UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"Implementación de un laboratorio de Análisis de malware".

Proyecto de tesis para obtener el titulo de

Ingeniero en Computación

Presentan:

Ramírez Gutiérrez Rubén Omar.
Reyes Fuentes Oscar Alberto.

Director de Tesis:

M.I. Jaquelina López Barrientos

Ciudad Universitaria.
Octubre de 2009.

AGRADECIMIENTOS.

A mis padres Antonia y José.
Por su guía, cariño, comprensión y apoyo incondicional.

A Silvia.
Por tu apoyo, compresión y amor.

A mis hermanos José Antonio y Gustavo.
Por su compañía y el apoyo que me brindan. Sé que cuento con ustedes siempre.

A todos mis amigos.
Que estuvieron conmigo y compartimos tantas aventuras, experiencias, y desveladas.

Oscar Alberto Reyes Fuentes.

Tengo un sentimiento encontrado, por una parte me siento muy contento por haber terminado un proyecto que me costo mucho trabajo finalizarlo, por esta razón agradezco a todas las personas que estuvieron a mi lado, mis padres, mis hermanas, mis amigos, compañeros de trabajo, a la M.I. Jaquelina y sobre todo a ti Oscar por aguantar todo el proceso.

Por otro lado quisiera agradecer a Diana, que aunque ya no estamos juntos fue parte fundamental en mi crecimiento y fuente de inspiración durante estos últimos 4 años.

Rubén Omar Ramírez.

ÍNDICE

Introducción.

El término Malware proviene de la asociación de dos palabras de la lengua inglesa: **Mal**icious Soft**ware**. Las cuales se asocian a todo aquel software que tiene el propósito de causar un daño. Los objetivos de éste software malicioso van desde una simple recolección de información personal de usuarios (para poder venderla a otras compañías), hasta el uso de recursos de forma remota o bien dañar la estructura del sistema operativo afectado. Estas metas se encuentran estrictamente relacionadas con la persona que diseña cada malware; algunos lo hacen por simple ocio, mientras que la mayoría lo hacen bajo la premisa de obtener un beneficio económico. Se pueden encontrar amenazas tales como; Gusanos, Troyanos, Backdoors, Spywares, RootKits, Exploits, Dialers, etc.

Los virus informáticos tienen básicamente la función de propagarse o replicarse con diferentes objetivos que bien pueden ir desde una simple broma, realizar daños importantes en los sistemas o bloquear redes generando tráfico inútil. Los gusanos son similares a los virus, pero estos no dependen de archivos portadores para poder contaminar otros sistemas. Pueden modificar el sistema operativo con el fin de ejecutarse automáticamente como parte de un proceso de inicialización del sistema. Por otra parte los troyanos son programas capaces de alojarse en computadoras y permitir el acceso a usuarios externos, a través de una red local o de Internet, con el fin de recabar información o controlar remotamente la máquina, pero sin afectar el funcionamiento de ésta[1].

Una puerta trasera (o backdoor) es un software que permite el acceso al sistema de la computadora ignorando los procedimientos de autenticación. Un exploit es aquel software que ataca una vulnerabilidad particular de un sistema operativo. Los exploits no son necesariamente maliciosos, ya que generalmente son creados por investigadores de seguridad informática para demostrar que existen vulnerabilidades en los sistemas; Siendo estos componentes comunes de los programas maliciosos como los gusanos informáticos.

Los rootkit, son programas que son insertados en una computadora después de que algún atacante ha ganado el control de algún sistema. Éstos generalmente incluyen funciones para ocultar los rastros del ataque, como es borrar las bitácoras de entradas o encubrir los procesos del atacante. Estos pueden incluir puertas traseras, permitiendo al atacante obtener de nuevo acceso al sistema o también pueden tener exploits para atacar otros sistemas.

El spyware, tiene como función más común recopilar información sobre el usuario u organizaciones sin que estos tengan conocimiento y distribuirlo a empresas publicitarias u otras interesadas, también se pueden emplear en círculos legales para obtener evidencias de sospechosos de delitos, tal y como sucede en los casos de piratería de software.

Los programas Bot o botnet actuan introduciéndose como un usuario más, pero adquiere el poder del operador, lo que le permite administrar ilícitamente y de manera eficiente algunas de las tareas.

Uno de los aspectos que más comúnmente se usan para clasificar el malware es su nivel de peligro o daño potencial para el equipo afectado. El malware, tiene variados comportamientos, que van desde la simple publicidad no autorizada hasta la actividad claramente maliciosa, lo cual pone en riego el cumplimiento de los tres principios de la seguridad de la información (confiabilidad, integridad y autenticidad).

[1] http://www.eset-la.com/centro-amenazas/1996-importancia-actualizaciones

Entre las actividades más típicas que se pueden encontrar en un Malware, destacan las siguientes:

- **Vectores de replicación**: Muchos tipos de Malware tienen la capacidad de explorar las listas de contactos que tiene el usuario afectado en su equipo y auto enviarse a todos los usuarios de ésta. Es un método típico de propagación de virus y gusanos de correo electrónico, aunque este comportamiento también puede repetirse en una red P2P, en sistemas de mensajería instantánea, incluso hasta en una red local.
- **Explotación de una vulnerabilidad de software**: Generalmente, cada vez que se da alerta de un problema del sistema operativo o de otro de sus componentes importantes (como clientes de correo electrónico y navegadores de Internet), se crea un malware que es capaz de explotar dicha vulnerabilidad, en muchos casos para intentar tomar control remoto de la máquina afectada. He aquí la importancia por parte del usuario de mantenerse al tanto de cada una de estas publicaciones y de instalar de manera correcta los parches necesarios.
- **Ingeniería social**: Es sabido que una gran cantidad de malware no puede funcionar sin la interacción del usuario. Muchos virus y gusanos informáticos pueden venir encapsulados en un archivo ejecutable que si no es abierto por el usuario no reviste daño alguno. Por lo tanto, muchos de estos programas malignos necesitan de la cooperación de la famosa “capa 8”, la cual consiste en el error humano, por lo tanto este tipo de malware suele venir adjunto en correos electrónicos tentadores, con remitentes que a veces son nuestros propios contactos.
- **Robo de contraseñas**: Uno de los principales objetivos que persigue un delincuente informático es alcanzar la contraseña de administrador de los equipos que busca atacar, para poder tener acceso sin restricciones al sistema. Por lo tanto una gran cantidad de malware está diseñado para cumplir esta misión.
- **Polimorfismo**: El malware es una amenaza en evolución. Estos programas son capaces de cambiar su forma para evitar ser destruidos o neutralizados por el software de protección. Cuando se lanzan los parches de protección a una vulnerabilidad que elimina un malware específico, los creadores de este último intentan hacer ingeniería inversa, de tal manera de mejorar su creación. Así comienza una brutal carrera entre el atacante y el atacado.
- **Secuestro de información**: Este tipo de malware, conocidos recientemente como **ransomware,** se caracterizan por impedir el acceso del usuario a sus propios documentos hasta que sea pagada una determinada cantidad monetaria.

Tomando en cuenta estos conceptos, se puede plantear las siguientes preguntas básicas que permiten comenzar a delimitar el objetivo de este trabajo, ¿Qué es lo que sucede actualmente con la seguridad informática?, ¿Qué beneficios se obtienen al contar con un laboratorio de malware?, ¿Cuál es el proceso de implementación del laboratorio?

Existen cuatro aspectos básicos de la gestión de seguridad, que son; el uso no autorizado de terminales informáticas, el número de incidentes internos y externos documentados, tipologías de ataque y por último, las acciones tomadas ante intrusiones y compromisos en los equipos.

Uno de los principales intereses en los ataques de malware es el económico. Por lo tanto las empresas tienen que considerar la importancia del resguardo de su

información y de esta manera obtener una interacción entre las tecnologías de información y los usuarios.

El malware se perfila como la mayor fuente de pérdidas económicas en las empresas. En segunda posición, aparecen los accesos no autorizados, seguidos de los daños económicos relacionados con la pérdida de hardware móvil. El cuarto factor es el robo de propiedad industrial, sumando estos cuatro factores el 74% de las pérdidas financieras totales en las empresas.

Durante el año 2008, hubo un gran movimiento en torno a los códigos maliciosos, a tal punto de quedar en evidencia que en la actualidad representan un negocio altamente redituable para quienes se dedican a lucrar con su creación y/o propagación.

A esta situación se le suma el fenómeno de las redes sociales, que cada día va en aumento en la sociedad y desde la perspectiva en materia de seguridad de la información, constituyen uno de los focos de ataque más aprovechado por usuarios sin escrúpulos. Las redes sociales como MySpace, Facebook, Twitter, Orkut y hi5 empiezan a ser cada vez más populares y en consecuencia comienzan a ver la luz los primeros códigos maliciosos que intentan explotarlas[2].

Además, no sólo se han transformado en fuentes de información donde los delincuentes intentan recolectar diferente tipo de datos sino que también lo utilizan como vínculo para llegar hasta los usuarios más desprevenidos a través de falsos perfiles y desplegando todo tipo de publicidad (splog). Durante el primer trimestre del 2008, se masifica la utilización de un tipo de malware cuyas primeras variantes fueron conocidas en el año 2006: el **Rogue**[3]. Es un malware que se caracteriza por simular ser una herramienta de seguridad, comienza a masificarse y a tomar notoriedad utilizando, en la mayoría de los casos, coberturas como programas antivirus. Generalmente responden a códigos maliciosos que bajo la excusa de eliminar todas las amenazas que se encuentran en el sistema, terminan descargando otros programas dañinos e infectando al usuario con diversa cantidad de otros malware.

Con el crecimiento del uso de dispositivos de almacenamiento que se conectan a través del puerto USB (cámaras fotográficas, teléfonos celulares, MP3 Player, memorias USB, entre otros), se crea otro canal de ataque altamente explotado por el malware a través de un archivo autorun.inf, que permite su ejecución de manera automática, diseminándose por cuanto dispositivo se conecte en un equipo comprometido.

Estas amenazas reciben el nombre genérico de INF/Autorun y durante este año han obteniendo una alta tasa de propagación, sobre todo en grandes organizaciones y universidades donde se permite el uso de estos dispositivos.

Otras cuestiones importantes a destacar, son la evolución, en cuanto a la eficacia, de las técnicas de Ingeniería Social utilizadas por el gusano Nuwar debido a la diversidad de métodos empleados por este; y el surgimiento de redes avanzadas llamadas **Fast-Flux**, que consiste en la modificación continua de la estructura de sitios web maliciosos y son utilizadas para la ejecución de diferentes ataques vía web.

Por otro lado se empieza a incrementar el ataque vía web llamado Drive-by-Download. Esta técnica se refiere a la inyección de código dentro del código fuente de

[2] Redes sociales utilizadas para propagar malware, Cristian Borghello, ESET.
[3] Cronología de los virus informaticos. Cristian Borguello, ESET.

la página vulnerada para que, cuando el usuario ingrese a la misma, sea redireccionado a otra con contenido malicioso[4].

En octubre y noviembre de 2008 se conoce la propagación de dos gusanos muy peligrosos: Gimmiv y Conficker, ambos se propagan a través de vulnerabilidades encontradas en plataformas Microsoft Windows, logrando un alto porcentaje de infección en pocas horas en el caso de Conficker. Los problemas de seguridad generados por estos gusanos, dejaron una vez mas en evidencia cuan importante es mantener reglas claras en cuanto a la implementación adecuada de actualizaciones de seguridad. A final de año apareció otra técnica de ataque llamada ClickJacking, El ClickJacking es una metodología de ataque que aprovecha vulnerabilidades en los navegadores web para redirigir a los usuarios a sitios maliciosos.[5]

El análisis de malware se puede abordar para su estudio desde dos perspectivas diferentes:

La primera, requiere obtener el binario del malware, una vez que se cuenta con éste, se utiliza alguna técnica de ingeniería inversa para obtener el código fuente y así poder predecir cuáles serán los efectos que dañaron el equipo de cómputo, esta técnica es cada vez más difícil de utilizar debido a que los binarios de los malware más actuales tienen técnicas de ofuscación que no permiten utilizarse de una manera rápida.

La otra forma de analizar un malware es a partir del comportamiento que éste tenga en una computadora, en esta técnica hay varias formas de operar, una es utilizando máquinas virtuales, de esta manera un malware se puede identificar que está siendo ejecutado en un entorno de investigación y simplemente decida entonces no actuar, aquí es precisamente donde el presente proyecto encuentra el reto a vencer, el cual consiste en ejecutar el malware desde una máquina real pero sin los inconvenientes de tener que estar instalando el sistema operativo cada vez que se ejecute el malware. El contar con un laboratorio de análisis de malware presenta una serie de ventajas en el estudio y desarrollo de la seguridad informática, entre las que destacan: el previo filtrado de información que ingresa a los sistemas, así como una herramienta sobre todo de apoyo para el análisis forense.

Adquirir las evidencias sin alterar ni dañar el original. La forma ideal de examinar un sistema consiste en detenerlo y examinar una copia de los datos originales, es importante tener en cuenta que no se puede examinar un sistema presuntamente comprometido utilizando las herramientas que se encuentran en dicho sistema pues éstas pueden estar afectadas. La cadena de custodia documenta el proceso completo de las evidencias durante la vida del caso, quién la recogió y dónde, quién y cómo la almacenó, quién la procesó, etc.

El Análisis Forense tiene como principales objetivos dar respuesta a las preguntas que se suelen hacer ante un ataque informático:

- ¿Quién realizó el ataque? En este punto es necesario ver las direcciones de red o bitácoras de acceso para determinar el posible atacante, esto es, desde dónde se realizó el posible ataque.

[4] http://www.eset-la.com/centro-amenazas/1792-drive-by-download-infeccion-web
[5] http://www.eset-la.com/company/1983-clickjacking-nueva-tecnica-ataque-navegadores-web

- ¿Cómo se realizó el ataque? Esto es, qué vulnerabilidad o falla fue empleada para ser explotada y qué técnica utilizó para acceder al sistema, como tercer punto nos preguntamos.

- ¿Qué hizo el atacante? Esto se refiere a las acciones que realizó el atacante una vez que accedió al sistema, que archivos, carpetas o si hubo modificaciones en el sistema operativo y por último tenemos la principal pregunta.

- ¿Cómo se realizó el ataque? Este punto es base ya que se necesita de herramientas, para poder llegar a cada una de las evidencias que dejó el atacante y así determinar la forma de realización del ataque.

- ¿Para qué accedió el intruso al sistema? Por lo general, la mayoría de atacantes tiene como objetivo la utilización de computadoras y redes para preparar ataques violentos o para coordinar y llevar a cabo actividades terroristas que amenazan la seguridad a escala mundial, los delitos relacionados con la posesión o distribución de pornografía infantil, la falsificación y fraude de datos bancarios o simplemente con el tráfico de cualquier producto que pueda mercarse a escala mundial muestran un panorama complejo,

Aunque es preferible mantener un sistema actualizado y bien protegido frente a ataques, muchas veces es difícil mantener un entorno complejo completamente protegido, por lo que es necesario tener una formación en este tipo de técnicas que permitan analizar y comprender el análisis de una intrusión informática.

La importancia de estas preguntas es tener conocimiento del tipo de ataque que hace el malware y de esta manera mejorar herramientas de protección para poder contrarrestar amenazas.

Una de las dificultades es la obtención de sistemas de pruebas (previamente atacados) que se puedan emplear para ensayar y conocer las herramientas de análisis existentes, así como de pruebas que permitan conocer el nivel de conocimientos sobre esta materia[6].

Por otro lado viendo que en la mayoría de empresas importantes, siguen utilizando como sistema operativo Microsoft Windows. Debido a la cantidad de equipos que hay distribuidos en el mundo con este sistema operativo, de los cuales la gran cantidad de software malicioso está diseñado para trabajar sobre esta plataforma y además de que existen muchos servicios que utilizan este sistema operativo, algunos ejemplos son servidores web, servidores de correo, etc.

El promedio de pérdidas anuales estimado de algunas empresas encuestadas fue de más de 2 millones de dólares americanos. En los meses recientes se ha producido una lluvia de ataques a entornos informáticos, muchos de ellos a través de Internet y a equipos con el sistema operativo Microsoft Windows. Sin embargo, éstos son sólo los problemas de seguridad más notorios a los que se enfrentan las empresas en la actualidad.

Por todo lo antes expuesto es que se tomó la decisión de realizar un trabajo de investigación serio y formal de manera particular en este sistema operativo.

[6] http://www.faqs.org/rfcs/rfc3227.html

El objetivo del presente trabajo de tesis consiste en construir un laboratorio de análisis de malware aplicando metodologías correspondientes al análisis forense hacia un equipo con sistema operativo Windows de forma automática utilizando herramientas propias y de terceros, para la obtención de información que revele el comportamiento del malware después de su ejecución en el sistema.

Con la implementación de este laboratorio se pretende conocer el comportamiento preciso del malware y con ello brindar a otros investigadores y desarrolladores una herramienta que indique de manera puntual en qué dirección deben desarrollarse nuevas medidas para contrarrestar la acción del software malicioso, este proyecto se basará en la utilización de herramientas de terceros que evalúen diversas características.

Para alcanzar el objetivo planteado, en el primer capítulo se presenta una introducción del contexto actual en el que se encuentra ubicado el malware, comenzando con una breve descripción de algunos programas maliciosos, los alcances de ataques de cada uno, así como la respectiva consecuencia de los ataque. En el capítulo 2 se realiza un análisis previo del laboratorio de malware, éste consta en la revisión de sistemas operativos dentro de los cuales va implementado el ataque del malware, éstos abarcan el tipo en el cual se realizará el laboratorio, el sistema en donde se llevara acabo un respaldo y la restauración del mismo, así como el sistema operativo que servirá como plataforma para desplegar resultados (servidor Web) y como firewall. También se realizan pruebas con herramientas propias y de terceros para la obtención de datos necesarios que permiten el análisis de malware. En el capítulo 3 se construye finalmente el laboratorio de malware, éste es armado en módulos en el que cada uno hace una revisión de puntos clave atacados frecuentemente por códigos maliciosos, estos rubros son llaves de registro, procesos, conexiones a redes, cambio de nombres en dns, etc., en este capítulo se configuran también las herramientas para el buen funcionamiento del laboratorio de malware éstas están constituidas por canales de comunicación, accesos remotos al laboratorio, administración de firewall, la administración del equipo de respaldo y una vez concluido el desarrollo entramos a la etapa de pruebas que es el resultado del capítulo 4 éste se basa en el diseño y desarrollo de software mediante el espiral de 4 vueltas, que consiste en realizar corrección de errores conforme las pruebas realizadas. Por último en el capítulo 5 se presentan las conclusiones obtenidas después de haber desarrollado el presente trabajo.

Capítulo1

Antecedentes y características del malware.

En este capítulo se plantea de manera general el contexto actual del tema, ,se definen conceptos básicos necesarios para adentrarnos en el fascinante mundo del código malicioso (malware), detallando cada una de las variantes de este y explicando las características más significativas, además se presenta una definición de vulnerabilidad, así como las técnicas de analizarla, se detalla el ciclo de vida y se mencionan las formas más comunes de como pueden ser aprovechadas a la hora de implementar código malicioso.

1. Introducción al malware.

1.1 Malware.

La palabra malware proviene de las palabras (malicious, software). Este es un programa que está diseñado para insertar virus, gusanos, troyanos, spyware o incluso los bots, para tratar de conseguir un objetivo como lo puede ser información sobre el usuario o sobre una máquina.

Malware es todo código malévolo instalado en una computadora y puede dar al atacante un grado verdaderamente alarmante de control sobre el sistema, red o datos, del usuario sin su conocimiento. Pertenecen a la clasificación de malware todos aquellos virus, gusanos y código malévolo entregado a través de los navegadores web y de los clientes de correo electrónico, backdoors, caballos de Troya y rootkits entre otros a nivel usuario. Cada uno de estos objetos son descritos en los siguientes puntos de este capítulo.

Una de los aspectos fundamentales en la propagación de malware es el medio de distribución y de manera puntual es el navegador, en esta parte encontramos código móvil que toma la forma de un programa Java, un script de JavaScript, Visual Basic Scripts (VBScripts) y controles ActiveX, en otras palabras el código móvil es un programa ligero descargado de un sistema remoto y ejecutado localmente con mínima o nula intervención del ser humano.

Este programa puede ser descargado del servidor, donde reside el código de la aplicación a la computadora del usuario y es ejecutado. Esta tendencia se localiza en páginas web donde se muestran noticias o menús de interacción.

Como son programas pequeños y sencillos les permite atravesar la red rápidamente y ejecutarse en un servidor o una máquina sin requerir de la intervención del usuario, una vez obtenido el código móvil, éste se ejecuta con fines para los cuales fue creado, una vez contraído este código hace que el sistema realiza algo que no se desea.

Un intruso puede utilizar el código malicioso móvil para una diversidad de actos molestos para las víctimas, incluyendo el monitoreo de actos de navegación, obteniendo acceso no autorizado al sistema de archivos, infectando la máquina con un caballo de Troya, secuestrando el navegador web para visitar sitios que el usuario no tenía previsto visitar. También existen muchas formas para las que se puede utilizar, en el instante en que se ejecuta este tipo de programas se aprovecha todas las limitantes de seguridad para que el malware funcione correctamente.

La forma más tradicional en que se puede propagar malware mediante navegadores web es con scripts, ya que los desarrolladores recurren a este tipo de herramientas para mejorar la apariencia de un sitio, como habilitar efectos de botones, procesar elementos de una forma o adaptar la apariencia de una página de acuerdo a las propiedades del navegador del usuario. El código que implementa esta funcionalidad está escrito en lenguajes como JavaScript, VBscript o algún otro.

Cuando visitamos una página web, el navegador descarga automáticamente y ejecuta su código móvil. Esto es por que los scripts están dentro de las páginas siendo parte del código HTML incluidos en etiquetas especiales. Estas etiquetas script indican el comienzo del código y especifican el lenguaje en el cual está escrito una vez que la función ha sido declarada. Este código es capaz de interactuar con el contenido de la página web de la que se origina.

Otro método simple es la negación de servicio, consiste en que el usuario no pueda trabajar adecuadamente. Los desarrolladores de malware implementan ataques de negación de servicios al consumir los recursos disponibles del sistema hasta que el sistema llegue a inutilizarse.

1.2 Virus.

Un virus es una secuencia de código que se inserta en un archivo ejecutable denominado Host, de forma que al ejecutar el programa también se ejecuta el virus; generalmente esta ejecución implica la copia del código viral – o una modificación del mismo – en otros programas. El virus necesita obligatoriamente un programa donde insertarse para poderse ejecutar, por lo que no se puede considerar un programa o proceso independiente.

Los virus informáticos tienen básicamente la función de propagarse, o replicarse con objetivos que van desde una simple broma hasta realizar daños importantes en los sistemas, o bloquear redes generando tráfico inútil.

Las características de los virus son:

- La incapacidad de funcionar como ejecutables por sí solos. Esta es la razón por la cual éste se anexa a sí mismo a otros programas. Un virus es un parásito que se pega al inicio del código de un programa.

- Un portador de virus, conocido como huésped, puede ser un programa ejecutable estándar o un archivo de datos que puede contener comandos tipo macro o simplemente se puede atar en instrucciones de bajo nivel almacenadas en el sector de arranque de un disco que le indica al sistema operativo cómo iniciar.

- La autorreplicación que se refiere a la habilidad de crear automáticamente copias de sí mismo sin requerir de un operador humano para duplicarlo. Esta capacidad es la que le permite propagarse a través de archivos, directorios, discos y sistemas.

- La acción maligna es realizada por la parte del virus que se conoce como payload, éste puede ser programado para hacer cualquier cosa que un programa en ejecución pueda hacer en el ambiente de la víctima. Las acciones que puede hacer por el payload incluyen la corrupción o eliminación de archivos, enviar información sensitiva al autor del virus o a un remitente arbitrario, además de proveer un acceso a la máquina infectada a través de una puerta trasera.

- La acción de multiplataforma se refiere a que el objetivo de los virus no solo son para las máquinas con sistema operativo Windows. Estos también tienen como objetivos otros sistemas operativos. Linux, Solaris y algún otro tipo Unix.

Como se había mencionado anteriormente para que un virus funcione necesita atarse a un programa huésped. La manera de infección de los archivos ejecutables estándar es muy frecuente ya que estos programas son lanzados directamente como parte de una rutina de uso del sistema. Al atarse a un archivo ejecutable, el virus se asegura que será ejecutado. Los sistemas operativos tienen varios tipos de ejecutables, por ejemplo, los sistemas Unix incluyen binarios y una variedad de tipos de scripts que podrían ser infectados por un virus. Por otro lado Windows, soporta dos tipos de ejecutables, cada uno un huésped potencial, para los virus estos son los archivos COM y los EXE.

Los archivos COM contienen una imagen binaria de lo que debe ser cargado directamente en memoria y ejecutado por la computadora.

Los archivos EXE son más difíciles de infectar, estos archivos que se ejecutan de forma nativa cumplen el formato Portable Ejecutable (PE).

Además de los archivos ejecutables, los virus también pueden intentar introducirse a sí mismos en el núcleo del sistema operativo conocido como kernel. Existen muchas formas que puede tomar un virus cuando infecta un sistema, programa, archivo ejecutable. Algunos de estos métodos aplican tanto para archivos COM como para EXE. Las técnicas de infección más comunes son la compañía, sobre-escritura y las técnicas de agregado y pre-agregado.

La técnica de infección de compañía es la técnica más simple ya que el virus puede acompañar a un programa ejecutable, éste se nombra axial, para que el sistema operativo lo lance cuando se solicita la ejecución del programa original. Los especimenes que emplean este método de infección son llamados virus de compañía o de expansión y no modifican el código del ejecutable.

La técnica de infección de sobre escritura como su nombre lo indica reemplaza porciones de código del huésped. Una forma en la que el virus puede llevar a cabo esto es al abrir el programa objetivo en modo de escritura como si fuera un archivo de datos regular y entonces guarda una copia de sí mismo en el archivo, entonces cuando la víctima intenta lanzar el ejecutable, el sistema operativo ejecuta en su lugar el código del virus. Un virus de sobre escritura frecuentemente daña el huésped hasta hacerlo inoperable.

La técnica de infección de pre-agregado, el virus inserta su código al comienzo del programa que éste infecta. Esta es una técnica más elaborada que la empleada por los de sobre escritura y éste generalmente no destruye el programa huésped. Cuando es lanzado un programa con un virus de este tipo, el sistema operativo primero ejecuta el Código del virus debido a que éste se localiza al principio del ejecutable.

La técnica de infección de agregado, aquí el virus inserta su código al final del programa huésped. Para que el virus agregado se ejecute, éste necesita modificar el comienzo del huésped para crear un salto a la sección del archivo donde reside el

código del virus. Después de que el virus realiza sus acciones, regresa el control al programa infectado. Este método de infección, como la técnica del pre-agregado, usualmente no destruye el ejecutable infectado.

La infección de sectores de arranque, se refiere a que cuando se prende una computadora ésta primero ejecuta un conjunto de instrucciones que realizan el hardware y permiten al sistema operativo iniciar. El Código que implementa estas acciones es parte del programa del BIOS que está embebido en los circuitos de la máquina. Debido a que el BIOS no tiene la capacidad de cargar un sistema operativo, éste localiza el primer sector de arranque en el primer disco duro y ejecuta un pequeño programa almacenado en él llamado registro de arranque maestro (MBR – master boot record).

Sin embargo, el MBR tampoco conoce como cargar el sistema operativo, esto se debe a que el equipo puede tener múltiples particiones y varios sistemas operativos instalados, cada uno con sus propios requerimientos de inicio. Al sector de arranque ubicado al comienzo de cada partición se le conoce como el sector de arranque de la partición (PBS – partition boot sector)

A los virus que toman ventaja de la naturaleza ejecutable del contenido del MBR y el PBS se les conoce como virus de sector arranque y lo que hacen es que éstos se atan a sí mismos a uno de los sectores de arranque. Una computadora infectada con un virus de sector de arranque ejecutará el código del virus cuando la máquina inicie.

1.3 Gusanos.

Los gusanos son similares a los virus, pero estos no dependen de archivos portadores para poder contaminar otros sistemas. Pueden modificar el sistema operativo con el fin de ejecutarse automáticamente como parte del proceso de inicialización del sistema.

Un gusano es una pieza de código que se auto replica y propaga a través de las redes y usualmente no requiere de la interacción humana.

Un gusano golpea una computadora, la controla y la usa para buscar y conquistar otros sistemas vulnerables. Cuando este nuevo objetivo está bajo control del gusano, la propagación continúa cuando el gusano brinca a la nueva víctima y sigue buscando más sistemas vulnerables.

Aunque la red es la característica intrínseca que define a los gusanos, es importante reconocer que muchos gusanos se propagan sin la interacción del usuario. Usualmente explotan alguna falla en un sistema y la explotan en forma automatizada.

Los intrusos utilizan los gusanos debido a que ofrecen la escala que no puede ser alcanzada con otro tipo de ataques. Los gusanos utilizan el poder inherente de las grandes redes distribuidas y usan esto para redes indeterminadas. Los intrusos emplean estas capacidades de los gusanos para alcanzar numerosos objetivos, incluyendo el control de un vasto número de sistemas, haciendo el rastreo más difícil y ampliando los daños.

Los componentes de un gusano son la cabeza utilizada para penetrar el objetivo. El motor de propagación de indica su destino. El algoritmo de selección de objetivo y el motor de búsqueda funcionan como pequeños giroscopios para guiar al gusano hacia su destino. El payload es la parte más peligrosa y la que causa el peor daño en el objetivo.

La cabeza del gusano es la parte esencial para que éste pueda tener acceso en la máquina de la víctima. Ellos entran a su objetivo usando la cabeza, una pieza de código que explota una vulnerabilidad en el sistema objetivo. Este exploit, cargado dentro de la cabeza podría penetrar el sistema usando un número muy amplio de errores posibles en el objetivo (victima).

Las técnicas más populares son las siguientes:

- **Exploit de buffer overflow,** este error surge ya que se comenten errores al escribir programas. Ya que no se revisan los tamaños de alguna pieza de datos antes de moverla entre varios buffers de memoria. Esto puede dejar una vulnerabilidad de buffer overflow en el programa. Para explotar este error, un gusano envía más datos al programa que los que esperaba originalmente, desbordando el buffer y corrompiendo varias estructuras críticas de memoria.
- **Ataque de archivos compartidos,** este tipo de ataque se realiza usando archivos compartidos en Windows o NFS de Unix, los usuarios pueden leer o escribir archivos transparentemente a través de la red.
- **Gusanos de correo electrónico**: Suelen propagarse a través de los mensajes valiéndose de la utilización de ciertos programas clientes, reenviándose automáticamente a los contactos de la libreta de direcciones. Algunos de los gusanos más recientes (SirCam, Nimda, Klez, BugBear), pueden incluso, llegar a enviarse a cualquier dirección de correo que encuentren en caché, con lo cual, si en una página visitada se ha incluido una dirección de correo, puede ser utilizada por el gusano, al tener éste la capacidad de rastrearlas.
- **Gusanos de IRC:** Estos se propagan a través de canales de IRC (Chat), empleando habitualmente para ello al mIRC y al Pirch. En este apartado cabe recomendar tener precaución con las transferencias que uno acepte.
- **Gusanos de VBS** (Visual Basic Script): Son gusanos escritos o creados en Visual Basic Script y para su prevención es importante considerar la sugerencia de hacer visibles todas las extensiones en nuestro sistema (para poder identificar y rechazar los archivos que vengan con doble extensión, como es el caso de anexos de correos infectados por SirCam).
- **Gusanos de Windows 32**: Son Gusanos que se propagan a través de las API (Application Program Interface o Application Programming Interface) de Windows, las cuales son funciones pertenecientes a un determinado protocolo de Internet. Las API corresponden al método específico prescrito por un sistema operativo o por cualquier otra aplicación de aplicación mediante el cual un programador que escribe una aplicación puede hacer solicitudes al sistema operativo o a otra aplicación.

Los errores comunes de configuración son los conjuntos de exploits usados por los intrusos, para obtener acceso involucra la explotación de una variedad de errores comunes de configuración. Varios administradores de sistemas y usuarios cometen los mismos errores al configurar sus equipos, permitiendo alguna forma de acceso que ellos nunca contemplaron.

En lo que refiere al motor de propagación, es la parte que se activa después de que la cabeza actúa, el gusano debe transferir el resto de su cuerpo al sistema. En algunos casos, la cabeza en sí lleva al gusano completo a la víctima. Si el exploit de la cabeza puede ser usado para llevar código, un gusano eficiente puede cargar todo su contenido dentro de la cabeza.

El motor de búsqueda de un gusano es un algoritmo que utiliza direcciones generadas por el motor de selección, el gusano busca a través de la red para identificar víctimas. Utilizando el motor de búsqueda, el gusano envía uno o más paquetes a una víctima potencial para medir si el exploit del gusano funcionará. Cuando es encontrada una nueva víctima el gusano se propaga a ella y el proceso completo de propagación se repite una y otra vez. La cabeza abre la puerta, el gusano se propaga, el payload se ejecuta, se seleccionan nuevas víctimas, y entonces la búsqueda comienza nuevamente.

Ya que se han mencionado los componentes para construir un gusano, los obstáculos que encuentran los gusanos y las estrategias que utilizan para evadir barreras son las siguientes. Al entender las dificultades que los gusanos encuentran para llegar a su objetivo podemos comenzar a idear cómo poder defendernos de este malware.

1.4 Troyanos.

Los Troyanos son programas capaces de alojarse en computadoras y permitir el acceso a usuarios externos, a través de una red local o de Internet, con el fin de recabar información o controlar remotamente a la máquina, pero sin afectar el funcionamiento de ésta.

Existen troyanos unifuncionales y multifuncionales, los primeros se refieren al tipo de troyano que solo realiza una función, como abrir una puerta trasera, robar una contraseña, espiar un equipo o bajar información. Mientras tanto los troyanos multifuncionales son aquellos que realizan una o más funciones al mismo tiempo. Algunos creadores de virus prefieren crear troyanos multifuncionales en vez de paquetes de troyanos unifuncionales, ya que se puede realizar muchas tareas con tan solo un troyano, ahora mencionaremos algunos tipos de troyanos y sus principales características.

- **Una familia de troyanos se dedica a robar contraseñas**, por lo general, las contraseñas para entrar al sistema de los equipos de las víctimas. Estos troyanos buscan los archivos del sistema que contienen información confidencial tales como contraseñas y números de acceso a Internet para luego enviar esta información a una dirección de correo electrónico contenida en el cuerpo del

troyano. La información secuestrada será usada por el dueño o usuario del programa ilegal.

  - Algunos troyanos pueden robar otro tipo de información, detalles de la configuración del sistema (memoria, espacio libre, detalles del sistema operativo), detalles del cliente de correo electrónico, direcciones IP, detalles de inscripción, contraseñas de juegos en línea.

- **Los troyanos clickers** remiten a los equipos de las víctimas a determinados sitios web o recursos de Internet. Los clickers también envían a los navegadores determinadas instrucciones o reemplazan los archivos del sistema donde se guardan las direcciones de Internet.
  - Los clickers suelen usarse para elevar la posición de determinados sitios en las clasificaciones con objetivos publicitarios, organizar ataques DoS contra el servidor o sitio especificado o para conducir a la víctima hacia un recurso infectado, donde será atacada por otros programas maliciosos (virus o troyanos).
- **Los troyanos de descarga o downloaders**, descargan e instalan nuevos programas maliciosos o publicitarios en el equipo de la víctima. Luego el downloader ejecuta los nuevos programas maliciosos o los registra para ser ejecutados automáticamente, de conformidad con las exigencias del sistema operativo local. Todo esto se hace sin que el usuario se dé cuenta y sin su consentimiento.
- **Los troyanos droppers** se utilizan para instalar otros programas maliciosos en los equipos de las víctimas. Los droppers instalan su carga útil sin mostrar ninguna notificación o bien mostrando un mensaje falso sobre un error en un archivo comprimido o en el sistema operativo. El nuevo programa malicioso es dejado en una ubicación específica o en un disco local para luego ser ejecutado.
  - Los hackers usan estos programas para alcanzar dos objetivos, como ocultar o disimular la instalación de otros troyanos o virus o engañar a las soluciones antivirus que son incapaces de analizar todos los componentes.
- **Lo troyanos proxies** funcionan como servidores proxy y proporcionan acceso anónimo a Internet desde los equipos de las víctimas. Hoy en día estos troyanos son muy populares entre los spammers que siempre necesitan de equipos adicionales para hacer sus envíos masivos. Los programadores de virus con frecuencia incluyen proxies-troyanos en sus paquetes de troyanos y venden las redes de equipos infectados a los spammers.
- La familia de **los troyanos spies** incluye una variedad de programas espías y key loggers, que monitorean la actividad del usuario en el equipo afectado y luego envían esta información a su dueño. Los espías troyanos recolectan varios tipos de información.

1.5 Backdoors.

Una puerta trasera(o Backdoor) es un software que permite el acceso al sistema de la computadora ignorando los procedimientos normales de autenticación.

Una puerta trasera es un programa que permite a un atacante brincar los controles normales de seguridad, como son cortafuegos, filtros de acceso de Ip o

cualquier configuración de seguridad que contenga un servidor, de esta manera un atacante obtiene el acceso definido.

Los tipos de puertas traseras son:

- **Por escalamiento local de privilegios**. Esta puerta trasera permite a un atacante con acceso al sistema, cambiar sus privilegios a nivel de administrador o root. Con estos privilegios el atacante puede reconfigurar el equipo y acceder a cualquier archivo almacenado.

- **Ejecución remota de comandos individuales**. Usando este tipo de puerta trasera el atacante puede enviar mensajes al equipo destino para ejecutar un comando a la vez. La puerta trasera ejecuta el comando y regresa la salida al atacante.

- **Acceso remoto por líneas de comando**. Conocido también como shell remoto, este tipo de puerta trasera permite al atacante introducir comandos directamente en una terminal del equipo víctima a través de la red. El atacante puede utilizar todas las facilidades de un shell, como ejecutar y automatizar tareas, puesto que este simula tener acceso al teclado del sistema.

- **Por control remoto de la GUI** *(Graphical User Interface)*, En lugar de utilizar comandos en un shell, algunas puertas traseras permiten controlar la interfase de usuario. Control de los movimientos del mouse, insertar teclazos en una aplicación. Con el control de la GUI el atacante puede observar también todas las actividades del usuario en el equipo comprometido.

Una instalación de una puerta trasera, se debe realizar por principio de cuentas en el equipo víctima. Si el atacante tuvo acceso en cualquier momento podría haberla instalado para el acceso a futuro, otra forma de instalarlas es por medio de la explotación de una vulnerabilidad, también puede ser por configuraciones inseguras de algunos sistemas. Otra manera de instalar puertas traseras es mediante virus o gusanos.

Un aspecto importante es que las puertas traseras se ejecutan con los mismos permisos del usuario que inició la ejecución, es decir, si la puerta se inició en una sesión de root o administrador, el atacante obtendrá acceso remoto de root.

Para sistemas operativos Windows, hay 3 formas de iniciar una puerta trasera. La primera de ellas es mediante fólderes de inicio automático de aplicaciones. Aquí el atacante deposita una liga en estos fólderes hacia un archivo que es la puerta trasera.

En la parte de los registros, en el registry de Windows también se pueden incluir aplicaciones para ser ejecutadas automáticamente.

Por tareas programadas o task scheduler. Un ataque puede programar la ejecución de la puerta trasera en ciertos horarios y fechas o cuando ocurra un cierto evento.

El inicio de las puertas traseras en Unix es más variado y hay más formas de iniciarlas automáticamente, esto es, desde modificar archivos, agregarlos a cierto directorio o agregarlos a la lista de ejecución programada (crontab).

Cuando un sistema Unix es inicializado se ejecutan una serie de programas y scripts. El primer proceso que se levanta es el demonio INIT. El archivo /etc/initab contiene un script que indica qué aplicaciones iniciar, un atacante podría agregar una línea a este archivo para abrir la puerta trasera.

En Unix el inittab usualmente determina qué otros scripts lanzar, en general son los scripts que inician los demonios de correo, servidor web, ambiente gráfico, etc. Y se encuentran ubicados en /etc/rc.d o /etc/init.d

En un sistema Unix mínimo se encuentran 20 o más archivos, cada uno con una longitud de 10 a 50 líneas, proveyendo de un buen lugar para insertar puertas traseras.

1.6 Exploits.

Un exploit es un programa o técnica que aprovecha una vulnerabilidad. Los exploits dependen de los sistemas operativos y sus configuraciones, de las configuraciones de los programas que se están ejecutando en una computadora y de la LAN donde están. Pero para poder explotar un sistema es necesario tener una vulnerabilidad. Una 'vulnerabilidad' es un error en la configuración del servicio o bien un error en la programación del servicio. Pueden estar creados en cualquier lenguaje. Los exploit para Windows suelen estar hechos en C y/o ensamblador, mientras que los de Unix pueden estar hechos también en perl por ejemplo, aunque suelen estar hechos en C.

Exploit es el término con el que se denomina en el entorno "hacker" a un método concreto de usar un error de algún programa (bug) para entrar en un sistema informático. Generalmente, un exploit suele ser un programa que se aprovecha de algún error del sistema operativo, por ejemplo, para obtener los privilegios del administrador y así tener un control total sobre el sistema. Aunque como decíamos, un exploit no tiene por qué ser un programa, por definición es simplemente una forma de sacar provecho de un error en un programa (explotar un error).

Para su uso no se puede establecer una regla general porque cada uno ataca un problema diferente y por lo tanto es una buena costumbre leer el código fuente para saber cómo compilarlo y ejecutarlo con éxito.

Se puede diferenciar a los exploit en locales y remotos. Los exploits locales actúan en la máquina en la que están. Por tanto si se desea atacar otra máquina, se denomina exploit remoto, este primero se tendrá que subir el código y luego conseguir ejecutarlo para que funcione allí ya que de otra manera actuará donde se ejecute.

Un acceso a otra maquina es mediante un puerto este es un punto de acceso a los servicios que se tienen en una computadora, estos están divididos en privilegiados y no privilegiados. Cuando el puerto tiene 16 bits, hay 65535 números disponibles. Los números del 1 al 1023 tienen puertos privilegiados y son reservados para servicios que corre el usuario root, del puerto 1024 hasta el 65535 son puertos no privilegiados y son

utilizados para cualquier servicio. Las aplicaciones más conocidas se escuchan en los siguientes puertos.

Port	State	Service
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
25/tcp	open	smtp
70/tcp	open	gopher
79/tcp	open	finger
80/tcp	open	http

Una shellcode es básicamente una serie de instrucciones en ensamblador que hace algo de lo cual sacamos provecho, como ejecutar un /bin/sh para obtener, copiar y poner un shell con suid root. También se utiliza para poder ejecutar un código después de haber sobrescrito la dirección de retorno de un programa/función mediante un overflow o cualquier otro método válido. Este método funciona colocando en el valor de la dirección de retorno la dirección de sector de memoria donde se encuentra nuestra shellcode.

1.7 Spyware.

Los programas espía o spyware son aplicaciones que recopilan información sobre una persona u organización sin su conocimiento. La función más común que tienen estos programas es la de recopilar información sobre el usuario y distribuirlo a empresas publicitarias u otras organizaciones interesadas, pero también se han empleado en círculos legales para recopilar información contra sospechosos de delitos, como en el caso de la piratería de software. Además pueden servir para enviar a los usuarios a sitios de Internet que tienen la imagen corporativa de otros, con el objetivo de obtener información importante.

