preferentemente considerar algún esquema de contraseñas dinámicas Ej.: S/Key o SecurID.
- Activar la caducidad de las contraseñas de manera de renovarlas con frecuencia.

5.3. Flooding

Descripción Las organizaciones hacen uso extensivo de los servicios de Internet a través de su propia infraestructura, con el objetivo de mejorar sus operaciones o reducir sus costos. Ya sea publicando páginas web, utilizando un servidor de correo electrónico propio o brindando acceso remoto para teletrabajo, se crean canales abiertos de comunicación. Estos canales o vías de ingreso virtual pueden ser utilizados en forma maliciosa atacando su disponibilidad.

Este ataque tiene como principal objetivo saturar las conexiones al puerto SSH (port 22) abriendo el mayor número de sockets posibles en la víctima, llegando a provocar de esta manera una denegación de servicio.

Software utilizado

- La distribución Linux usada para realizar el ataque es BACKTRACK 2, versión Live.
- Para el ataque se utilizó OCTOPUS, programa que satura las conexiones a un puerto específico, dirigiendo su ataque al puerto 22 (SSH).
- Se utilizó el compilador GCC versión 4.2.1 para generar código ejecutable.

Precondiciones Dado que se trata de un ataque desde un host remoto (perteneciente a otra red), no corresponde su aplicación en la Topología I.

- La máquina víctima debe estar encendida y funcionando en red.
- El puerto que se ataca debe estar abierto y visible para el atacante.

Ataque El atacante inicia el ataque ejecutando el comando

octopusInf 192.168.2.102 22

Con esto se pretende inundar de conexiones el puerto 22 de la máquina cuya IP es 192.168.2.102.

Durante el ataque se abren n conexiones al puerto 22 utilizando para ello n puertos del atacante, provocando la denegación de servicio del puerto que es atacado, impidiendo el acceso a este servicio a cualquier usuario, y provocando una gran lentitud en el desempeño de la máquina virtual.

Resultados y análisis De los datos obtenidos por las herramientas dispuestas en la honeynet, se puede concluir que el NIDS no aporta datos sobre el ataque, porque el patrón de tramas que circula por la red virtual es considerado como válido, pues corresponde a intentos de establecer una conexión con un servicio.

Por otro lado, cuando llegan las peticiones el HIDS detecta que el usuario que está intentando establecer la conexión con el servicio no está ingresando los datos de identificación (usuario y contraseña). Este no es un dato menor, ya que si tantas conexiones en tan poco tiempo acusan ese problema es muy probable que se esté frente a un ataque.

Contramedidas

- Utilizar un firewall para restringir el acceso al puerto para el uso externo o bien cerrarlo.
- Nunca utilizar el servicio con usuario *root* (administrador).
- Configurar el IDS para tomar medidas frente a esta clase de ataques. En este caso OSSEC provee la posibilidad de efectuar "respuestas activas", utilizando para ello un demonio llamado ossec-excd. Por ejemplo permite efectuar: denegación de acceso al host; ejecutar determinadas reglas de firewall (cerrar el servicio para determinada dirección IP), etc.
- Utilizar OSSEC configurando el demonio ossec-maild, para que frente a algún problema alerte al administrador mediante el envío de un mail en el momento que ocurre el problema.
- No utilizar el servicio.

5.4. Linux Key Logger

El ataque tiene como principal objetivo capturar toda la información que la víctima ingrese por el teclado con el fin de obtener información sensible, dándole al atacante la posibilidad de realizar posteriormente ataques más específicos, como pueden ser: acceder a cuentas de correo electrónico, conocer las páginas que visita el usuario, acceder al *home banking* utilizando usuario y contraseña legítimos, entre otros.

Descripción Un *keylogger* es una herramienta que se encarga de registrar las pulsaciones que se realizan sobre el teclado, para guardarlas en un archivo o enviarlas a través de la red. El registro de lo que se teclea se puede hacer tanto por hardware como por software. Los sistemas comerciales disponibles incluyen dispositivos que pueden conectarse al cable del teclado y al teclado mismo. La escritura de aplicaciones para realizar keylogging es simple y, como cualquier programa computacional, puede ser distribuido masivamente a través de un troyano o como parte de un gusano.