Puede tener acceso por ejemplo a: correo electrónico y contraseña; dirección IP y DNS; teléfono, país; páginas que se visitan, qué tiempo se está en ellas y con qué frecuencia se regresa; qué software está instalado en el equipo y cuál se descarga; qué compras se hacen por Internet; tarjeta de crédito y cuentas de banco.

Los programas espía pueden ser instalados en una computadora mediante un virus, un troyano que se distribuye por correo electrónico, como el programa Magia Lantern desarrollado por el FBI, o bien puede estar oculto en la instalación de un programa aparentemente inocuo.

Los programas de recolección de datos instalados con el conocimiento del usuario no son realmente programas espías si el usuario comprende plenamente qué datos están siendo recopilados y a quién se distribuyen.

Los cookies son un conocido mecanismo que almacena información sobre un usuario de Internet en su propia computadora, y se suelen emplear para asignar a los visitantes de un sitio de Internet un número de identificación individual para su reconocimiento subsiguiente. Sin embargo, la existencia de los cookies y su uso generalmente no están ocultos al usuario, quien puede desactivar el acceso a la

información de los cookies. Sin embargo, dado que un sitio Web puede emplear un identificador cookie para construir un perfil del usuario y éste no conoce la información que se añade a este perfil, se puede considerar a los cookies una forma de spyware. Por ejemplo, una página con motor de búsqueda puede asignar un número de identificación individual al usuario la primera vez que visita la página, y puede almacenar todos sus términos de búsqueda en una base de datos con su número de identificación como clave en todas sus próximas visitas (hasta que el cookie expira o se borra). Estos datos pueden ser empleados para seleccionar los anuncios publicitarios que se mostrarán al usuario, o pueden ser transmitidos (legal o ilegalmente) a otros sitios u organizaciones.

1.8 Botnets.

La palabra bot es abreviatura de robot. Robots o programas automatizados que son con frecuencia utilizados en el Internet. Estos robots son utilizados para realizar búsquedas o respuestas de algún patrón automáticamente. Un buen programador puede crear su propio bot o personalizar uno existente, ayudando con esto a esconder el bot de las medidas básicas de seguridad y a su fácil propagación.

Los bots son procesos, con los cuales ejecutan un número de operaciones automáticas. El control que toman estos bots se basa frecuentemente en el envío de comandos que el atacante envía a través de un canal. La administración del bot requiere de un mecanismo de autenticación y autorización, lo que significa que solamente el dueño puede hacer uso del canal.

Una importante ventaja de los bots es su capacidad de propagarse rápidamente hacia otros equipos. Cuidadosamente planeado el proceso de infección ayuda a obtener mejores resultados en un mejor tiempo. Una estructura más compleja de bots es la llamada botnet que es un número de n de bots conectados a un solo canal y en espera de comandos.

En muchos casos, los bots infectan a usuarios caseros, servidores de universidades y redes de pequeñas compañías. Esto se debe a que en muy pocos casos estos equipos son constantemente monitoreados, y algunas veces los dejan totalmente desprotegidos.

La razón de esto es por la carencia de políticas de seguridad. Otro de los motivos es debido a que los usuarios caseros con conexión ADSL no son debidamente informados del riesgo al que están expuestos y por lo tanto no hacen uso de software que mitigue este riesgo.

Los botnets son usadas frecuentemente para generar ataques DDoS (Dinamic Denial of Services, Negación de servicios Dinámica por sus siglas en inglés) un intruso puede controlar un extenso número de equipos comprometidos desde un equipo remoto.

Un ataque de DDoS es una variante del ataque Flooding DoS, este consiste en enviar muchas peticiones a un servido y acabar con sus recursos, de esta manera el objeto saturá la red utilizando todo el ancho de la banda disponible. La mejor manera para lanzar este tipo de ataque es controlando diferentes equipos, así cada host tendrá

su propio ancho de banda y con ésta pueden concentrar todo en un solo objetivo, distribuyendo así el ataque.

Las botnets son un medio ideal para los spammers. Ellos utilizan medios para intercambiar direcciones de correo electrónico y para controlar equipos zombies ocupados para el envío masivo de spam como para los ataques DDoS. Un simple correo spam podría ser enviado por la botnet para ser distribuido, de esta manera se envían a cualquier contacto así es como se realiza un ataque por medio de un bot.

Los bots también pueden ser eficientes medios para manejar el arte del sniffing que su acción es capturar tráfico y detectar posible información valiosa.

Los equipos comprometidos a través de los bots pueden ser utilizados como repositorios para la distribución ilegal de software, música, videos, herramientas, etc.

1.9 Vulnerabilidades del software.

En términos de computación y/o seguridad de sistemas, una Vulnerabilidad es una falla en la configuración o en la instalación de un programa (software) o de un dispositivo (hardware), que permite a una persona u otro equipo electrónico, obtener acceso de manera no autorizada al programa, datos o dispositivo.

El término 'Vulnerabilidad' se menciona a menudo en conexión con la seguridad de las computadoras en muchos diferentes contextos.

En su sentido más amplio, el término 'Vulnerabilidad' se refiere a la violación de una política de seguridad. Esto puede deberse a reglas de seguridad inadecuadas o a problemas dentro del mismo software. En teoría, todos los sistemas de computadoras tienen vulnerabilidades, de cuya seriedad depende que sean o no usados para causar un daño al sistema.

Según la terminología de MITRE's CVE[1] (un grupo de investigación y desarrollo):

Una vulnerabilidad universal es un estado en un sistema de computadoras o un grupo de sistemas que permite que un atacante:

- Ejecute órdenes como otro usuario.
- Tenga acceso a los datos de acceso restringido.
- Hacerse pasar por otra entidad.
- Conduzca una denegación de servicio.

MITRE cree que cuando un ataque se hace posible por una debilidad o una política de seguridad inapropiada, es mejor describirla como "exposición".

Una exposición es un estado en un sistema de computadoras (o grupo de sistemas) que no es una vulnerabilidad universal, pero:

[1] http://cve.mitre.org/about/terminology.html

- Permite que un atacante reúna información sobre las actividades del sistema.
- Permite que un atacante disimule sus actividades.
- Incluye una función que se comporta como se espera, pero puede ser fácilmente puesta en peligro.
- Es un punto primario de entrada que un atacante puede intentar usar para obtener acceso al sistema o a los datos.
- Un problema de acuerdo a alguna política de seguridad razonable.

Cuando se trata de obtener acceso no autorizado al sistema, un intruso usualmente primero realiza un escaneo de rutina (o investigación) del blanco, reúne cualquier dato 'expuesto' y luego explota las debilidades o vulnerabilidades de las políticas de seguridad.

Por esta razón, las vulnerabilidades y exposiciones son aspectos importantes que deben ser considerados cuando se asegura un sistema contra el acceso no autorizado.

1.9.1 Análisis de vulnerabilidades[2].

Para analizar vulnerabilidades existen diversos caminos entre los que destacan los siguientes:

- Ingeniería inversa.

  SWIFI (software-implemented fault injection).
  Análisis de flujo de información.
  Diferencia en binarios.
  Depurador de aplicaciones.

- Análisis estático.

  Flujo de entradas
  Herramientas de auditoria de código.

1.9.2 El ciclo de vida de la vulnerabilidad.

Una vulnerabilidad independientemente de su naturaleza cumple con un ciclo de vida, y no importa que ésta sea descubierta en un software de código abierto o cerrado, en este ciclo de vida no hay tiempos predefinidos, ya que los tiempos van a ser dictados por la complejidad de la vulnerabilidad y la calidad y disponibilidad de la información.

Se han definido cinco etapas en el ciclo de vida de la vulnerabilidad, cabe aclarar que en ocasiones, se altera el orden o se fusionan entre sí varias de ellas, las etapas son las siguientes (ver figura 1.1)

- Descubrir la vulnerabilidad.

[2] http://cve.mitre.org/docs/vuln-trends/vuln-trends.pdf

- Anunciar la vulnerabilidad.
- Divulgación de la misma
- Liberación del parche o actualización.
- Usuarios instalan el parche.

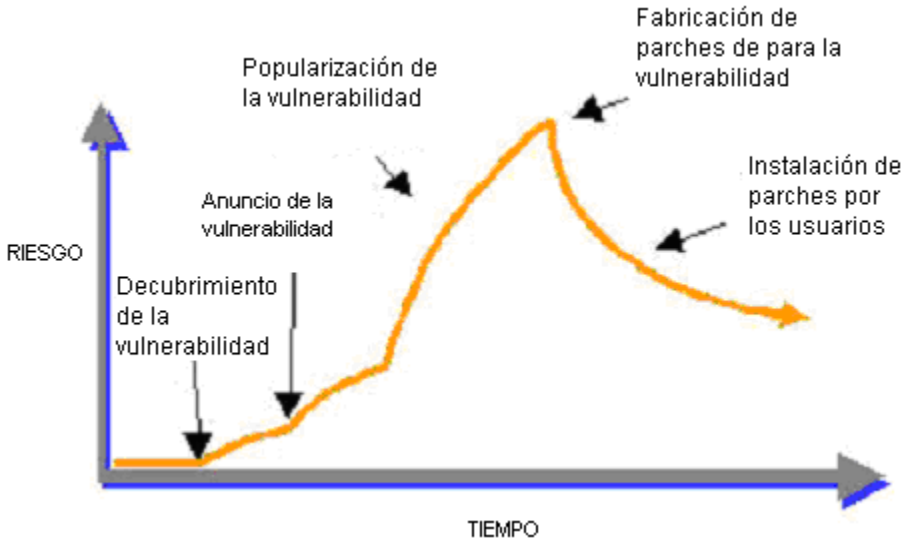

Figura 1.1: Ciclo de vida de las Vulnerabilidades.

Cabe recalcar que los hackers no crean las vulnerabilidades ellos simplemente las descubren y las explotan, y como antes se mencionó, en gran medida se deben a una pobre ingeniería de software y a utilizar implementaciones de lenguajes inseguros como C, sin las debidas precauciones.

En la primera fase, se comienzan a hacer evidente la existencia de algún problema, aquí es necesario mencionar que la vulnerabilidad existe pero aun no puede ser explotada, regularmente sólo un círculo muy pequeño conoce la existencia de ésta.

En la segunda fase, es el periodo de tiempo que pasa entre el descubrimiento de la vulnerabilidad y antes de ser anunciada, en esta fase los analistas hacen las pruebas suficientes para determinar el entorno sobre el que puede replicarse.

En la tercera fase, se da a conocer la vulnerabilidad, aquí el riesgo se incrementa ya que terceros pueden desarrollar pruebas de concepto. Regularmente este sería el proceso que se tendría que realizar para obtener un exploits, pero es necesario mencionar que el día que el descubridor de la vulnerabilidad genere la prueba de concepto sin haber anunciado la vulnerabilidad tendremos un exploit del día cero, al cual los desarrolladores del software afectado difícilmente podrán responder con el parche o actualización en un tiempo razonable.

En la cuarta fase, se empiezan a liberar herramientas automáticas para explotar dicha vulnerabilidad, es decir aquella persona que obtenga dicha herramienta y la

ejecute sobre un sistema vulnerable podrá dañar al sistema (scriptkid), en este punto del ciclo de vida la capacidad de explotación de dicha vulnerabilidad se vuelve exponencialmente crítica, el fabricante emite un parche o actualización de la aplicación o sistema operativo que resuelva esta vulnerabilidad.

Quinta fase, esta fase se enfoca en el tiempo entre la liberación e instalación del parche o actualización por parte de los administradores pude ser crítico, y el tiempo de explotación de la vulnerabilidad es altamente crítico, y el ciclo puede repetirse con el mismo software u otro.

Desgraciadamente el ciclo que a estas alturas se denomina romper-parchar no resuelve la seguridad integral de las aplicaciones, además de resultar extremadamente costoso, encontramos lo siguiente:

- Solo se parchan los huecos o vulnerabilidades conocidas, pero los crackers pueden localizar otros problemas que no se reportan a los desarrolladores.

- Los parches solo reparan el síntoma de un problema, pero no hacen nada en la dirección correcta, entender el cambio en el ciclo de vida del software.

- La mayoría de las veces las actualizaciones no son aplicadas por el administrador del sistema, en forma oportuna.

Mientras no se cambie la visión de que la seguridad existe o es necesaria solo cuando alguno de sus productos presenta problemas este ciclo se volverá a repetir una y otra vez.

1.9.3 Ejemplos de vulnerabilidades.

1.9.3.1 Buffer overflow.

Lenguajes de programación como C permite al desarrollador crear espacios de almacenamiento en memoria en tiempo de ejecución, en dos diferentes secciones: en la pila o stack, y en la memoria libre o también llamada heap. Generalmente el uso de la memoria heap se utiliza a través de mecanismos dinámicos como un new o un mallow, en cambio la asignación de las variables no estáticas, así como el paso de parámetros se realiza en la sección de la pila, el desarrollador debe verificar que cuando se escriban datos sobre los buffer, estos no traten de escribir datos fuera de esta porción de memoria, es necesario mencionar que el leguaje C tiene desgraciadamente muchas funciones que son vulnerables, entre las que destacan el strcpy, scan, y todas aquellas que manejen cadenas.

Si los datos exceden la capacidad del buffer, estos simplemente terminaran afectando el funcionamiento del programa, como a continuación se explica, un usuario malicioso puede tomar ventajas de un buffer overflow sobrescribiendo la pila con código propio del atacante, y regularmente se aprovecha para invocar un shell con los privilegios de root o del administrador del sistema.

1.9.3.2 Cross Site Scripting(XSS).

También se le conoce como programación de sitios cruzados, y se identifica por las siglas XSS, y produce cuando el usuario introduce datos malintencionados en un sitio Web, por ejemplo enviando el mensaje malintencionado a un grupo de noticias.

1.9.3.3 Race Conditiong.

Esta vulnerabilidad se produce cuando varios procesos están compitiendo por un proceso, y entre dos operaciones, a uno de ellos se le cuela otro que cambia las características del recurso. Otra definición es la siguiente, el Race Conditiong se genera cuando dos o más procesos acceden a un recurso compartido sin control, de tal manera que el resultado combinado de este acceso depende del orden de llegada, de las órdenes.

Los anteriores ejemplos de vulnerabilidades son solo algunos, y se da una breve explicación debido a que son muy ilustrativos y fáciles de comprender.

Capítulo 2

Análisis de sistemas operativos y herramientas de terceros.

En este capítulo se mencionan las características de los sistemas operativos que se utilizan en el presente proyecto de tesis, así como la metodología a seguir para la realización de un análisis forense en un sistema Windows Online, además de la evaluación de herramientas que pueden ayudar en la recolección de datos a la hora de realizar un análisis de este tipo y las cuáles pueden ayudar a realizar el trabajo de una manera semiautomática.

2. Análisis preliminar y de sistemas operativos.

2.1 Windows.

El sistema operativo Windows , introdujo algunas modificaciones respecto a sus predecesores, como el sistema de archivos NTFS 5, la capacidad de cifrar y comprimir archivos, así como las mejoras en el sistema de componentes COM, introduciendo COM+ que unificó en un solo paquete de servicios integrados, la tecnología COM y MTS de Windows NT4, con nuevas ventajas. Este sistema fue el primer intento de Microsoft por juntar su versión MS-DOS (Windows 95, 98, ME) y NT (3.51, 4). Esta versión ha tenido mucho éxito en empresas, que todavía hoy la usan, pero entre los usuarios de hogares no tuvo mucho éxito.

Los requerimientos mínimos para Windows son un Pentium 166 MHz, 64 Mb de RAM y 2Gb de disco duro, con espacio libre de al menos 1 Gb. A partir de la versión de Windows 2000 las mejoras destacadas son la estabilidad del sistema y el aumento en seguridad respecto a las versiones anteriores de Windows, muy criticadas por sus continuas fallas. Hoy en día Windows 2000 sigue considerándose por muchos el mejor sistema de la marca Microsoft. Actualmente Microsoft ha publicado 4 service packs corrigiendo la mayoría de errores y aumentando todavía más su estabilidad y seguridad.

La familia de servidores Microsoft Windows va más allá de proporcionar lo esencial, tal como archivar, imprimir y comunicarse. Está diseñada y construida específicamente para permitir a las empresas utilizar económica y confiablemente tecnologías emergentes para mejorar la rentabilidad del negocio e incrementar su agilidad en un mercado en constante evolución como lo es el electrónico.

Hasta los servidores más confiables se encuentran fuera de línea en ocasiones, solamente por mantenimiento periódico. Para aplicaciones críticas, los negocios requieren un mecanismo de respaldo para asegurarse que los usuarios no sean interrumpidos mientras un servidor no está disponible. Las ediciones de Advanced Server y Datacenter Server de la familia Windows Server le permiten incrementar la disponibilidad de su sistema utilizando tecnologías de agrupación incluidas en su sistema operativo, que le permitirán acoplar servidores para manejar tareas específicas.

Para asegurarse de que puede utilizar los más actuales y poderosos dispositivos, Microsoft Windows es compatible con una amplia gama de hardware y periféricos.

2.1.1 Problemas de seguridad para equipos Windows.

Aún cuando siga considerándose el sistema más seguro de Microsoft, Windows aun tiene demasiadas amenazas de seguridad, de las cuales se están analizando, ya que es el sistema operativo más utilizado en las empresas. A medida que evolucionan los sistemas de Información (TI), también lo hacen las amenazas a la seguridad que estos pueden sufrir. Para proteger su entorno de forma eficaz contra los ataques, se necesita conocer con detalle los peligros con los que puede encontrarse.

Muchas organizaciones no tienen en cuenta el factor del lugar donde se realizan los ataques, pues asumen que un ataque grave sólo puede venir del exterior

normalmente, a través de su conexión a Internet. En CSI/FBI Computer Crime and Security Survey, en una encuesta realizada sobre delitos y seguridad informáticos de CSI/FBI, el 31% de los encuestados citaron sus sistemas internos como un punto de ataque frecuente. Sin embargo, muchas empresas pueden no estar al corriente de que se están dando ataques internos, básicamente porque no comprueban si existen[1].

No existe un entorno de información (TI) totalmente seguro y al mismo tiempo útil. Al examinar su entorno, deberá:

- Evaluar los riesgos que sufre actualmente
- Determinar un nivel de riesgo aceptable
- Detener el riesgo a ese nivel o por debajo del mismo.
- Los riesgos se reducen aumentando la seguridad de su entorno.
- Cuanto más alto sea el nivel de seguridad de una organización, más costosa resultará su implementación y más posibilidades habrá de que se reduzca su funcionalidad.
- Tras evaluar los posibles riesgos, quizá tenga que reducir el nivel de seguridad para conseguir aumentar la funcionalidad y reducir el costo.

Para entender los principios de la administración de riesgos, es necesario entender algunos términos básicos utilizados en el proceso de administración de riesgos. Estos incluyen recursos, amenazas, vulnerabilidades, explotaciones y contramedidas.

Un recurso es cualquier elemento de su entorno que intente proteger. Puede tratarse de datos, aplicaciones, servidores, ruteadores e incluso personas. El objetivo de la seguridad es evitar que sus recursos sufran ataques. Una parte importante de la administración de riesgos consiste en determinar el valor de sus recursos.

Una amenaza es una persona, un lugar o un elemento que puede tener acceso a los recursos y dañarlos. En la tabla 2.1 se muestran distintos tipos de amenazas con ejemplos.

Tipo de amenaza	Ejemplos
Natural y física	Fuego, agua, viento, terremoto Corte eléctrico
No intencionada	Empleados no informados Clientes no informados
Intencionada	Atacantes Terroristas Espías industriales Gobiernos Código malicioso

Tabla 2.1: Amenazas a entornos informáticos.

[1] http://www.gocsi.com/forms/fbi/csi_fbi_survey.jhtml

Una vulnerabilidad es un punto en el que un recurso es susceptible de ser atacado. Se puede interpretar como un punto débil. Las vulnerabilidades se dividen a menudo en las categorías que se muestran en la tabla 2.2.

Tipo de vulnerabilidad	Ejemplos
Física	Puertas sin cerrar
Natural	Sistema antiincendios averiado
De hardware y software	Software antivirus no actualizado
De medios	Interferencia eléctrica
De comunicación	Protocolos no cifrados
Humana	Procedimientos no seguros de asistencia técnica

Tabla 2.2: Vulnerabilidades en entornos informáticos.

Una amenaza que se aprovecha de una vulnerabilidad de su entorno puede tener acceso a un recurso. Este tipo de ataque se denomina explotación de recursos y se puede realizar de varias maneras. En la tabla 2.3 se incluyen algunas de las más comunes.

Tipo de explotación	Ejemplo
Explotación de vulnerabilidad técnica	Ataques a fuerza bruta Desbordamiento del búfer Problemas de configuración Ataques repetidos Secuestro de sesión
Obtención de información	Identificación de dirección Identificación de sistema operativo Exploración de puertos Búsqueda de servicios y aplicaciones Exploración de vulnerabilidades Análisis de respuestas Enumeración de usuarios Destrucción de documentos Fuga de dispositivos inalámbricos Ingeniería social
Denegación de servicio	Daño físico Eliminación de recursos Modificación de recursos Saturación de recursos

Tabla 2.3: Explotaciones en entornos informáticos.

Cuando una amenaza utiliza una vulnerabilidad para atacar un recurso, las consecuencias pueden ser graves. En la tabla 2.4 se muestran algunos de los resultados de explotaciones que se produce en su entorno y algunos ejemplos.

Resultados de una explotación	Ejemplos
Pérdida de confidencialidad	Acceso no autorizado Reasignación de privilegios Personificación o robo de identidad
Pérdida de integridad	Daños en datos Desinformación
Pérdida de disponibilidad	Denegación de servicio

Tabla 2.4: Resultados de explotaciones.

Para reducir el riesgo en su entorno, debe usar una estrategia de defensa en profundidad para proteger los recursos de amenazas externas e internas. El término defensa en profundidad describe la aplicación de contramedidas de seguridad con el fin de formar un entorno de seguridad cohesivo sin un solo punto de error.

Con el despliegue de varias capas de seguridad, que ayudan a garantizar, si es que se pone en peligro una capa, las otras ofrecerán la seguridad necesaria para proteger sus recursos. Por ejemplo, que el servidor de seguridad de una organización esté en peligro, no debe significar que un atacante pueda tener acceso sin trabas a los datos más confidenciales de la organización. En el caso ideal, cada capa debe proporcionar diferentes formas de contramedidas para evitar que se utilice el mismo método de explotación en varias capas.

En la figura 2.1 se muestra una estrategia de defensa en profundidad eficaz:

Figura 2.1: Estrategia de defensa en profundidad.

2.1.1.1 Defensa de datos.

Para muchas empresas, uno de los recursos más valiosos son los datos. Si estos datos cayeran en manos de la competencia o sufrieran daños, tendrían problemas importantes.

A nivel de cliente, los datos almacenados localmente son especialmente vulnerables. Si se roba un equipo portátil, es posible realizar copias de seguridad, restaurar y leer los datos en otro equipo, aunque el delincuente no pueda conectarse al sistema.

Los datos se pueden proteger de varias formas, incluido el cifrado de datos mediante EFS (Encrypting File Service) o sistemas de cifrado de otros fabricantes y la modificación de las listas de control de acceso discrecional en los archivos.

2.1.1.2 Defensa de aplicaciones.

Como una capa de defensa más, el refuerzo de las aplicaciones es una parte esencial de cualquier modelo de seguridad. Muchas aplicaciones utilizan el subsistema de seguridad de Windows para proporcionar seguridad. No obstante, es responsabilidad del programador incorporar la seguridad en la aplicación para proporcionar una protección adicional a las áreas de la arquitectura a las que la aplicación puede tener acceso. Una aplicación existe en el contexto del sistema, de modo que siempre se debe tener en cuenta la seguridad de todo el entorno al considerar la seguridad de una aplicación.

Se debe probar en profundidad el cumplimiento de la seguridad de cada aplicación de la organización en un entorno de prueba antes de permitir que se ejecute en una configuración de producción.

2.1.1.3 Defensa de hosts.

Debe evaluar cada host del entorno y crear directivas que limiten cada servidor sólo a las tareas que tenga que realizar. De este modo, se crea otra barrera de seguridad que un atacante deberá superar antes de poder provocar algún daño.

Un modo de hacerlo consiste en crear directivas individuales en función de la clasificación y el tipo de datos que contiene cada servidor. Por ejemplo, las directivas de una organización pueden estipular que todos los servidores Web son de uso público y, por lo tanto, sólo pueden contener información pública. Sus servidores de bases de datos están designados como confidenciales de la empresa, lo que significa que la información debe protegerse a cualquier precio. De ahí las clasificaciones que se muestran en la tabla 2.5.

Tabla 2.5: Clasificación de servidores.

Valor	Definición
De uso público	La distribución de este material no está limitada. Incluye información de marketing, materiales de venta e información seleccionada para uso público. Los datos incluidos en servidores de Internet públicos deben ser de uso público.

Sólo para uso interno	La revelación de esta información es segura para la distribución interna, pero puede provocar daños considerables a la organización si se hace pública. Debe colocarse por lo menos un servidor de seguridad entre esta información e Internet.
Confidencial de la empresa	La revelación de esta información puede provocar daños graves a la organización en conjunto. Esta información es del tipo más confidencial y sólo se expone si es absolutamente necesario. Deben colocarse por lo menos dos servidores de seguridad entre esta información e Internet.

2.1.1.4 Defensa de redes.

Si dispone de una serie de redes en la organización, debe evaluarlas individualmente para asegurarse de que se ha establecido una seguridad apropiada. Si un ruteador sufre un ataque eficaz, puede denegar el servicio a partes enteras de la red.

Debe examinar el tráfico admisible en sus redes y bloquear el que no sea necesario. También puede considerar el uso de IPSec para cifrar los paquetes en sus redes internas y SSL para las comunicaciones externas. Asimismo, debe supervisar la existencia de detectores de paquetes en la red, que sólo deben usarse bajo controles estrictos.

2.1.1.5 Defensa de perímetros.

La protección del perímetro de su red es el aspecto más importante para detener los ataques externos. Si su perímetro permanece seguro, la red interna estará protegida de ataques externos. La organización debe disponer de algún tipo de dispositivo de seguridad para proteger cada punto de acceso a la red. Es necesario evaluar cada dispositivo, decidir qué tipos de tráfico se permiten y desarrollar un modelo de seguridad para bloquear el resto del tráfico.

Los servidores de seguridad son una parte importante de la defensa del perímetro. Necesitará uno o más servidores de seguridad para asegurarse de minimizar los ataques externos, junto con la auditoría y la detección de intrusiones para estar seguro de detectar los ataques en caso de que se produzcan.

También debe recordar que, para las redes que permiten el acceso remoto, el perímetro puede incluir los equipos portátiles del personal e incluso los equipos domésticos. Deberá asegurarse de que estos equipos cumplen con los requisitos de seguridad antes de que se conecten a la red.

2.1.1.6 Seguridad física.

Hablamos de un lugar inseguro, por que los usuarios no autorizados pueden obtener acceso físico a los equipos. Un ataque de denegación de servicio muy eficaz puede ser simplemente quitar el sistema de alimentación de un servidor o las unidades de disco. El robo de datos puede producirse si alguien roba un servidor o incluso un equipo portátil.

Debe considerar la seguridad física como un elemento fundamental para su estrategia de seguridad global. Una prioridad principal será la de establecer una seguridad física para las ubicaciones de los servidores. Puede tratarse de salas de servidores del edificio o de centros de datos enteros.

También debe tener en cuenta los accesos a los edificios de la organización. Si alguien puede tener acceso a un edificio, puede tener oportunidades para llevar a cabo un ataque sin ni siquiera tener que conectarse a la red. Estos ataques pueden incluir:

La denegación de servicio (por ejemplo, conectar un equipo portátil a la red como servidor DHCP o desconectar la alimentación de un servidor), el robo de datos (por ejemplo, robar un equipo portátil o detectar paquetes en la red interna), la ejecución de código malicioso (por ejemplo, activar un gusano desde el interior de la organización). El robo de información de seguridad crítica (por ejemplo, cintas de copia de seguridad, manuales de funcionamiento y diagramas de red), como parte de la estrategia de administración de riesgos, debe determinar el nivel de seguridad física apropiado para su entorno. A continuación se enumeran algunas de las posibles medidas de seguridad física.

- Establecer seguridad física para todas las áreas del edificio (esto puede incluir tarjetas de acceso, dispositivos biométricos y guardias de seguridad)
- Requerir a los visitantes que vayan acompañados en todo momento
- Requerir a los visitantes que firmen un registro de entrada de todos los dispositivos informáticos
- Requerir a todos los empleados que registren cualquier dispositivo portátil de su propiedad
- Fijar físicamente todos los equipos de escritorio y portátiles a las mesas
- Requerir que se registren todos los dispositivos de almacenamiento de datos antes de sacarlos del edificio
- Ubicar los servidores en salas separadas a las que sólo tengan acceso los administradores
- Conexiones a Internet, alimentación, sistemas antiincendios, etc. redundantes
- Protección contra desastres naturales y ataques terroristas
- Establecer seguridad para las áreas en las que se puede dar un ataque por denegación de servicio.

2.1.1.7 Directivas y procedimientos.

Una directiva es un conjunto de reglas, políticas o normas que rigen la seguridad de la entidad indicando de manera clara lo que esta permitido hacer y lo que no.

- **directiva de seguridad [security policy]**

Directiva activa establecida por el administrador que genera mediante programación los permisos concedidos para todo el código administrado basándose en los permisos solicitados por el código. No se permite la ejecución del código que requiere más permisos de los que concederá la directiva.

- **permisos concedidos [granted permissions]**

Permisos que se concederá al código, determinados por la directiva de seguridad, y que le permiten el acceso a recursos y le brindan su identidad. Los permisos concedidos están determinados tanto por los permisos solicitados como por lo que permite la configuración de la directiva de seguridad.

- **permisos solicitados [requested permissions]**

Permisos especificados opcionalmente en un ensamblado que representan los permisos mínimos necesarios, deseados opcionalmente, y que siempre se rechazan para todo el código del ensamblado. Si no hay ninguna solicitud, se concede al código el máximo que permita la directiva de seguridad.

- **directiva de versiones [version policy]**

Reglas que especifican a qué versión de los ensamblados dependientes hay que realizar el enlace. Las directivas de versiones se expresan mediante archivos de configuración.

Un procedimiento es el proceso que debe seguir para cumplir cada una de las directivas

2.2 Linux.

El sistema operativo Linux no es **Unix**. Es más bien un clon de Unix, o por lo menos así lo han expresado siempre sus defensores. Tanto los sistemas Windows NT como Linux son sistemas operativos con micronúcleo (*microkernel*). Los entusiastas de Linux argumentan que este núcleo, en el S. O. (sistema operativo) Linux, está escrito desde cero, sin haberlo copiado de ninguna parte. No obstante, en el proceso de reescritura mucha de la forma y método de Unix se ha transferido, como se transmiten los genes a parientes no tan cercanos. Los comandos de Unix se han transferido sin mayores cambios, la esencia misma de Unix está presente en Linux. Sin embargo, esta pequeña diferencia se torna importante. Actualmente este sistema operativo evoluciona en una forma descentralizada, donde no existe una organización que pueda determinar monolíticamente el camino o sentido hacia donde avanzar tecnológicamente con el producto.

La gratuidad de Linux se basa en modos de licenciamiento que no involucran transferencia monetaria alguna. Sin embargo existen restricciones, ya que también existe un contrato que limita y especifica las obligaciones de las partes, al igual que una EULA (*End User License Agreement*) de Microsoft. La restricción más importante es la distribución del código fuente en algunos casos, o bien la prohibición de guardarse las modificaciones para sí, sin darlas a conocer a la comunidad. Microsoft, en cierto modo y en ciertos casos, también tiene *software* de uso libre, tal como la Embedded Visual Tools, que se compone de IDE más compiladores, herramientas de depuración, emuladores binarios para los dispositivos móviles y documentación completa, la cual no tiene absolutamente ningún costo monetario. Microsoft, sin

embargo, no distribuye el código fuente todavía, aunque Redmond ha comenzado tibiamente a liberar código en ciertos campos que considera viables para el aporte directo de la comunidad.

Las características más importantes de Linux son once y se presentan a continuación:

- **Multitarea.**

  Linux desde su concepción fue diseñado como un sistema operativo multitarea, lo que le permite ejecutar varios programas a la vez, de forma que no tiene que esperar a que termine uno para empezar otro. La multitarea está controlada por el Sistema Operativo (S.O.) y no por las aplicaciones, por lo que es muy difícil que el fallo de un programa "cuelgue" el sistema por una mala utilización de los recursos del equipo.

- **32 bits reales.**

  Linux permite aprovechar toda la potencia del procesador, corre a 32 bits reales en un procesador intel o amd, y a 64 bits en los nuevos procesadores que están llegando al mercado. Esto le confiere al sistema rapidez, eficacia, seguridad y fiabilidad.

- **Multiusuario.**

  Linux es un sistema operativo capaz de responder, simultáneamente, a las solicitudes de varios usuarios que empleen el mismo servidor, incluso con necesidades distintas. Además proporciona los elementos necesarios para garantizar la seguridad y privacidad de los datos entre los diferentes usuarios.

- **POSIX.**

  POSIX es un estándar de la industria que asegura una calidad mínima en ciertas partes del S.O. y asegura la compatibilidad a nivel de código. De esta forma los programas POSIX que funcionan en un Unix no tienen ningún problema para compilarse y ejecutarse en Linux.

- **Estabilidad.**

  Linux es robusto, por lo que si un programa falla no interrumpirá el trabajo de los demás. Entraremos al sistema, desbloquearemos el programa y podremos seguir utilizando el sistema sin ningún problema. Esta característica permite que el sistema funcione durante periodos muy largos de tiempo sin necesidad de parar y volver a arrancar.

- **Libre.**

  Como disponemos del código fuente, podemos hacer cualquier modificación sin tener que esperar a que alguien nos envíe un "Service Pack" para solucionarlo. En el caso de que no sepamos arreglar el fallo podremos contratar a cualquier empresa para que lo arregle, aún cuando la empresa que nos vendió el programa haya cerrado o no le interese resolver nuestro problema, ya que se conoce el código fuente.

- **Adaptable o flexible.**

Linux es un S.O. que evoluciona rápidamente adaptándose a las novedades del mercado y solucionando rápidamente los problemas que puedan surgir, además se puede personalizar tanto como uno quiera ya que ahora mismo hay comunidades autónomas que han hecho su propia distribución.

- **Sistema de archivos.**

  .
  Linux puede operar con una gran variedad de sistemas de archivos, pudiéndolos leer y operar con ellos. Por ejemplo: FAT, VFAT, OS2/FS, ISO9660, ReiserFS, etc.

- **Multiplataforma.**

  Linux es soportado por los sistemas computacionales independientemente del microprocesador que lleven instalado (386, 486, Pentium, Pentium Pro, Pentium II, Pentium III, Pentium 4, AMD 64, Amiga, Atari, Alpha, PowerPC, SPARC, RISC, etc.).

- **Red.**

  Linux fue desarrollado desde sus comienzos para trabajar en red. Su protocolo principal es TCP/IP, aunque soporta una gran variedad de protocolos como SLIP/PPP, PLIP, NFS, Telnet, TNP, SMTP, IPX, AppleTalk, etc. Además es capaz de mediar entre todo tipo de redes, permitiendo trabajar en red con equipos que utilicen sistemas operativos como Windows 98 o XP sin ningún problema.

- **Entorno Gráfico.**

  Linux puede trabajar con o sin entorno gráfico. Por ejemplo para funcionar de manera óptima en equipos con poca memoria o en servidores donde el entorno gráfico consume recursos innecesariamente. Si por el contrario queremos usar un entorno de ventanas, existen un sinfín de gestores (ICEwin y otros) y de entornos de escritorio (KDE y GNOME son los más populares) que permiten al usuario doméstico trabajar de una forma intuitiva.

2.3 El disco duro.

El disco duro es el principal almacenamiento secundario no volátil que dispone un sistema computacional para guardar información, como pueden ser programas, archivos del usuario, etc; con el concepto de no volátil nos referimos a que no se perderán los datos al apagar el sistema como es el caso de la memoria principal o RAM.

2.3.1 Geometría.

En términos generales, un disco duro consiste en uno o más platos o superficies magnéticas en donde regularmente emplean las dos caras de cada plato, con excepción del primero y el último, montados junto con otros sobre un eje común. Para cada superficie existe una cabeza lectora/grabadora montada en un brazo que puede desplazarse en sentido radial, en otras palabras, acercándose o alejándose del centro del disco, que gira constantemente a gran velocidad. En cada superficie, los

datos se almacenan en pistas, organizadas como círculos concéntricos. Cada pista, a su vez, está dividida en porciones llamadas sectores. Visto por sectores, el funcionamiento de los discos es similar al de las cintas magnéticas, ya que en cada uno la información se almacena de manera secuencial. La diferencia consiste en que en el disco la cabeza sí puede ir directamente de una pista a otra y, una vez en una pista, puede dejar pasar sectores hasta que llegue al sector deseado.

Como se muestra en la figura 2.2 el disco visto desde arriba, todas las pistas de los diferentes platos que lo componen están alineadas, y se conocen como cilindros. Un cilindro, entonces, es a un disco completo lo que una pista es a una de sus caras. Así, un disco que tenga 8 superficies con 1024 pistas, de esta manera se tiene 1024 cilindros aunque el número total de pistas será 8 x 1024

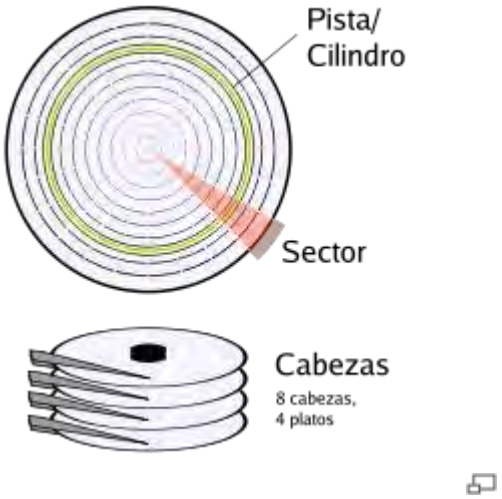

Figura 2.2: Disco duro.

Dando un pequeño Resumen de este tema tenemos lo siguiente:

- **Plato**
  Cada uno de los discos que hay dentro del *disco duro*.

- **Cara**
  Cada uno de los dos lados de un *plato.*

- **Cabeza**
  Número de cabezal; equivale a dar el número de *cara*, ya que hay un cabezal por *cara*.

- **Pista**
  Una circunferencia dentro de una *cara*; la *pista* 0 está en el borde exterior.

- **Cilindro.**
  Conjunto de varias *pistas*; son todas las circunferencias que están alineadas verticalmente (una de cada *cara*).

- **Sector**
  Cada una de las divisiones de una pista. El tamaño del sector es fijo, siendo el estándar actual 512 bytes. Antiguamente el número de sectores por pista era fijo,

lo cual desaprovechaba el espacio significativamente, ya que en las pistas exteriores pueden almacenarse más sectores que en las interiores.

2.4 Análisis de herramientas de respaldo.

2.4.1 Comando dd.

El comando dd crea una copia de imagen de bits (o *descarga del disco*) de los archivos y particiones. En algunas ocasiones dd es utilizado par realizar una copia de seguridad a bajo nivel, para llevar a cabo una auditoria, Siempre que sea posible es conveniente realizar una copia de los datos del equipo a bajo nivel, de forma que se tenga la información completa del estado del sistema cuando se detecto el ataque. Si es posible el análisis posterior de los datos se debería realizar sobre la copia (con el equipo apagado/desconectado). En este proyecto será utilizado para restaurar el MBR.

Cuando se utiliza el comando dd se deben tener algunas consideraciones importantes:

- Tener mucho cuidado al especificar el dispositivo en el que se desea escribir porque los datos de la partición o disco destino serán destruidos por completo, reemplazados por datos que no pueden ser convenientes.

- Debemos tener en cuenta también los tamaños de la partición o disco destino debe ser igual en tamaño (o en todo caso mayor) que la partición o disco origen.

- Es conveniente realizar una copia de seguridad de los datos importantes y tener a mano un disco de arranque de Linux por si se tiene un problema al cambiar el MBR de la computadora.

Este comando copia un archivo, de entrada estándar a salida estándar por defecto, opcionalmente cambiando tamaño de bloques de entrada salida y realizando diversas conversiones, con las siguientes opciones.

- if =ARCHIVO leer la entrada del archivo indicado.
- of =ARCHIVO dirigir la salida al archivo indicado.
- Ibs =BYTES leer la entrada en bloques de BYTES bytes.
- obs =BYTES grabar la salida en bloques de BYTES bytes.
- bs =BYTES leer y grabar entrada y salida en bloques.
- conv =CONVERSION[,CONVERSION]... convertir según argumentos.
- Las opciones numéricas admiten los multiplicadores b para 512, k para 1024.
- Los argumentos de conversión se separan por comas sin espacios; incluyen:
- ascii convierte EBCDIC a ASCII.
- ebcdic convierte ASCII a EBCDIC.
- ibm convierte ASCII a EBCDIC alternativo.
- block embroca cada línea de entrada en 'cbs' bytes, reemplaza nueva línea por espacio, rellena con espacios.
- unblock reemplaza espacios finales de cada bloque con nueva línea.
- lcase convierte mayúsculas a minúsculas
- ucase convierte minúsculas a mayúsculas.
- notrunc no truncar el archivo de salida.

2.4.2 Ghost.

Una de las funciones más importantes de la herramienta ghost es la de realizar copias de respaldo de todo lo que tenga en su equipo: música digital, fotografías, documentos financieros, aplicaciones, configuraciones, sistema operativo, etc., en un solo paso.

También recupera el sistema y los datos aun cuando no puede reiniciar el sistema operativo, efectúa copias de respaldo incrementales que permiten ahorrar tiempo y espacio, efectúa copias de respaldo sobre la marcha, sin necesidad de volver a reiniciar el sistema, realiza copias de respaldo de casi todos los medios, entre los que se incluyen unidades de CDR/RW y DVD+-R/RW, dispositivos USB y FireWire (IEEE 1394), además de unidades Iomega, Zip y Jaz.

Actualmente otras funciones que tiene ghost son las de crea de forma automática un programa inicial de copia de respaldo basado en la configuración del equipo, detectar automáticamente los dispositivos de almacenamiento, analiza el sistema y durante la instalación, ofrece consejos sobre la mejor manera de realizar las copias de respaldo.

También monitorea y optimiza de manera automática el espacio del disco de la copia de respaldo, desencadena la realización de copias de respaldo de los sucesos clave, como por ejemplo, las instalaciones de un nuevo programa o los inicios de sesión de usuarios. Crea nuevas copias de respaldo con el botón “Backup Up Now” siempre que lo desee, cifra las copias de respaldo para que resulten más seguras, tiene una interfase basada en tareas que simplifica la administración y el monitoreo, permite visualizar de forma eficaz todas las copias de respaldo programadas, además del nivel de protección de respaldo asignado a cada una de las unidades del equipo.

2.5 Análisis forense.

"Conjunto de conocimientos médicos relacionados con la administración de la justicia, especialmente identificación personal, diagnóstico de la o las causas de la muerte, etc."[2]

2.5.1 Análisis forense informático.

"El análisis forense es una metodología de estudio ideal para el análisis posterior de incidentes, mediante el cual se trata de reconstruir cómo se ha penetrado en el sistema, a la par que se valoran los daños ocasionados."[3]

"Aplicación de métodos, protocolos y técnicas suficientemente legales para reunir, analizar y preservar la información computacional".[4]

[2] Según Enciclopedia Encarcarta
[3] Según Wikipedia www.wikipedia.com
[4] Según el manual del investigador de crimen de alta tecnología www.lci.ulsa.mx/Material/pdf/seguridad_2002.pdf

Siendo su objetivo reconstruir los hechos pasados con base en la evidencia que es recolectada y preservando la información que pueda ser usada como evidencia en un proceso judicial, es necesario señalar que aun no existen estándares aceptados, aunque algunos proyectos están en desarrollo como los siguientes:

- C4PDF (Código de Prácticas para Digital Forensics), de Roger Carhuatocto.
- Open Source Computer Forensics Manual, de Matías Bevilacqua Trabado.
- Training Standards and Knowledge Skills and Abilities de la International Organization on Computer Evidence.

Además se les reconoce a Dan Farmer y Wietse Venema, los creadores del Forensics Toolkit, y como los pioneros del análisis forense. Actualmente, Brian Carrier es probablemente uno de los mayores expertos mundiales en el tema.

El proceso forense a grandes rasgos se compone de las siguientes etapas:

- Recolección de evidencia.
- Análisis de la evidencia.
- Reportar resultados.

El análisis forense se determina bajo los siguientes puntos:

- **Para determinar cual fue el fallo, un análisis forense de calidad debe de poder responder las siguientes preguntas.**
  - **¿Quién ha realizado el ataque?**
    Dirección IP de los equipos implicados en el ataque.
  - **¿Cómo se realizó el ataque?**
    Vulnerabilidad o falla empleada para acceder al sistema.
  - **¿Qué hizo el atacante?**
    Qué acciones realizó el atacante una vez que accedió al sistema.
  - **¿Por qué medio accedió al sistema el agresor?**
    Una pregunta que suele ser más difícil de responder es la anterior.
- **Mejorar los esquemas de seguridad.**

  En la manera en que los administradores se den cuenta de los elementos utilizados para violar su sistema, estos deberán de responder mejorando la protección, y evitando que al menos pueda ocurrir otro incidente utilizando un patrón de ataque ya conocido, por ellos.

- **Retroalimentación al equipo de respuesta a incidentes.**

  El equipo de respuesta a incidentes debe evaluar su desempeño una vez que haya terminado éste y tratar de mejorar los puntos en los que salió con una calificación deficiente.

- **Deslindar responsabilidades.**

  Al deslindar responsabilidades cada quien tendrá la parte justa de esta, es necesario recordar que en algunas ocasiones la responsabilidad no podrá recaer en nadie de dentro de la organización, ya que si el análisis determina que fue un método novedoso, por ejemplo vulnerabilidades de día cero, es teóricamente imposible poder estar preparado para un evento de esa magnitud.

Las dos situaciones posibles que nos podemos encontrar a la hora de hacer un análisis forense son las siguientes: sistema “muerto” se define de la siguiente manera, es aquel sistema que está apagado y no esta en un estado volátil con procesos en ejecución, desgraciadamente al estar el sistema en esta condición ciertas características muy importantes se pierden como son la información cargada en la memoria RAM, los procesos en ejecución, y variables volátiles, ahora veamos la otra situación que suele también presentarse, sistema “vivo” se define de la siguiente manera, es aquel sistema que esta encendido, con los procesos en ejecución, con variables volátiles cargadas y listo para interpretar las ordenes que el operador le indique, ahora veamos el comparativo de un sistema muerto y un sistema vivo, tabla 2.6.

Estado del Sistema	Características
Sistema Muerto	-Sistema estático. -Fácil duplicación. -No se han perdido datos estáticos como pueden ser los del disco duro. -Menos cosas que recolectar -Se puede iniciar el análisis forense de inmediato. -No se puede recuperar información de los procesos, ni ningún tipo de información volátil
Sistema Vivo	-Sistema dinámico. -Duplicación más difícil. -Variables desconocidas. -Más cosas que recopilar. -Memoria de ejecución -Procesos. -Variables dinámicas -El atacante puede darse cuenta de la actividad del administrador, y entorpecer la labor de este

Tabla 2.6: división de sistemas para análisis forense.

El análisis en un sistema muerto suele emplearse en mayor medida en sistemas del tipo Unix, debido a que resulta más fácil hacer la copia de un sistema y luego realizar el análisis en otro sistema “sano”, esto es principalmente debido a que en la mayoría de los Unix son compatibles entre si, específicamente en su sistema de archivos lo cual permite que el montaje de una partición se ha una tarea sencilla de realizar, además de las múltiples posibilidades que este tipo de sistema operativo nos brinda a la hora de montar una partición, entre las que destacan que no se pueda ejecutar los binarios de esa partición, que no se puedan modificar los llamados tiempos MAC, otro punto que destaca a la hora del análisis es el poderoso shell que tienen los sistemas Unix, el cual resulta bastante útil a la hora de andar buscando archivo ocultos, directorios ocultos y demás patrones que puedan ayudarnos a la hora de hacer un análisis forense, esto solo es una parte además hay que agregar que hay múltiples herramientas que corren en sistemas Unix como es el caso de TCT, o Autopsy, por todo lo antes expuesto el análisis de un sistema Unix muerto se realiza comúnmente en un Unix en cambio el sistema de archivos de un Windows en especial cuando utiliza el sistema de archivos NTFS el cual resulta bastante engorroso y poco fiable el uso, al menos al tratar de montar una partición de este tipo en un sistema Unix, por eso se prefiere hacer el análisis de un sistema Windows en vivo.

A continuación veremos como hacer el análisis forense a un sistema Windows, debido a que no existe una metodología estandarizada claramente definida, el análisis pueden ser en algunos puntos diferente, debido principalmente al enfoque que el especialista le de a este.

Los pasos a seguir para realizar un forense en un sistema Windows en estado vivo son regularmente los que a continuación se mencionan y que se muestran en la figura 2.3 y se describen a detalle en la tabla 2.7

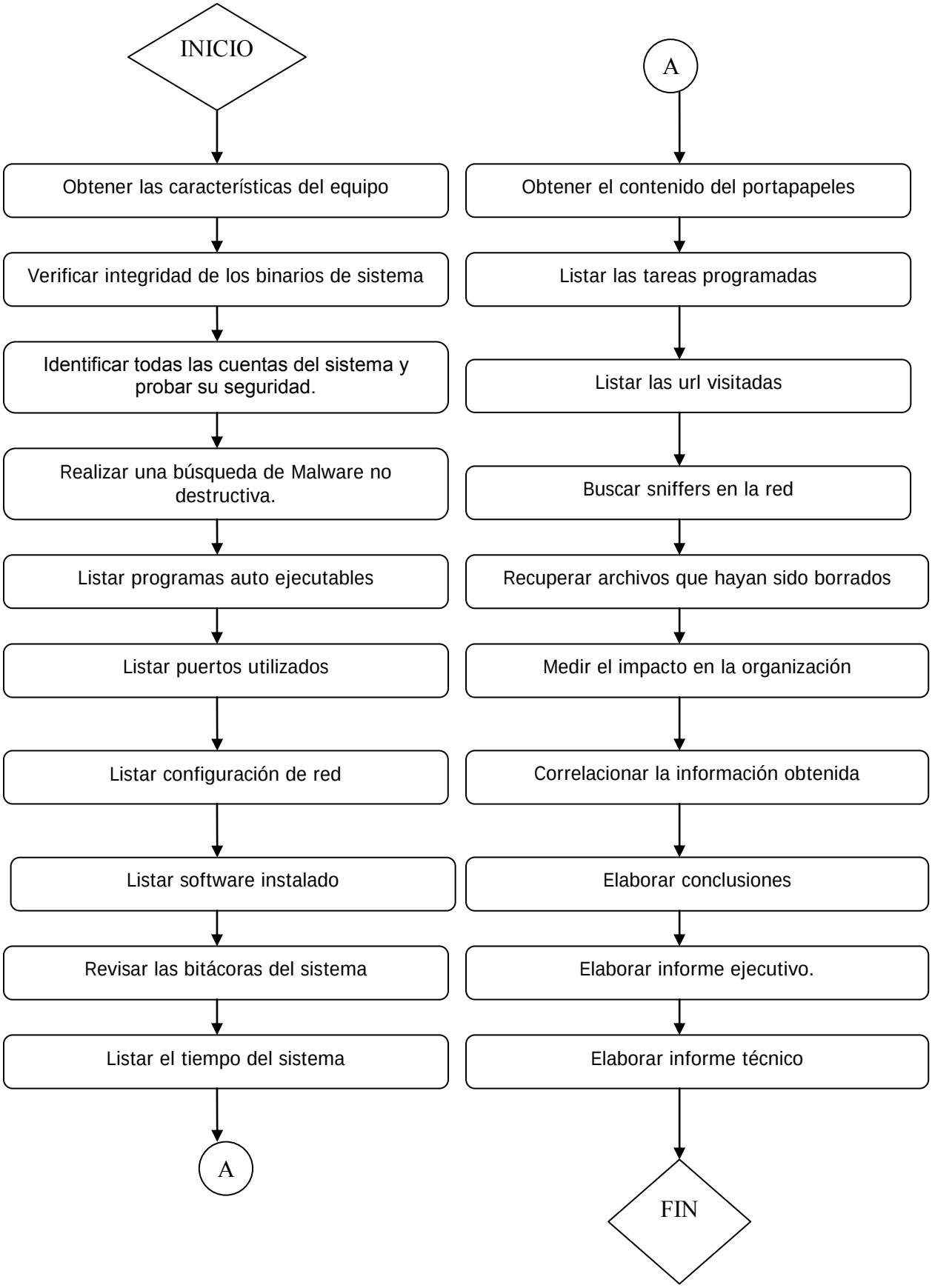

Figura 2.3: Pasos a seguir en un análisis forense de un sistema vivo Windows.

Tabla 2.7: Pasos a seguir en un análisis forense de un sistema vivo Windows.

Metodología a seguir en un análisis forense en sistemas Windows.
1. Se deben describir a fondo las principales características del sistema operativo en cuestión, cuál es la versión, el huso horario, enlistar el software instalado en el sistema que se está analizando, además de las actualizaciones que tiene instaladas el sistema
2. Se debe realizar una verificación de la integridad de los archivos de sistema, así como la fecha de creación, modificación y último acceso, ya que esto nos permitirá posteriormente realizar un diagrama de tiempos de las actividades que realizo el intruso mientras tenia el control del sistema.
3. Debemos de identificar y listar todas las cuentas que tenga el sistema, así como las características que éstas presentan, además de probar la seguridad de las contraseñas de las cuentas; esto se puede realizar con una herramienta como John The Ripper
4. El siguiente paso es realizar una búsqueda de Malware, esto se puede realizar utilizando cualquier antivirus, pero cabe aclarar que solo lo vamos a buscar, y no a borrar del sistema
5. El siguiente punto es listar los programas auto arrancables y la lista de servicios que presenta el equipo
6. Listar los puertos abiertos TCP y UDP.
7. Listar las características que presenta la configuración de red, y al igual que la de los documentos compartidos del sistema.
8. Listar software que no este firmado digitalmente, instalado en el sistema.
9. Debemos revisar las bitácoras de sistema (EventView) y además las bitácoras que haya de otros servicios que estén habilitados en el sistema, como ejemplo de una aplicación que tiene sus propias bitácoras tenemos a apache; regularmente las bitácoras es uno de los primeros objetivos del intruso.
10. Listar el tiempo del sistema.
11. Listar el contenido del portapapeles
12. Listar las tareas programadas
13. Listar las url visitadas por los usuario del sistema; lo anterior se logra abriendo los archivos index.dat, y volcando el historial de internet.
14. Recolectar evidencia de la red, esto se puede hacerse mediante el uso de un sniffer.
15. Utilizar alguna herramienta de recuperación de archivos, esto con el fin de obtener posibles herramientas que hallan sido ejecutadas y borradas del sistema, regularmente estas herramientas tienen el propósito de esconder la intrusión y/o atacar a otros

sistemas desde el sistema comprometido (rootkits y exploits)
16. A partir de las evidencias encontradas, debemos de tratar de medir el impacto que pudo tener en la organización.
17. Lo siguiente que debemos de hacer al realizar un análisis forense, es tratar de correlacionar los datos obtenidos, y tratar de armar una línea de tiempo de lo que pudo haber ocurrido. ¿Qué es una línea de tiempo?
18. Elaborar nuestras conclusiones.
19. Elaborar y presentar un informe ejecutivo
20. Elaborar y presentar un informe técnico

Como se puede apreciar son muchos los puntos que hay que considerar antes de poder emitir un juicio que plasme de manera fiel lo que ocurrió dentro del sistema.

Entre las principales modificaciones que realiza un malware en un sistema Windows, son las que se mencionan dentro de la tabla 2.8.

Tabla 2.8: Modificaciones que realiza un malware a un sistema Windows.

Posibles Modificaciones que realiza un malware a un sistema Windows.
Hacer modificaciones al registro de Windows.
Hacer copias de si mismo, en el sistema de archivos, dicho de otra manera altera el sistema de archivos
Cambiar el tiempo del sistema
Crear usuarios
Modificar los servicios que tiene el equipo.
Cambiar la configuración de la tarjeta de red.
Almacenar variables en el portapapeles.
Crear tareas programadas, que se ejecuten cada X tiempo.
Cambiar la configuración de los recursos compartidos.

Abrir puertos.
Crear procesos
Levantar o encender servicios.
Consumir el ancho de banda que tengamos destinado para ese equipo.

Cabe aclarar que estas no son todas las modificaciones que pueden realizarse al ejecutar un malware en un sistema Windows, pero resultan ser las de mayor frecuencia, por esta razón es importante considerarlas dentro del objeto de estudio de este proyecto.

2.5.2 Tiempos MAC.

Los tiempos MAC son potencialmente la información más valiosa para un análisis forense, los MACtimes se refieren a tres atributos de tiempo de los archivos (mtime, atime y ctime) que podemos encontrar en los sistemas de archivos usados por sistemas operativos como Unix, NT y otros, en los sistemas Microsoft, estos son descritos como LastWriteTime, LastAccessTime y CreationTime.

Muchas veces, estas marcas de tiempo son subutilizadas, a pesar de que pueden ser fundamentales en la investigación de intrusiones, esto es debido a la gran complejidad del análisis y que solo existe rudimentarios métodos para realizar el análisis de estas marcas de tiempo.

Los tiempos MAC se modifican de la siguiente manera:

- Atime.- El hecho de leer un archivo, generalmente cambia este tiempo.
- Mtime.- Este tiempo cambia cuando se modifica el contenido de un archivo.
- Ctime.- Mantiene un registro de cuándo se cambió la meta información de un archivo, la meta información son los datos del archivo como propietario, grupo, permisos, etc.
- Dtime.- En algunos sistemas de archivos, existe este tiempo, este tiempo indica la fecha en la cual fue borrado.

Un aspecto particular que cabe destacar de los tiempos MAC es el siguiente:

Cuando un archivo es copiado, el mtime de un archivo copiado es el mismo que el del original, mientras que el atime y el ctime pueden cambiar, esta situación puede crear la ilusión de que se creó después de que fue modificado.

La importancia de estas marcas de tiempo reside en que no solamente se pueden obtener estos tiempos de un sistema vivo, también se puede montar el disco duro de una máquina y para poder obtener estas marcas.

Al utilizar estas marcas de tiempo (MACtime) se debe ser extremadamente cuidadoso al recolectar esta información, ya que su existencia suele ser muy efímera,

por lo cual es preferible utilizar un duplicado de los datos y no realizar este análisis sobre el original, si la alternativa anterior no fuera posible, otra forma es montar el sistema de archivos a analizar como sólo lectura.

Una línea de tiempo es la explicación en orden cronológico de los hechos tal y como fueron ocurriendo dentro del sistema.

2.5.3 Herramientas utilizadas en un análisis forense para un sistema Windows.

En los puntos anteriores se hizo hincapié en la recolección de datos para poder llevar a cabo un análisis forense exitoso, así, algunas de las herramientas más utilizadas para llevar a cabo esta tarea se pueden apreciar en la tabla 2.9

Tabla 2.4.3: Listado de herramientas.

Nombre	Descripción
Las siguientes herramientas sirven para observar los puertos que el sistema está usando, además, pueden ser usadas para encontrar puertas ocultas en el sistema.	
Nport	Herramienta que se ejecuta desde línea de comandos, la cual permite identificar los puertos abiertos desconocidos y sus correspondientes aplicaciones asociadas, funciona en sistemas Windows 2000 y XP, se puede descargar desde www.foundstone.com, es un freeware.
Open Ports	Es una herramienta usada para verificar el estado de todos los puertos ya sean de protocolo TCP o UDP de un sistema informático, es una herramienta pequeña de solo 24 kb y opera desde línea de comandos, se puede descargar de la siguiente dirección http://www.diamondcs.com.au/openports/.
Currports	Despliega la lista de todos los puertos TCP/IP y UDP abiertos en el sistema local, y asocia al proceso que abre el puerto, se puede descargar desde www.nirsoft.net
Las siguientes herramientas permiten verificar la configuración de los dispositivos de red que tienen nuestros equipos, y algunas de ellas permiten observar dispositivos en estado promiscuo, importante propiedad en la detección de sniffers.	
PromiscDetect	Es una herramienta que revisa todas las interfaces de red verificando si están en modo promiscuo o no, en caso de que estuviera en modo promiscuo algún dispositivo lanzará una advertencia debido a que posiblemente se halla un sniffer instalado en el equipo analizado, esta herramienta se ejecuta desde línea de comandos, y se puede descargar desde www.ntsecurity.com
IP List	Es una herramienta que muestra todas las interfaces de red, así como sus características (broadcast, máscara de red, ip).
Las siguientes herramientas permiten observar los procesos de sistema, y sus archivos asociados.	
Process Explorer	Es una herramienta de interfaz gráfica la cual permite verificar qué programa abre un archivo o directorio, y además muestra información sobre procesos y subprocesos, la podemos descargar sin costo desde www.sysinternals.com, es un freeware.
Pslist	Herramienta que se ejecuta desde línea de comandos, y es equivalente al comando ps de los sistemas Unix, permite visualizar información detallada sobre procesos y memoria utilizada, se puede descargar desde www.sysinternals.com

A continuación se enumeran algunos comandos del propio sistema operativo que suelen ser muy útiles al realizar un análisis forense.	
Net	Es parte del sistema operativo, y se ejecuta desde línea de comandos, al ejecutarlo podemos obtener los usuarios del sistema, las carpetas compartidas, y además de las tareas programadas del sistema.
Reg	Es un comando propio del sistema operativo, el cual al ejecutarlo se puede consultar, añadir, y cambiar llaves del registro de Windows, es un comando del sistema operativo y funciona desde línea de comandos.
Schtask	Utilizando este comando se pueden saber las tareas programadas que tiene el sistema que se está analizando, es un comando del sistema operativo y se ejecuta desde la línea de comandos o el prompt del sistema operativo. Además se pueden programar archivos binarios para que se ejecuten periódicamente en un horario específico comúnmente llamadas tareas, desde aquí podemos observar, agregar y quitar tareas programadas del sistema.
Con las siguientes herramientas se puede obtener información de sistema, como pueden ser los ejecutables que se inician de forma automática al arrancar el sistema, información de rootkits, el tiempo que tiene la maquina sin ser apagada..	
Autoruns	Es una herramienta que puede operar desde línea de comandos o desde interfaz grafica, la cual provee información de las llaves del registro en especial de los elementos que se activan al iniciar el sistema, se puede descargar libremente de www.sysinternals.com
Psinfo	Obtiene la descripción completa del sistema al cual se le esta practicando el análisis forense. http://www.microsoft.com/technet/sysinternals/utilities/psinfo.mspx
Uptime	Esta herramienta operada desde línea de comandos nos arroga el número de minutos que el sistema ha permanecido en activo, desde la última vez que fue encendido.
RootkitRevealer	Es una herramienta que funciona por medio de la interfaz grafica o en otras palabra desde ventanas y su función es tratar de establecer si la maquina analizada tiene instalado un RootKit., es un freeware el cual se puede descargar desde www.sysinternals.com

2.5.4 Herramientas de auditoría.

2.5.4.1 Encase (Edición Forense).

La policía, el gobierno, los militares y los investigadores corporativos confían en EnCase Edición Forense para ejecutar exámenes informáticos delicados y conclusivos. Guiados por nuestras relaciones con investigadores en el mundo entero, El programa EnCase ha sido optimizado para manejar la complejidad cada vez mayor de las configuraciones y capacidades informáticas. El programa EnCase soporta un amplio rango de sistemas operativos, archivos y periféricos que son el desafío de los investigadores forenses diariamente. Como una herramienta seleccionada por la policía, el programa EnCase ha soportado numerosos desafíos en las cortes de

justicia, demostrando su confiabilidad y exactitud. Recientemente, el Instituto Nacional para Estándares y Tecnología (NIST) concluyó que EnCase Imaging Engine (motor de creación de imágenes de discos) opera con mínimos defectos.

La clave para computación forense es la capacidad de adquirir y analizar datos rápidamente. EnCase le permite a los investigadores manejar fácilmente largos volúmenes de evidencia computacional, la visualización de todos archivos relevantes, incluyendo aquellos eliminados y el espacio no utilizado. La incomparable funcionalidad del programa de EnCase permite a los investigadores llevar satisfactoriamente el proceso completo de investigación computacional, incluyendo reportes personalizados de búsquedas y sus ubicaciones.

El programa EnCase ejecuta adquisiciones de medios produciendo un duplicado binario exacto de los datos del medio original. EnCase verifica este duplicado generando valores de hash MD5 en ambos medios (el original y el archivo imagen o "archivo de evidencia"). Adicionalmente, a cada 64 sectores de la evidencia se le asigna un valor CRC. Estos valores CRC son verificados cada vez que la evidencia es accesada.

EnScript es un macro lenguaje de programación incluido en el programa EnCase. Emulando características de Java y C++, EnScript le permite al investigador construir scripts personalizados para necesidades específicas de la investigación y/o automatización de tareas rutinarias complejas. Mediante la automatización de cualquier tarea investigativa, EnScript no solamente puede salvar días de investigación, si no semanas del tiempo de análisis.

El programa soporta las siguientes configuraciones dinámicas de discos: Spanned, Mirrored, Striped, RAID 5 y básico. Con el ingreso de un mínimo de información, por parte del investigador, el programa EnCase puede detectar automáticamente la configuración de los discos y conectar todas las particiones, mientras que se conservan intactas las áreas libres y de arranque para futuras búsquedas.

EnCase le permite a los investigadores analizar y pre visualizar simultáneamente múltiples bloques de datos adquiridos. Los investigadores pueden utilizar búsquedas globales de palabras claves, análisis de hash, análisis de firmas de archivos y filtros específicos de archivos para analizar rápidamente la evidencia.

Al igual que existen muchas formas de medios digitales, existen muchas otras formas de adquisición de medios. EnCase incluye cables para puertos paralelos y cable de red cruzado para Windows y DOS. Ambos métodos permiten al programa "escribir bloques" para ser colocados en el medio sospechoso, asegurando que el medio original no sea alterado.

Los siguientes sistemas de archivos son actualmente soportados por EnCase: FAT12 (disco flexible), FAT16, FAT32, NTFS, HFS, HFS+, Sun Solaris UFS, EXT2/3, Reiser, BSD FFS, Palm, CDFS, Joliet, UDF e ISO 9660.

2.5.4.2 SleutKit.

Se trata de un conjunto de herramientas de análisis forense que permiten analizar imágenes de sistemas de archivos *ext2,* así como el análisis de otros tipos de sistemas de archivos, como *ntfs, ext3,* FAT, Ext2/3, NTFS, UFS1, y UFS2. Al soportar diversos sistemas de archivos, es posible realizar el análisis forense sobre equipos

que contenía sistemas operativos como Windows. En estos casos, el análisis variará, debido a que los sistemas de archivos como FAT algunas características distintas al sistema de archivos de Linux *ext2*.

Incluye para su implementación algunas herramientas del conjunto *The Coroner's Toolkit (TCT)*.

Algunas características del sleuthkit esta ordenado en capas. Hay una *capa de datos* que se refiere a cómo la información se almacena en un disco y una *capa donde encontramos a los meta datos, los* cuáles son como los inodos y directorios. Los comandos que se ocupan de la capa de datos se prefijan con la letra *d*, y los comandos que se ocupan de la capa de los meta datos se prefijan con la letra *i.*

Algunos de los comandos en sleuthkit son:

- dcat : Ve el contenido de un bloque.
- Dls: Lista los bloques sin referencia. Hace búsquedas de palabra clave más eficientes. Consigue una lista de bloques sin referencia
- Dcalc: Te dice donde están los bloques sin referencia.
- Dstat: Detalles sobre un bloque dado.
- Icat: Ver el contenido de un archivo dado tu valor del inodo .No enumera directorios, enumera el contenido.
- Ils: Enumera los fragmentos del archivo en un disco.
- Istat: Información sobre un número del inodo.

Cabe mencionar que sleuthkit se puede utilizar con el programa Autopsy, el cual brinda una interfaz gráfica utilizando el navegador de internet, volviéndolo un poco más amigable

2.6 Lenguajes de programación.

Lenguaje de programación es el nombre genérico que se aplica a cualquier lenguaje disponible para escribir programas para una computadora. Si no se incluye el ensamblador, este concepto es idéntico al del lenguaje de alto nivel. De forma aparentemente paradójica, escribir un programa en algún lenguaje de programación es la última etapa del proceso conocido comúnmente como "programar", ya que los pasos anteriores en orden son entender el problema, analizarlo y programarlo en pseudo código.

Lenguaje de alto nivel se llama así a los lenguajes de programación que están "por encima" (en poder expresivo) del lenguaje máquina y del lenguaje ensamblador. Existe una gran cantidad y diversidad de estos lenguajes, pero para cada uno de ellos debe existir un compilador que traduzca el programa escrito en él lenguaje de la máquina donde se intente ejecutarlo.

2.6.1 Perl.

Perl (Practical Extraction and Report Languaje) es un lenguaje de programación surgido a inicios de los noventas, que busca antes que nada el facilitar la elaboración de tareas comunes en sistemas tipo Unix, que se aplican sobre grandes cantidades de información (por lo regular texto) por lo que se requiere que sean de alto rendimiento, su autor, Larry Wall.

Perl toma características del C, del lenguaje interpretado shell, AWK, sed, Lisp y, en un grado inferior, muchos otros lenguajes de programación.

Perl surgió como una opción para una gran cantidad de herramientas de Unix en las cuales basa su propia sintaxis, buscando el mínimo sacrificio de su desempeño por una máxima facilidad de programación e integración, sigue la filosofía de mantener un ambiente que sea capaz de detectar y corregir pequeñas omisiones del programador, y de proporcionarle una forma abreviada de realizar múltiples tareas.

Las plataformas donde Perl se ha desarrollado más son los servidores Unix, por sus necesidades de administración y lo robusto de su manejo de memoria y de procesos además de la facilidad de Perl para realizar los así llamados CGIs, interfaces para comunicar recursos del servidor con un servicio de intranet

Perl no establece ninguna filosofía de programación (de hecho, no se puede decir que sea orientado a objetos, modular o estructurado aun cuando soporta directamente todos estos paradigmas), los objetivos que se tuvieron en cuenta al diseñar la sintaxis de Perl fueron la facilidad de aprendizaje y de uso y la claridad de código.

Perl es software libre y está licenciado bajo la Licencia Artística y la GNU General Public License. Existen distribuciones disponibles para la mayoría de sistemas operativos. Está especialmente extendido en Unix y en sistemas similares, pero ha sido portado a las plataformas más modernas.

2.6.2 Programación en shell.

Un sistema Unix esta estructurado por niveles, en el primer nivel se encuentra el hardware, que consiste en discos, terminales, el CPU, memoria y demás dispositivos. Corriendo sobre el hardware esta el sistema operativo; su función es controlar el hardware y proporcionar una interface de llamada al sistema a todos los programas. Estas llamadas permiten a los programas del usuario crear y manear procesos, archivos y otros recursos.

En adición al sistema operativo y al sistema de librerías, Unix tiene una serie de programas estándar; estos incluyen el procesador de comandos (shell), compiladores, editores, procesadores de texto y utilerías de manejo de archivos. Son estos programas los que el usuario invoca desde una terminal.

El shell es un intérprete de comandos y su trabajo es la interacción de un usuario Unix y el sistema en si, por lo tanto un shell es un programa que interpreta y ejecuta los comandos conforme se proporcionan desde una terminal. No se requiere ningún privilegio especial para ejecutar un shell; para el kernel es un programa más.

La programación en shell se basa en el uso de las herramientas del sistema, por otro lado el Unix y sus clones se considera un sistema operativo que cuenta con bastantes herramientas de proceso y filtrado de textos, también de control de procesos, entre otras. Por ello, permite automatizar procesos repetitivos, que hechos a mano serían engorrosos y lentos.

Para poder programar en shell, hay que conocer el mayor número posible de herramientas del sistema, como pueden ser el grep, wc, sort, tr, sed, cut y awk (el awk es casi un lenguaje de programación aparte), aunque puede servir perfectamente como complemento para la programación en shell.

Capítulo 3

Diseño e implementación del laboratorio de malware.

En este capítulo se explica cómo se realiza el diseño, así como la implementación del laboratorio de malware, aplicando todos los conocimientos de ingeniería de programación, seguridad informática, redes, sistemas operativos y estructuras de datos

El diseño del laboratorio se realiza a partir de la documentación y/o boletines de seguridad, que mediante la elaboración de algunos scripts de auditoría se determinan los cambios realizados en el sistema operativo, por parte del malware. Los scripts de auditoría se crean utilizando herramientas propias o de terceros para obtener información que revele el comportamiento de un malware después de su ejecución en el laboratorio.

3 Diseño e implementación del laboratorio de malware.

3.1 Ingeniería de software.

> *"Disciplina tecnológica y administrativa dedicada a la producción sistemática de productos de programación, que son desarrollados y modificados a tiempo y dentro de un presupuesto definido".* [1]

> *"El establecimiento y uso de sólidos principios de ingeniería, orientados a obtener, económicamente, software que sea fiable y funcione eficientemente sobre máquinas reales".*[2]

Para el desarrollo de este proyecto, se ha determinado que el paradigma que de manera más satisfactoria cumple con las necesidades del proyecto es el llamado en **espiral**, ya que al ser este el mismo equipo de desarrollo que el del cliente, es posible la creación de prototipos, para después ser probados y finalmente generar las correcciones necesarias que permitan obtener un producto terminado. Para llegar a este resultado es necesario dar cuatro vueltas del espiral. A pesar de que este paradigma es algo lento, es bastante funcional para proyectos de investigación ya que ofrece una constante retroalimentación.

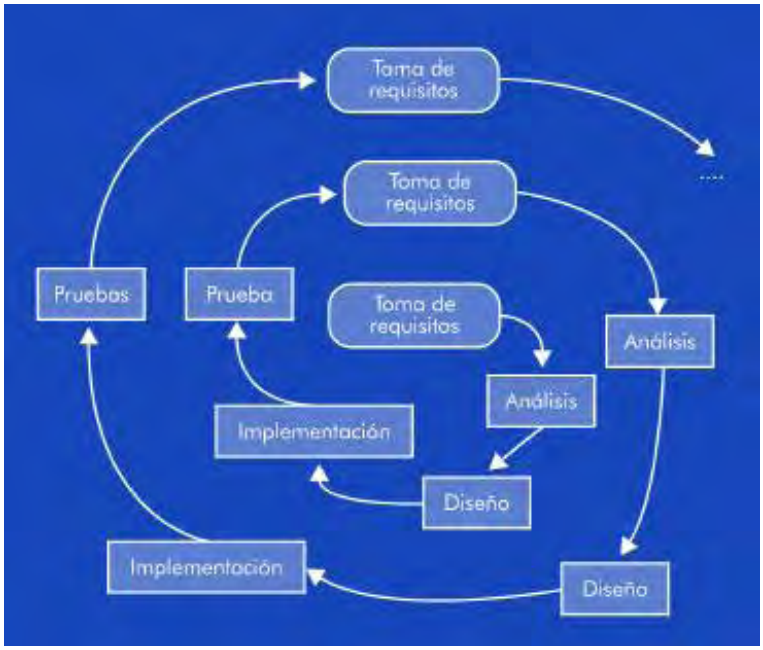

Figura 3.1: Paradigma en espiral.

3.2 Descripción del laboratorio de malware.

El análisis de requisitos es la primera etapa en el desarrollo de cualquier proyecto de software, aquí se tiene que listar las tareas necesarias para cumplir el objetivo de analizar los cambios realizados en un sistema operativo Windows al ejecutar un binario desconocido. La mayor parte de los requerimientos los obtendremos de las acciones habituales de un malware y otros serán requeridos al ir implementando este laboratorio.

[1] Richard Fairley
[2] Fritz Bauer

3.2.1 Requerimientos.

Figura 3.2: Requerimientos del laboratorio de malware.

3.3 Análisis del problema.

Se pretende en esta etapa del proyecto plasmar por medio de diagramas de flujo, de mapas metales y de diagramas físicos, un sistema que cumpla con los objetivos descritos en el apartado de requerimientos. Siempre tomando en cuenta que muchos de estos diagramas serán utilizados en la codificación de nuestros scripts y en la implementación del laboratorio, por lo tanto es necesario realizarlo detallada y cuidadosamente.

3.3.1 Diagrama lógico.

El diagrama lógico es la representación de tareas mediante la utilización de símbolos. La programación por diagramas lógicos, sustituye con bloques normalizados algunas funciones secuenciales típicas, este sistema queda reservado a aplicaciones en las que solo intervengan variables booleanas y algunos bloques secuenciales elementales.

Para explicar el funcionamiento del laboratorio se utiliza un diagrama lógico donde se explican los procedimientos que se siguen en estricto orden del proyecto (véase la figura 3.3)

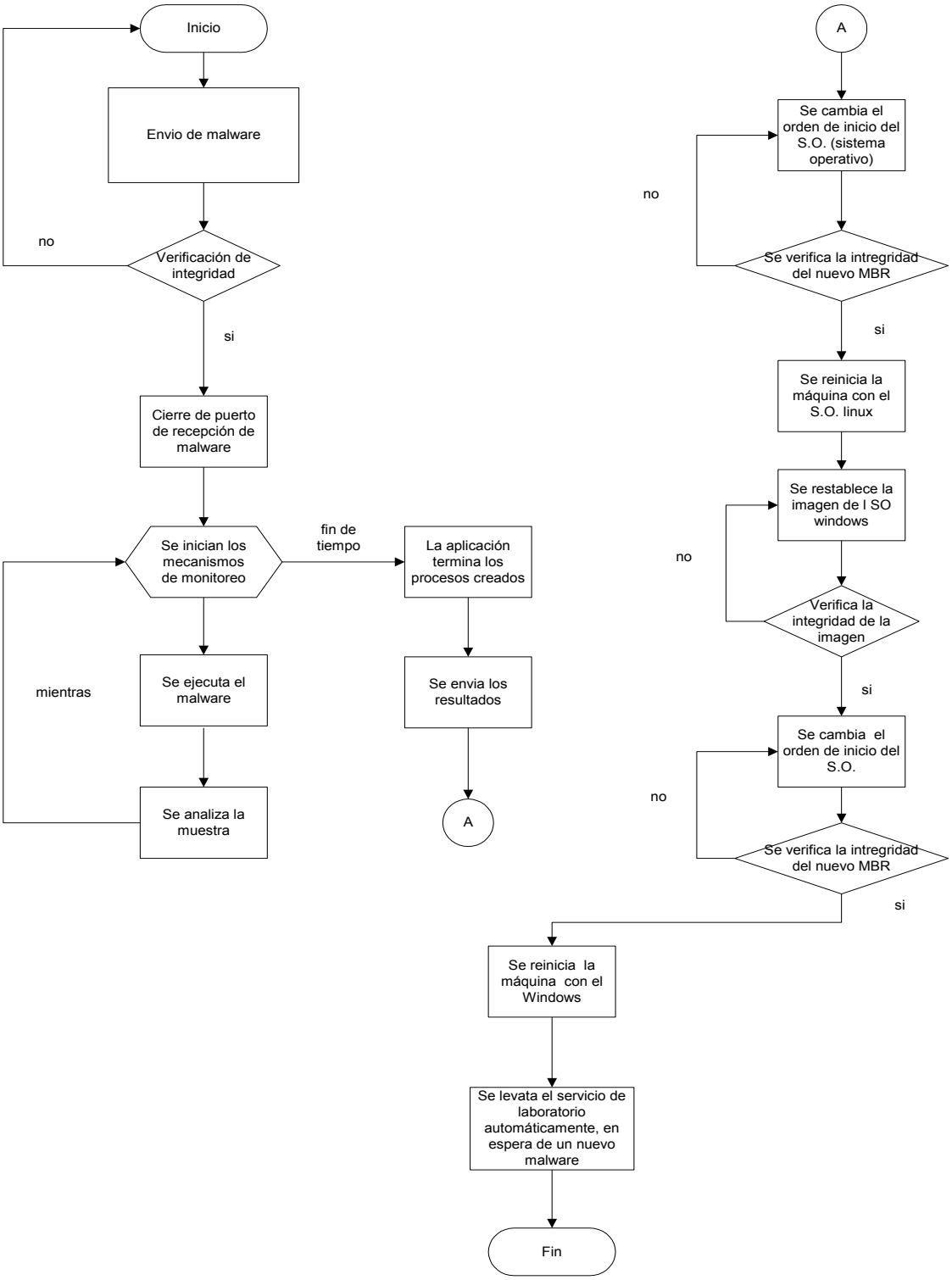

Figura 3.3: Diagrama lógico del laboratorio de malware.

3.3.2 Mapa mental.

Es un diagrama lógico usado para representar las palabras, ideas, las tareas u otros artículos ligados y dispuestos alrededor de una palabra clave o de una idea central. Los elementos se arreglan intuitivamente según la importancia de los conceptos y se organizan en las agrupaciones, las ramas, o las áreas. Para el proyecto, se utiliza un mapa mental para permitir una mejor organización y fácil entendimiento de todos los elementos empleados en el laboratorio, como se muestra en la figura 3.4.

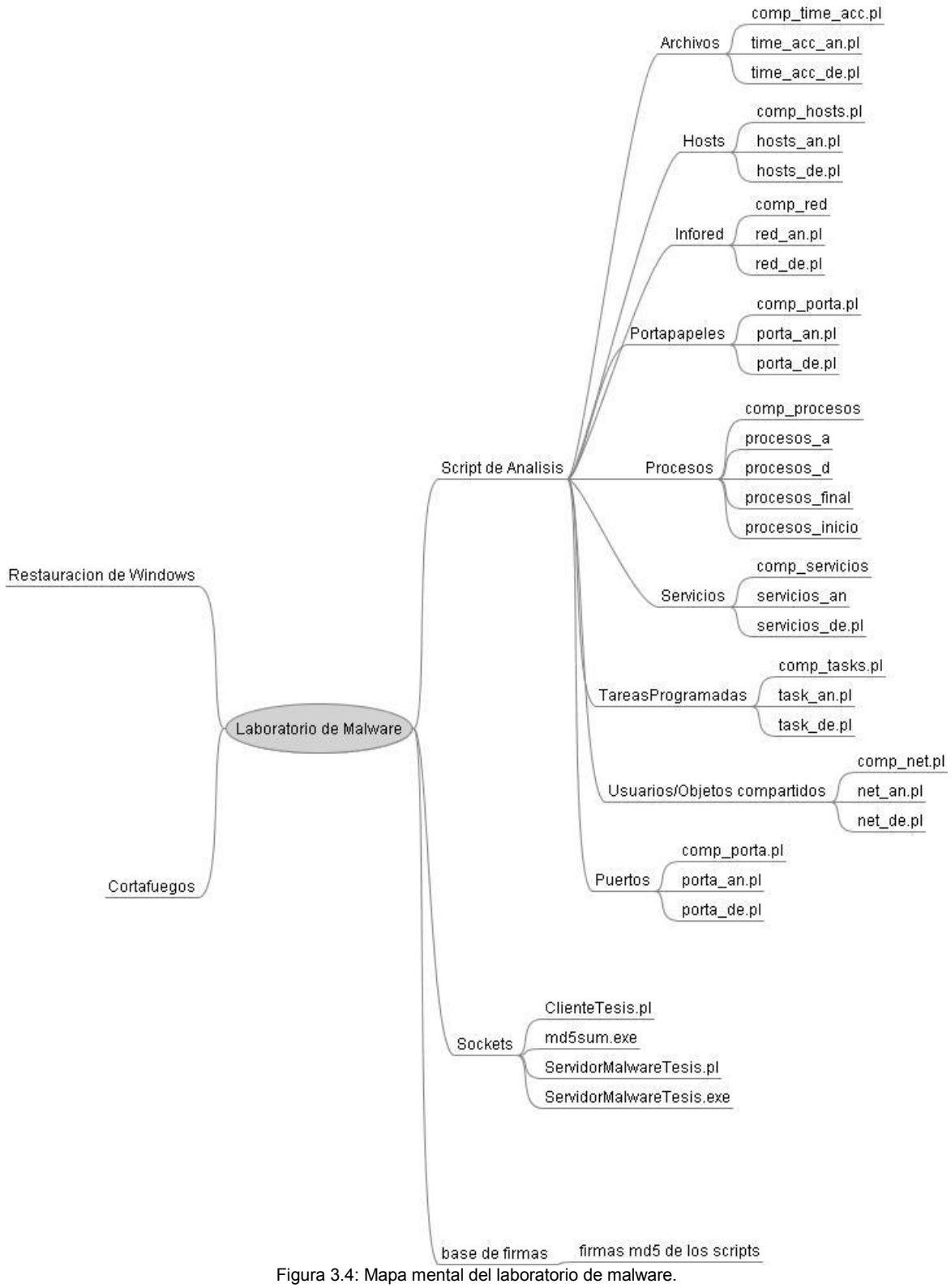

Figura 3.4: Mapa mental del laboratorio de malware.

3.3.3 Diagrama físico.

Un diagrama físico es la descripción del proyecto, la figura 3.5 muestra de manera general como se encuentra instalado el laboratorio, donde la computadora de la izquierda representa el laboratorio, la figura del centro es un servidor web integrado con un firewall siendo éste el único punto de acceso al servidor malware y la máquina

de los usuarios que está representada por la figura de la derecha la cual se comunica vía SSH al servidor web.

Figura 3.5: Diagrama físico del Laboratorio de malware.

3.4 Implementación de la solución.

En está etapa del proyecto se presenta la codificación y documentación de los scripts, así como la configuración de los servicios utilizados, todos estos creados para cumplir con los objetivos de este trabajo. Esta documentación además tiene el fin de servir como un antecedente en el caso de futuras actualizaciones.