A veces se afirma que la utilización de "teclado virtual" evita la captura de las pulsaciones del teclado al requerir únicamente clicks del mouse. Sin embargo, ya que los eventos de mensajes del teclado deben ser enviados al programa externo para que se escriba el texto, en principio cualquier keylogger podría registrar lo escrito mediante un teclado virtual. Por otro lado, existe malware que graba videos del movimiento del teclado mientras opera en un teclado virtual, lo cual lo inutiliza completamente como medida de prevención.

Software utilizado Para realizar el ataque se utilizó el software Linux Key Logger (lkl), que es un capturador de teclado que funciona en espacio de usuario bajo el sistema operativo Linux.

lkl captura las pulsaciones del teclado ocultando el puerto 0x60 y crea un archivo de log con todos los datos recolectados, previa traducción a código ASCII

utilizando un archivo de mapa del teclado. Además ofrece la facilidad de enviar por e-mail al atacante los datos recolectados, luego de llegar a algún tamaño de archivo previamente configurado.

Precondiciones

- El atacante es capaz de instalar y configurar en la máquina víctima un programa para que se ejecute el programa LKL cada vez que el usuario ingresa al sistema.
- El usuario víctima puede enviar correo electrónico.
- El usuario víctima usa su computadora en forma habitual y además navega por Internet visitando páginas interesantes para el atacante, como aquellas de *home banking* o webmail.

Estas precondiciones son válidas para ambas topologías.

Ataque El ataque se desarrolla utilizando el programa LKL y el archivo de mapeos de código ASCII KEYMAP. El comando ejecutado es:
lkl -l -k /root/Desktop/lkl/keymaps/it_km -o log.file -m atacante@mail.com; donde:

-l	determina que se haga log del puerto del teclado (0x60).
-k	establece la ruta del "keymap" que viene con la aplicación.
-o	especifica el nombre y extensión del archivo log que se va a generar.
-m	especifica la dirección de mail donde se enviarán los datos recogidos por la aplicación.

Por defecto, la aplicación envía datos al atacante cada 1 KB de información recopilada y continúa su funcionamiento.

Resultados y análisis SNORT no detecta ningún tipo de anomalía aparente, debido a que el único tráfico de red es la salida de un mail. OSSEC tampoco encuentra ningún patrón irregular que permita deducir algún evento de seguridad.

De los datos obtenidos por las herramientas dispuestas en la honeynet, se puede concluir que el NIDS no aporta datos sobre el ataque, pues el patrón de tramas que circula por la red virtual es considerado como válido por el IDS; y efectivamente lo es, ya que corresponde simplemente al envío de mensajes de email de un usuario válido.

Con respecto al HIDS, las herramientas de detección instaladas y la disposición de las topologías no fueron suficientes para detectar el *daemon* que está a la escucha del puerto del teclado; por lo tanto, no pudo ser identificado el ataque.

Finalmente, se puede concluir que para este conjunto de herramientas, los resultados son equivalentes en ambas topologías y no son suficientes para detectar el ataque.

Contramedidas Algunas medidas que se pueden tomar para detectar la existencia de un keylogger son:

- Acceder a un programa monitor de procesos para ver lo que está siendo ejecutado por la computadora en ese momento. En forma complementaria, es aconsejable contar con software anti-rootkit para detectar procesos ocultos.
- Utilizar software anti-spyware o anti-keylogging, que detectan a los keyloggers conocidos mediante el uso de firmas.

Existen algunas medidas que dificultan o impiden el ataque:

- Usar un firewall para evitar la transmisión del material registrado sobre la red.
- Ciertas prácticas permiten "engañar" a los keylogger, evitando la captura de la información al usar la secuencia de copiar/pegar texto.

6. Conclusiones y trabajo futuro

El resultado concreto de este trabajo es el diseño e implementación de honeynets virtuales. Para lograrlo, se ha aplicado un conjunto de herramientas de seguridad a dos diferentes topologías. Para demostrar el funcionamiento de cada instancia de honeynet, se ha preparado un conjunto de ataques que fueron finalmente ejecutados en cada topología seleccionada.