3.4.1 Creación de scripts de auditoría.

Un script es conjunto de instrucciones que permiten la automatización de tareas creando pequeñas utilidades, para el desarrollo del laboratorio de malware es necesario crear scripts de auditoría, ya que se tiene que hacer una comparación del sistema antes de ser infectado por el malware y después de su infección, así de este modo determinar los cambios que se realizaron en el sistema, que bien pueden ser en usuarios, llaves de registro, procesos, cambio de tiempo, en portapapeles, etcétera.

3.4.1.1 Cambios de tiempo en el equipo y contenido del portapapeles.

Para la creación del script de cambio de tiempo en el equipo es necesario utilizar la herramienta uptime**,** ésta es operada desde línea de comandos (véase figura 3.6) nos arroja el número de horas, minutos y segundos que el sistema ha permanecido en activo, desde la última vez que fue encendido (script 3.1).

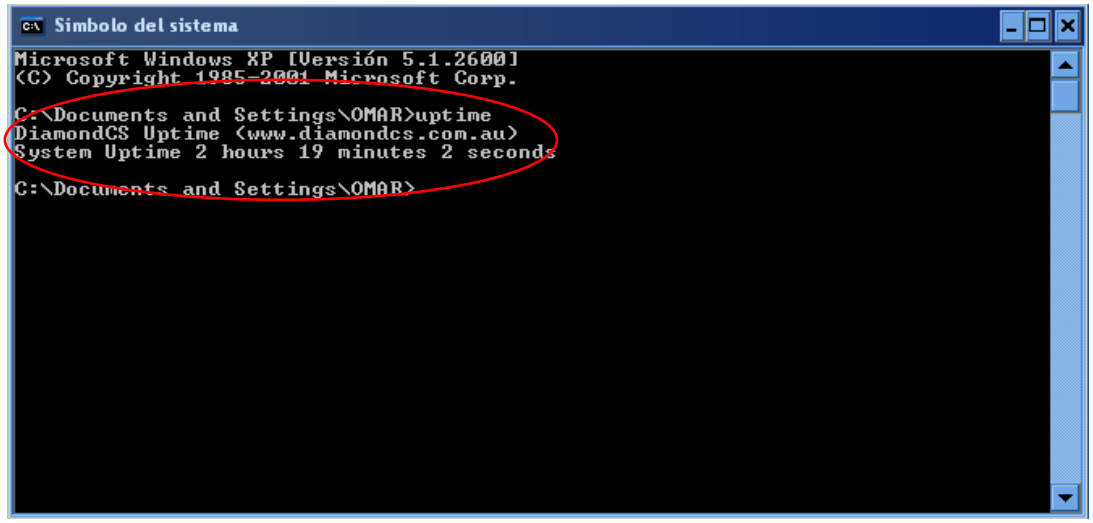

Figura 3.6: Ejecución del comando uptime.

Por otro lado una de las vulnerabilidades más grandes de una máquina es el acceso al portapapeles, un ejemplo muy nombrado de este ataque es mediante Internet Explorer que permite capturar los datos del portapapeles desde una página web. Aún que el uso malicioso de esta característica ya fue denunciado a finales del 2002, aún hoy día la configuración de la mayoría de sistemas con Internet Explorer permiten de forma transparente esta función.

El objeto clipboardData es el responsable de esta funcionalidad, y fue incorporada en la versión 5 de Internet Explorer. Básicamente admite tres métodos para interactuar con los datos del portapapeles, "getData" para capturar la información, "setData" escribe datos, y "clearData" para borrar el portapapeles.

Evidentemente tiene usos prácticos, lo cierto es que también puede ser utilizada de forma maliciosa. De hecho resulta muy simple diseñar una página web que intentara capturar los datos del portapapeles de todos los visitantes.

Si se puede suponer un problema de seguridad sobre el portapapeles, ya depende exclusivamente del grado de sensibilidad de la información que cada usuario suele copiar en el portapapeles en el día a día. Algunos ejemplos cotidianos son datos de hojas de cálculo, párrafos del procesador de textos, una conversación por mensajería instantánea que queríamos almacenar o incluso una contraseña que hemos copiado para pegar en un formulario de autenticación.

Si el usuario cree que los datos que suele copiar al portapapeles son sensibles, puede modificar la configuración de Internet Explorer para que avise o directamente rechace, cualquier intento de captura por parte de una página web.

Otros navegadores no contemplan esta funcionalidad y por tanto sus usuarios no requieren en esta ocasión ninguna configuración adicional para evitar un potencial uso malicioso de esta técnica.

El siguiente script ocupa un módulo de perl llamado clipboard, este módulo permite interactuar con el portapapeles de Windows, así como obtener el contenido, colocar, vaciar o bien estar en hibernación mientras se hacen cambios.

```
use strict;
my $arch_por="porta_de.txt";
use Win32::Clipboard;
open (MAN,"+>$arch_por");

###########TIEMPO DE ANALISIS#################################
my $salidaAnalisis=" ";
my @analisis=`uptime.exe`;
foreach $salidaAnalisis(@analisis){
   if($salidaAnalisis !~ /DiamondCS|Free|_|interfaces|ADDRESS|^$/){
            printf MAN $salidaAnalisis;
   }
}
##########CAMBIOS EN EL PORTAPAPELES#########################
 my $CLIP=Win32::Clipboard();

 my $algo=$CLIP->Get();
 printf MAN "El portapapeles contiene: $algo\n";
```

Script 3.1: Información del portapapeles y cambio de tiempo en el equipo.

Para la función **Win32::Clipboard->Get()**, que se utiliza en el script anterior, tenemos como resultado el contenido del portapapeles, no importando el formato, esto

se refiere a que la función get() trabaja con diferentes formatos de una manera general como son texto, mapa de bits o archivos, se hace referencia por que esta herramienta también tiene funciones que especifica cada uno de los formatos mencionados.

3.4.1.2 Usuarios con inicio de sesión por red.

El siguiente script (script 3.2) detecta qué usuarios han accedido al equipo por una sesión de red, la herramienta que se utiliza es **"net"**, la cual se describe a continuación.

```
use strict;
my $arch1="usrnet_de.txt";
open (MAN,"+>$arch1");
#################USUARIOS DEL SISTEMA#############
my @usuarios=`net users`;
my $usuario;
foreach$usuario(@usuarios){
        if($usuario!~/\-|comando|correctamente|^$/){
                printf MAN $usuario;
        }
}

#################RECURSOS COMPARTIDOS##############

my @compartidos=`net share`;
my $recurso;
foreach$recurso(@compartidos){
        if($recurso!~/\-|comando|correctamente|^$/){
                $recurso =~ s/(\(|\)|\+|\$|\\)/ /g;
                printf MAN $recurso;
        }
}
```

Script 3.2: Inicio de sesión por red.

3.4.1.3 Descripción de la herramienta net.

 # net user: Agrega o modifica cuentas de usuario o simplemente muestra información acerca de las cuentas que están en el equipo local o en el dominio. Como se muestra en la figura 3.7.

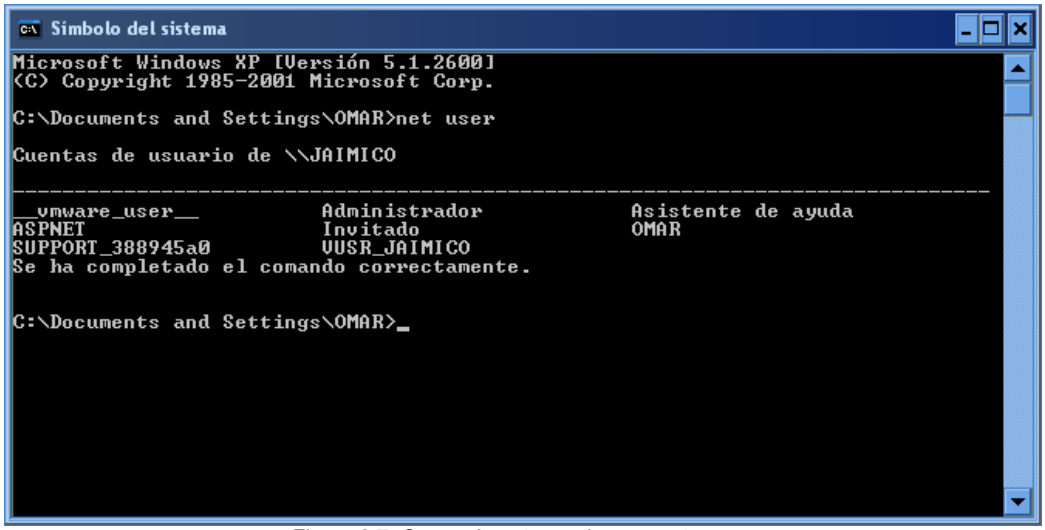

Figura 3.7: Comando net con el argumento user.

 # net share: Administra los recursos compartidos. Cuando se usa sin parámetros, net share presenta información acerca de todos los recursos compartidos del equipo local, como se muestra en la figura 3.8.

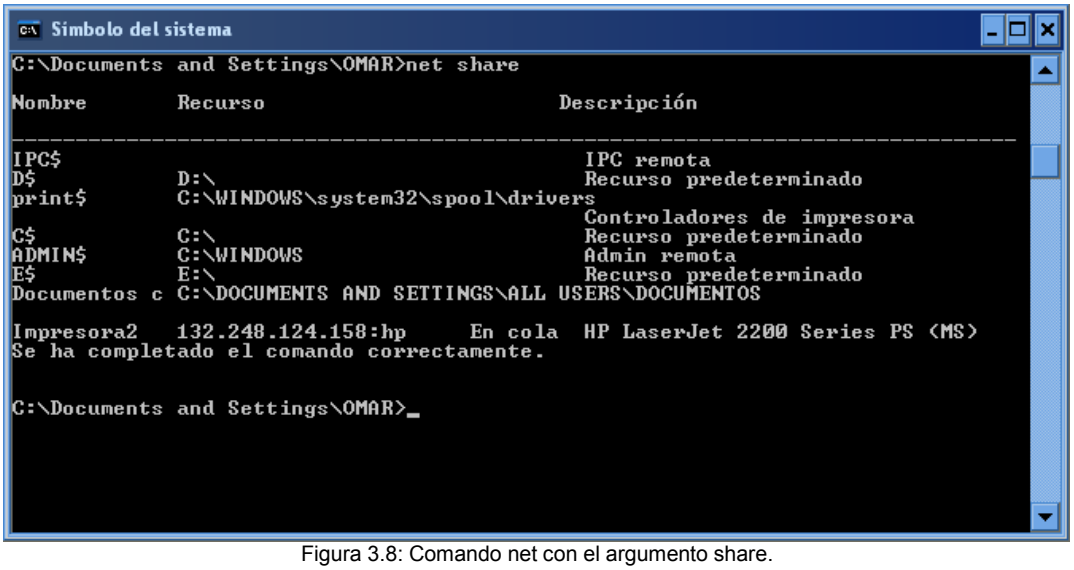

Figura 3.8: Comando net con el argumento share.

3.4.1.4 Procesos creados por el malware.

El siguiente script monitorea los procesos que genera el malware, para esta herramienta, se utilizan varios ejecutables como son **tlist**, **pslist, pmdump y kill**, en el monitoreo de procesos fue necesario realizar 3 diferentes scripts que serán descritos.

```
use strict;

#Programa que obtiene los procesos en el sistema antes de la ejecución del malware

 open(FH1,">procesos_inicio.txt");
 print FH1 `tlist`;
 close (FH1);
```

Script 3.3 listado de procesos.

La herramienta **tlist** Muestra una lista de IDPs (identificador de proceso), nombres y procesos ejecutándose sobre el equipo local como se muestra la figura 3.9.

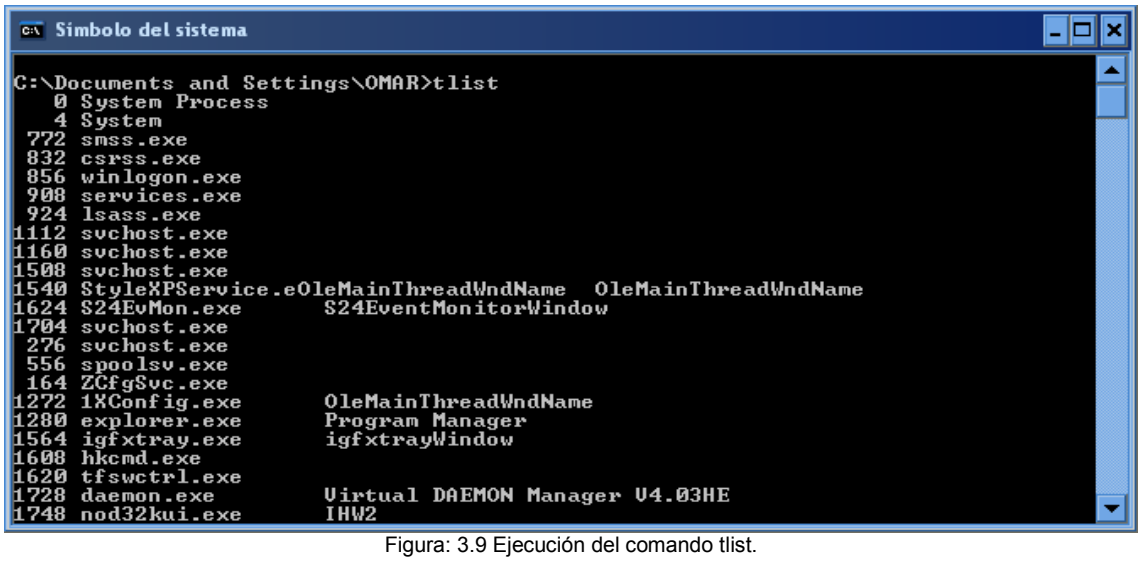

Figura: 3.9 Ejecución del comando tlist.

Por lo que en el script 3.3, se hace es un listado de los procesos con sus respectivos valores que inicialmente se encuentran corriendo en el sistema.

```
#programa que obtiene todos los procesos
#que se estana ejecutando en el sistema despues de la ejecucion de malware
#usando la herramieta pslist

use strict;
my $arch_pro_de="arch_pro_de.txt";

open(AR,"+>$arch_pro_de") or die "No se puede abrir: $!\n"; #Abre el archivo

my @linea=`pslist.exe`;    #@linea, contiene las lineas de la ejecucion del comando
my @linea2;
my $nom_proceso;

#dejamos en limpio el archivo
print AR "";

  print "-------------------------------------\n";
   print "\n\nInformación del Sistema:\n";
            foreach(@linea){
                        @linea2 = split(/\ /, $_);
                        $nom_proceso=$linea2[0];
                                    if($nom_proceso !~ /PsList|Copyright|Sysinternals|^\s$|Process|Name/){
                                                print AR $nom_proceso."\n";
                                    }
            }

if(close(AR)==1){
            print "Archivo creado correctamente";
            }
```

Script 3.4: Creación de procesos.

El script 3.4 utiliza la herramienta pslist, que describiremos en seguida:

El comando "**plist**" por defecto muestra la información de todos los procesos que están funcionando. La información esta enumerada respectivamente e incluye el tiempo que el proceso se ha ejecutado, la cantidad de tiempo en que éste ha permanecido en el núcleo, el usuario y la cantidad de memoria física que el sistema operativo ha asignado al proceso. Ver figura 3.10

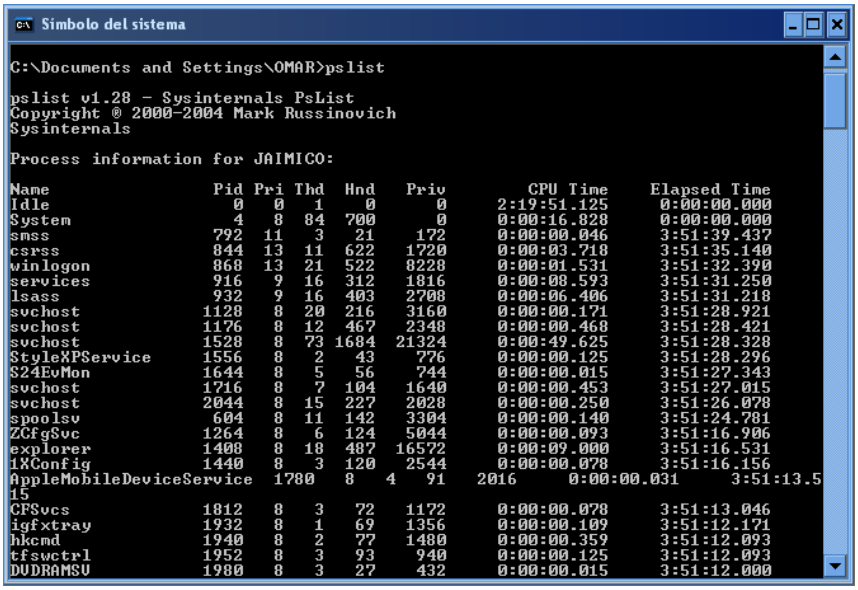

Figura 3.10: Comando pslist.

```
use strict;

####Programa que regresa los procesos creados después de la ejecución del malware
sleep(10);

my @procesos;
my @procesos1;
my @procesos2;
my $procesos1;
my $procesos2;
open(FH0,">procesos_final.txt");
print FH0 `tlist`;
close(FH0);
open(FH1,"<procesos_inicio.txt");
open(FH2,"<procesos_final.txt");
open(FH3,">procesos.txt");
my @procesos1=<FH1>;
my @procesos2=<FH2>;
my $n;
foreach $procesos2 (@procesos2){
    $n=0;
    foreach $procesos1 (@procesos1){
            if($procesos2=~/$procesos1/){
            $n=$n+1;
            }
            if($procesos2=~/TLIST/ || $procesos2 =~ /tlist/ || $procesos2 =~ /perl.exe/ || $procesos2 =~ /defservidor.exe/ ||
$procesos2 =~ /ejec_finales.pl/ ){
            $n=$n+1;
            }
    }
     if ($n eq 0){
     print FH3 "$procesos2";
     }
}
close(FH1);
close(FH2);
close(FH3);

open(FH4,"<procesos.txt");
my @procesos=<FH4>;
my $procesos;

foreach  $procesos (@procesos){
if ($procesos=~/\d+/){

print "$&\n";
`pmdump $& imagen_procesos.$&`;

`kill $&`;

}
}
```

Script 3.5 Procesos después de ejecución de malware.

El último script (script 3.5) tiene la función de monitoreo de procesos, utiliza 3 comandos para hacer la comparación con respecto a los procesos iniciales, el primer comando es **tlist,** este realiza una comparación con respecto de los procesos finales que no resultan en el archivo de inicio, después se utiliza el comando **pmdump** el cual crea una imagen del proceso, ya que PMDump es una herramienta que permite descargar el contenido de la memoria de un proceso a un archivo sin parar el proceso y el último comando que se utiliza es **kill** el cual elimina los procesos, la estructura para este comando es **kill PID.** Entonces el script, lo que hace es verificar los

procesos actuales en base a los iniciales, de los procesos diferentes captura la imagen y después los elimina.

3.4.1.5 Servicios.

Para listar los servicios que se generan en el sistema, es necesario implementar la herramienta **listservices**, la cual permite ver todos los servicios que se están ejecutando en el sistema con el siguiente formato, se listan 3 columnas; en la primera tenemos el nombre del servicio, en la segunda vemos el estado del servicio este solo tiene 2 formas presentes (stopped y running) y por ultimo tenemos se observa la descripción del servicio (véase figura 3.11).

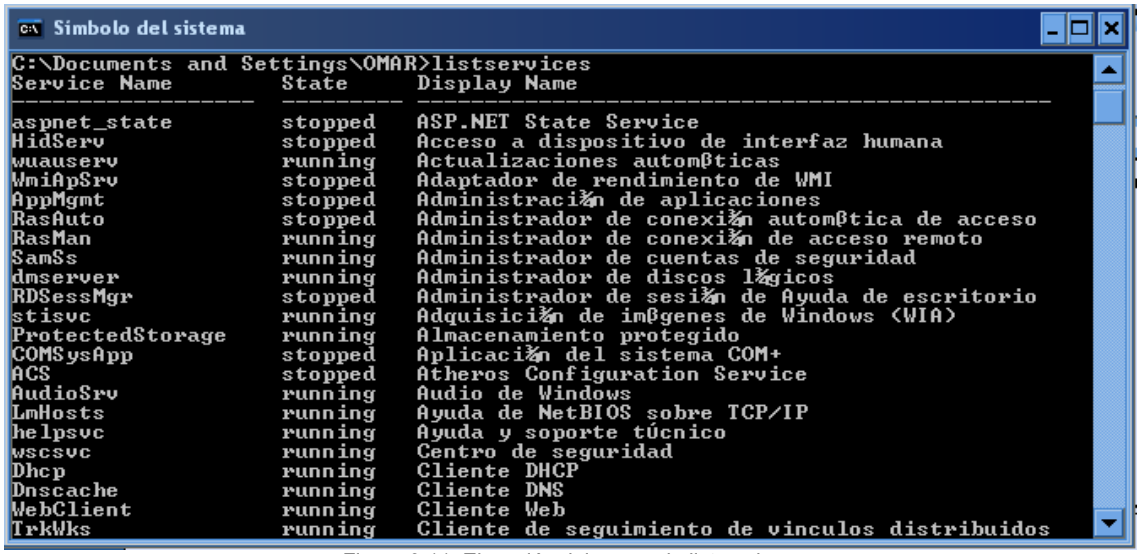

Figura 3.11: Ejecución del comando listservices.

El siguiente script (script 3.6) funciona al ejecutar la herramienta ***listservices***, ya antes mencionada y se guarda en un documento de texto llamado *servicios_anclon.txt* en el cual se imprime toda la salida del comando pero eliminando algunas líneas, que no son necesarias, como encabezados o notas de pie que aparecen en la ejecución del comando.

Script 3.6: Lista de servicios

```
use strict;
my @servicios;
my $salidaServicios;
my @lectura;
my $red;
open (MAN,"+>servicios_anclon.txt");

##########LISTA DE SERVICIOS Y SU ESTADO ACTUAL#############

my @servicios=`listservices.exe`;
foreach $salidaServicios(@servicios){
   if($salidaServicios !~ /This|more|full|Service|-|^$/){
            printf MAN $salidaServicios;
   }
}
close(MAN);

#############INICIA EDICION DE LA SALIDA###########
open (LEC,"servicios_anclon.txt");
@lectura=<LEC>;

 open (WRT,"+>servicios_de.txt");
```

```
foreach $red(@lectura){
        $red =~ s/(\(|\)|V|\+|\$)/ /g;
        if($red =~/[a-zA-Z0-9]/){
                printf WRT "$red";
        }
}
```

3.4.1.6 Información de red.

Para el siguiente análisis de información que es la parte de configuración de red, es necesario utilizar dos herramientas estrechamente ligadas, éstas son **iplist** y **promiscdetect,** la primera muestra todas las interfaces de red, así como sus características (broadcast, mascara de red, ip, véase figura 3.12), mientras que la segunda herramienta es un detector de sniffers, comprueba si alguno de los dispositivos de red que están instalados en la máquina, se está ejecutando en modo promiscuo, lo que generalmente es un signo de tener un sniffer instalado en la computadora, hablar de modo promiscuo es un modo de operación en el que una computadora conectada a una red compartida captura todos los paquetes, incluyendo los paquetes destinados a otras computadoras. Es muy útil para tareas de supervisión, sin embargo presenta un riesgo de seguridad dentro de una red de producción.

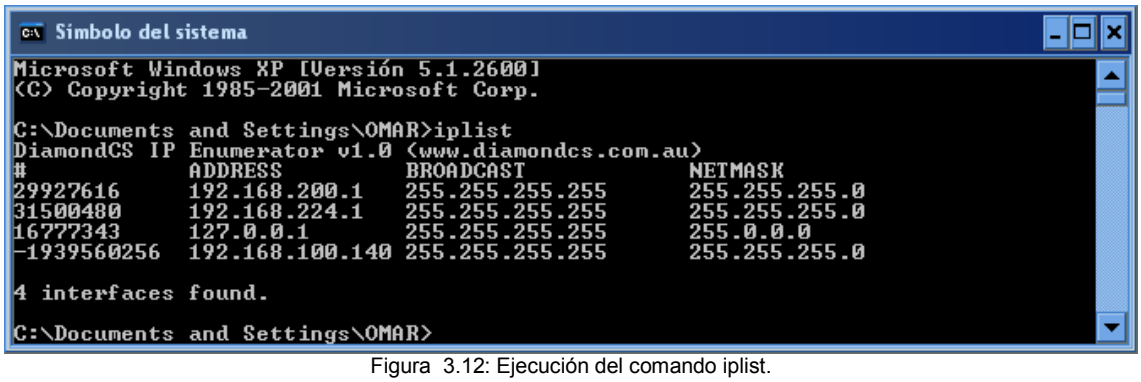

Figura 3.12: Ejecución del comando iplist.

El siguiente script despliega en un archivo la salida de los dos comandos antes mencionados pero únicamente con la información necesaria sin caracteres especiales para poder facilitar la forma de análisis.

Script 3.7: Información de Red.

```
use strict;
my $arch1="red_de.txt";
open(MAN,"+>$arch1");

###########INFORMACION DE RED#################################
my $salidaRed;
####### Comando para listar las ip's asignadas a la maquina ####
my @red=`iplist.exe`;

foreach $salidaRed(@red){
   if($salidaRed !~ /DiamondCS|Free|_|interfaces|ADDRESS|^$/){
        printf MAN $salidaRed;
   }
}
my $salidaPromiscua;
my @promiscuo=`promiscdetect.exe`;
foreach $salidaPromiscua(@promiscuo){
        if($salidaPromiscua !~ /PromiscDetect|- http:|^$/){
        $salidaPromiscua =~ s/\(//g;
        $salidaPromiscua =~ s/\)//g;
```

```
        if($salidaPromiscua =~/[a-zA-Z0-9]/){
                printf MAN $salidaPromiscua;
        }

        }
}
```

3.4.1.7 Llaves de registro.

Para el siguiente script es necesario realizar una búsqueda de llaves de registro para este paso se usara un modulo de perl llamado “Win32::Registry”, este modulo está basado en el comando **reg** del sistema operativo**.**

Este comando realiza operaciones de adición, cambio, importación, exportación y otras operaciones en la información de las subllaves del Registro y los valores de las entradas del Registro, también se especifica la ruta de acceso completa de la subllaves, las llaves raíz válidas son **HKLM**, **HKCU**, **HKCR**, **HKU** y **HKCC**.

Para este caso de estudio, se ocuparan algunas rutas en especifico, como se había mencionado al inicio del capítulo y se proporcionan las direcciones de las llaves que son usualmente comprometidas por el malware.

El script 3.8, muestra la programación para obtener las subllaves mediante el método **GetValues** perteneciente al modulo Win32::Registry, este programa toma todos y cada uno de los valores de las subllaves especificadas en la ruta o path, así como su tipo.

En la programación de este script es necesario hacer algunos arreglos, para ordenar el tipo de llave (%RegType), junto con unas conversiones que permitan desplegar el valor de las subllaves.

Script 3.8: Información de las llaves de registro.

```
#Programa que revisa las lleves de Registro en WIN2000
use strict
use Win32::Registry;
my %RegType = (
                                        0 => 'REG_0',
                                        1 => 'REG_SZ',
                                        2 => 'REG_EXPAND_SZ',
                                        3 => 'REG_BINARY',
                                        4 => 'REG_DWORD',
                                        5 => 'REG_DWORD_BIG_ENDIAN',
                                        6 => 'REG_LINK',
                                        7 => 'REG_MULTI_SZ',
                                        8 => 'REG_RESOURCE_LIST',
                                        9 => 'REG_FULL_RESOURCE_DESCRIPTION',
                                        10 => 'REG_RESSOURCE_REQUIREMENT_MAP'
                                        );
my $PATH="Software\\Microsoft\\Windows\\CurrentVersion\\";
my @llaves=("Run","RunOnce","RunOnceEx","policies","Setup");

my $Register;
my $RegType;
my $RegValue;
my $RegKey;
my $value;
my %values;
open(FH,">llaves.txt");
foreach (@llaves) {
```

```
$Register=$PATH.$_;
$HKEY_LOCAL_MACHINE->Open($Register,$hkey)|| die $!;

$hkey->GetValues(\%values);

foreach $value (keys(%values))
        {
        $RegType        = $values{$value}->[1];
        $RegValue       = $values{$value}->[2];
        $RegKey  = $values{$value}->[0];
        next if ($RegType eq '');
        $RegKey  = 'Default' if ($RegKey eq '');      #nombre por defecto de los nombres de la subllave
        print FH "$Register.$RegKey";
        print FH " ($RegType{$RegType}) : ";
        SWITCH:
                {
                        if ($RegType == 4)
                        {printf FH "Ox%1x \n", unpack("L",$RegValue); last SWITCH; }
                if ($RegType == 5)
                        {printf FH "Ox%1x", unpack("N",$RegValue); last SWITCH; }
                if ($RegType < 8 )
                        {printf FH "$RegValue\n"; last SWITCH; }
                print FH"\n";
                }

        }
$hkey->Close();
}
close(FH);

open(FH2,"<llaves.txt");
open(FH3,">llaves_de.txt");
@llaves=<FH2>;

foreach $llave (@llaves){
        if($llave=~/^$/){
                next;
        }
        $llave=~s/\\/_/g;
        $llave=~s/\//_/g;
        $llave=~s/(\[|\])/ /g;
        $llave=~s/(\(|\))/ /g;
        $llave=~s/\"/ /g;
        $llave=~s/:/ /g;
        $llave=~s/\./ /g;
        $llave=~s/!/ /g;
        $llave=~s/^[             ]*//g;
        print FH3 $llave;

}
close(FH2);
close(FH3);
```

3.4.1.8 Tareas programadas.

Ésta sección hace uso de la herramienta **schtasks**, ya que sirve para calendarizar comandos, programas y se ejecuten periódicamente o a una hora especifica. Agrega y quita tareas del programa, inicia y detiene tareas a petición, también muestra y cambia tareas programadas. Utilizando este comando con la opción query, podemos saber las tareas programadas que tiene el sistema que se está analizando, la sintaxis del comando es la siguiente.

Un ejemplo de schtasks o schtasks /query, por defecto despliega la tabla en el formato que se muestra en la figura 3.13.

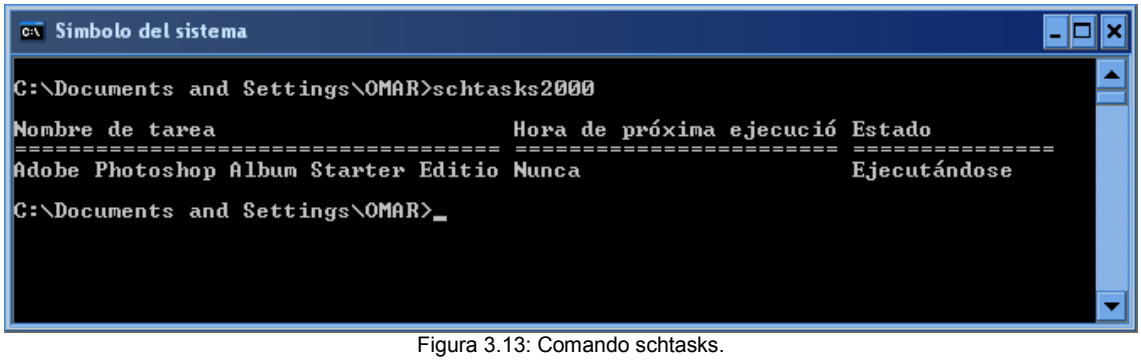

Figura 3.13: Comando schtasks.

Para el análisis de este comando, es necesario quitarle algunas líneas y caracteres especiales y así hacer una comparación optima.

```
use strict;

open(AUX1,"+>task_de.txt");
my @tareas_a=();
@tareas_a=`schtasks2000`;

foreach (@tareas_a){
        if($_ !~ /Nombre de tarea|=|^$/){
                $_ =~ s/(\:|\/|\,)/ /g;
    print AUX1 "$_";
    }
}

close(AUX1);

exit;
```

Script 3.9: Información de tareas programadas.

3.4.1.9 Cambios en el sistema de archivos.

Para ver los cambios en el sistema de archivos que realiza el malware ejecutado en el laboratorio, se escogió una herramienta llamada integrit la cual es del tipo hids.

Un HIDS (Host-based Intrusion Detection System), es un tipo de IDS, su funcionamiento se basa en recolectar información contenida en el hosts en un momento dado y guardar esta información en una "bases de datos" para posteriormente realizar la comparación. Este tipo de ids depende de la confiabilidad de estos registros, de nada sirve instalarlo en una máquina que haya sido comprometida.

Estas herramientas tienen la capacidad de alertarnos sobre modificaciones en archivos o directorios, además de nuevos elementos y los más avanzados sobre elementos borrados. Como ejemplo de ids tenemos a Tripwire que presenta como desventaja ser software comercial, una alternativa fue utilizar una herramienta con licenciamiento GNU llamada **integrit**.

Integrit es un IDS del tipo HIDS escrita por L. Cashin, la versión usada es la 4.1, esta versión es capaz de detectar archivos y/o directorios modificados, y nuevos elementos, está desarrollada totalmente en lenguaje C, fue diseñada para trabajar sobre S.O. que admitan el estándar de archivos POSIX como son: Linux, BSD.

Aunque es posible exportarla a ambientes Windows, utilizando un emulador como Cygwin, que es la mejor alternativa para el presente laboratorio.

Para poder hacer uso de esta herramienta se realizaron los siguientes pasos:

- Instalar la herramienta cygwin.
- Instalar integrit y vi utilizando la herramienta de instalación que viene en cygwin.
- Hacer una copia del Unix virtual a la unidad C de la siguiente manera.
  - cp /usr/sbin/integrit.exe /cygdrive/c
- Agregar la librería cygdrive1.dll al directorio C:/WINDOWS/System32/
- Verificar el funcionamiento de la herramienta intregrit con el siguiente comando (véase en la figura 3.14).

  C:\integrit.exe –h

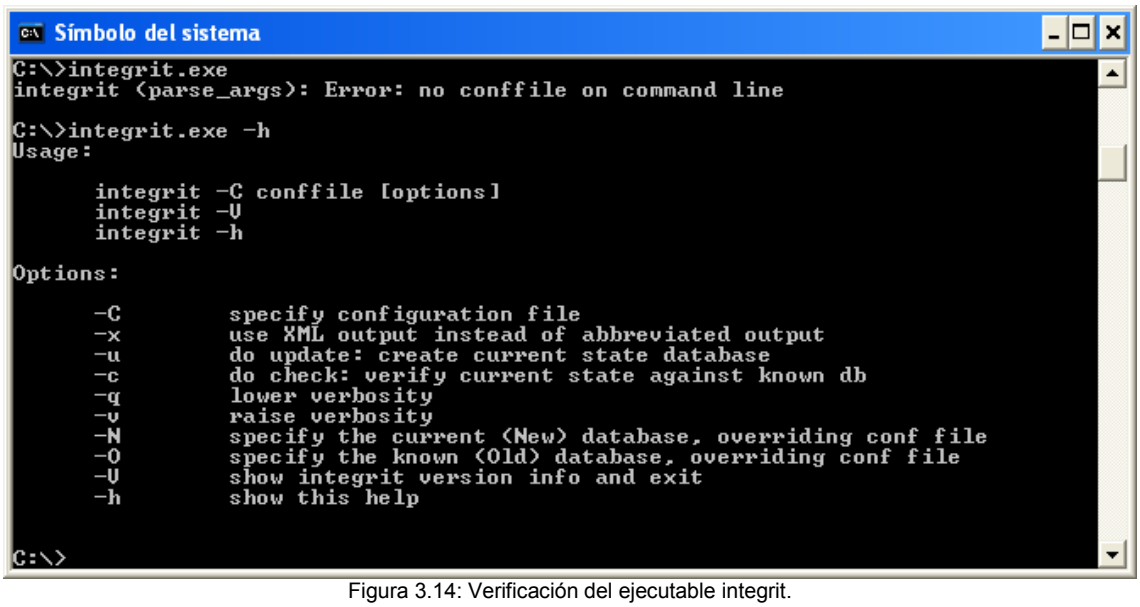

Figura 3.14: Verificación del ejecutable integrit.

Se creó un archivo de configuración, al cual se nombro integrit.conf (veasé Archivo de configuración1), y tiene el siguiente contenido:

root=/home/ ← Archivo Raíz desde donde se empezara el análisis
known=known.cdb ← Archivo donde se guarda la base creada en la primera ejecución.
current=current.cdb ← Archivo donde se guarda la ultima base creada.
¡/dev/nst0 ← Con ! excluimos del análisis a un archivo o directorio.

El archivo de configuración de esta herramienta se muestra de la siguiente manera:

```
root=C://
known=known.cdb
current=current.cdb
!C:///Documents\ and\ Settings/All\ Users/Datos\ de\ programa/Microsoft/Network/Downloader/qmgr0.dat
!C:///Documents\ and\ Settings/All\ Users/Datos\ de\ programa/Microsoft/Network/Downloader/qmgr1.dat
!C:///Documents\ and\ Settings/Administrador/NTUSER.DAT
!C:///Documents\ and\ Settings/Administrador/ntuser.dat.LOG
!C:///Documents\ and\ Settings/Administrador/Configuración\ local/Datos\ de\ programa/Microsoft/Windows/UsrClass.dat.LOG
!C:///Documents\ and\ Settings/Administrador/Configuración\ local/Datos\ de\ programa/Microsoft/Windows/UsrClass.dat
!C:///WINNT/SoftwareDistribution/EventCache
!C:///WINNT/system32/config/software.LOG
!C:///WINNT/system32/config/default.LOG
!C:///WINNT/system32/config/default
!C:///WINNT/system32/config/SECURITY
!C:///WINNT/system32/config/security
!C:///WINNT/system32/config/SECURITY.LOG
!C:///WINNT/system32/config/security.log
!C:///WINNT/system32/config/SYSTEM.ALT
!C:///WINNT/system32/config/system.alt
!C:///WINNT/system32/config/SYSTEM
!C:///WINNT/system32/config/system
!C:///WINNT/system32/config/SAM
!C:///WINNT/system32/config/sam
!C:///WINNT/system32/config/SAM.LOG
!C:///WINNT/system32/config/SOFTWARE
!C:///WINNT/system32/config/software
!C:///WINNT/system32/config/DEFAULT
!C:///WINNT/system32/config/AppEvent.Evt
!C:///WINNT/system32/config/SysEvent.Evt
!C:///WINNT/system32/wbem
!C:///WINNT/system32/NtmsData
!C:///WINNT/security/logs/scepol.log
!C:///WINNT/Debug/PASSWD.LOG
!C:///WINNT/Debug/ipsecpa.log.last
!C:///WINNT/Debug/ipsecpa.log
!C:///WINNT/Debug/oakley.log.sav
!C:///WINNT/Debug/oakley.log
!C:///WINNT/inf
!C:///WINNT/Tasks/SA.DAT
!C:///WINNT/SchedLgU.Txt
!C:///WINNT/CSC/00000001
!C:///WINNT/CSC/ShellIconCache
!C:///WINNT/WindowsUpdate.log
!C:///WINNT/SoftwareDistribution/ReportingEvents.log
!C:///WINNT/SoftwareDistribution/DataStore/Logs
!C:///WINNT/SoftwareDistribution/DataStore/DataStore.edb
!C:///WINNT/osiris/private/auth.db
!C:///Documents\ and\ Settings/Administrador/Configuración\ local/Temp/jusched.log
!C:///Documents\ and\ Settings/Administrador/Configuración\ local/Historial/History.IE5
!C:///Documents\ and\ Settings/Administrador/Configuración\ local/Archivos\ temporales\ de\ Internet/Content.IE5/index.dat
!C:///Documents\ and\ Settings/Administrador/Cookies/index.dat
!C:///Documents\ and\ Settings/Administrador/Reciente
!C:///Documents\ and\ Settings/All\ Users/Documentos/DrWatson
!C:///Documents\ and\ Settings/Administrador/Datos\ de\ programa/Microsoft/Internet\ Explorer/Desktop.htt
!C:///Documents\ and\ Settings/Administrador/ntuser.ini
!C:///cygwin/home/Administrador
!C:///integrit
!C:///scripts
!C:///pagefile.sys
```

Archivo de Configuración1: Configuración de integrit

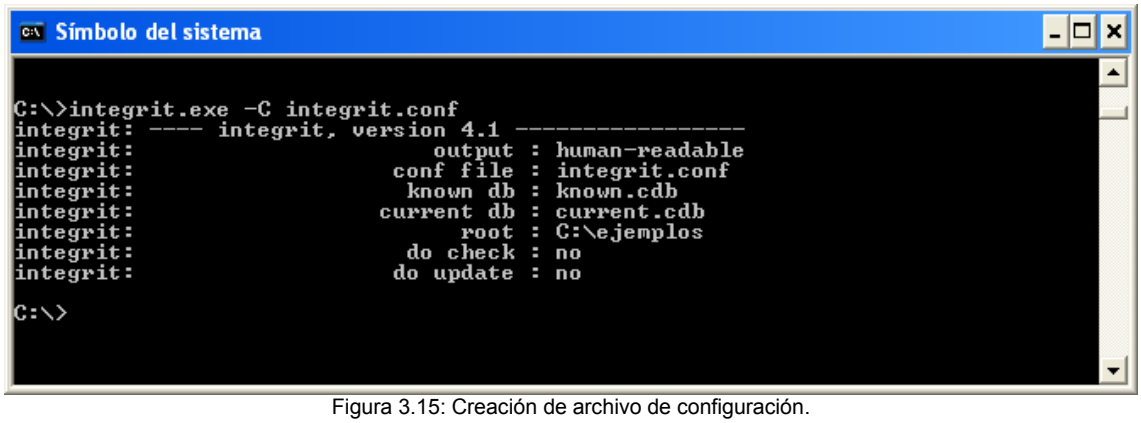

Figura 3.15: Creación de archivo de configuración.

Utilizando el siguiente comando se creara la base de firmas con la que se compara en las siguientes ejecuciones de esta herramienta:

C:\>Integrit.exe –C integrit.conf –u

De esta manera, se obtiene la primera base de datos de los archivos que se tienen en el laboratorio, como se muestra en la figura 3.16.

Figura 3.16: Creación de la base de datos de los archivos del sistema.

Después se deberá copiar la base de datos

#cp current.cdb known.cdb

Para verificar la integridad del sistema de archivos ejecutar el siguiente comando, (véase en la figura 3.17).

#Integrit.exe –C integrit.conf –c

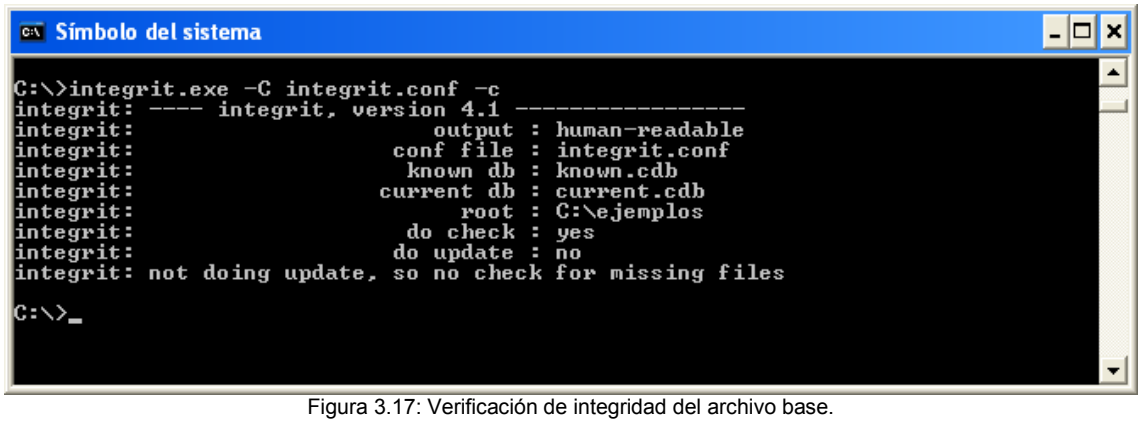

Figura 3.17: Verificación de integridad del archivo base.

Un ejemplo donde se observa el uso de un archivo y la creación de otro se muestra en la figura 3.18. El primer cambio mostrado, aparece con la marca **changed,** este es un archivo que se modifico después de crear la base de datos de los archivos, mientras que al segundo se marca con la marca **new** que es un nuevo archivo.

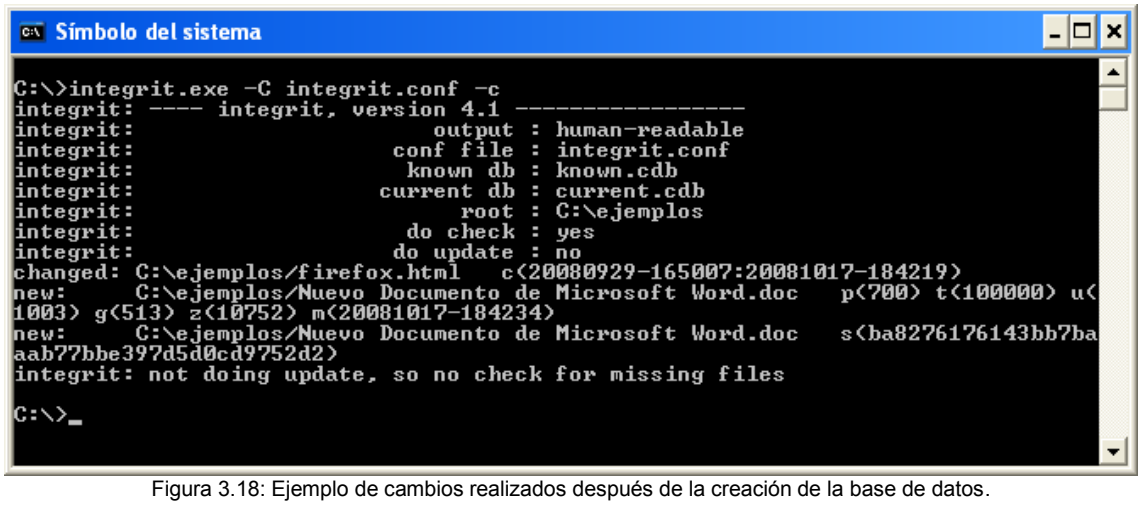

Figura 3.18: Ejemplo de cambios realizados después de la creación de la base de datos.

Nota: Este HIDS fue desarrollado para ambiente Unix, por lo que solo se debe utilizar vi, para editarlo.

3.4.1.10 Informe de puertos abiertos.

Un puerto es una forma genérica de denominar a una interfaz por la cual diferentes tipos de datos pueden ser enviados y recibidos. Dicha interfaz puede ser física, o a nivel software, Los puertos de comunicaciones TCP/IP se numeran desde el 1 al 65535. Los puertos que van desde el 1 al 1023 se denominan “well known ports” o puertos bien conocidos y están reservados para determinados estándares de comunicación (Web, FTP, Telnet...) o para procesos con permisos de administrador. El resto de puertos desde el 1024 al 65535 se denominan puertos azarosos y son utilizados por diversas aplicaciones.

El hecho de que sea necesario mapear uno o más puertos para hacer funcionar ciertas aplicaciones en un determinado sistema, depende principalmente de dos factores: del software o aplicación que se quiere ejecutar en el sistema y el hardware de red y/o conexión a Internet que usa ese sistema.

Para está parte se emplea la herramienta **openports**, la cual es útil para la detección de troyanos y software maligno, que se suelen conectar a un sistema a

través de uno de estos puertos que se tienen abiertos. Openports considera todos los puertos abiertos del TCP y del UDP en el sistema, también muestra toda la información sobre los puertos que estén abiertos, el tipo de conexión, el nombre y el ID del proceso, números de puerto local y remoto, IP de conexión remota y estado, otra función que realiza es que puede cerrar cualquier conexión TCP no deseada directamente desde la aplicación; o bien exportar la lista de conexiones a otro formato para analizarla más tarde.

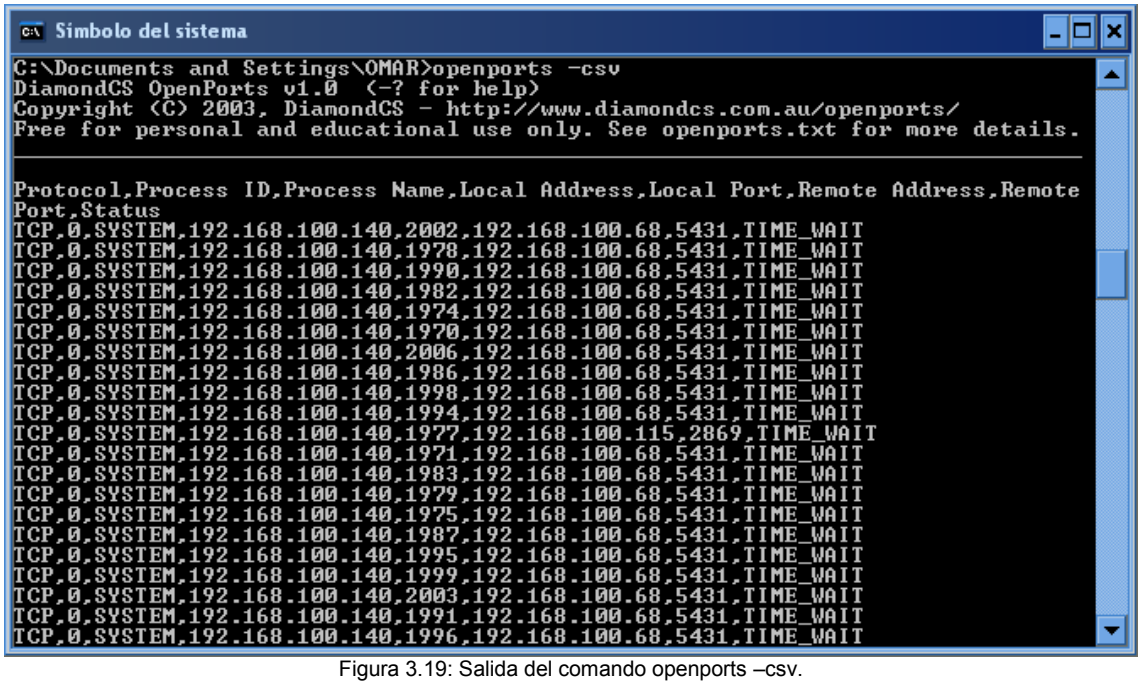

Figura 3.19: Salida del comando openports –csv.

Para la realización del script, solo es necesario eliminar campos y líneas no útiles para el comparativo, el script es el siguiente (script 3.11).

```
##########PUERTOS UTILIZADOS Y SU ESTADO ACTUAL###############
open (MAN,"+>ptos_usa_an.txt");

my @comando=`openports.exe -csv`;
foreach $salidaCompleta(@comando){
   if($salidaCompleta !~ /DiamondCS|Free|_|^$/){
            @openports=split(/,/,$salidaCompleta);
            printf MAN $openports[3]." ".$openports[4]." ".$openports[0]." ".$openports[7];
   }
}
```
Script 3.11: Información de puertos abiertos.

La salida que tenemos de este comando es el propietario de la aplicación, la ip origen, el protocolo y el puerto.

3.4.1.11Script de comparación de archivos.

Al contar con todos los archivos de los resultados de cada uno de los módulos del laboratorio y que son necesarios para el análisis, se requiere llevar a cabo la comparación del resultado final con lo obtenido en la base para ello se construye el script de comparación, este script se usa con la misma base, lo que cambia en cada uno de los puntos a revisar, son variables y archivos, por lo que el algoritmo de comparación es el mismo y se presenta así (véase script 3.12).

Script 3.12: Script de comparación.

```
########COMPARA FINA2 CON FINAL, LO QUE CAMBIO O NO EXISTE
########EN FINAL CON FINAL2 LO IMPRIME
open(FH,"archivo_de.txt")||die"no puede abrir $!";
open(FHI,"archivo_an.txt")||die"no puede abrir $!";
open (MAN,"+>archivo_final.txt");

my $array1;
my $array2;
my $array3;
my $n;
my @cadenas=<FH>;
my @cadenas1=<FHI>;
my $firma1=`md5sum archivo_de.txt`;
my $firma2=`md5sum archivo_an.txt`;

if($firma1 == $firma2){

        print MAN "No Hubo Cambios en XXXXXXXXXXXX\n";
}

else{
        print MAN "=================================== ====================== ===============\n";
        print MAN "                                    Diferencias En Los ARCHIVOS\n";
        print MAN "=================================== ====================== ===============\n";

        print MAN  "Elementos nuevos o modificados\n";
        print MAN "=================================== ====================== ===============\n";
        foreach     $array1(@cadenas){
                        #si el archivo es diferente de vacio.
                        if($#cadenas1>0){
                                foreach $array2(@cadenas1){
                                        if($array1=~$array2){
                                                $n=0;
                                                next;
                                        }
                                        $n+=1;
                                        if($n==($#cadenas1+1)){
                                                print MAN "$array1";
                                                last;
                                        }
                                        }
                                        $n=0;
                        }
                }

                #si el archivo original esta vacio.
                if($#cadenas1<=0){
                                foreach $array3(@cadenas){
                                        print MAN $array3;
                                        }
                                }
        print MAN "\n\n";
        print MAN "Elementos borrados o modificados \n";
        print MAN "=================================== ====================== ===============\n";

        foreach $array2(@cadenas1){
                        #si el archivo es diferente de vacio.
                        if($#cadenas>0){
                                foreach $array1(@cadenas){
                                        if($array1=~$array2){
                                                $n=0;
                                                next;
                                        }
                                        $n+=1;
                                        if($n==($#cadenas+1)){
```

```
                                                print MAN "$array2";
                                                last;
                                        }
                                        }
                                        $n=0;
                        }
                }
                        if($#cadenas<=0){
                                foreach $array3(@cadenas1){
                                        print MAN $array3;
                                        }
                                }

}
```

Este script está dividido en 4 partes que se describen a continuación:

- La primera parte consta de la comparación de los archivos por medio de la firma md5 está parte es integridad de información, ya que si en los archivos finales no se genero ningún cambio, esto se refleja al comparar las firmas md5 y son iguales, en caso de lo contrario, ya sea que se haya suscitado un cambio mínimo en el archivo, es necesario realizar otra búsqueda.

- La segunda comparación de este script, es realizar un barrido del archivo final, con respecto al archivo inicial, en está parte lo que se hace es buscar línea por línea y hacer una comparación, donde se detecta un cambio, se imprime la línea diferente.

- El tercer método de comparación es cuando se tiene el archivo original vacío y en el archivo final se encuentran cambios, aquí se imprime todo el archivo final.

- Y por último, se hace la comparación cuando se modificó el archivo inicial.

3.4.1.12 Envío de información al servidor de análisis.

Teniendo los datos recopilados por los diversos scripts de auditoría, y las imágenes de los procesos de malware, ahora es necesario enviar estos datos a un servidor, por lo que se utiliza SSH. El destino de los datos será el servidor que tiene la función de firewalls, Como se comento anteriormente, la segunda función de este servidor es para web, el cual publica los resultados.

Procedimiento para envío de información:

- Instalar un cliente de SSH para Windows, la herramienta más factible es PuTTY. Que se puede descargar desde la siguiente liga (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) se recomienda descargar e instalar el putty-0.60-installer.exe, ya que este provee todos los binarios necesarios para está tarea. Solo basta ejecutar el programa.

- Incluir los binarios de PuTTY en el path del sistema.

- Generar la llave privada y pública, para esto se da un clic en el PuTTYgen y seguir los pasos que a continuación se describen:

- Dar clic en Generate y mover el mouse hasta que el programa indique que ha terminado la generación de las llaves.
- Guardar la llave publica dando clic en el botón Save public key.para esto de clic en el botón Save private key. No se deberá poner contraseña a la llave.
- Pegar la llave pública en el archivo authorized_keys2, esto es del lado del servidor de SSH (Linux), para el proyecto se encuentra en /root/.ssh/ authorized_keys2.
- Reiniciar el demonio del SSH /etc/init.d/sshd restart en el Linux.

- Terminada la configuración del SSH utilizando llaves como medio de autenticación, ahora solo basta probar. La intención es enviar un conjunto de archivos a otro servidor sin la necesidad de teclear una contraseña, esto se hará utilizando el siguiente comando:

  C:\Documents and Settings\Oscar\Escritorio>pscp -i sauro-private.ppk -r Tesis root@192.168.1.12:

- Como se puede observar se utiliza el secure copy versión PuTTY y las opciones son **–i** para utilizar llaves y el parámetro que se necesita para la ubicación y el nombre de la llave privada, que en este caso se llama **llave_priv.ppk**, la opción **–r** la cual se utiliza para enviar un directorio y por último **usuario@ipdelservidor:** .Al emplear el anterior comando se manda la carpeta **Tesis** al servidor **192.168.1.11**, utilizando para esto el usuario **root** y como directorio destino **home de root.**

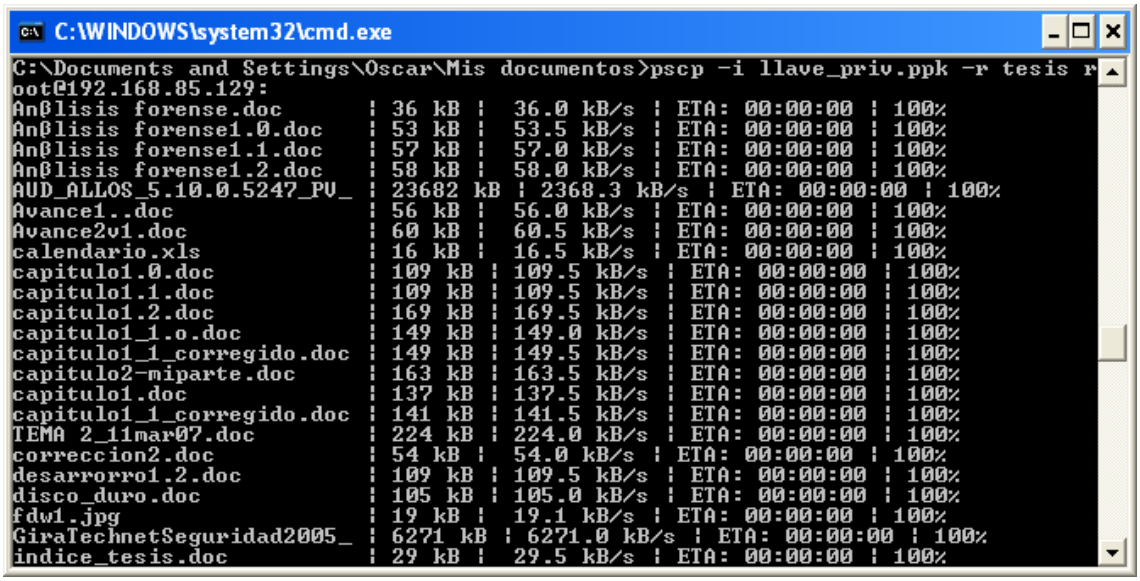

Figura 3.20: Salida del envío de información.

- Se adjunta el archivo de configuración del servicio de SSH:

```
Port 22
Protocol 2
SyslogFacility AUTHPRIV
#PermitRootLogin no
X11Forwarding no
UsePrivilegeSeparation yes
Banner /etc/ssh/banner
```

```
Subsystem       sftp    /usr/libexec/openssh/sftp-server
RSAAuthentication yes
#AllowUsers admin bases
```

3.5. Descripción de la herramienta dd.

Copia un archivo, de entrada estándar a salida estándar por defecto, opcionalmente cambia el tamaño de los bloques de entrada salida y realiza diversas conversiones.

if	=	ARCHIVO leer la entrada del archivo indicado.
of	=	ARCHIVO dirigir la salida al archivo indicado.
bs	=	BYTES leer y grabar entrada y salida en bloques.
conv	=	CONVERSION[,CONVERSION]... convertir según argumentos.

3.6 Creación de scripts de comunicación.

Esta arquitectura consiste básicamente en que un programa “el cliente” realiza peticiones a otro programa “el servidor” que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras (véase figura 3.21).

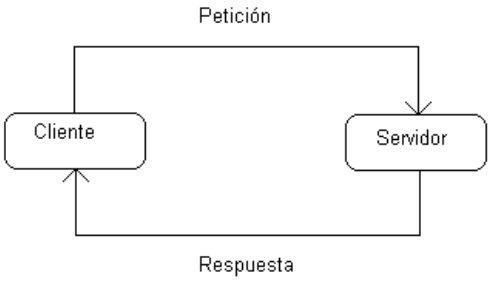

Figura 3.21: Diagrama de cliente servidor.

3.6.1 Scripts de cliente.

Como primer punto, el script verifica que exista el archivo configuraciones.txt, en caso contrario termina el proceso, de no ser así analiza el archivo obteniendo datos como son la ip y el puerto por el cual el servidor está atendiendo peticiones, suponiendo que el servicio este apagado, cerrara la conexión. Si existe un servidor escuchado, preguntara el nombre del archivo a enviar, verificara que dicho archivo exista, de ser positivo, se inicia el envío del mismo, en el caso contrario, se requiere que se confirme el nombre del archivo y pedirá el nombre nuevamente. A continuación determinar la firma del archivo enviado y la entregara al servidor, si ambas firmas son iguales, aparece un mensaje de envío exitoso de no ser así pedirá volverá a enviar el archivo.

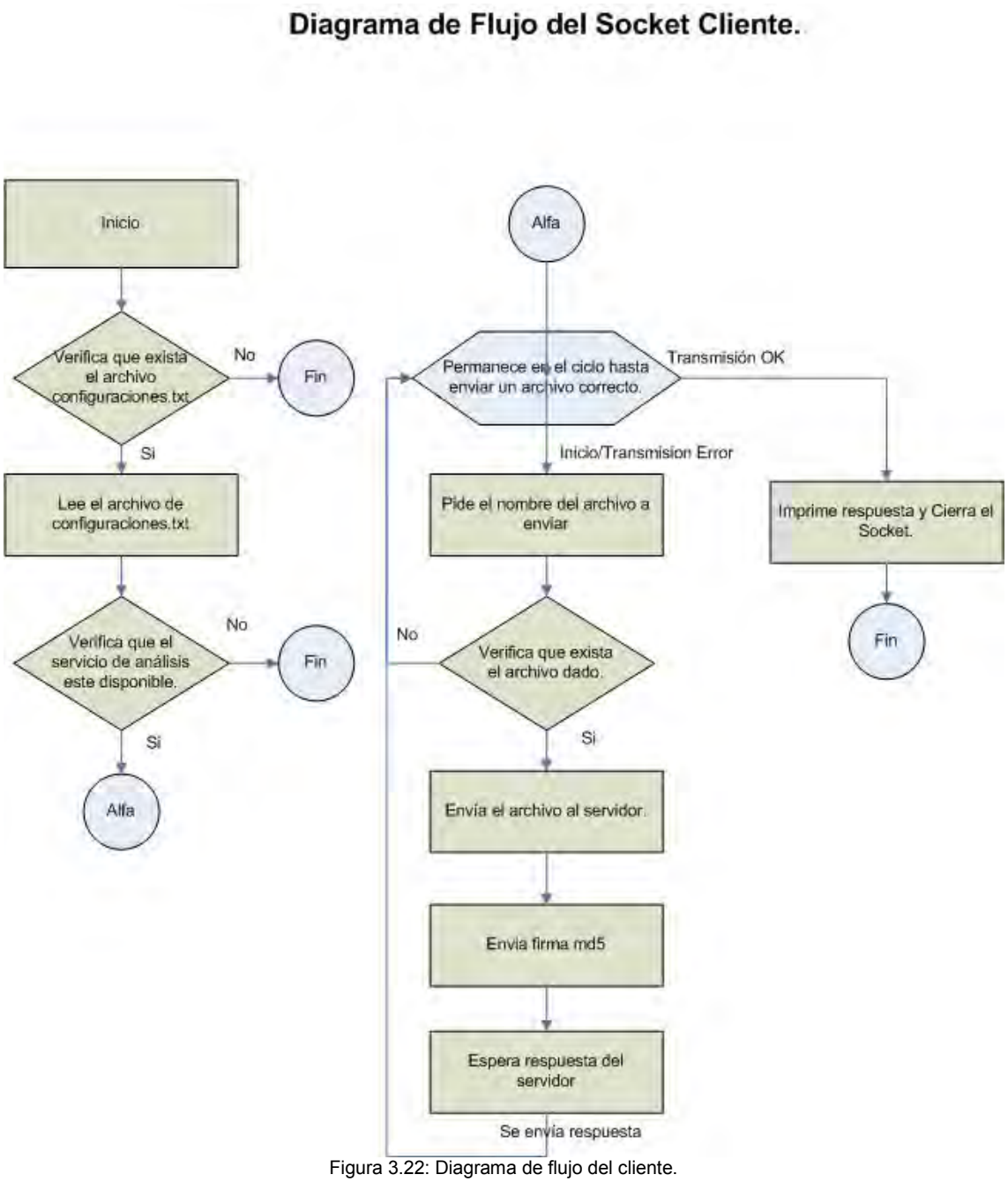

Figura 3.22: Diagrama de flujo del cliente.

Script 3.13: Script de comunicación Cliente.

```
#!/usr/bin/perl -w
# servidor
#use strict;
use IO::Socket;
my $name;
my @lista;
$n=0;

#############################################################################
#En el siguiente bloque se obtienen los parametros necesarios para iniciar la
#comunicacion con el servidor de malware

                        open(ML,"configuracion.txt");
                        @conf=<ML>;
                        foreach(@conf){
```

```
                                chomp($_);
                                if($_!~/\#/){
                                        chomp($_);
                                        @lista=split(/:/,$_);
                                }
                        }
                        print "#################################################\n";
                        print "#     Universidad Nacional Autonoma de Mexico    #\n";
                        print "#             Facultad de Ingenieria             #\n";
                        print "#         Analisis de Malware Automatico         #\n";
                        print "#                 Cliente 1.0                    #\n";
                        print "#################################################\n";
                        print "\nLa ip de su servidor  es :$lista[0] \n";
                        print "El puerto habilitado  es :$lista[1] \n";
                        #print "El tiempo de analisis es :$lista[2] \n";


$sock = new IO::Socket::INET (PeerAddr => $lista[0],
                        PeerPort => $lista[1],
                        Proto    => 'tcp',
                                        Type     => SOCK_STREAM
                       );
                                        die "El socket no se pudo crear. Reason: $!\n" unless $sock;
print "Cual es el nombre del archivo a enviar:";
#Se verifica la existencia del archivo y en su caso se transmite al servidor
while($n!=5){
        $name = <STDIN>;
        chomp $name;
        if(-e $name){
        open(IN,"$name") || die "No se pudo abrir el archivo $name";
        binmode(IN);
        autoflush $sock 1;
        while(<IN>){
                print $sock  $_;
                print "\nTransmitiendo....";
        }
        close (IN);
        close ($sock);
        $n=5;
        }
        else{
                $n=0;
                print "  El archivo $name no existe \n";
                print "  Verifica el nombre del archivo \n";
                print "\nCual es el nombre del archivo a enviar: ";
        }
}

#############################################################################
#Se verica la coherencia de la transmision

print "\nVerificando transmision espere...\n";
sleep(2);
my $firma=`md5sum $name`;
my @fir2=split('\ ',$firma);
print "FIRMA malware \n$fir2[0]\n";
sleep(1);
print "\nEnviando md5.";
$sock = new IO::Socket::INET (PeerAddr => $lista[0], #ip de la maquina servidor
                        PeerPort => $lista[1],
                        Proto    => 'tcp',
                                        Type     => SOCK_STREAM
                       );

print $sock  $fir2[0];
close ($sock);
```

```
sleep(3);
#Se recibe la respuesta del servidor

$socket_r = IO::Socket::INET->new(PeerAddr => $lista[0],
                                  PeerPort => $lista[1],
                                  Proto    => "tcp",
                                  Type     => SOCK_STREAM)
                                    or die "no se pudo crear conexion de lectura\n";

                while(<$socket_r>){

                                        print "$_"; #IMPRIMIENDO RESPUESTA

                                }
                close($socket_r);
                close(ML);
```

3.6.2 Scripts de servidor.

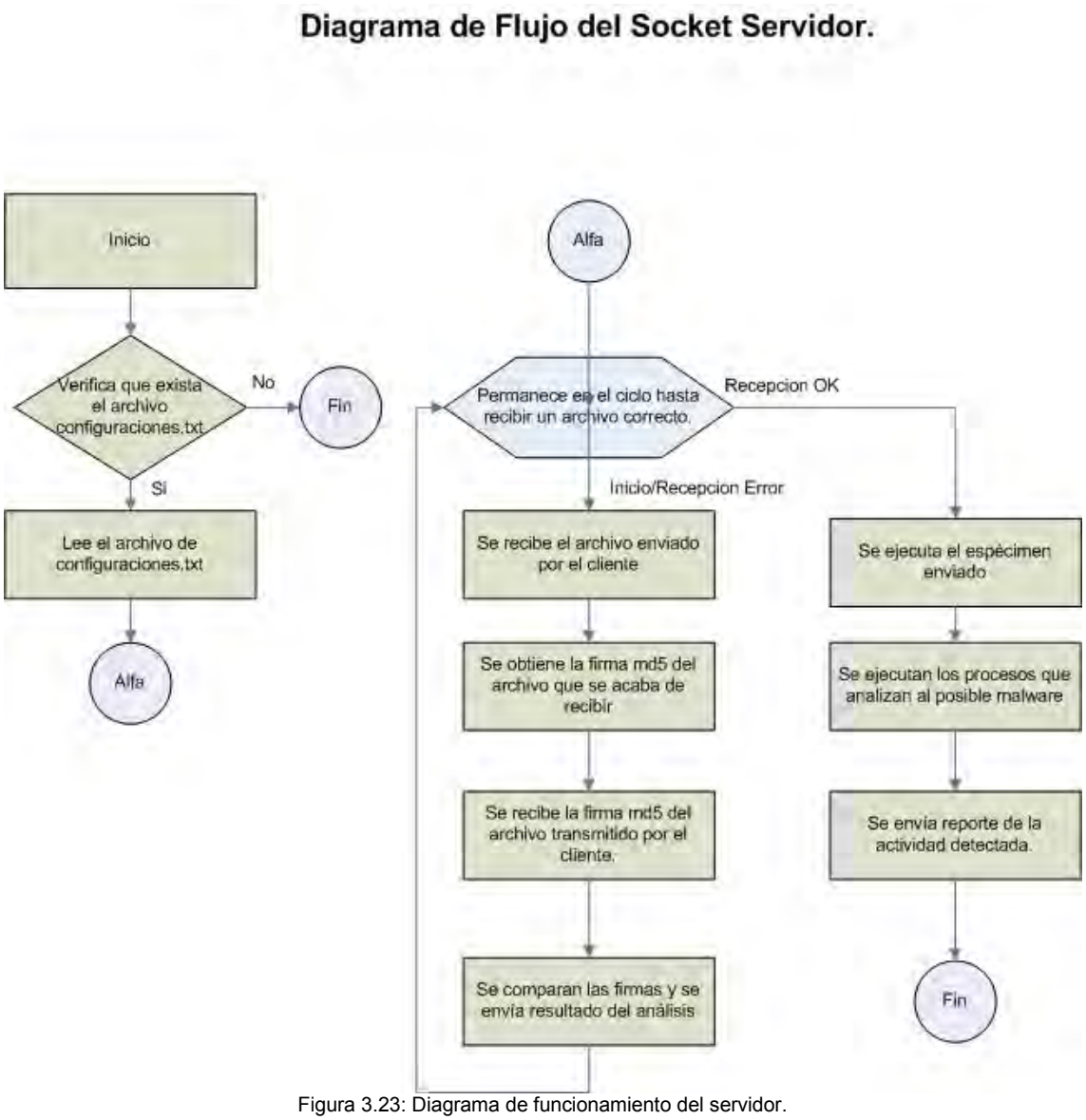

Figura 3.23: Diagrama de funcionamiento del servidor.

Script 3.14: Script de comunicación servidor.

```
#!/usr/bin/perl
# servidor
#use strict;
use IO::Socket;
my @lista;
$n=0;
#Obtenemos los parametros que el usuario del laboratorio puede manipular como son
#Direccion IP, el puerto TCP por donde se inicia la comunicacion
#Ademas del tiempo de analisis del archivo que se llama configuracion

                open(ML,"configuracion.txt");
                @conf=<ML>;
                foreach(@conf){
                        chomp($_);
                        if($_!~/\#/){
                                chomp($_);
                                @lista=split(/:/,$_);
                        }
                }
                print "#########################################################\n";
                print "#     Universidad Nacional Autonoma de Mexico     #\n";
                print "#             Facultad de Ingenieria              #\n";
                print "#         Analisis de Malware Automatico          #\n";
                print "#                  Servidor 1.0                   #\n";
                print "#########################################################\n";
                print "\nLa ip de su servidor es :$lista[0] \n";
                print "El puerto habilitado es :$lista[1] \n";
                print "El tiempo de analisis es:$lista[2] \n";


while($n!=5){ #Permanece en el ciclo hasta que obtengas una transmision correcta
#este script recibe la transmision, y crea el archivo que se llama savanback
################################################################################
        #`perl pservidor.pl`;

$sock = new IO::Socket::INET (LocalHost => $lista[0],
                        LocalPort => $lista[1],
                Proto    => 'tcp',
                Listen   => 10,
                                Type     => SOCK_STREAM
                );
die "El socket no se pudo crear. Reason: $!" unless $sock;
open(SAVAN,">savanback.exe");
binmode(SAVAN);
while ($new_sock = $sock->accept()) {
   while (defined ($buf = <$new_sock>)) {
     print SAVAN $buf || die("$!");
     #print  $buf || die();

   }
close ($sock);
}
#close ($sock);
close(SAVAN);

################################################################################
#En el siguiente bloque se verifica que la trasmision este correcta

        print "\nComparando las firmas md5\n";
        sleep(2);
        my $firma=`md5sum savanback.exe`;
        my @fir2=split('\ ',$firma);
        my $recibe;
        print "FIRMA malware \n$fir2[0]\n";

#Este socket se abre para recibir la firma md5 que proporciona el cliente
```

```
$sock = new IO::Socket::INET (LocalHost => $lista[0], #ip de la maquina servidor
                              LocalPort => $lista[1],
                        Proto    => 'tcp',
                        Listen   => 10,
                                        Type     => SOCK_STREAM
                        );
while ($new_sock = $sock->accept()) {
   while (defined ($buf = <$new_sock>)) {
                print $buf || die("$!");
                $recibe=$buf;
   }
close ($sock);
}

print "\nDe nuevo md5:$recibe";
print "\nEnviando respuesta.";

$servidor_w = IO::Socket::INET->new(LocalAddr => $lista[0],
                                                LocalPort => $lista[1],
                           Type     => SOCK_STREAM,
                           Reuse    => 1,
                                                Proto    => 'tcp',
                           Listen   => 10 )
                or die "No se pudo crear socket de escritura\n";

            while ($escritura = $servidor_w->accept()) {
                        if($fir2[0]==$recibe){
                                    print $escritura "\nTransmision correcta OK!\n";
                                    $n=5;
                        }
                        else{
                                    print $escritura "\nError en la trasmision!! reenvialo \n ";
                        }
                        close($servidor_w);
            }
}

close(ML);
$hijo = fork();

#En el siguiente bloque se ejecuta el especimen a analizar
if($hijo != 0){
                        print "\nEJECUTANDO EL MALWARE RECIBIDO!!!!\n";
                        #system("savanback.exe");

            }

            #proceso de escritura
            else{
                        sleep($lista[2]); #tiempo configurable por el administrador
                        print "\nEjecutando el analisis";
                        #invocar el script de analisis
                        #`perl ejec_finales.pl`;
                        #matar procesos
                        # sleep(10);
                        #`perl Procesos.pl`;
                         print "\nREINICIO DEL SISTEMA!!!!!!";
                        #`psshutdown -r -t 60`;
            }
```

El script 3.14 Verifica que exista el archivo configuraciones.txt, en caso de no existir el socket no se crea y termina este, cuando existe lo analiza, obtiene la ip y el puerto por el cual debe atender las peticiones de los clientes, permaneciendo a la escucha. Recibe el archivo enviado por el cliente obtiene la firma md5 del archivo recibido, espera a que el cliente le envíe la firma del archivo enviado y compara ambas firmas, en caso de ser iguales enviara una bandera que cierra la comunicación con el

cliente y comienza con la ejecución del malware para después iniciar con los procesos de monitoreo, como punto final enviará un reporte de la actividad realizada y detectada por el malware.

3.7 Implementación de firewall.

Un firewall (o firewall en inglés), es un elemento de hardware o software utilizado en una red de computadoras para prevenir algunos tipos de comunicaciones prohibidas por las políticas de red, las cuales se fundamentan en las necesidades del usuario.

La característica principal de un firewall es crear un punto único de control de acceso para la entrada y salida de tráfico de una red. Esta herramienta adecuadamente configurada resulta bastante efectiva para añadir protección a una instalación informática, bajo ninguna circunstancia debe de subestimarse la seguridad de la red con está herramienta, ya que podría ser vulnerable ante los atacantes. La seguridad en profundidad, debe de estar dividida en múltiples capas, esta metodología es óptima para lograr una red más segura ante ataques tanto internos como externos

La configuración correcta de firewalls se basa en grandes conocimientos de los protocolos de red y de la seguridad de la computadora. Errores pequeños pueden dejar a esta herramienta de seguridad carente de sus principales funciones.

3.7.1 Ventajas de un firewall.

- **Protege de intrusiones**.- Solamente entran a la red las personas autorizadas basadas en la política de la red en base a las configuraciones.
- **Optimización de acceso**.- Identifica los elementos de la red internos y optimiza que la comunicación entre ellos sea más directa. Esto ayuda a reconfigurar los parámetros de seguridad.
- **Protección de información privada**.- Permite el acceso solamente a quien tenga privilegios a la información de cierta área o sector de la red.
- **Protección contra virus**.- Evita que la red se vea infestada por nuevos virus que sean liberados.

Hay dos maneras de implementar las políticas en un firewall:

- Política por defecto ACEPTAR: en principio todo lo que entra y sale por el firewall se acepta y solo se rechaza lo que se señala explícitamente.
- Política por defecto DENEGAR: todo esta denegado, y solo se permite pasar por el firewall aquellos que están autorizados explícitamente.

3.7.2 Estudio de reglas de firewall (iptables).

IPtables es un sistema de firewall vinculado al kernel de Linux que se ha extendido enormemente a partir del kernel 2.4 de este sistema operativo. Al igual que su antecesor ipchains, un firewall de iptables no es como un servidor que se inicia, detiene o bien que se pueda caer por un error de programación.

Iptables está integrado con el kernel, es parte del sistema operativo. ¿Cómo se pone en marcha? Realmente lo que se hace es aplicar reglas. Para ello se ejecuta el comando iptables, el cual permite añadir, borrar o crear reglas. De esta manera un firewall de iptables no es un simple script de shell en el que se van ejecutando las reglas de firewall.

3.7.2.1 Creación de reglas de firewall.

Véase apéndice A.

Capítulo 4

Pruebas de Laboratorio.

En este capítulo se muestran las pruebas realizadas a todos y cada uno de los puntos que se han especificado en el capítulo 3. Estas pruebas servirán para desarrollar y mejorar el laboratorio ya que esta basado en el paradigma de desarrollo de software de espiral, por lo que es necesario crear prototipos, probarlos y generar correcciones para concluir el proyecto con la menor cantidad de posibles errores.

4 Pruebas de laboratorio.

La finalidad de estas pruebas es llegar a establecer un modelo de desarrollo de software basado en el espiral, ya que para este proyecto es necesario generar 4 vueltas, dentro de las cuales se especifican cada uno de los puntos realizados para cada una de estas, sin perder los objetivos planteados anteriormente.

4.1 Primera vuelta de pruebas.

4.1.1 Pruebas de sistema operativo para laboratorio y equipo de servicios (Firewall y Servidor Web).

Para contar con un respaldo del sistema operativo Windows, es necesario realizarlo con la herramienta dd (duplicate disk) por su eficiencia en la transferencia de datos desde un dispositivo o un archivo hacia un dispositivo, archivo o cualquier file system.

Procedimiento, para el respaldo con la herramienta dd:

- Iniciar el proceso usando el sistema Linux
- Ejecutar el comando dd con la siguiente sintaxis:
  - # dd if=/dev/hda1 of =<nombre Archivo>
- Al archivo de salida generar la firma digital usando el comando md5sum.
- md5sum archivo de salida, este valor se almacena para compararlo después con la firma digital obtenida del sistema Windows.
- Repetir este procedimiento desde el punto donde se utiliza dd, cambiando el nombre del archivo de salida.
- Verificar las dos firmas digitales y en caso de ser idénticas realizar el siguiente punto, en caso contrario se debe repetir este procedimiento desde el punto numero dos.
- Reiniciar el sistema operativo.
- Iniciar la computadora utilizando el sistema Windows que deseamos probar.

Procedimiento para probar el funcionamiento de lilo:

En la actualidad muchas distribuciones de Linux no utilizan lilo, como bootloader por default, pero es factible instalarlo en la mayoría de las distribuciones.

4.1.1.1 Procedimiento para instalación de lilo.

- Obtener el paquete de lilo, de la url antes indicada.
- Resolver todas las dependencias de paquetes que se pudieran presentar, el número de dependencias que estará dado por que tan mínimo haya sido la instalación del Linux.

- Configurar este gestor de arranque en el archivo /etc/lilo.conf
- Ejecutar el comando /sbin/lilo.
- Verificar si esta instalado. Debe de aparecer una pantalla roja al reiniciar el sistema, que nos permitirá escoger los diferentes sistemas operativos que tengamos instalados en dicho equipo.
- Se escoge cualquier sistema operativo y si entramos, procedemos con otro sistema operativo hasta que todos los sistemas operativos que tenga, hayan sido probados, se puede usar cualquier sistema operativo entonces el lilo esta configurado y funcionando correctamente en otro caso se debe de reconfigurar el archivo de configuración de lilo hasta que se pueda ingresar a todos los sistema instalados.

4.1.2 Pruebas de sistema operativo (Windows) para instalación de laboratorio.

Sistema Operativo	Resultado
Windows XP	OK
Windows XP SP1	OK
Windows XP SP2	OK
Windows 2000	OK
Windows 2000 SP4	OK
Windows 2003	La herramienta dd no pudo funcionar correctamente.

Tabla 4.1: Pruebas de funcionalidad de el binario dd.

4.1.3 Pruebas de sistema operativo (Linux) para instalación del Sistema de respaldo/restauración del equipo Windows.

Distribución Linux	***Resultado***
Slackware(Linux)	Ok
Fedora 7	Falta
OpenSuse	Ok
Mandrake 10	Ok
RedHat 7.3	No se pudieron resolver las dependencias debido a que es una versión antigua.

Tabla 4.2: Pruebas de funcionalidad con el gestor de arranque lilo.

4.1.4 Pruebas de funcionamiento de lilo entre sistemas operativos.

Sistema Operativo	***Windows XP***	***Windows XP SP1***	***Windows XP SP2***	***Windows 2000***	***Windows 2000 SP4***	***Windows 2003***
Slackware (Linux)	OK	OK	OK	OK	OK	OK
Fedora 7	No funciona	No funciona	No funciona	No funciona	No funciona	No funciona
OpenSuse	OK	OK	OK	OK	OK	No funciona
Mandrake 10	OK	OK	OK	OK	OK	No funciona
RedHat 7.3	No funciona	No funciona	No funciona	No funciona	No funciona	No funciona

Tabla 4.3: Pruebas de funcionalidad en conjunto.

Para cumplir con los objetivos y después del análisis minucioso que se realizan para cada uno de los sistemas operativos, se elabora el laboratorio de malware en un sistema operativo **Windows 2000 profesional** en conjunto con **Linux Mandrake 10** que permita crear los respaldos del sistema operativo del laboratorio ya que este sistema es el optimo en el manejo del MBR (Master Boot Record).

Una vez instalados los Sistemas Operativos, se ejecutan las pruebas de respaldo del sistema que contiene el laboratorio, para ello es necesario utilizar el comando "dd" de la siguiente manera:

 # dd if=/dev/hda1 of=SOWin_<fecha>

La manera de comprobar la integridad de los datos es generando firmas md5 a los dos archivos, de la siguiente forma.

 # md5sum /dev/hda1

 # md5sum SOWin_<fecha>

Estas dos firmas deben de ser iguales, en caso contrario, se debe realizar otra vez el respaldo. En la primera prueba que se realizó, se obtuvo un resultado satisfactorio.

 # md5sum /dev/hda1
1897c75e2641e8618b3e94426a5c2dd1

 # md5sum SOWin_01092008
1897c75e2641e8618b3e94426a5c2dd1

4.1.5 Primera programación de scripts.

Siguiendo con los objetivos de este laboratorio se probaron algunas herramientas para generar las primeras pruebas de los scripts, de los cuales se tienen los siguientes resultados.

4.1.5.1 Análisis de archivos.

Para analizar archivos se utilizo el script 4.1

Script 4.1: Tiempo de acceso a los archivos.

```
#busca los tiempos de acceso de los archivos
#del sistema y los almacena en el archivo time_de.txt
use Win32::OLE('in');
use constant wbemFlagReturnImmediately => 0x10;
use constant wbemFlagForwardOnly => 0x20;

$computer = ".";
$objWMIService = Win32::OLE->GetObject
    ("winmgmts:\\\\$computer\\root\\CIMV2") or die "WMI connection failed.\n";
$colItems = $objWMIService->ExecQuery
    ("SELECT * FROM CIM_DataFile","WQL",wbemFlagReturnImmediately |
wbemFlagForwardOnly);

$time_de="time_de.txt";
open(TIME_A,"+>$time_de") or die "No se puede abrir:$0 \n";


foreach my $objItem (in $colItems)
{
       $fname=$objItem->{FileName};
       $creationd=$objItem->{CreationDate};
       $lastmod=$objItem->{LastModified};
```

```
        print TIME_A "$fname \t";
         print TIME_A "$creationd \t";
        print TIME_A "$lastmod \n";
}

close(TIME_A);

open (LEC,"$time_de");
@lectura=<LEC>;

 open (WRT,"+>time_de2.txt");
 foreach$time(@lectura){
        $time =~ s/(\(|\)|\+|\.)//g;
        $time =~ s/(\*)+//g;
        if($time =~/[a-zA-Z0-9]/){
                printf WRT "$time";
        }
}
close (WRT);
```

Este script solo determinaba el tiempo de acceso a los archivos, por lo que no muestra una salida clara de lo que se tenía como fin, que era desplegar todos y cada uno de los archivos del sistema, además que el tiempo de ejecución de este es aproximadamente de 35 minutos. Para lo cual es necesario tomar una medida más precisa que permita realizar esta tarea más rápidamente

4.1.5.2 Análisis de hosts.

En esta primera prueba aún no se consideraba analizar este punto, pero según las circunstancias y revisando algunos ataques recientes de spyware, troyanos o algún otro malware, se determino que se tenia que incluir en el análisis.