El resultado de la experimentación permitió:

- Visualizar la explotación de las vulnerabilidades expuestas.
- Confirmar que los IDS detectaron la mayoría de los patrones de ataques.
- Encontrar información sobre el atacante (Ej. IP y FQDN).
- Confirmar que utilizar herramientas que muestren los eventos de seguridad ocurridos de una manera comprensible y ordenada nos ayuda a implementar distintos tipos de solución para cada problema encontrado.
- Conocer cuáles son las vulnerabilidades del sistema y establecer a futuro políticas de seguridad que minimicen los riesgos para que alguno de estos ataques sea llevado a cabo.
- Descubrir que las herramientas dispuestas no fueron suficientes para la detección de la ejecución del ataque en ninguna de las topologías propuestas. (Ej: Durante el ataque utilizando linux key-logger).
- Confirmar que el uso de distintas topologías de red tiene una gran incidencia en cuanto a la seguridad. (Ej: Para la topología II el ataque de envenenamiento de caché arp no es implementable por cuestiones de funcionamiento de la propia topología.)

El resultado de la presente experiencia puede ser tomado como punto de partida para realizar otros casos de estudio utilizando honeynets. Algunos de estos casos de estudio podrían ser:

- Exponer una arquitectura honeynet a Internet, con el objetivo de visualizar, estudiar, clasificar y documentar los ataques encontrados, ya sean éstos ataques conocidos o no.

- Implementar una arquitectura honeynet dentro de alguna organización pequeña o mediana replicando el funcionamiento de la infraestructura existente, con todos los servicios necesarios y/o existentes: Ej: Servidor de correo electrónico, controladores de dominio, servidores web, DNS, DHCP, etc.
- Realizar un conjunto de ataques que explote las vulnerabilidades existentes, para finalmente proponer y documentar un conjunto de políticas de seguridad que ayude a mejorar la seguridad en dicha red.
- Implementar una arquitectura honeynet distribuida y enviar toda la información recolectada, utilizando una red privada virtual, al servidor de base de datos central que almacena toda la información de lo ocurrido en cada uno de los honeypots remotos. De esta manera, analizar las distintas tendencias de los atacantes en cada sitio remoto.

A modo de conclusión general, el diseño e implementación de honeynets no es un problema trivial y existen múltiples maneras de realizarlo. Diversas elecciones definen el tipo de honeynet a implementar (primera generación, segunda generación, honeynets virtuales, honeynets distribuidas, etc.), la definición de los grados de libertad y restricciones que se puedan aplicar para cada despliegue (i.e. conjunto de herramientas y políticas de seguridad que deban ser respetadas), y por supuesto, el conjunto de servicios que serán expuestos.

Referencias

1. Plummer David C. Rfc-826: An ethernet address resolution protocol - or - converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware. 1982.
2. Apache Community. The apache software foundation. http://www.apache.org, 1999.
3. Snort Community. Snort (ids/ips). http://www.snort.org, 1998.
4. Bellard Fabrice. Qemu. http://www.qemu.org, 2005.
5. López de Vergara J. Gallego E. Honeynets: Aprendiendo del atacante, 2004.
6. D. Gonzalez Gomez. Sistemas de detección de intrusiones versión 1.01, cap. 4. http://www.dgonzalez.net/pub/ids/html/index.htm, 2003.
7. Lyon Gordon. Nmap. http://nmap.org/, 1997.
8. LBNL's Network Research Group. arpwatch. http://www-nrg.ee.lbl.gov/.
9. JoMo-Kun. Medusa. http://www.foofus.net/jmk/medusa/medusa.html.
10. Cid Daniel B.; Panda Partha; Baig Bilal; Rockburn Lynn. Ossec: Open source host-based intrusion detection system. http://www.ossec.net, 2008.
11. Valleri Marco Ornaghi Alberto. Ettercap. http://ettercap.sourceforge.net, 2001.
12. Openwall Project. John the ripper. http://www.openwall.com/john/.
13. The Honeynet Project. Know your enemy - learning about security threats, 2002.
14. Davoli Renzo. Virtual square. http://wiki.virtualsquare.org, 2006.
15. Tanenbaum Andrew S. *Redes de computadoras.* Pearson Educación, 2003.
16. Inc. Sun Microsystems. Mysql: Open source database. http://www.mysql.com/, 1995.
17. Symmetrix Technologies. Snort report. http://www.symmetrixtech.com, 2007.
18. Volobuev Yuri. *Redir games with ARP and ICMP.* 1997.