4.1.5.3 Análisis de información de red.

Para este script no hubo ninguna modificación por lo cual se siguen utilizando las herramientas **iplist** y **promiscdetect**.

4.1.5.4 Análisis de portapapeles.

En la parte del portapapeles el script sigue siendo el mismo, se utiliza la librería de perl Win32::Clipboard().

4.1.5.5 Análisis de procesos.

La primera versión que se tenia de este script era con la herramienta **tlist**, véase en el script 4.2.

```
use strict;

#Programa que obtiene los procesos en el sistema antes de la ejecución del malware

 open(FH1,">procesos_inicio.txt");
 print FH1 `tlist`;
 close (FH1);
```

Script 4.2: script procesos iniciales

El siguiente script obtuvo los procesos con más detalle y no solo desplegaba un listado, para ello se utilizó la herramienta pslist y se programó como se muestra en el script 4.3.

```
#usando la herramieta pslist


use strict;
my $arch_pro_an="arch_pro_an.txt";

open(AR,">$arch_pro_an") or die "No se puede abrir: $!\n"; #Abre el archivo

my @linea=`pslist.exe`;    #@linea, contiene las lineas de la ejecucion del comando
my @linea2;
my $nom_proceso;

#dejamos en limpio el archivo
print AR "";

   print "-------------------------------------\n";
    print "\n\nInformación del Sistema:\n";
        foreach(@linea){
               @linea2 = split(/\ /, $_);
               $nom_proceso=$linea2[0];
                      if($nom_proceso !~
/PsList|Copyright|Sysinternals|^\s$|Process|Name/){
                                print AR $nom_proceso."\n";
                      }
        }

if(close(AR)==1){
       print "Archivo creado correctamente";
       }
```

Script 4.3: Script de listado detallado de procesos.

La propuesta es buena, sin embargo se tiene que hacer dos script para un mismo objetivo, lo que le resta eficiencia al proyecto, éste análisis genera un retraso en el tiempo, de esta manera es necesario encontrar un proceso más directo de optimizar este problema.

4.1.5.6 Análisis de servicios.

Los Servicios se obtienen con una herramienta llamada “listservices.exe” de esta manera no genero ningún cambio, ya que se obtenían los resultados planteados para este punto.

4.1.5.7 Análisis de tareas programadas.

Se realizan las pruebas con la herramienta “schtasks2000”, en este punto también se llego a los objetivos planteados, por lo tanto no se realizó ningún cambio sobre el script.

4.1.5.8 Análisis de usuarios y objetos compartidos de red.

En este modulo tampoco se hicieron cambios en el script ya que la herramienta “share” desplegaba todo los resultados que se tenían planteados como objetivo.

4.1.5.9 Análisis de puertos.

No se tenía programado este punto por lo que fue necesario implementar una herramienta que pudiera facilitar esta búsqueda y generar una salida confiable.

4.1.5.10 Análisis de llaves de registro.

Este punto fue muy delicado, ya que el primer script que se realizó solo mostraba algunas de las llaves de registro y la mayoría de ellas no se encontraban en el sistema operativo, ya que era un script general para todas las familias de los Windows (Windows 2000, Windows XP, Windows 2003), por lo tanto mostraban demasiados errores, la búsqueda se realizó tal y como se muestra en el script 4.4.

Esta tarea utiliza la herramienta **reg query,** es efectiva, sin embargo ocupa muchos arreglos que pueden efectuar errores al momento de programar el script.

Script 4.4: Llaves de registro.

```
#Programa que revisa las lleves de Registro en WIN2000
use strict;
my @llave_HKLM;
my @llave_HKCU;
my $llave;
my $i;
my @llaves;
my
@llaves=("Run","RunServices","RunOnce","RunOnceEx","RunServicesOnce","Polices\\Explorer\
\Run","RunOnce\\Setup");
my $HKLM="HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\";
my $HKCU="HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\";

open(FH,">llaves.txt");
for ($i=0;$i<7;$i++){

       $llave_HKLM[$i]=$HKLM.$llaves[$i];
       print FH "-----$llave_HKLM[$i]-----"."\n";
       print FH `reg query $llave_HKLM[$i]`."\n";
       $llave_HKCU[$i]=$HKCU.$llaves[$i];
       print FH "-----$llave_HKCU[$i]-----"."\n";
       print FH `reg query $llave_HKCU[$i]`."\n";
}

#############Otras LLaves#############
print FH "\n-----
HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Winlogon\\Userinit-----\n";
print FH `reg query
"HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Winlogon\\Userinit"`."\n";

print FH "\n-----HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Windows-----\n";
print FH `reg query
"HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Windows"`."\n";

print FH "\n-----HKLM\\System\\CurrentControlSet\\Control\\Session Manager\\KnownDLLs---
--\n";
print FH `reg query "HKLM\\System\\CurrentControlSet\\Control\\Session
Manager\\KnownDLLs"`."\n";

print FH "\n-----HKLM\\System\\ControlSet001\\Control\\Session Manager\\KnownDLLs-----
\n";
print FH `reg query "HKLM\\System\\ControlSet001\\Control\\Session
Manager\\KnownDLLs"`."\n";

print FH "\n-----HKCU\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Windows\\load----
-\n";
print FH `reg query
"HKCU\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Windows\\load"`."\n";

print FH "\n-----HKCU\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Windows-----\n";
print FH `reg query
"HKCU\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Windows"`."\n";
close (FH);
```

```
open(FH2,"<llaves.txt");
open(FH3,">llaves_inicio.txt");
@llaves=<FH2>;

foreach $llave (@llaves){
        if($llave=~/^$/){
                next;
        }
        $llave=~s/\\/\//g;
        $llave=~s/(\[|\])/ /g;
        $llave=~s/\./,/g;
        print FH3 $llave;

}
close(FH2);
close(FH3);
```

Otra particularidad de este script es que muestra muchos caracteres especiales en su salida, de esta manera al hacer un análisis con Perl causa errores de sintaxis.

4.1.5.11 Algoritmo de comparación de archivos de salida.

Este script es el más importante en el laboratorio ya que es el motor que realiza la comparación entre el archivo que se tiene como configuración base antes y después de la ejecución del malware, el primer prototipo que se tenía para este punto solo realizaba una comparación de una vuelta, es decir, únicamente comparaba el archivo final contra el archivo inicial, como se muestra en el script 4.5.

```
########COMPARA FINA2 CON FINAL, LO QUE CAMBIO O NO EXISTE
########EN FINAL CON FINAL2 LO IMPRIME
open(FH,"servicios_de.txt")||die"no puede abrir $!";
open(FHI,"servicios_an.txt")||die"no puede abrir $!";

open (MAN,"+>servicios_final.txt");

my @cadenas=<FH>;
my @cadenas1=<FHI>;

my $n=0;

print MAN "=================================== ========================
===============\n";
print MAN "                      Diferencias entre servicios\n";
print MAN "=================================== ========================
===============\n";
foreach $array1(@cadenas){
        foreach $array2(@cadenas1){
                if($array1=~/$array2/){
                        $n=0;
                         next;
                }
                $n+=1;
                if($n==($#cadenas1+1)){
                        print MAN "$array1 \n";
                        last;
                }
          }
           $n=0;
        }

        print MAN "\n";
```

Script 4.5: Comparación de archivos final contra inicial.

4.1.6 Resultados de la primera vuelta de pruebas.

En esta primera etapa se busca obtener como resultado al infectar el laboratorio con el keyloger (xey0pkom88c.exe) lo que se observa en la Tabla Muestra1, este espécimen es utilizado ya que es una muestra controlada.

```
xey0pkom88c.exe

información conocida del malware

Nombre del Código Malicioso: xey0pkom88c.exe
Aliases 1: xey0pkom88c.exe Archivo
binario: xey0pkom88c.exe
Firma MD5: 62c860914a771056a475b7182e4617e1
Firma SHA1: e98694b373ec0fa2b0ac0fba13eef1c30ca94dcc
Tamaño: 41191
Tipo de Archivo: application/x-msdownload
Clasificación: Troyano Llaves de registro:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run SWNETSUP

Información general:
Proceso malicioso: xey0pkom88c.exe
Path: c:\windows\xey0pkom88c.exe

Llaves editadas en el registro de Windows:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run SWNETSUP c:\windows\xey0pkom88c.exe

Archivos maliciosos creados por el malware:
Created: c:\windows\xey0pkom88c.exe
Created: C:\WINDOWS\setup.txt

Características generales:

1. Genera direcciones IP Aleatoria para escanear su puerto 3127/TCP
2. Asegura su activacion editando la llave del registro "RUN" de Windows
3. Genera un registro de todo lo que se teclea en el sistema comprometido. Esta bitacora
la aloja en el archivo setup.txt
4. Los Archivos creados por el malware son ocultos.
```

Tabla de Muestra1: Keyloger xey0pkom88c.exe

El análisis tardo aproximadamente 40 minutos una vez ejecutado el script final, de lo cual se obtuvo el siguiente resultado.

Tabla de Resultado1: Resultado de keyloger xey0pkom88c.exe

```
Archivos:

No hubo cambio en Archivos

Infamación de Red:

=================================== ========================
                          Diferencias la RED
=================================== ========================
Elementos nuevos o modificados
=================================== ========================
2046863552   192.168.0.122   255.255.255.255      255.255.255.0

Portapapeles y tiempo de acceso:

=================================== ========================
                          Diferencias entre Protapapeles y tiempo de acceso
=================================== ========================
Elementos nuevos o modificados
=================================== ========================
System Uptime 0 hours 43 minutes 41 seconds


Procesos:

############################################################
                                    Procesos Nuevos
############################################################
```

```
Name                 Pid     Pri    Thd     Hnd      Priv          CPU Time     Elapsed Time
xey0pkom88c          696   110564    16648    3220     3404     7852      56    47


Puertos abiertos:
=================================== ========================
                          Diferencias En Los Puertos
=================================== ========================
192.168.0.237 2828 TCP CONNECTING
192.168.0.237 2818 TCP CONNECTING
192.168.0.237 2922 TCP CONNECTING
192.168.0.237 2791 TCP CONNECTING
192.168.0.237 2821 TCP CONNECTING
0.0.0.0 2806 TCP LISTENING
0.0.0.0 2810 TCP LISTENING
0.0.0.0 2818 TCP LISTENING
0.0.0.0 2814 TCP LISTENING
0.0.0.0 2822 TCP LISTENING
0.0.0.0 2826 TCP LISTENING
0.0.0.0 2919 TCP LISTENING
0.0.0.0 2791 TCP LISTENING
0.0.0.0 2830 TCP LISTENING
0.0.0.0 2923 TCP LISTENING
0.0.0.0 2795 TCP LISTENING
0.0.0.0 30167 UDP LISTENING

Servicios:

No hubo cambios en los servicios

Tareas programadas:

No hubo cambios en las Tareas Programadas.

Usuarios y redes compatidas:

No hubo cambios en usuarios y redes compartidas.
```

En conclusión, los scripts realizados son muy tardados y poco eficientes, ya que no muestran claramente algunos puntos, como las llaves de registro, los procesos, los archivos. Siendo estos tres factores puntos claves, por lo que no se puede confiar en la veracidad de este resultado.

Es necesario tener un módulo que ejecute los otros scripts de forma automática y que arroje los resultados a otro servidor para que éste los muestre de manera más clara, también se tienen que se optimizar los scripts y así realizar el análisis más eficiente, ya que no es viable que tarde más de 40 minutos en la ejecución final.

4.2 Segunda Vuelta de pruebas.

4.2.1 Corrección de errores de programación del laboratorio de malware.

En esta parte del proyecto se realizan las correcciones de los errores encontrados en la primera prueba del laboratorio.

4.2.1.1 Análisis de archivos.

Después de buscar una herramienta precisa se encontró la llamada “Integrit.exe”, que se especificó en el capítulo 3. Se utiliza ejecutando el siguiente comando.

```
`integrit.exe -C integrit.conf -c > archivos_de.txt`;
```

Para hacer la comparación, solo se despliega la parte de cambios y nuevos archivos, corriendo el comando integrit de la siguiente manera..

```
`grep -E "changed:|new:" archivos_de.txt >
C:/scripts/salidas/archivos_final.txt`;
```

Obteniendo una eficiencia en este script, ya que detecta todos y cada uno de los archivos que se tienen en el sistema operativo y hace la comparación en menos tiempo a diferencia del script anterior, este proceso se realiza aproximadamente en 8 minutos, reduce en 35 minutos el tiempo de ejecución.

4.2.1.2 Análisis de hosts.

Se creó un programa para imprimir el hosts, como se muestra en el script 4.6.

```
use strict;

my @str_hosts=`cat C:\\WINNT\\system32\\drivers\\etc\\hosts`;
my $arch_hosts="hosts_orig.txt";
open(MAN,"+>$arch_hosts");

        my $lin_hosts;
        foreach $lin_hosts(@str_hosts){
                $lin_hosts =~ s/\(//g;
                $lin_hosts =~ s/\)//g;
                if($lin_hosts =~/[a-zA-Z0-9]/){
                printf MAN $lin_hosts;
                }
}
```

Script 4.6: Análisis de Hosts.

Este script, únicamente realiza una replica del archivo hosts, quitando algunos caracteres especiales, de este modo cumple con el objetivo de realizar la comparación de archivos.

4.2.1.3 Análisis de procesos.

Se modifican los scripts de procesos al realizar una comparación más rápida y eficiente, se programo un arreglo en el que se obtiene como resultado el nombre del proceso y el PID (identificador), como se muestra en el script 4.7.

```
use strict;
my $arch_pro_an="arch_pro_an.txt";
my @cmd_pslist=`pslist.exe|tr -s " "`;
my @proceso;

open(AR,">$arch_pro_an") or die "No se puede abrir: $!\n"; #Abre el archivo

foreach(@cmd_pslist){
        @proceso = split(/\ /, $_);
        if($proceso[0] !~ /PsList|Copyright|Sysinternals|^\s$|Process|Name/){
                print AR $proceso[0].":".$proceso[1]."\n";
        }
}
```

Script 4.7: Análisis de Procesos.

La comparación de procesos, se realiza de igual forma con el algoritmo de comparación que se utiliza para los scripts anteriores, solo que el extra en este script, es que se detalla el proceso diferente, además genera la imagen con la herramienta

pmdump y lo elimina del sistema operativo con la herramienta **pskill**, como se muestra en el script 4.8.

Script 4.8: Comparación de procesos.

```
my $n=0;
@procesos_originales=`cat arch_pro_an.txt`;
@procesos_finales=`cat arch_pro_de.txt`;
open(FH3,"> procesos.txt");
print FH3
"################################################################################".
"\n";
print FH3 "                                    Procesos Nuevos
       "."\n";
print FH3
"################################################################################".
"\n";
print FH3 "Name                 Pid    Pri    Thd    Hnd     Priv          CPU Time
Elapsed Time"."\n";

foreach $pf(@procesos_finales){
       $n=0;
       foreach $po(@procesos_originales){
              if(($pf =~
/$po/)||($pf=~/perl/)||($pf=~/pslist/)||($pf=~/tr/)||($pf=~/cmd/)){
              $n=$n+1;
              }
       }
       if ($n==0){
              @nuevos = split(/:/, $pf);
              `pmdump $nuevos[1] imagen_procesos.$nuevos[1]`;
              @detalles=`pslist -m $nuevos[1]`;
              print FH3 $detalles[3];
              `pskill -t $nuevos[1]`;
       }
}
print FH3 "\n Busque las imagenes de los procesos \n";
close FH3;

close(FH);
close(FHI);
close(MAN);
```

4.2.1.4 Análisis de llaves de registro.

Esta modificación utiliza un nuevo módulo de perl, llamado Win32::Registry, este modulo extrae el valor de las llaves de registro desde el path que se le asigne, esta herramienta es muy rápida y solo se utilizan arreglos de datos, que comparado con el script anterior, es más sencillo de programar y únicamente quitando los caracteres especiales, para así realizar la comparación sin que se tenga error alguno.

Script 4.9: Análisis de llaves de registro.

```
       use Win32::Registry;
my %RegType = (
                      0 => 'REG_0',
                      1 => 'REG_SZ',
                      2 => 'REG_EXPAND_SZ',
                      3 => 'REG_BINARY',
                      4 => 'REG_DWORD',
                      5 => 'REG_DWORD_BIG_ENDIAN',
                      6 => 'REG_LINK',
                      7 => 'REG_MULTI_SZ',
                      8 => 'REG_RESOURCE_LIST',
                      9 => 'REG_FULL_RESOURCE_DESCRIPTION',
                      10 => 'REG_RESSOURCE_REQUIREMENT_MAP'
                      );

my $PATH="Software\\Microsoft\\Windows\\CurrentVersion\\";
my @llaves=("Run","RunOnce","RunOnceEx","policies","Setup");

my $Register;
my $RegType;
my $RegValue;
```

```
my $RegKey;
my $value;
my %values;
open(FH,">llaves.txt");
foreach (@llaves) {

$Register=$PATH.$_;
$HKEY_LOCAL_MACHINE->Open($Register,$hkey)|| die $!;

$hkey->GetValues(\%values);

foreach $value (keys(%values))
        {
        $RegType       = $values{$value}->[1];
        $RegValue      = $values{$value}->[2];
        $RegKey        = $values{$value}->[0];
        next if ($RegType eq '');  #do not print FHdefault value if not assigned
        $RegKey        = 'Default' if ($RegKey eq '');      #name the default key
        print FH "$Register.$RegKey";
        print FH " ($RegType{$RegType}) : ";

        SWITCH:
                {
                if ($RegType == 4)
                        {printf FH "Ox%1x \n", unpack("L",$RegValue); last SWITCH; }
                if ($RegType == 5)
                        {printf FH "Ox%1x", unpack("N",$RegValue); last SWITCH; }
                if ($RegType < 8 )
                        {printf FH "$RegValue\n"; last SWITCH; }
                print FH"\n";
                }
        }
$hkey->Close();
}
close(FH);

open(FH2,"<llaves.txt");
open(FH3,">llaves_an.txt");
@llaves=<FH2>;

foreach $llave1 (@llaves){

        if($llave1=~/^$/){
                next;
        }
        $llave1=~s/^[  ]*//g;
        $llave1=~s/\\/_/g;
        $llave1=~s/\//_/g;
        $llave1=~s/(\[|\])/ /g;
        $llave1=~s/(\(|\))/ /g;
        $llave1=~s/\"/ /g;
        $llave1=~s/:/ /g;
        $llave1=~s/\./ /g;
        $llave1=~s/!/ /g;

        print FH3 $llave1;

}
close(FH2);
close(FH3);
```

4.2.1.5 Script de comparación de archivos de salida.

En este script se realizo una modificación, ya que en el original solo daba por resultado la comparación del archivo final contra el archivo inicial, la modificación que se realizó es la comparación de ambos archivos, es decir, se revisa el archivo final contra el archivo inicial y viceversa, se decidió ejecutar este algoritmo ya que los virus afectan la información que ya se tenia en la ejecución inicial. Este proceso se muestra en el script 4.10.

Script 4.10: Programacion de comparación.

```
########COMPARA FINA2 CON FINAL, LO QUE CAMBIO O NO EXISTE
########EN FINAL CON FINAL2 LO IMPRIME
open(FH,"archivo_de.txt")||die"no puede abrir $!";
open(FHI,"archivo_an.txt")||die"no puede abrir $!";
open (MAN,"+>archivo_final.txt");

my $array1;
my $array2;
my $array3;
my $n;
my @cadenas=<FH>;
my @cadenas1=<FHI>;
my $firma1=`md5sum archivo_de.txt`;
my $firma2=`md5sum archivo_an.txt`;

if($firma1 == $firma2){

        print MAN "No Hubo Cambios en XXXXXXXXXXXX\n";
}

else{
        print MAN "=================================== ========================
===============\n";
        print MAN "                        Diferencias en XXXXXXXXXXXXXXXXX\n";
        print MAN "=================================== ========================
===============\n";

        print MAN  "Elementos nuevos o modificados\n";
        print MAN "=================================== ========================
===============\n";
        foreach $array1(@cadenas){
                        #si el archivo es diferente de vacio.
                        if($#cadenas1>0){
                                foreach $array2(@cadenas1){
                                        if($array1=~$array2){
                                                $n=0;
                                                next;
                                        }
                                        $n+=1;
                                        if($n==($#cadenas1+1)){
                                                print MAN "$array1";
                                                last;
                                        }
                                        }
                                        $n=0;
                        }
                }

                #si el archivo original esta vacio.
                if($#cadenas1<=0){
                                foreach $array3(@cadenas){
                                        print MAN $array3;
                                        }
                                }
        print MAN "\n\n";
        print MAN "Elementos borrados o modificados \n";
        print MAN "=================================== ========================
===============\n";

        foreach $array2(@cadenas1){
                        #si el archivo es diferente de vacio.
                        if($#cadenas>0){
                                foreach $array1(@cadenas){
                                        if($array1=~$array2){
                                                $n=0;
                                                next;
                                        }
                                        $n+=1;
                                        if($n==($#cadenas+1)){
                                                print MAN "$array2";
                                                last;
                                        }
                                        }
                                        $n=0;
                        }
                }
                        if($#cadenas<=0){
```

```
                                foreach $array3(@cadenas1){
                                        print MAN $array3;
                                        }
                                }

}
```

4.2.1.6 Script de Automatización.

Se diseñó un script para solucionar uno de los problemas que se mencionaban en las pruebas de la primera vuelta, para poder automatizar la ejecución de los scripts y enviar los resultados a la máquina que funcionará como servidor web, para ello véase el script 4.11.

Script 4.11: Script de automatización.

```
#Archivos Se ejecutó una vez la actualizacion de archivos
`integrit.exe -C integrit.conf -c > archivos_de.txt`;
`perl C:\\scripts\\portapapeles\\porta_de.pl`;
`perl C:\\scripts\\infored\\red_de.pl`;
`perl C:\\scripts\\hosts\\hosts_de.pl`;
`perl C:\\scripts\\llavereg\\llaves_de.pl`;
`perl C:\\scripts\\procesos\\procesos_de.pl`;
`perl C:\\scripts\\servicios\\servicios_de.pl`;
`perl C:\\scripts\\tareasprogra\\task_de.pl`;
`perl C:\\scripts\\usrcomp\\net_de.pl`;
`perl C:\\scripts\\puertos\\ptos_usa_de.pl`;

#--------------------------------------
#ahora se hace la comparacion

#compara tiempo de acceso a los archivos del sistema
#crea un archivo de resultados: times_acc.txt para los tiempos de acceso
`grep -E "changed:|new:" archivos_de.txt > C:/scripts/archivos_final.txt`;

#realiza un analisis del portapapeles al final porta_final.txt
`perl C:\\scripts\\portapapeles\\comp_porta.pl`;

#analiza una red  red_final.txt
`perl C:\\scripts\\infored\\comp_red.pl`;

##analiza cambios en el archivo de hosts del sistema
`perl C:\\scripts\\hosts\\comp_hosts.pl`;

##analiza cambios en las llaves del registro de windows
`perl C:\\scripts\\llavereg\\comp_llaves.pl`;

#compara procesos, genera un archivo con los resultados de la comparacion
#llamado: arch_pro_cre.txt
`perl C:\\scripts\\procesos\\comp_proc.pl`;

#servicios -- servicios_final.txt
`perl C:\\scripts\\servicios\\comp_servicios.pl`;

#compara tareas programadas
#crea un archivo de resultados llamado: task_new.txt, para las tareas programadas
`perl C:\\scripts\\tareasprogra\\comp_tasks.pl`;

#carpeta de usuario net_final.txt
`perl C:\\scripts\\usrcomp\\comp_net.pl`;

#puertos usando en ptos_final.txt ptos_final.txt
`perl C:\\scripts\\puertos\\comp_puertos.pl`;


# envio de informacion

`pscp -i nueva2.ppk -r salidas root@10.0.0.10:`;
`pscp -i nueva2.ppk -r imagen_procesos* root@10.0.0.10:`;
```

4.2.2 Resultados de la segunda vuelta.

Realizados los cambios en esta segunda vuelta, se ejecutó el mismo malware que se utilizó en la primera prueba, esperando los resultados de Tabla de Muestra1 (véase en la página 86), teniendo como salida lo que aparece en la Tabla Resultado2 de segunda vuelta.

Tabla Resultado2: Keyloger xey0pkom88c.exe de segunda vuelta.

Archivos:

```
changed: C:///WINNT/SoftwareDistribution/SelfUpdate/Default/wsus3setup.cab   m(2
0080317-125906:20081026-151726) c(20080317-125906:20081026-151726)
changed: C:///WINNT/SoftwareDistribution/WuRedir/9482F4B4-E343-43B6-B170-9A65BC8
22C77/wuredir.cab   m(20080210-151742:20081026-151724) c(20080210-151742:2008102
6-151724)
new:     C:///WINNT/A7ScL1zQmT.exe   p(755) t(100000) u(1000) g(513) z(41181) m(
20081026-152134)
new:     C:///WINNT/A7ScL1zQmT.exe   s(d8ca85d418b4d365e0368a5de51d76ad99c5caa5)

new:     C:///WINNT/setup.txt   p(644) t(100000) u(1000) g(513) z(182) m(2008102
6-152832)
new:     C:///WINNT/setup.txt   s(a64b61312e96b060cac610fbab91455e7dbc347e)
changed: C:///Documents and Settings/All Users/Documentos   l(3:4)
changed: C:///Recycled/INFO2   s(f23e75df7b2bcbc49e66202c793a4ea758f37f42:57c1ac
205347f4c48fa00a415b7366cd17494ea9)
changed: C:///Recycled/INFO2   m(20081015-151952:20081026-151612) c(20081015-151
952:20081026-151612)
new:     C:///Recycled/Dc1.txt   p(644) t(100000) u(1000) g(513) z(350) m(200810
26-151602)
new:     C:///Recycled/Dc1.txt   s(eb7ddc5357c4ba730b698e36d2cb4e0bcfa54b28)
integrit: not doing update, so no check for missing files

Hosts:

No hubo cambios en el archivo /etc/hosts

Infomacion de Red:

================================== ========================
                          Diferencias la RED
================================== ========================
Elementos nuevos o modificados
================================== ========================
2046863552   192.168.0.122   255.255.255.255      255.255.255.0


Elementos borrados o modificados
================================== ========================
-318723904   192.168.0.237   255.255.255.255      255.255.255.0

Llaves de Registro:
================================== ========================
                          Diferencias Llaves de Registro
================================== ========================
Elementos nuevos o modificados
================================== ========================
Software_Microsoft_Windows_CurrentVersion_Run PCSCAN  REG_SZ    C _WINNT_A7ScL1zQmT exe
Software_Microsoft_Windows_CurrentVersion_RunOnce PCSCAN  REG_SZ    C _WINNT_A7ScL1zQmT
exe
Software_Microsoft_Windows_CurrentVersion_RunOnceEx PCSCAN  REG_SZ    C
_WINNT_A7ScL1zQmT exe
Software_Microsoft_Windows_CurrentVersion_policies PCSCAN  REG_SZ    C _WINNT_A7ScL1zQmT
exe
Software_Microsoft_Windows_CurrentVersion_Setup PCSCAN  REG_SZ    C _WINNT_A7ScL1zQmT
exe

Portapapeles y tiempo de acceso:

================================== ========================
                          Diferencias entre Protapapeles y tiempo de acceso
================================== ========================
Elementos nuevos o modificados
================================== ========================
System Uptime 0 hours 30 minutes 16 seconds
```

```
Elementos borrados o modificados
================================== ========================
System Uptime 0 hours 13 minutes 32 seconds

Procesos:

############################################################
                                  Procesos Nuevos
############################################################
Name                 Pid     Pri    Thd     Hnd      Priv          CPU Time     Elapsed Time
xey0pkom88c          696  110564   16648    3220     3404     7852     56   47


Puertos abiertos:
================================== ========================
                           Diferencias En Los Puertos
================================== ========================
Elementos nuevos o modificados
================================== ========================
192.168.0.237 2828 TCP CONNECTING
192.168.0.237 2818 TCP CONNECTING
192.168.0.237 2922 TCP CONNECTING
192.168.0.237 2791 TCP CONNECTING
192.168.0.237 2821 TCP CONNECTING
192.168.0.237 2809 TCP CONNECTING
192.168.0.237 2801 TCP CONNECTING
192.168.0.237 2916 TCP CONNECTING
192.168.0.237 2873 TCP CONNECTING
192.168.0.237 2928 TCP CONNECTING
192.168.0.237 2923 TCP CONNECTING
192.168.0.237 2790 TCP CONNECTING
192.168.0.237 2869 TCP CONNECTING
192.168.0.237 2826 TCP CONNECTING
192.168.0.237 2819 TCP CONNECTING
192.168.0.237 2805 TCP CONNECTING
192.168.0.237 2830 TCP CONNECTING
192.168.0.237 2831 TCP CONNECTING
192.168.0.237 2811 TCP CONNECTING
192.168.0.237 2919 TCP CONNECTING
192.168.0.237 2794 TCP CONNECTING
192.168.0.237 2798 TCP CONNECTING
192.168.0.237 2793 TCP CONNECTING
192.168.0.237 2812 TCP CONNECTING
192.168.0.237 2918 TCP CONNECTING
192.168.0.237 2917 TCP CONNECTING
192.168.0.237 2924 TCP CONNECTING
192.168.0.237 2820 TCP CONNECTING
192.168.0.237 2832 TCP CONNECTING
192.168.0.237 2920 TCP CONNECTING
192.168.0.237 2823 TCP CONNECTING
192.168.0.237 2829 TCP CONNECTING
192.168.0.237 2804 TCP CONNECTING
192.168.0.237 2925 TCP CONNECTING
192.168.0.237 2788 TCP CONNECTING
192.168.0.237 2813 TCP CONNECTING
192.168.0.237 2930 TCP CONNECTING
192.168.0.237 2816 TCP CONNECTING
192.168.0.237 2825 TCP CONNECTING
192.168.0.237 2921 TCP CONNECTING
192.168.0.237 2864 TCP CONNECTING
192.168.0.237 2929 TCP CONNECTING
192.168.0.237 2799 TCP CONNECTING
192.168.0.237 2824 TCP CONNECTING
192.168.0.237 2810 TCP CONNECTING
192.168.0.237 2814 TCP CONNECTING
192.168.0.237 2807 TCP CONNECTING
192.168.0.237 2934 TCP CONNECTING
192.168.0.237 2817 TCP CONNECTING
192.168.0.237 2806 TCP CONNECTING
192.168.0.237 2796 TCP CONNECTING
192.168.0.237 2802 TCP CONNECTING
192.168.0.237 2827 TCP CONNECTING
192.168.0.237 2789 TCP CONNECTING
192.168.0.237 2833 TCP CONNECTING
192.168.0.237 2803 TCP CONNECTING
```

```
192.168.0.237 2872 TCP CONNECTING
192.168.0.237 2792 TCP CONNECTING
192.168.0.237 2926 TCP CONNECTING
192.168.0.237 2932 TCP CONNECTING
192.168.0.237 2795 TCP CONNECTING
192.168.0.237 2800 TCP CONNECTING
192.168.0.237 2927 TCP CONNECTING
192.168.0.237 2822 TCP CONNECTING
192.168.0.237 2797 TCP CONNECTING
192.168.0.237 2808 TCP CONNECTING
192.168.0.237 2931 TCP CONNECTING
192.168.0.237 2933 TCP CONNECTING
0.0.0.0 2815 TCP LISTENING
0.0.0.0 1043 TCP LISTENING
0.0.0.0 1051 TCP LISTENING
0.0.0.0 1037 TCP LISTENING
0.0.0.0 1050 TCP LISTENING
0.0.0.0 2927 TCP LISTENING
0.0.0.0 2799 TCP LISTENING
0.0.0.0 2803 TCP LISTENING
0.0.0.0 2931 TCP LISTENING
0.0.0.0 2869 TCP LISTENING
0.0.0.0 2873 TCP LISTENING
0.0.0.0 2807 TCP LISTENING
0.0.0.0 2811 TCP LISTENING
0.0.0.0 2819 TCP LISTENING
0.0.0.0 2788 TCP LISTENING
0.0.0.0 2823 TCP LISTENING
0.0.0.0 2916 TCP LISTENING
0.0.0.0 2920 TCP LISTENING
0.0.0.0 2827 TCP LISTENING
0.0.0.0 2792 TCP LISTENING
0.0.0.0 2796 TCP LISTENING
0.0.0.0 2831 TCP LISTENING
0.0.0.0 2924 TCP LISTENING
0.0.0.0 2800 TCP LISTENING
0.0.0.0 2928 TCP LISTENING
0.0.0.0 2804 TCP LISTENING
0.0.0.0 2932 TCP LISTENING
0.0.0.0 2808 TCP LISTENING
0.0.0.0 2816 TCP LISTENING
0.0.0.0 2812 TCP LISTENING
0.0.0.0 2820 TCP LISTENING
0.0.0.0 2917 TCP LISTENING
0.0.0.0 2824 TCP LISTENING
0.0.0.0 2789 TCP LISTENING
0.0.0.0 2921 TCP LISTENING
0.0.0.0 2793 TCP LISTENING
0.0.0.0 2828 TCP LISTENING
0.0.0.0 2797 TCP LISTENING
0.0.0.0 2832 TCP LISTENING
0.0.0.0 2925 TCP LISTENING
0.0.0.0 2801 TCP LISTENING
0.0.0.0 2929 TCP LISTENING
0.0.0.0 2933 TCP LISTENING
0.0.0.0 2805 TCP LISTENING
0.0.0.0 2809 TCP LISTENING
0.0.0.0 2817 TCP LISTENING
0.0.0.0 2813 TCP LISTENING
0.0.0.0 2821 TCP LISTENING
0.0.0.0 2918 TCP LISTENING
0.0.0.0 2790 TCP LISTENING
0.0.0.0 2825 TCP LISTENING
0.0.0.0 2922 TCP LISTENING
0.0.0.0 2829 TCP LISTENING
0.0.0.0 2794 TCP LISTENING
0.0.0.0 2864 TCP LISTENING
0.0.0.0 2798 TCP LISTENING
0.0.0.0 2833 TCP LISTENING
0.0.0.0 2926 TCP LISTENING
0.0.0.0 2802 TCP LISTENING
0.0.0.0 2930 TCP LISTENING
0.0.0.0 2934 TCP LISTENING
0.0.0.0 2872 TCP LISTENING
0.0.0.0 2806 TCP LISTENING
0.0.0.0 2810 TCP LISTENING
0.0.0.0 2818 TCP LISTENING
```

```
0.0.0.0 2814 TCP LISTENING
0.0.0.0 2822 TCP LISTENING
0.0.0.0 2826 TCP LISTENING
0.0.0.0 2919 TCP LISTENING
0.0.0.0 2791 TCP LISTENING
0.0.0.0 2830 TCP LISTENING
0.0.0.0 2923 TCP LISTENING
0.0.0.0 2795 TCP LISTENING
0.0.0.0 30167 UDP LISTENING

Elementos borrados o modificados
================================== ========================
192.168.0.237 68 UDP LISTENING

Servicios:

No hubo cambios en los servicios

Tareas programadas:

No hubo cambios en las Tareas Programadas ...

Usuarios y redes compatidas:

No hubo cambios en usuarios y redes compartidas.
```

En este caso los resultados fueron mucho más cercanos a los que mostraba el resultado esperado, sin embargo genero archivos nuevos, otros modificados, redes diferentes, abrió algunos puertos más de los esperados y así como la generación de llaves de registro, por lo tanto se considera un resultado satisfactorio.

Concluido este punto es necesario crear una segunda prueba esta se realiza con un malware llamado **navscnr**, los resultados esperados se observan en la Tabla Muestra2: malware navscnr

```
navscnr

información conocida del malware

Nombre del Código Malicioso: navscnr.exe
Aliases 1: navscnr.exe
Archivo binario: navscnr.exe
Firma MD5: ad99bfacd166264ca50ccd40bf4670e5
Firma SHA1: 1d9be31b9122ccf05049b9656457a700a9ce292a
Tamaño: 214528
Tipo de Archivo: application/x-msdownload
Clasificación: Botnet Llaves de registro:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Norton Antiviral

Scanner Información general:
Proceso malicioso: navscnr.exe
Path: ~\WINDOWS\system32\navscnr.exe
Llaves editadas en el registro de Windows:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Norton Antiviral Scanner
Archivos maliciosos creados por el malware:

Created: C:\WINDOWS\system32\navscnr.exe
Características generales:
Este malware abre un puerto TCP aleatorio, lo modifica en cada ejecución. Además utiliza
el puerto 69 (tftp) UDP y se comunica con un servidor irc a través del puerto destino
7997. Intenta propagarse a través del puerto 1197 TCP y se conecta a un sitio remoto por
medio del puerto 7997.

En conclusión, este malware cuenta con características de gusano por su método de
propagación, de puerta trasera por abrir un puerto en el sistema comprometido y de bot
por tener y conectarse a un servidor irc remoto.

Este malware se activa mediante un servicio llamado Norton Antiviral Scanner y se puede
conectar a distintos dominios.
```

Tabla muestra2: Malware navscnr.

Se ejecuta el script y se obtiene como resultado lo impreso en la siguiente Tabla Resultado3: resultado malware navscnr.

Tabla Resultado3: Resultado malware navscnr.

```
Analisis de los scripts

Archivos:
new:     C:///WINNT/system32/navscnr.exe   p(755) t(100000) u(1000) g(513) z(214528)
m(20081026-150416)
new:     C:///WINNT/system32/navscnr.exe   s(608a628b413674995b5dbb239cb5a96232faad55)
new:
C:///WINNT/SoftwareDistribution/Download/1fb165f79840aeef7b62fd1df1d872762ed04dad
p(644) t(100000) u(1000)g(513) z(18446) m(20081026-155100)
new:
C:///WINNT/SoftwareDistribution/Download/1fb165f79840aeef7b62fd1df1d872762ed04dads(d4cb7
0e81ff64904c06eab1398ffc696cee38300)
changed: C:///WINNT/SoftwareDistribution/SelfUpdate/Default/wsus3setup.cab   m(20080317-
125906:20081026-154940) c(20080317-125906:20081026154940)
changed: C:///WINNT/SoftwareDistribution/WuRedir/9482F4B4-E343-43B6B170-
9A65BC822C77/wuredir.cab   m(20080210-151742:20081026-154938) c(20080210151742:20081026-
154938)
changed: C:///Documents and Settings/All Users/Documentos   l(3:4)
changed: C:///Documents and Settings/tesis1/Configuración local/Temp   l(14:13)

Hosts:
No hubo cambios en el archivo /etc/hosts

Red:
No hubo cambios en la red

Llaves de registro:
================================== ========================
                          Diferencias Archivos hosts
================================== ========================
Elementos nuevos o modificados
================================== ========================
Software_Microsoft_Windows_CurrentVersion_Run Norton Antiviral Scanner  REG_SZ    C
_WINNT_system32_navscnr exe
Software_Microsoft_Windows_CurrentVersion_RunOnce Norton Antiviral Scanner  REG_SZ    C
_WINNT_system32_navscnr exe
Software_Microsoft_Windows_CurrentVersion_RunOnceEx Norton Antiviral Scanner  REG_SZ
C _WINNT_system32_navscnr exe
Software_Microsoft_Windows_CurrentVersion_policies Norton Antiviral Scanner  REG_SZ    C
_WINNT_system32_navscnr exe
Software_Microsoft_Windows_CurrentVersion_Setup Norton Antiviral Scanner  REG_SZ    C
_WINNT_system32_navscnr exe
Elementos borrados o modificados
================================== ========================

Portapapeles:
================================== ========================
                          Diferencias entre Protapapeles y tiempo de acceso
================================== ========================
Elementos nuevos o modificados
================================== ========================
System Uptime 0 hours 5 minutes 1 seconds
Elementos borrados o modificados
================================== ========================
System Uptime 0 hours 3 minutes 20 seconds

Procesos:
##############################################################
                                  Procesos Nuevos
##############################################################
Name                 Pid    Pri   Thd    Hnd     Priv          CPU Time    Elapsed Time
navscnr             1084   33828   14840    1856    2024    11659      7   24

Busque las imagenes de los procesos

Puertos:
================================== ========================
                          Diferencias En Los Puertos
================================== ========================
```

```
Elementos nuevos o modificados
================================== ========================
0.0.0.0 51417 TCP LISTENING
Elementos borrados o modificados
================================== ========================
192.168.0.237 1037 TCP ESTABLISHED

Servicios:
No hubo cambios en los servicios

Tareas Programadas:
No hubo cambios en las Tareas Programadas

Usuarios y redes compartidos:
================================== ========================
                          Diferencias Archivos hosts
================================== ========================
Elementos nuevos o modificados
================================== ========================
Elementos borrados o modificados
================================== ========================
```

Estos resultados muestran claramente las modificaciones que se realizaron al sistema, donde se ve que, los procesos, llaves de registro y archivos, son muy apegados al resultado esperado, de lo que no se tiene una salida clara es en el análisis de puertos donde se pone el puerto 51417 como puerto en escucha.

En conclusión se determina que este tipo de programación es lo más confiable para nuestro laboratorio, ya que es eficiente, rápido y confiable, por lo tanto para el siguiente punto es necesario modificar el envío de datos y presentación del proyecto.

4.3 Tercera vuelta de pruebas.

Al contar con el laboratorio de malware en funcionamiento, únicamente resta reparar los detalles, entre los cuales hay que destacar la utilización de un servidor, tal como se menciona en el capítulo 3, el cual será un bastión con una doble funcionalidad como servidor web y como firewall, para ello es necesario generar una política de seguridad.

Otro punto importante en el proyecto es la presentación que se debe de tener hacia terceros, por éste motivo es necesario generar una página web, donde se muestren los resultados de cada espécimen punto por punto.

4.3.1 Complemento de script de salida formato HTML.

Para hacer la presentación de los resultados se deben transformar los archivos de salida de texto a formato html, para este cambio se utiliza una herramienta de terceros llamada “A2hCons.exe”, esta herramienta se utiliza como se muestra el script 4.12.

Script 4.12: Complemento para el cambio de formato.

```
############## [ Generacion de Salida Con Formato HTML ] ################

`A2hCons.exe <archivo>_final.txt`;

sleep(5);

open(HTML,"<archivo>_final.html")||die"no puede abrir $!";

open (SALFIN,"+>C:/scripts/salidas/<archivo>_final_lab.html");

my @html=<HTML>;
```

```
my $salfin;

foreach $salfin(@html){

       $salfin =~s/^(This HTML file was).*/Laboratorio de Malware/g;
       $salfin =~s/^\(This message is omitted in the registered version\)/Proyecto de
Tesis/g;
      print SALFIN $salfin;
}

close(HTML);
close(SALFIN);
```

De esta manera obtenemos una salida en formato html, como se muestra en la figura 4.1, para cada uno de los puntos propuestos en la revisión del espécimen.

En este caso se muestra la salida del archivo final que revisa el hosts, al ejecutar un troyano llamado “launcher” este llega mediante el correo electrónico y lo que hace es cambiar la ip de la dirección de un banco, engañando al cliente con una página falsa que al momento de que el cliente escribe sus datos, éste recibe una página de error o una página falsa, de esta manera el receptor toma las credenciales (nombre de usuario o numero de cuenta y contraseñas) para poder hacer un fraude.

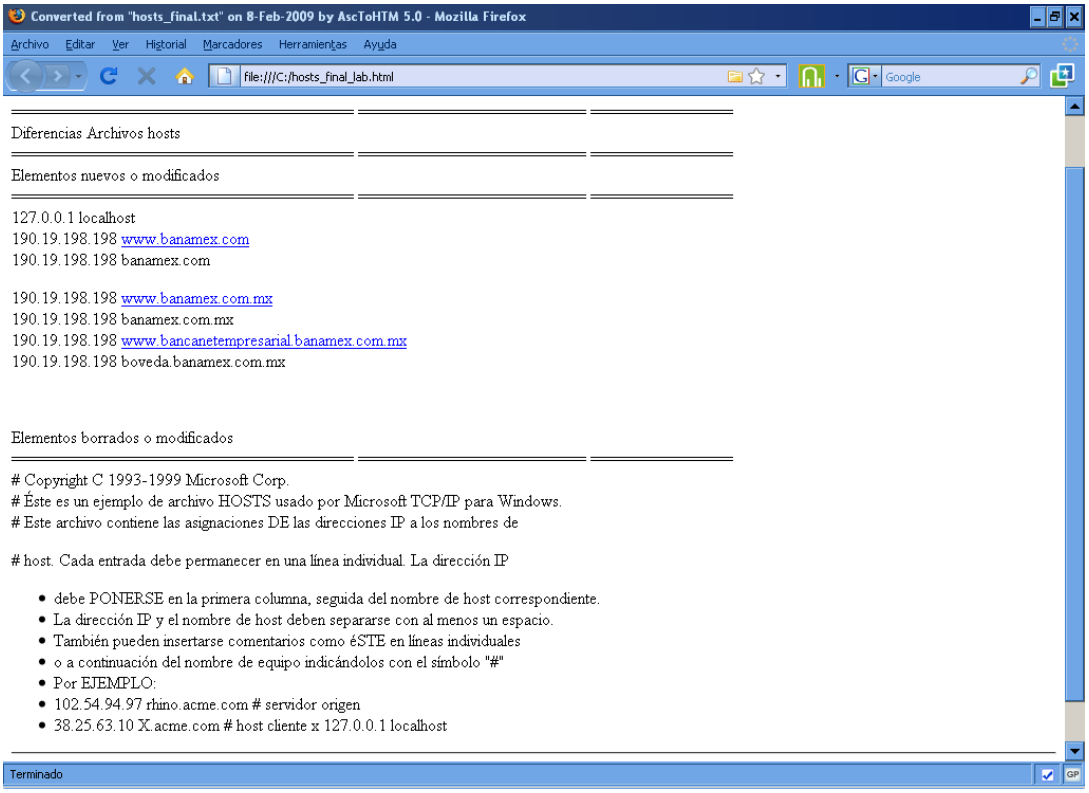

Figura 4.1: Salida del cambio del archivo hosts_final.txt.

4.3.2 Definición de la red para el proyecto.

Ya con todos los elementos y el laboratorio finalizado, lo único que hace falta es definir la red, para ello es necesario tener dos redes una pública y una privada, la primera es la que viene desde cualquier máquina hacia el equipo bastión, que es donde se muestra la salida de los resultados de los especímenes, mientras que la segunda red siendo privada solo tiene una comunicación directa entre el bastión, que es el equipo que contiene los ejecutables del malware y el laboratorio de malware, esta configuración se asignó tal y como se muestra en la figura 4.2.

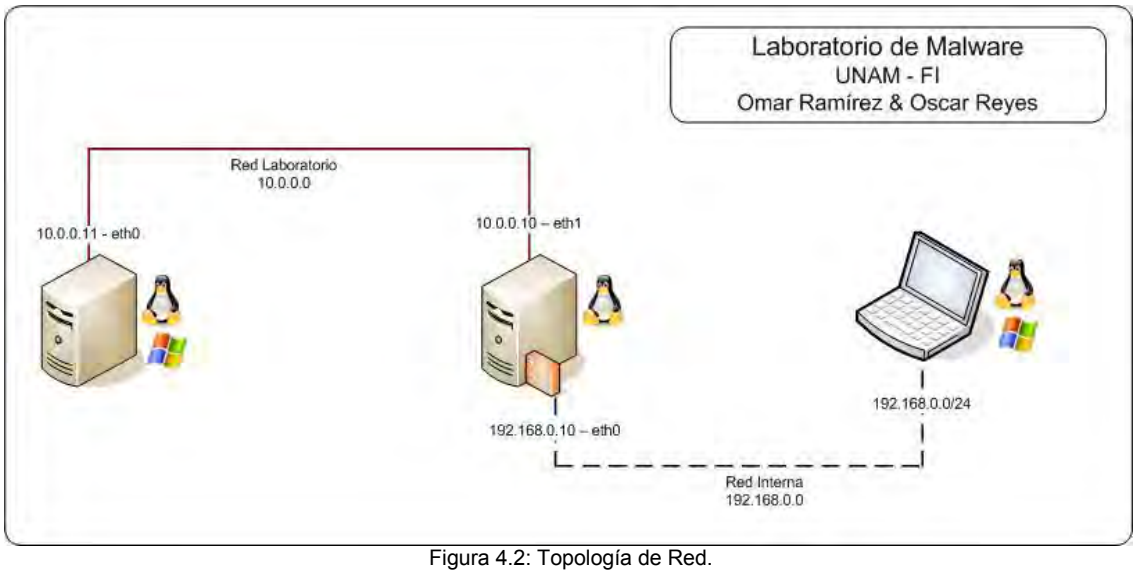

Figura 4.2: Topología de Red.

4.3.3 Definición de reglas de firewall.

Ya definida la red se tienen que configurar las reglas del firewall, descritas en el capítulo 3, en estas reglas solo se tienen abiertos algunos puertos de transmisión de datos, algunas ip's y únicamente esta definido para ciertas direcciones en las políticas de seguridad, las políticas están en el Anexo 2.

4.4 Cuarta vuelta de pruebas.

En esta última vuelta de pruebas, se opera con el espécimen que se utilizó en los 2 primeros análisis (keyloger xey0pkom88c.exe), los cuales generaron los mismos resultados que en la segunda vuelta, a diferencia es que para esta vuelta se cuenta con un formato html listo para mostrarse a cualquier persona.

4.4.1 Imágenes de la salida.

En esta parte sólo muestran las imágines de la salida del análisis del malware keyloger ***xey0pkom88c.exe.***

Para fines prácticos solo se presentan las pantallas de los archivos que sufrieron algún cambio, los otros archivos no se modificaron, a diferencia de los resultados obtenidos en la segunda vuelta de pruebas, se detecta casi el mismo resultado, sólo modificaron los puertos e ip's en donde se trata de conectar el malware, este análisis surgió porque es un software malicioso que muta.

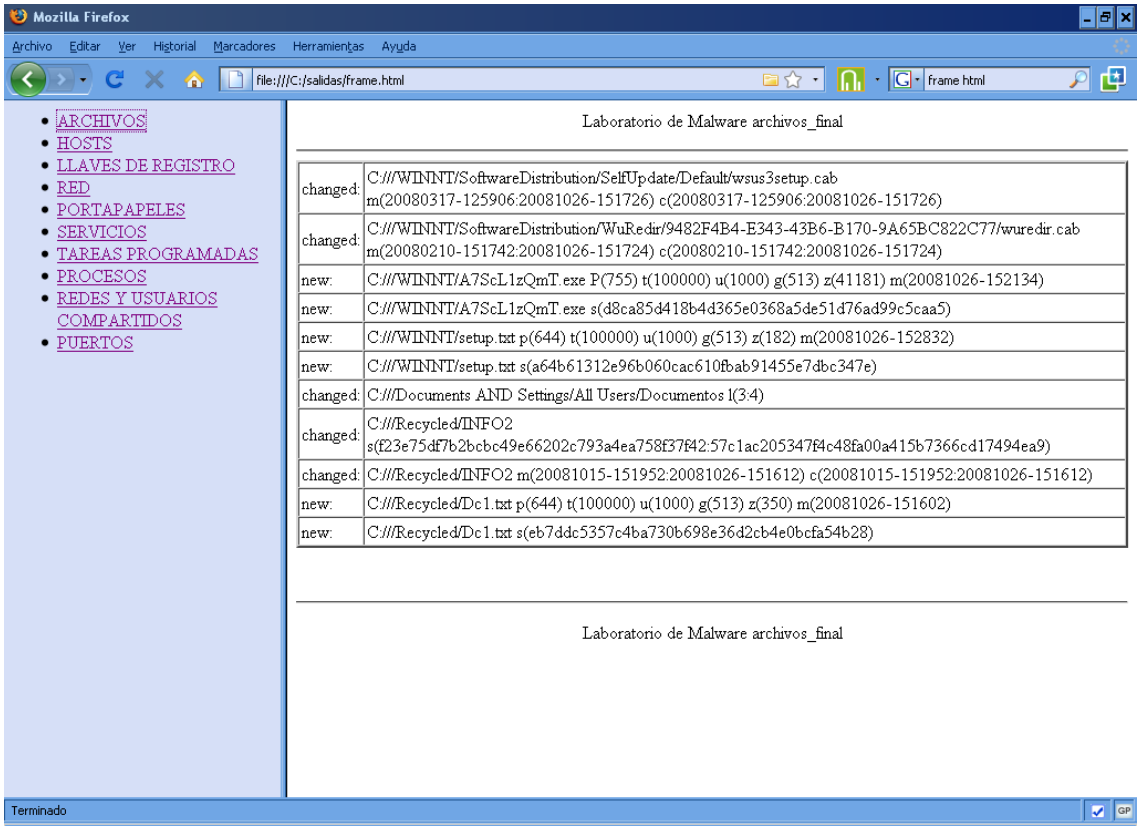

Figura 4.3: página de Archivos_final malware xey0pkom88c.exe

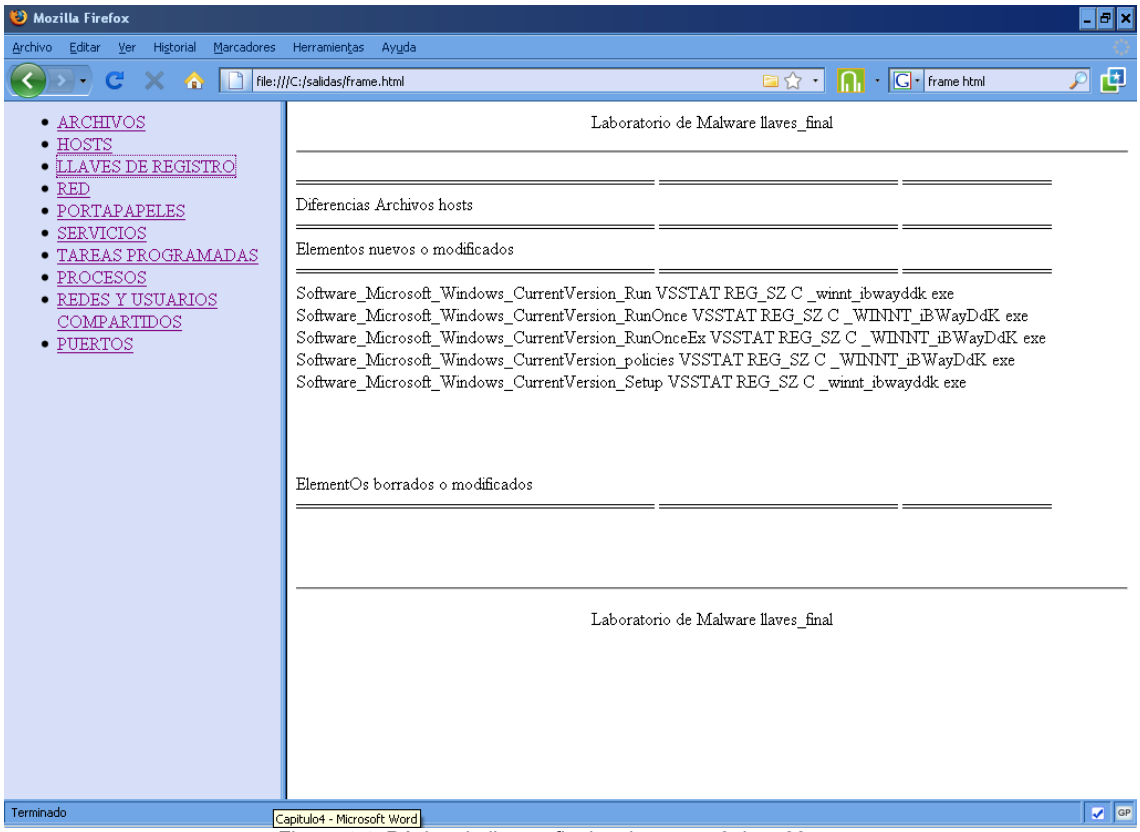

Figura 4.4: Página de llaves_final malware xey0pkom88c.exe.

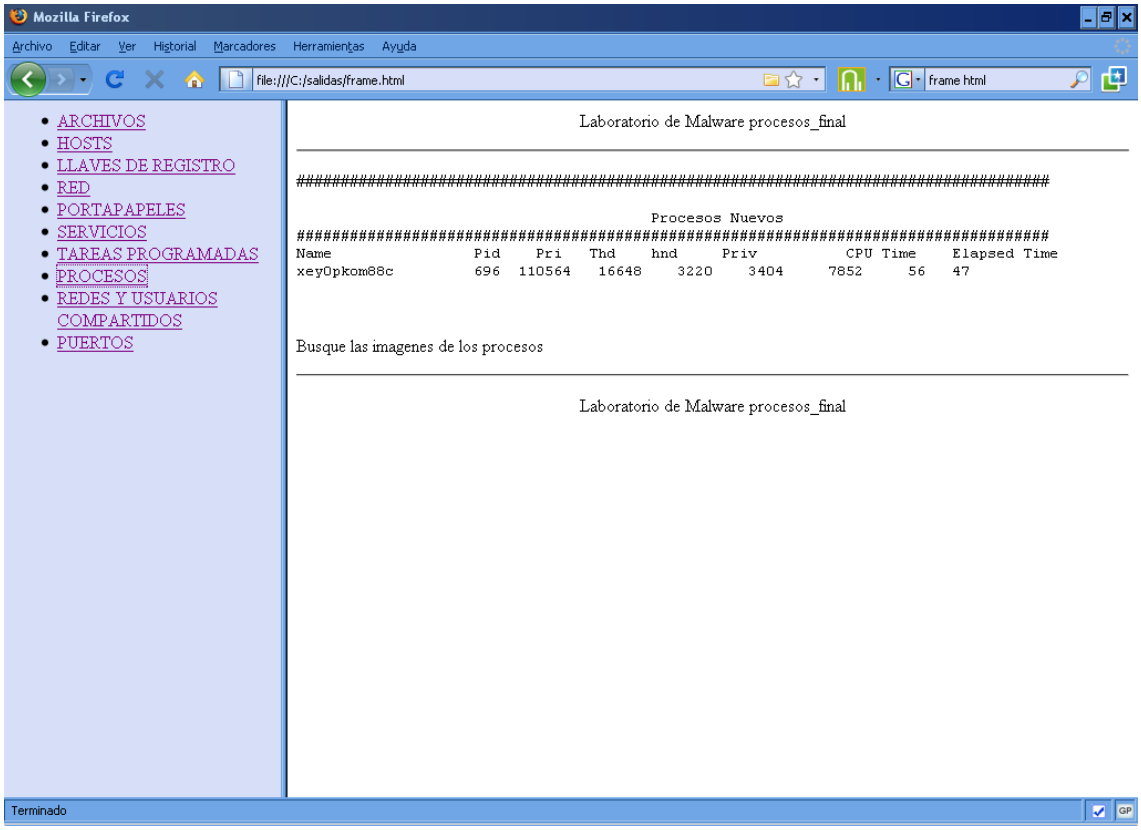

Figura 4.5: página de procesos_final malware xey0pkom88c.exe

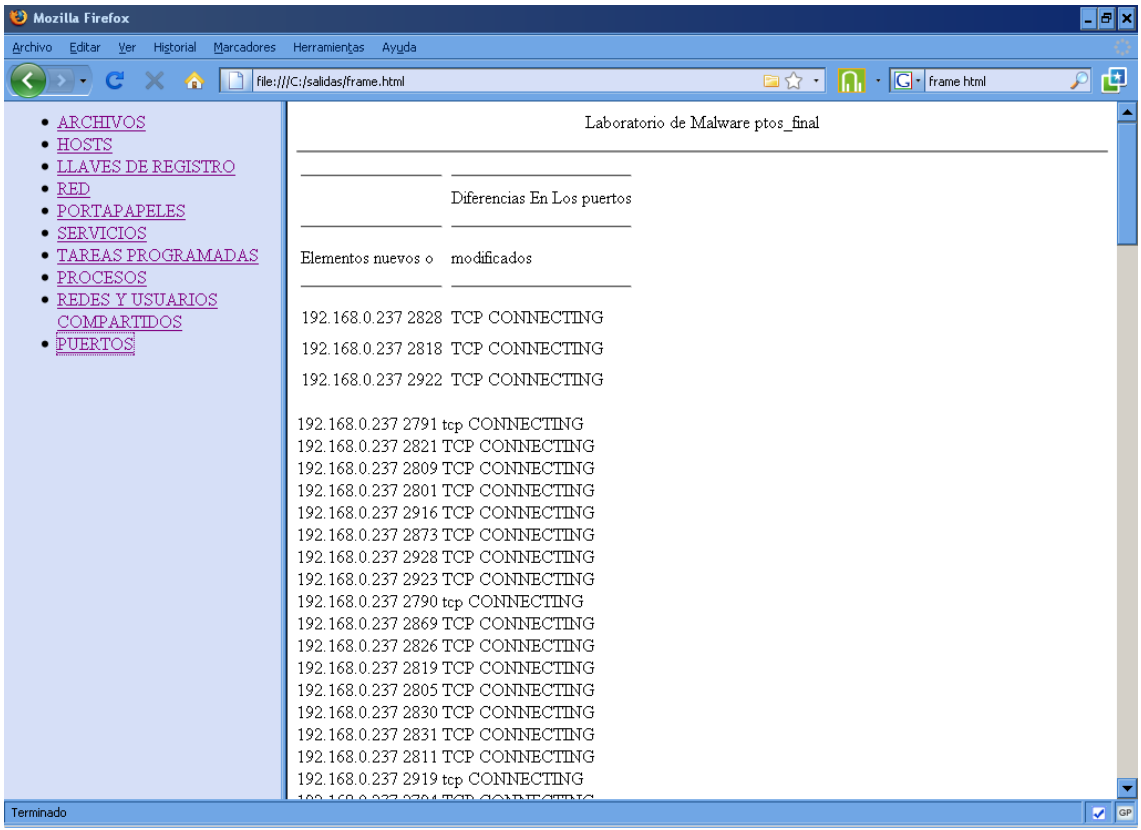

Figura 4.6: Página de puertos_final malware xey0pkom88c.exe.

4.4.2 Análisis de malware llamado TelcelSMS.

Para concluir este análisis se tomaron más especimenes no conocidos y se tienen los siguientes resultados.

Este malware llega mediante un correo electrónico, que trae un phishing scam y el encabezado del correo dice que es una nueva página para enviar mensajes SMS de manera gratuita como se muestra en la figura 4.7, este correo solicita que se realice la descarga de un software y sea instalado figura 4.8 y así es como el malware ingresa al sistema operativo y causa algunos estragos como se ve en la siguiente Tabla Muestra 3.

Tabla Muestra 3: Malware TelcelSMS.exe

```
DATOS DEL REGISTRO DE MALWARE

Nombre del Código Malicioso:  TelcelSMS.exe
Aliases 1:  TelcelSMS.exe.rar
Archivo binario:  TelcelSMS.exe
Firma MD5:  a6fe34e6e632e37d6dc3dcfe36d7c0fe
Firma SHA1:  1c1ccbe5fc1d128456ac1b9897b833979a2f3d3d
Tamaño:  461705
Tipo de Archivo:  application/x-msdos-program
Clasificación:  Troyano
Información general:  El malware viene dentro de un comprimido rar. Debido a que el
ejecutable está empaquetado con UPX no es posible su análisis en VMware.

Al ejecutar el malware se muestra un mensaje indicando al usuario una instalación
exitosa del mismo.

El código malicioso instala un troyano de la aplicación svchost.exe .

Modifica el archivo hosts del equipo, conteniendo lo siguiente:
127.0.0.1 localhost
127.0.0.1 www.banamex.com
127.0.0.1 banamex.com
127.0.0.1 banamex.com.mx
127.0.0.1 www.banamex.com.mx
127.0.0.1 www.bancanetempresarial.banamex.com.mx
127.0.0.1 bancanetempresarial.banamex.com.mx
127.0.0.1 www.bancanetempresarial.banamex.com
127.0.0.1 bancanetempresarial.banamex.com
127.0.0.1 boveda.banamex.com
127.0.0.1 www.boveda.banamex.com
127.0.0.1 www.boveda.banamex.com.mx
127.0.0.1 boveda.banamex.com.mx

Sistema Operativo, Versión:  Windows   XP
Sistema Operativo, Versión:  Windows   2000
Sistema Operativo, Versión:  Windows   2003

Vulnerabilidad relacionada:  Falta de actualizaciones

Método de propagación:  Vulnerabilidad asociada
```

Figura 4.7: Correo de propagación del malware TelcelSMS.exe

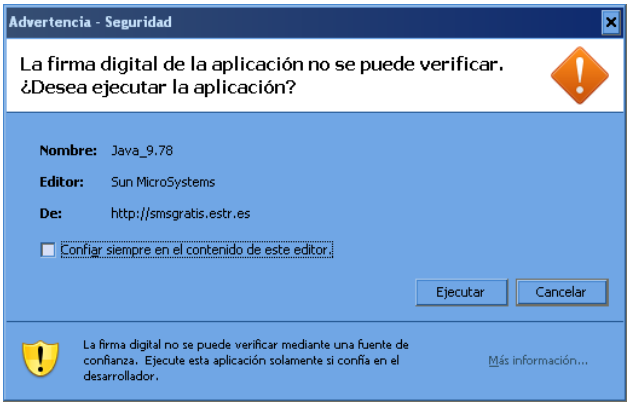

Figura 4.8: Wizard de instalación del malware TelcelSMS.exe

4.4.2.1 Análisis de malware llamado TelcelSMS con el laboratorio de malware.

Una vez analizado el troyano en el laboratorio, se obtienen los siguientes resultados mostrados en las siguientes imágenes, para fines prácticos solo se mostraran las pantallas que fueron modificadas.

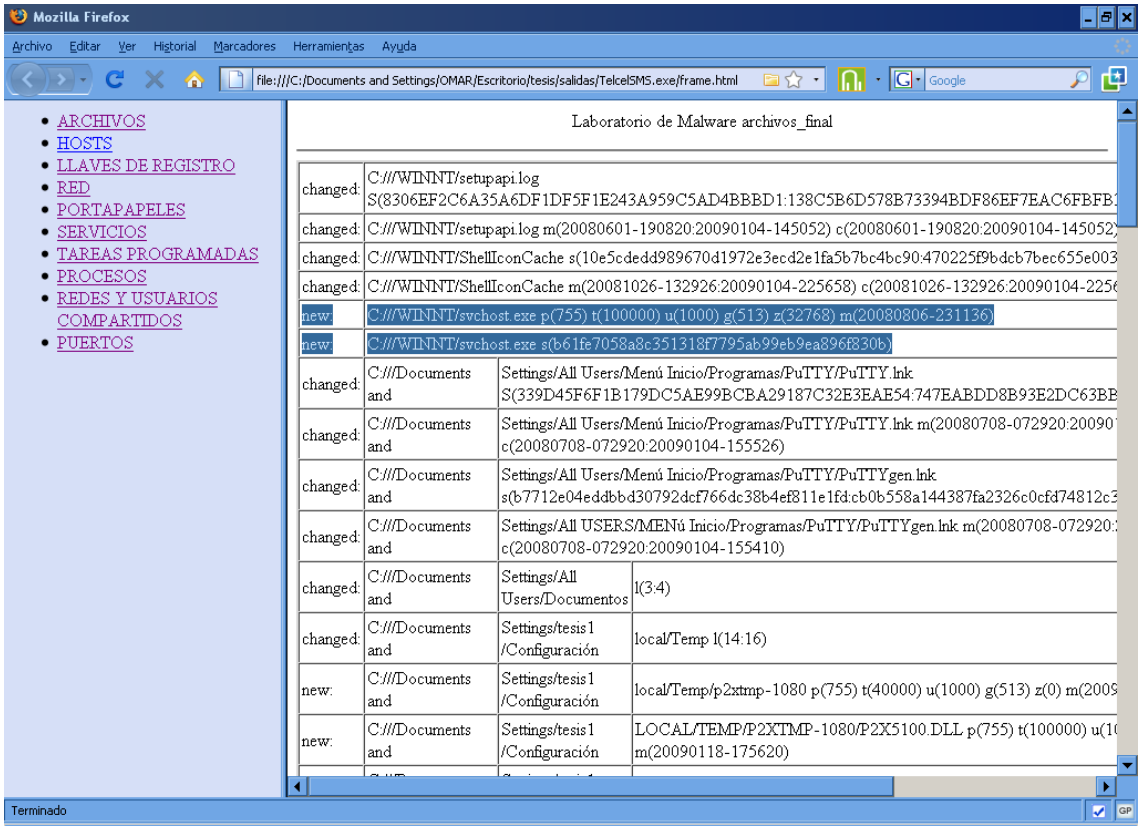

Figura 4.9: Análisis de archivos malware TelcelSMS.exe

De la figura 4.9 la parte subrayada, muestra la creación de los archivos svchost.exe, por lo tanto cumple con lo especificado en la documentación oficial.

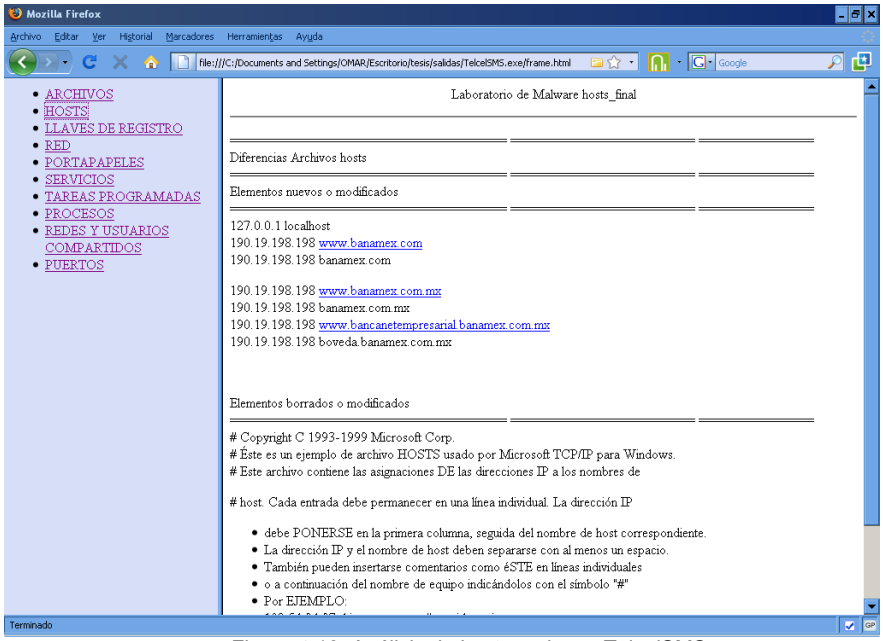

Figura 4.10: Análisis de hosts malware TelcelSMS.exe

Revisando el cambio de hosts, tenemos que de la figura 4.10, se puede observar que se genero un cambio en este archivo, a diferencia de la documentación oficial, el laboratorio detecta una ip 190.19.198.198 mientras que el original lo envía al

loopback (127.0.0.1), lo preocupante aquí es el servername que se utiliza, ya que se especificó anteriormente pertenece a una institución bancaria.

El cambio generado en las llaves del registro se analiza en la figura 4.11. Ya que estas se crean en todo el árbol estándar de tal manera que se ejecutan automáticamente el programa llamado “svchosts.exe”.

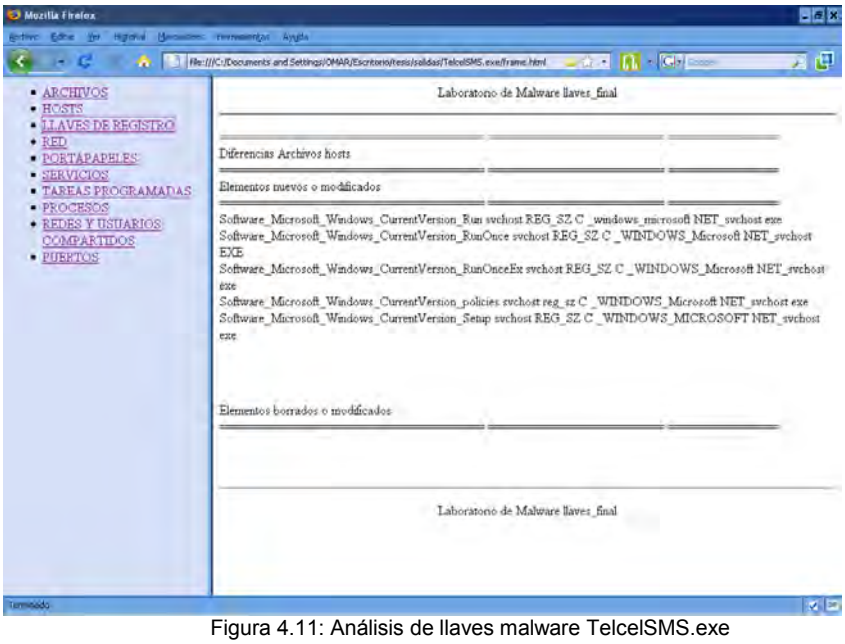

Figura 4.11: Análisis de llaves malware TelcelSMS.exe

De igual forma es posible analizar los procesos que se generan y así observar los siguientes procesos, **net.exe** y **svchosts.exe** como se muestran en la figura 12.

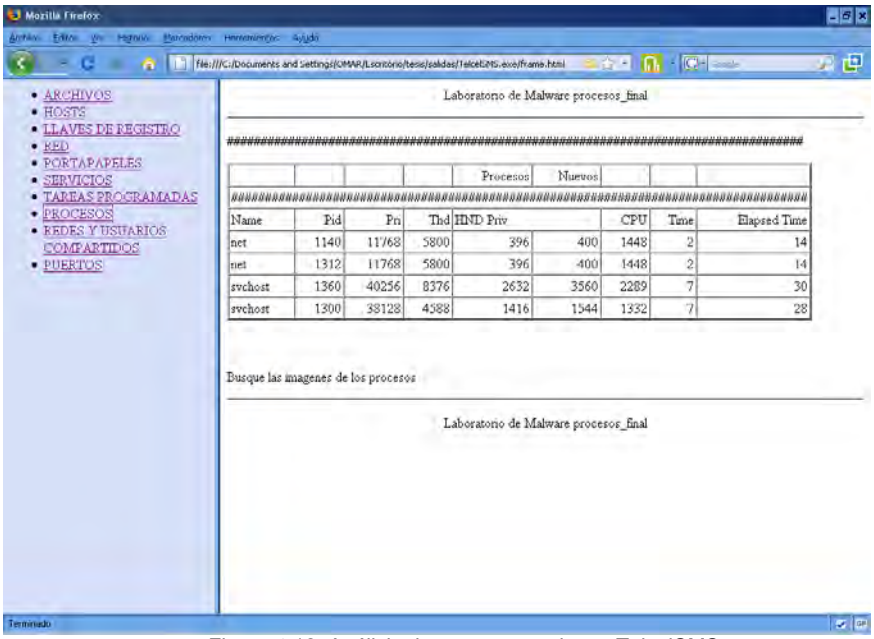

Figura 4.12: Análisis de procesos malware TelcelSMS.exe

Con éste ejemplo se da por finalizado el laboratorio, ya con la versión que se tiene para presentación hacia terceros, a forma de conclusión se considera un proyecto confiable y eficaz, ya que comparando con información oficial resulta ser muy similar.

4.5 Resumen de herramientas usadas en el laboratorio de malware.

A continuación se listan todas y cada una de las herramientas usadas para este proyecto. Ver figura 13.

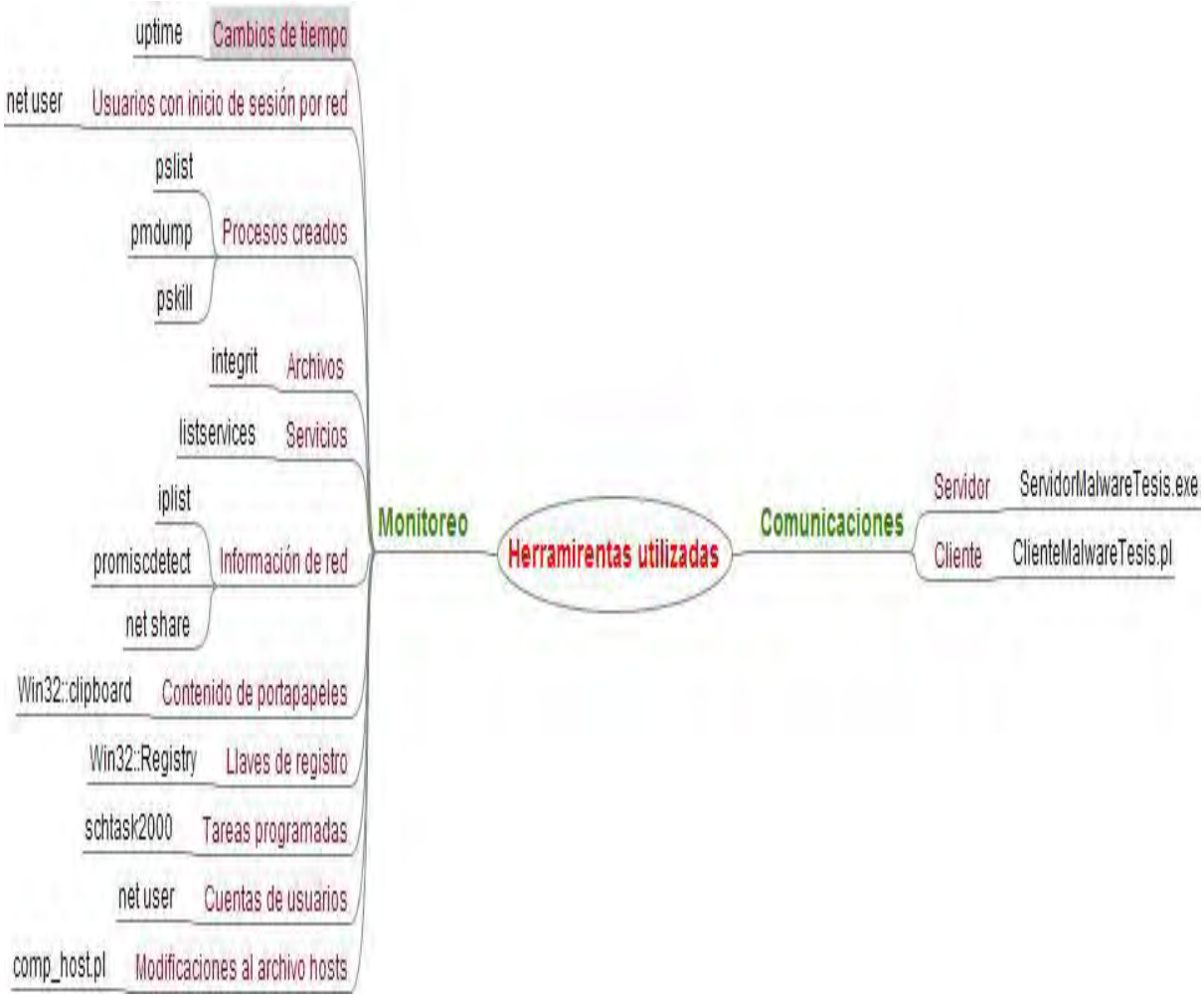

Figura 4.13: Resumen de herramientas utilizadas.

Conclusiones.

5. Conclusiones del proyecto.

Esta parte del proyecto se encuentra dividida principalmente en tres, las cuales especifican todos y cada uno de los resultados obtenidos durante el desarrollo de este trabajo, entre los puntos a destacar son la prevención de ataques de malware, el uso de herramientas de terceros y por último se agregaron algunos elementos clave para continuar con el desarrollo de éste y así en un futuro poder aplicarlo en sistemas operativos más recientes como son Windows 7 y Server2008.

El objetivo principal del proyecto, es realizar un laboratorio de análisis de malware utilizando herramientas de terceros y scripts, que permitan obtener información capaz de revelar el comportamiento del malware. A partir de la realización de pruebas con especimenes reales, los cuales arrojan resultados similares a los consultados en las bases de datos de otros laboratorios consolidados y reconocidos a nivel mundial, como es el UNAM CERT.

En base a la experiencia del manejo de los virus, se determino que no es factible realizar este laboratorio utilizando técnicas de virtualización, aunque es más fácil recuperar la puesta a punto del sistema antes de ser atacado, su uso es dudoso e inexacto ya que los resultados obtenidos por este método no contienen todos los elementos necesarios para que el malware se ejecute adecuadamente, además de que existen especímenes de malware que están programados para no actuar en caso de que el sistema operativo este instalado en un ambiente virtual, por lo que estos sistemas no puede ser un buen blanco de infección.

Otro de los objetivos que se planteo para este proyecto fue trabajar con una máquina real, dejando de lado el inconveniente del tiempo que se tiene que invertir en reinstalar, configurar y poner a punto el equipo cada vez que se prueba un nuevo malware, este punto de falla se resolvió al realizar una copia del sistema operativo original antes de ejecutar el malware, de esta manera el respaldo se realiza bit a bit con la herramienta “dd” mencionada en capítulos anteriores, por lo que el tiempo de recuperación de nuestro sistema operativo base es aproximadamente 20 minutos, este tiempo puede variar dependiendo de la capacidad del hardware del equipo (laboratorio).

El último punto es realizar las pruebas sin tener que invertir en herramientas con licenciamientos costosos, ya que esta solución es para investigación y prevención de ataques con un bajo costo de implementación.

En base a los requerimientos planteados en el capítulo 3, se destacan principalmente dos puntos claves en este proyecto, se tratan de herramientas que se adaptaron para obtener un mejor funcionamiento en el sistema operativo propuesto, estas herramientas son **integrit.exe, y schtasks2000.exe**.

La primera herramienta crea una pequeña base de datos de todos y cada uno de los archivos que se encuentran en el sistema operativo, comparando su integridad mediante el uso de firmas digitales. Para que se ejecute de manera eficiente y sin necesidad de integrar un emulador de Unix (cywin), se adapta esta herramienta a una plataforma Windows, para lo cual es necesario el desarrollo de un procedimiento que sea funcionalmente estable y así poder ser ejecutado desde línea de comandos, la especificación a detalle se encuentra en el capítulo 3.

La principal ventaja de la adaptación de esta herramienta es minimizar el tiempo de comparación de los archivos del sistema operativo hasta por 35 minutos, por tal motivo es de gran ayuda para el desarrollo del proyecto, de este modo la búsqueda se dio de manera eficiente al ser rápida y confiable en cada uno de los archivos.

Por otra parte la segunda herramienta realiza la función de búsqueda de tareas programadas, cabe señalar que no existía una herramienta, ni de terceros, ni de sistema operativo que nos brindara esta funcionalidad, por lo que fue necesario adaptarla de un sistema operativo diferente, en este caso se tomo de un Windows XP y se integro al sistema operativo usado en el laboratorio (Windows 2000), para realizar esta adaptación se modifico el binario a nivel hexadecimal.

Teniendo listas estas 2 modificaciones en el sistema, se realiza el laboratorio en un esquema de desarrollo en espiral, como se muestra en el capítulo 4, se realizaron pruebas con virus conocidos en un ambiente controlado y así obtener resultados esperados.

Para este proyecto, se realizaron pruebas con sistemas operativos de la marca Windows, como se ve en el capítulo 4, el motivo por el que se realizo en un Windows 2000 profesional, fue que se tenia en mente crear una herramienta para realizar el respaldo y restauración, esta versión tiene la opción de manipular el MBR.

En cuanto al servidor bastión, se tiene asegurado mediante cierre de puertos y la desactivación servicios innecesarios, instalación mínima, aplicación de políticas de seguridad en Sistemas Operativos, en aplicaciones tanto en el servidor de web, como en el SSH, para reducir los puntos vulnerables.

En la etapa de implementación, se utilizó la herramienta webmin, esta facilita la administración de un sistema Unix, ya que tiene una interfase de operación grafica que se provee vía web estableciendo un canal seguro con el protocolo https.

Esta herramienta facilita la administración de recursos del equipo, se utilizó para generar las reglas del firewall, la configuración del servidor de web (apache), configuración del SSH, además de todas estas utilidades, tiene la posibilidad de crear un respaldo de todos y cada uno de los archivos de configuración, para que en caso de perder esta información, permite regresar a la configuración inicial.

El laboratorio de análisis de malware trabaja de manera eficiente, confiable y segura, ya que la información que nos proporciona es muy similar a la obtenida de otras fuentes. Por lo tanto este punto se concluyó de manera satisfactoria.

En base a los resultados obtenidos, experiencia laboral y cultura adquirida en el desarrollo de la práctica en administración y seguridad de sistemas de información, es posible dar algunas sugerencias que mantengan el resguardo de información ante el ataque de cualquier tipo de malware, algunas de las cuales se mencionan a continuación.

Es importante destacar la necesidad de impulsar y fomentar una cultura de la seguridad informática aplicada a los usuarios, para así asegurar la confidencialidad de datos personales, ya que como bien se sabe en este mundo tecnológico es muy fácil obtener información y lucrar con ella, ante tal problema se recomienda hacer hincapié en algunos puntos donde se han detectado la mayor cantidad de ataques por malware,

que bien pueden ser mediante correo electrónico, mensajería instantánea o simplemente por la mala administración de los equipos de cómputo.

A continuación se brindan algunas recomendaciones para minimizar el daño que puede causar el malware, entre los ataques más comunes para usuarios que tienen conexión a internet.

Otro de los riesgos detectados con mayor frecuencia durante el desarrollo del proyecto es el uso de la banca electrónica, al identificar al usuario como un ente vulnerable al no tener los conocimientos, ni la cultura en cuanto a la seguridad y resguardo de su información.

Un ejemplo típico de estos ataques se encuentra especificado en el capítulo 4, donde se muestra un correo electrónico ordinario, que através de una invitación anuncia el servicio gratuito de SMS, para lo cual es necesario la descarga e instalación de un software de dudosa y desconocida procedencia, sin embargo para un usuario común este tipo de correspondencia es cotidiana, sin saber que se trata de un ataque.

Al realizar el análisis correspondiente en el laboratorio, este arrojo que se presentaba el caso de un envenenamiento al archivo que resuelve los nombres a nivel local dentro del equipo, el cual actúa modificando las direcciones IP de una institución bancaria por direcciones de los servidores del atacante.

Los servidores del atacante contienen una replica exacta de la pagina de uso del banco, entonces al escribir los datos privados, se envía la información a otro destino, generando de esta manera un fraude bancario y un uso malicioso y ventajoso de datos confidenciales.

Las recomendaciones para protegerse de este tipo de fraudes son:

- Realizar periódicamente actualizaciones del antivirus en la computadora del usuario. A través de programas llamados “antispyware” y “antiadware” que ayudan a proteger los sistemas en contra de usuarios maliciosos como los llamados “hackers”.

- Al solicitar soporte para una computadora, debe ser gente de tu confianza, ya que podrían instalar algún programa como los llamados “spyware”, los cuales espían todas las actividades que se realizan en el equipo durante el uso de Internet y también forzar a desplegar páginas de Internet inseguras, así como publicidad.

- Procurar no utilizar computadoras públicas en las que no se tenga completa seguridad o confianza, ya que pueden tener instalados programas que pueden copiar contraseñas y otros datos personales. Una computadora pública es la que utilizan otras personas como las que encuentran en los cafés internet, centros de negocios de hoteles, aeropuertos, universidades, entre otros.

- Ingresar a través del teclado siempre la dirección de la página web de la institución financiera directamente en el explorador. Nunca se debe de acceder a través de ligas o hipervínculos, o a través de ligas en correos electrónicos.” [1]

- Además de estas observaciones se recomienda que se firme como un usuario privilegiado solo para realizar tareas administrativas como son instalación de

[1] http://www.cnbv.gob.mx/recursos/folletorecom.pdf

programas, creación de usuarios, mientras que un usuario con privilegios limitados se utilice solo para realizar tareas cotidianas, como son navegar en internet, crear documentos de Word, etcétera.

- Se recomienda no instalar software de dudosa procedencia y los llamados P2P, que es un software que abre un canal de comunicación directo con una máquina de un posible atacante.

Como se puede observar, la mayor parte de los binarios que utiliza el malware, se propagan mediante correo electrónico, ya que los usuarios de un equipo de cómputo generalmente utiliza este medio para su comunicación, un ejemplo claro es el caso del ataque anteriormente mencionado en el capítulo 4, En el que la primera etapa para poder llevar acabo el ataque se da mediante el correo electrónico.

Para evitar este tipo de ataques, se recomienda.

- No abrir correos de remitentes desconocidos.
- No ejecutar archivos que estén adjuntos antes de ser analizados por un antivirus.
- Ignorar las ligas adjuntas a cualquier correo electrónico.

Por otro lado al ser un medio de comunicación los mensajeros instantáneos, se convierte en una importante y vulnerable fuente de propagación, tal y como se vio en el caso del correo electrónico, hay virus que utilizan estos canales para propagarse, recientemente se ha dado el caso de un virus que Kaspersky lo nombra como **IM-Worm.Win32.VB.az**, este espécimen envía un mensaje, regularmente en otro idioma diferente al español, donde te pide que veas unas fotos y envía un archivo *.zip, que al abrirlo se ejecuta un malware.

Recomendaciones para evitar este tipo de transmisión.

- No ejecutar archivos sin antes ser analizados por un antivirus.
- Ignorar las ligas adjuntas que venga en un mensaje de este de tipo de medios de comunicación.

En todos los casos es recomendable tener actualizado el sistema operativo con los últimos niveles de mantenimiento, tener antispyware, antivirus, firewall de host, todos estos configurados y actualizados a la última versión disponible.

Una buena práctica para evitar perdida de información es realizar respaldos en discos externos.

Para tener una mejor administración y seguridad en los equipos de cómputo, se recomienda tener particionado el disco duro en dos como mínimo, la primera de ellas es donde reside el sistema operativo y en la segunda se guardan todos los datos, archivos, documentos que el usuario crea en su utilización día a día.

Para evitar los ataques es necesario tomar conciencia del buen uso de las cuentas de acceso a cada sistema, para ello se debe contar con un control de contraseñas mediante las siguientes acciones:

- Debe ser mayor a 8 caracteres.
- Utilizar MAYUSCULA, minúsculas, números, caracteres especiales ($,@,*,$).

- No debe contener palabras de uso común o que puedan encontrarse en un diccionario, ni deberán estar basadas en información personal (fecha de nacimiento, números telefónicos u otra información de carácter publico).
- Deberá de cambiarse el password al menos cada mes.

Es así como este proyecto se convierte en un buen complemento para un análisis practico de todo el malware que se obtiene en las Honeypot o Honeynet, ya que este tipo de sistemas simulan ser vulnerables y los atacantes ya sea hackers o crackers caen fácilmente en estas trampas, de esta manera un atacante se conecta al servicio y trata de penetrar en él, este programa imita parcialmente un agujero de seguridad pero realmente no permite ganar completamente el control del sistema, registrando la actividad del atacante, posteriormente recoge información utilizada y realiza un análisis a los binarios.

En un futuro este trabajo puede ser implementado en otros sistemas operativos actuales, por tal razón una de los principales planteamientos de este proyecto es desarrollarlo de forma modular, ya que cada punto de búsqueda realiza funciones muy parecidas con herramientas muy similares para todos los sistemas operativos analizados, por ejemplo, la comparación en la ejecución de pruebas en un sistema operativo Windows XP, encontrándose pequeñas diferencias respecto a los resultados obtenidos en el laboratorio del Windows 2000, entre ellas es que el binario de la herramienta que se utiliza para encontrar las tareas programadas **schtasks.exe** sufrió una modificación en sus hexadecimales binarios y el comando dd de las UnixUtils no funciona desde el WindowsXP con los service pack.

Otro punto a destacar es el análisis, desarrollo y creación de una base de datos de cada uno de los especimenes que se ejecutan en el laboratorio, esta adaptación se propone para futuras generaciones que utilicen este proyecto.

Apéndice A.

Reglas de Firewall.

Estas son las reglas estipuladas en la herramienta iptables.

```
# Generated by iptables-save v1.3.6 on Mon Jan  5 05:29:39 2009
*raw
:PREROUTING ACCEPT [146:16269]
:OUTPUT ACCEPT [183:81807]
COMMIT
# Completed on Mon Jan  5 05:29:39 2009
# Generated by iptables-save v1.3.6 on Mon Jan  5 05:29:39 2009
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
# Completed on Mon Jan  5 05:29:39 2009
# Generated by iptables-save v1.3.6 on Mon Jan  5 05:29:39 2009
*mangle
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
# Completed on Mon Jan  5 05:29:39 2009
# Generated by iptables-save v1.3.6 on Mon Jan  5 05:29:39 2009
*filter
:FORWARD DROP [0:0]
:INPUT DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -p tcp -m tcp -s 192.168.0.100 -d 192.168.0.10 --dport 10000 -j ACCEPT
-A OUTPUT -p tcp -m tcp -s 192.168.0.10 -d 192.168.0.100 --sport 10000 -j ACCEPT
-A INPUT -p tcp -m tcp -d 192.168.0.10 --dport 22 -j ACCEPT
-A OUTPUT -p tcp -m tcp -s 192.168.0.10 --sport 22 -j ACCEPT
-A INPUT -p tcp -m tcp -s 10.0.0.11 -d 10.0.0.10 --sport 11110 -j ACCEPT
-A OUTPUT -p tcp -m tcp -s 10.0.0.10 -d 10.0.0.11 --dport 11110 -j ACCEPT
-A INPUT -p tcp -m tcp -s 10.0.0.11 -d 10.0.0.10 --dport 22 -j ACCEPT
-A OUTPUT -p tcp -m tcp -s 10.0.0.10 -d 10.0.0.11 --sport 22 -j ACCEPT
-A OUTPUT -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT
-A INPUT -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT
COMMIT
# Completed on Mon Jan  5 05:29:39 2009
```

Apéndice B.

Baseline de configuración de servidores Linux.

ID	Configuración	Implantación		Comentarios y/o Justificación
		SI	NO	
Seguridad Física del Servidor				
1	El servidor se debe ubicar en un site con acceso controlado.	✔		Si es necesario tener un acceso controlado, ya que este servidor es un laboratorio con uso especifico
2	Contar con UPS (Unninterrumpible Power Supply).	✔		Si, al manos tenerlo conectado en un nobrake
3	Identificar con etiqueta o algún otro medio al servidor.	✔		Si, al menos para identificarlo físicamente
4	Asignar contraseña al BIOS del servidor.	✔		Si, cuenta con ella
5	Deshabilitar la opción para bootear desde CD una vez que el servidor haya sido instalado.	✔		Si, para evitar algún ataque
Instalación del Servidor				
1	Contar con un esquema de particionamiento bien definido de acuerdo a las necesidades del sistema.		☒	No es necesario, ya que este equipo solo cuenta con un SO, y las herramientas que se instalan, no necesitan otro FS.
2	Contar con particiones independientes para cada file system.		☒	No es necesario, ya que este equipo solo cuenta con un SO, y las herramientas que se instalan, no necesitan otro FS.
3	Generar un kernel personalizado, no utilizar el que se instala por default. Deshabilitar la opción *firewall packet netlink device* para evitar conflictos con iptables. Habilitar los módulos necesarios de Netfilter (iptables). Hacer un aseguramiento general de la implementación de la pila TCP/IP como: Deshabilitar la opción de *ip_forwarding*. Deshabilitar la respuesta a paquetes icmp de broadcast. Deshabilitar los mensajes de redirección de IP (Puede ser modificada la tabla de ruteo local). Deshabilitar la opción de *syn_cookies* para evitar ataques de DoS:	✔		Se realizo esta configuración
4	Proteger el bootloader (grub) mediante el uso de contraseña.		☒	
5	Evitar dentro de lo posible la instalación del servidor X.	✔		
6	Actualizar el sistema en su totalidad con las últimas actualizaciones de seguridad (kernel y aplicaciones).	✔		Ya estan instaladas
7	Descargar los parches de actualización del sitio oficial de Debian, así como verificar el checksum y las firmas electrónicas de los paquetes descargados.			
8	Establecer esquema de cuotas en el sistema de archivos.		☒	No es necesario, ya que los archivos que se tienen no son muy grandes
Servicios				
1	Identificar qué servicios se están ejecutando en el servidor.	✔		Se lleva el control de los servicios
2	Deshabilitar aquellos servicios que no son necesarios (cups, kudzu, lpd, etc.).	✔		
3	Utilizar versiones seguras de los servicios si es que existen (ssh, sftp, https, etc.).	✔		Se utilizan las versiones seguras
4	De ser posible ejecutar los servicios con un usuario diferente a root.	✔		Se utilizara un usuario diferente a root para el login, por lo que se utilizara sudo
5	Habilitar el envío de eventos relevantes a bitácoras. Accesos a servicios, autenticaciones exitosas y fallidas, alta, baja y reinicio del servicio, intentos continuos de conexión, número de			

ID	Configuración	Implantación		Comentarios y/o Justificación
		SI	NO	
	conexiones activas, etc.			
6	Deshabilitar, de ser posible, el servicio de portmap. (Este servicio no se requiere si no se está utilizando NIS o NFS).			
7	Habilitar los mecanismos de control de acceso (iptables, tcpwrappers).		☒	
8	Sincronizar la hora y fecha del equipo con el servidor NTP.		☒	
Cuenta de root				
1	Habilitar el uso de contraseñas con MD5			
2	Las características de la contraseña serán: ▪ 16 caracteres como mínimo ▪ Emplear al menos un carácter especial. ▪ Emplear al menos una letra mayúscula o minúscula. ▪ Cambiar la contraseña mensualmente. ▪ Contener letras mayúsculas y letras minúsculas Contener *al menos* uno de los siguientes símbolos especiales: !, @, #, $, %, &, /, (,), ?, _, \, +, “ ▪ Contener *al menos* un dígito (0-9) <u>La contraseña no debe:</u> ▪ Estar basada en palabras que se encuentren en el diccionario ▪ Estar basada en información personal. Esto incluye palabras tales como nombres de familiares, nombres de mascotas, nombres de lugares, fechas de cumpleaños, números de teléfono. ▪ Estar basada en patrones simples de letras del teclado, tales como *aaaaa*, *qwerty*, *123321*, etc. ▪ Estar basada en cualquiera de las anteriores deletreadas hacia atrás. Por ejemplo: *oterces*, *elcaro*, etc.			T3s1s123
5	Hacer uso de la utilería *sudo* para control de acceso y ejecución de comandos administrativos.		☒	
6	Restringir el acceso directo con la cuenta de root excepto desde la consola del servidor.			
7	Verificar que los archivos o scripts que son ejecutados por el cron de root pertenezcan a él.			
Autenticación y usuarios				
1	Saber con exactitud cuántos usuarios están dados de alta en el sistema mediante un documento formal.			
2	Definir un criterio para la asignación de contraseñas a los usuarios se propone el siguiente: • Tener una longitud mínima de 8 caracteres • Contener letras mayúsculas y letras minúsculas • Contener *al menos* uno de los siguientes símbolos especiales: !,			

ID	Configuración	Implantación		Comentarios y/o Justificación
		SI	NO	
	@, #, $, %, &, /, (,), ?, _, \, +, “ • Contener *al menos* un dígito (0-9) La contraseña no debe: • Estar basada en palabras que se encuentren en el diccionario • Estar basada en información personal. Esto incluye palabras tales como nombres de familiares, nombres de mascotas, nombres de lugares, fechas de cumpleaños, números de teléfono. • Estar basada en patrones simples de letras del teclado, tales como *aaaaa*, *qwerty*, *123321*, etc. • Estar basada en cualquiera de las anteriores deletreadas hacia atrás. Por ejemplo: *oterces*, *elcaro*, etc.			
3	Deshabilitar los esquemas que no requieren de contraseña (borrar /etc/host.equiv, .rhost, etc).			
4	Emplear SSH para sesiones remotas			
5	No permitir el uso de sesiones remotas sobre canales no cifrados como rlogin, rsh, telnet, ftp, etc.			
6	Deshabilitar el acceso remoto como root.			
7	Realizar Password Cracking cada vez que un usuario nuevo es dado de alta en el Sistema.			
Monitoreo				
1	Verificar que el servicio de syslog esté activado.			
2	Contar con un servidor centralizado y remoto de bitácoras.			
3	Contar con algún mecanismo automático de monitoreo de bitácoras (logwatch, swatch, etc.).			
4	Contar con algún mecanismo de integridad de archivos del sistema (tipo tripwire).			
5	Instalar el monitoreo de accouting del sistema (uso de recursos del sistema).			
6	Llevar un control y monitoreo de los trabajos calendarizados (cron, at).			
7	Monitorear todos los servicios instalados en el servidor.			
8	Verificar el CheckSum de los binarios activos sean los correspondientes acorde al checksum indicado en la página del Fabricante (ej. Redhat)			
Red				
1	Asignar una dirección IP bien definida			
2	La asignación de la IP deber ser de manera estática.			
3	Configurar firewall de host (iptables) con política restrictiva..			
4	Configurar TCP-Wrappers de manera restrictiva y enviando alertas de intento de accesos			
5	Replicar las reglas de filtrado de paquetes en iptables y tcp wrappers.			
6	Verificar qué servicios de red están			

ID	Configuración	Implantación		Comentarios y/o Justificación
		SI	NO	
	activos (netstat –na| grep LISTEN). Validar que sean los correctos y/o autorizados.			
7	Deshabilitar el acceso como root vía SSH.			
8	Configurar el banner de acceso en SSH.			
9	Sólo acceder por el protocolo 2 de SSH.			
10	En caso de contar con algún sistema de archivos en red (NFS), habilitar algún mecanismo de control de acceso a este recurso. Por ejemplo: Filtrar el puerto 111 udp y 2049 tcp en el firewall o router del perímetro para que sólo accedan equipos de la red local. Montar los sistemas de archivos como solo lectura. Evitar exportar el sistema de archivos a todos, usar la opción *access=host*.			
11	Todas las sesiones gráficas remotas deberán hacerse utilizando SSH.			
	Recuperación			
1	Definir un esquema de respaldos, y contar con los procedimientos necesarios.			
2	Verificar el funcionamiento del esquema de respaldos. Probar si se puede recuperar la funcionalidad del sistema con los respaldos que se realizan.			
3	Se cuenta con el expertise y/o la documentación necesaria para reparar el servidor desde desperfectos físicos hasta la puesta en producción de los servicios.			
	Permisos			
1	Verificar que el umask default de todos los archivos sea 022.			
2	Revisar que el directorio y archivos bajo /var/log puedan ser sólo escritos por root			
3	Verificar que el archivo /etc/services/ pertenece a root y sus permisos son 644			
4	Verificar que el archivo /etc/login.defs pertenece a root y sus permisos son 600			
5	Verificar que el archivo /etc/crontab pertenece a root y permisos 600. Si es posible, deshabilitar el uso de cron a usuarios.			
6	Verificar que los permisos de los archivos /etc/motd y /etc/mtab sean 644			
7	Verificar que el archivo /etc/exports pertenece a root y sus permisos sean 644.			
8	Verificar que los permisos de los archivos /var/run/syslog.pid sean 644			
9	Considerar el deshabilitar el acceso a lectura a archivos de configuración del sistema. Dejar el permiso sólo a root. (p.ej. /etc/hosts.allow, etc.)			
10	Verificar que la imagen del kernel /boot/vmlinuz pertenece al grupo 0, su dueño es root y sus permisos sean 644.			
11	Verificar que los directorios /etc /usr/etc /bin /usr/bin /sbin /usr/sbin /tmp y /var/tmp pertenecen a root y el sticky-bit se encuentra sólo en los directorios /tmp y /var/tmp			
12	Verificar que todos los archivos bajo /dev sean archivos especiales (de carácter o bloque).			
13	Hacer una revisión de los archivos que cuentan con SUID y GID, validar que dichos archivos realmente lo requieren y eliminar el sticky-bit de aquellos en los			

ID	Configuración	Implantación		Comentarios y/o Justificación
		SI	NO	
	que no sea necesario. A continuación los que son necesarios: /bin/ping /bin/su /usr/bin/at /usr/bin/chage /usr/bin/chfn /usr/bin/chsh /usr/bin/crontab /usr/bin/gpasswd /usr/bin/newgrp /usr/bin/passwd /usr/sbin/postdrop /usr/sbin/postqueue /usr/sbin/utempter			

Apéndice C.

Webmin.

El presente procedimiento tiene la finalidad de llevar a cabo la instalación y puesta a punto del servicio Webmin en un servidor Linux desde cero.

El servidor Webmin es una interfaz basada en web para la administración de sistemas Unix. Usando cualquier navegador que soporte tablas y formularios (y Java para el módulo de gestión de archivos).

Webmin consiste en un servidor simple y compacto, así como un número de programas CGI que directamente ponen al día archivos de sistema como/etc/inetd.conf y/etc/passwd. El servidor web y todos los programas CGI son escritos en la versión 5 Perl, y no usan ningún módulo no estándar de Perl.

Los requisitos previos para llevar a cabo este procedimiento son contar con un servidor con Linux instalado, acceso a este servidor como usuario con privilegios administrativos y conexión a internet desde este servidor.

Cada vez más, Linux tiende hacia la simplificación y al manejo sencillo de archivos de configuración que permiten la aceptación total por parte del usuario final, y una herramienta que acerca mucho la administración del sistema operativo es Webmin, que si bien ya se explicó que es un servidor dentro de una máquina Linux que puede ser configurado utilizando cualquier navegador Web.

El desarrollador principal de Webmin es Jamie Cameron y ha estado al frente del proyecto desde sus inicios, la forma de distribución de Webmin así como su licencia de uso es BSD, esto significa que puede ser libremente distribuido y modificado tanto para usos comerciales como no comerciales. Esta filosofía está soportada y sustentada en el hecho de que Webmin utiliza el concepto de módulos (como en el caso de los plugins de Photoshop) y es por eso que cualquier persona los puede distribuir libremente bajo ninguna consideración de licencias.

Los sistemas operativos soportados por Webmin para su versión 1.480 son los siguientes:

Tabla C.1: Sistemas operativos soportados por Webmin.

AlphaCore Linux	Immunix Linux	Slackware Linux
APLINUX	Lanthan Linux	Slamd64 Linux
Asianux	LinuxPPC	SoL Linux
Asianux Server	Lycoris Desktop/LX	StartCom Linux
BigBlock	Mac OS X	Sun Java Desktop System
BSDI	Mandrake Linux	Sun Solaris
Caixa Magica	Mandrake Linux Corporate Server	SuSE Linux
Caldera OpenLinux	Mandriva Linux	SuSE OpenExchange Linux
Caldera OpenLinux eServer	Mepis Linux	SuSE SLES Linux
cAos Linux	MSC Linux	Tao Linux
Cendio LBS Linux	NeoShine Linux	Tawie Server Linux
CentOS Linux	NetBSD	ThizLinux Desktop
Cobalt Linux	OpenBSD	ThizServer
Coherent Technology Linux	OpenDarwin	TinySofa Linux
Conectiva Linux	openmamba Linux	Trustix
Corel Linux	OpenNA Linux	Trustix SE

Corvus Latinux	Oracle Enterprise Linux	TurboLinux
Cygwin	Oracle VM	Ubuntu Linux
Darwin	pclinuxos Linux	United Linux
Debian Linux	Playstation Linux	Ute Linux
DEC/Compaq OSF/1	Redhat Enterprise Linux	White Dwarf Linux
DragonFly BSD	Redhat Linux	Whitebox Linux
Endian Firewall Linux	Redhat Linux Desktop	Windows
FreeBSD	SCI Linux	X/OS Linux
Generic Linux	Scientific Linux	Xandros Linux
Gentoo Linux	SCO OpenServer	Yellow Dog Linux
Gralinux	SCO UnixWare	Yoper Linux
Haansoft Linux	Secure Linux	IBM AIX
HP/UX	SGI Irix	

Para el funcionamiento correcto de Webmin en su versión 1.480 éste contiene los módulos estándar de Perl que se describen a continuación:

Tabla C.2: Módulos Perl estándar utilizados por Webmin.

Nombre	Paquete a Descargar	Descripción
ADSL Client	adsl-client.wbm.gz	Establece un cliente PPP con el paquete RP-PPPoE
Apache Webserver	apache.wbm.gz	Configura todos las directivas y dependencias de Apache
BIND DNS Server	bind8.wbm.gz	Crea y edita dominios, registros DNS opciones BIND y vistas
BSD Firewall	ipfw.wbm.gz	Configura en firewall BSD utilizando IPFW mediante la creación y edición de reglas
Backup Configuration Files	backup-config.wbm.gz	Establece respaldos manuales o programados, así como la recuperación de archivos de configuración administrados por los módulos Webmin
Bacula Backup System	bacula-backup.wbm.gz	Configura Bacula para llevar a cabo respaldos manualmente o programados para uno o varios archivos
Bandwidth Monitoring	bandwidth.wbm.gz	Visualiza reportes del ancho de banda utilizado por host, puerto, protocolo y tiempo en un sistema operativo Linux
Bootup and Shutdown	init.wbm.gz	Ejecuta scripts que deben de ser ejecutados al momento de botear desde /etc/init.d o /etc/rc.local
CD Burner	burner.wbm.gz	Quema CDs desde imágenes ISO o directorios seleccionados
CVS Server	pserver.wbm.gz	Configura un sistema remoto accesible CVS en modo servidor, administra usuarios y permite manejar el repositorio
Change Language and Theme	change-user.wbm.gz	Permite al usuario de Webmin cambiar el idioma, tema y su password
Change Passwords	passwd.wbm.gz	Cambia el password de cualquier usuario del sistema
Command Shell	shell.wbm.gz	Ejecuta comandos del shell y visualiza su salida
Configuration Engine	cfengine.wbm.gz	Configura el programa CFengine para el chequeo y mantenimiento de varias opciones administrativas del sistema

Custom Commands	custom.wbm.gz	Crea botones para ejecutar comandos usados recientemente o editar archivos en el sistema
DHCP Server	dhcpd.wbm.gz	Maneja redes compartidas, subredes, hosts y grupos para ISC DHCPD
Disk Quotas	quota.wbm.gz	Ejecuta y edita usuarios y grupos para administrar el espacio en disco para archivos locales del sistema
Disk and Network Filesystems	mount.wbm.gz	Monta archivos de sistema y archivo swap usualmente configurados en /etc/fstab o /etc/vfstab
Dovecot IMAP/POP3	Server dovecot.wbm.gz	Configura el Dovecot IMAP y servidor de correo pop3
File Manager	file.wbm.gz	Permite ver, editar, cambiar permisos sobre los archivos y directorios en el sistema operativo anfitrión
Filesystem Backup	fsdump.wbm.gz	Permite el respaldo y recuperación de archivos del sistema utilizando el concepto de dump y restore, así como sus comandos propios
GRUB Boot Loader	grub.wbm.gz	Configura el booteo Linux grub que permite la selección de varios sistemas operativos y kernels a la vez
LDAP Client	ldap-client.wbm.gz	Configura el sistema como un cliente LDAP para usuarios y grupos
LDAP Server	ldap-server.wbm.gz	Maneja y mantiene el servidor OpenLdapy objetos en su base de datos
LDAP Users and Groups	ldap-useradmin.wbm.gz	Permite el manejo de usuarios y grupos almacenados en la base LDAP
Log File Rotation	logrotate.wbm.gz	Configura la rotación automática de archivos .log de Apache, Squid, Syslog y otros
Webmin Configuration	webmin.wbm.gz	Lleva a cano la configuración del Webmin, así como sus host, SSL, módulos instalados y temas
Webmin Servers Index	servers.wbm.gz	Despliega en forma indexada servidores para fácil conexión
Webmin Users	acl.wbm.gz	Crea usuarios Webmin y configura que módulos están disponibles, así como su acceso

Los módulos anteriormente mencionados son los más utilizados por Webmin y por la administración de servidores Linux pero existen más módulos con funciones específicas que se pueden descargar vía internet.

Por la gran flexibilidad del sistema Linux se puede configurar todo el sistema en general a través de la aplicación Webmin.

OBJETIVO

Al concluir el presente procedimiento el lector será capaz de instalar Webmin en cualquier servidor Linux, comprenderá la configuración de este servicio y será capaz de dejarlo funcionando para su posterior utilización.

ALCANCE

Cualquier usuario y todo tipo de servidores Linux

PROCEDIMIENTO

Requisitos previos para ejecutar el procedimiento.

Como se mencionó anteriormente, los requisitos previos son contar con acceso a un servidor con sistema operativo Linux el cual debe estar soportado dentro de los sistemas compatibles que se mencionaron anteriormente.

Otro requerimiento para llevar a cabo el procedimiento es contar con permisos de ejecución sobre archivos del sistema en el servidor donde se va a instalar Webmin (poseer permisos de sudo o la contraseña para convertirse en usuario root).

A continuación se describirá el procedimiento para instalar el servidor Webmin en una máquina Linux con Ubuntu 8.10.

Por último en el servidor donde se va a instalar Webmin debe tener instalado Perl versión 5 o superior.

Como primer paso es necesario descargar el paquete Webmin compatible para la distribución en la que se va a instalar, para la mayoría de las distribuciones se utiliza el paquete webmin-1.480.tar.gz y se puede descargar de la página http://www.webmin.com/download.html tal y como se muestra a continuación:

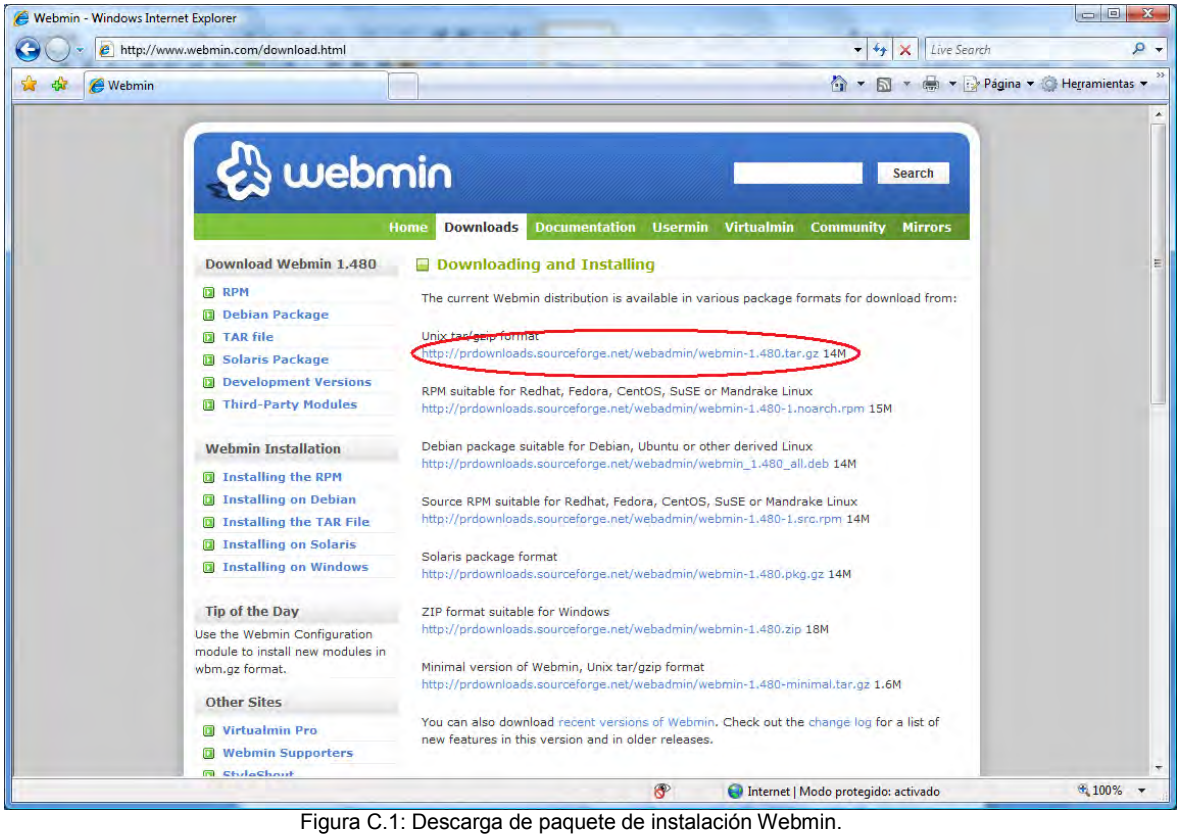

Figura C.1: Descarga de paquete de instalación Webmin.

Una vez descargado el paquete en el servidor donde se desea instalar Webmin se necesita descomprimir éste archivo, y para llevar a cabo esta instrucción es necesario convertirse en súper usuario mediante la instrucción sudo su – o su – si se cuenta con la contraseña de root. A continuación se muestra un ejemplo:

E ingresar la contraseña de root.

Posteriormente dirigirse al lugar donde se guardo el archivo mediante el comando cd /ruta donde está el archivo

Una vez ahí se podrá observar el archivo comprimido de Webmin mediante el comando ls –ltr el cual lista todos los archivos en esa carpeta, poniendo en último lugar la última modificación.

Para descomprimir el archivo webmin-1.480.tar.gz ya como root es necesario introducir el siguiente comando:

tar xzvf webmin-1.480.tar.gz el cual descomprimirá el archivo y creará una carpeta con el nombre webmin-1.480.tar.gz como se muestra a continuación:

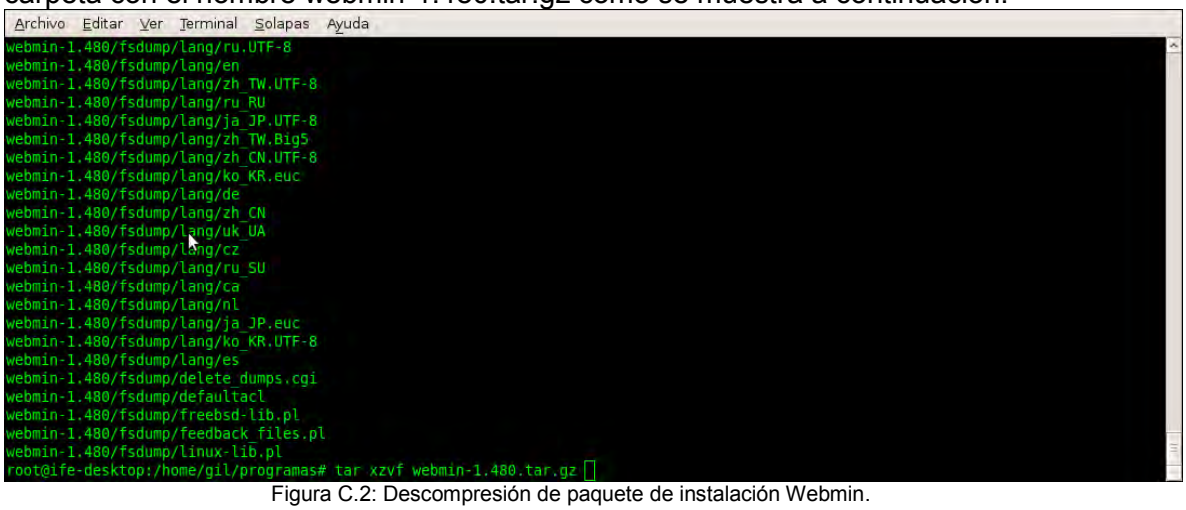

Figura C.2: Descompresión de paquete de instalación Webmin.

A continuación es necesario meterse en la carpeta recién creada (webmin-1.480) mediante la instrucción cd webmin-1.480 tal y como se muestra a continuación:

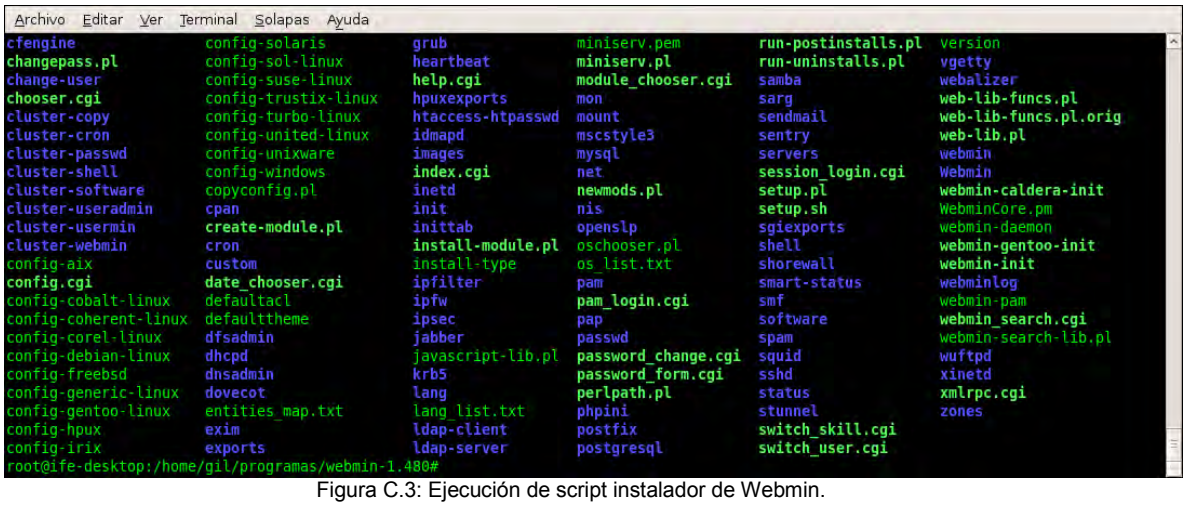

Figura C.3: Ejecución de script instalador de Webmin.

Una vez dentro de esta carpeta y como usuario root se debe ejecutar el comando “setup.sh” el cual llevará paso a paso la instalación de Webmin. Para ejecutar el comando de instalación se debe ejecutar la siguiente instrucción:
./setup.sh

Con lo cual el proceso de instalación solicitará unos parámetros para la instalación correcta de Webmin, tal y como se muestra a continuación:

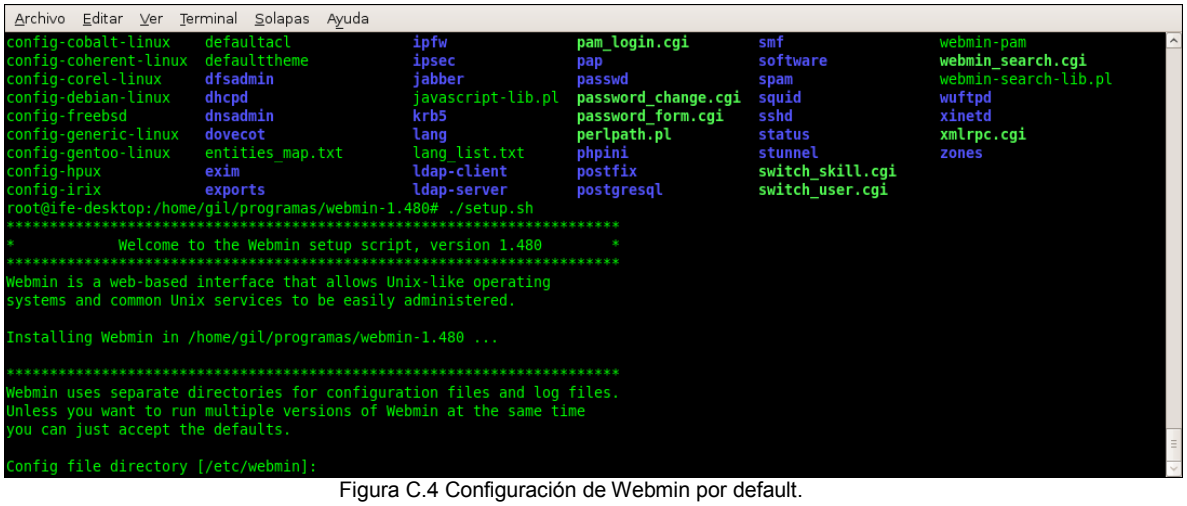

Figura C.4 Configuración de Webmin por default.

Y como se muestra en la imagen anterior, el instalador de Webmin solicita unos parámetros que son propios del correcto funcionamiento de Webmin, se pueden modificar estos parámetros o bien se pueden dejar por default. Para dejar los parámetros por default únicamente hay que dar “enter” en la función y con eso se dejan los parámetros básicos de Webmin.

Para esta instalación se dejaron los parámetros por default, como por ejemplo el archivo de configuración se deja en /etc/webmin y consulta la ejecución de Webmin al iniciar el servidor tal y como se muestra a continuación:

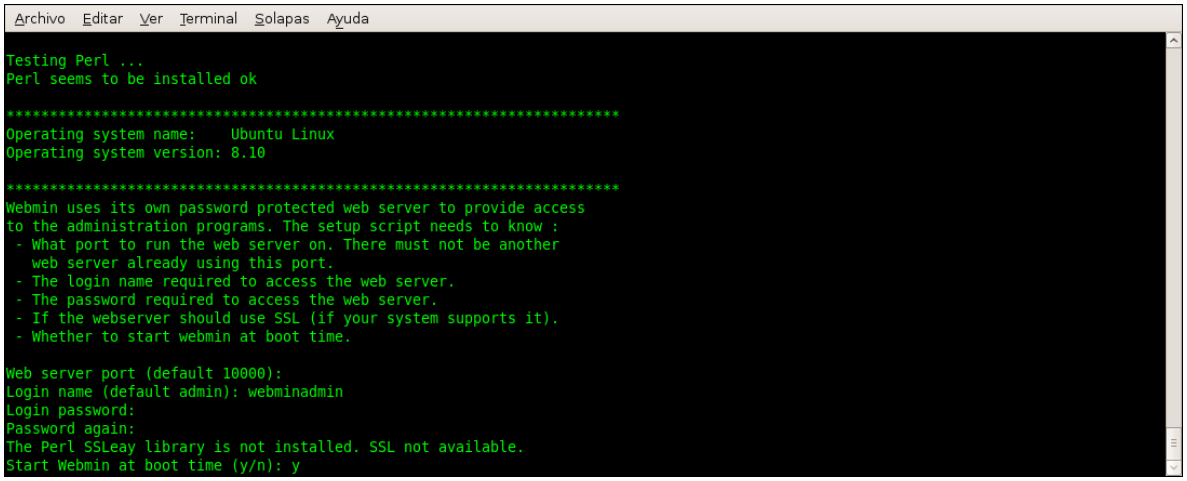

Figura C.5: Configuración de Webmin al iniciar servidor.

Para este caso es necesario introducir una de las opciones que se muestran (y para sí o n para no).

Un parámetro que vale la pena mencionar de manera aislada es el puerto por donde se va a comunicar Webmin con el navegador del usuario que consulta esta aplicación.

Por default Webmin escucha y envía información por el puerto 10000, en caso de que se desee seleccionar otro puerto únicamente hay que introducir el puerto deseado en el momento en que el instalador lo solicite como e s el caso que se muestra a continuación:

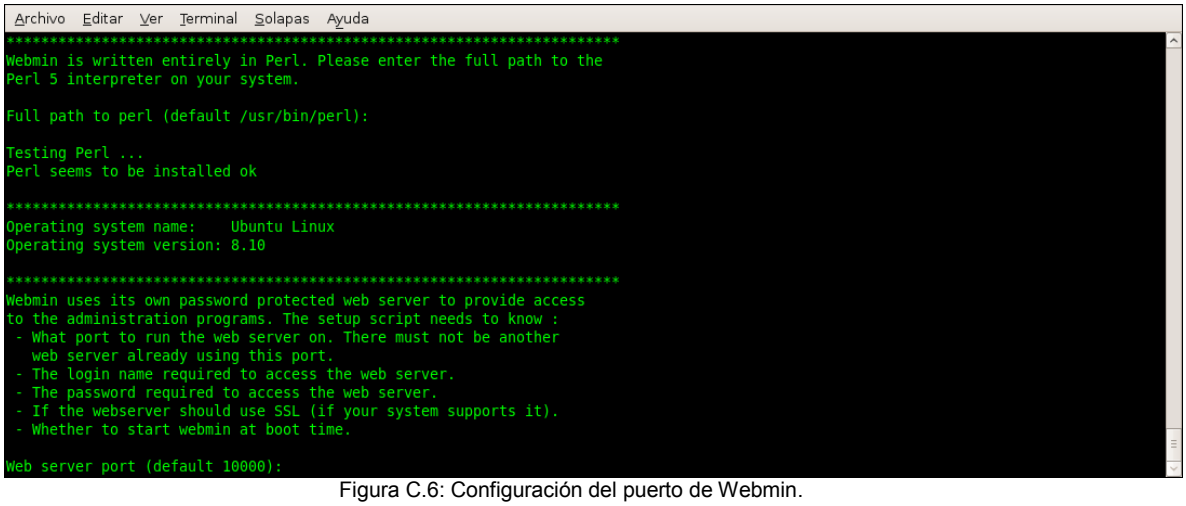

Figura C.6: Configuración del puerto de Webmin.

Si se da “enter" utilizará el puerto default (10000) y si se especifica otro puerto Webmin estará escuchando en el puerto especificado. Para esta configuración se dejó el puerto default.

Una vez introducidos todos los parámetros solicitados por el instalador de Webmin solicita un nombre de usuario y posteriormente una contraseña, una vez ratificada la contraseña Webmin utilizará este usuario especificado y esa contraseña para permitir el logueo a la aplicación y su utilización completa.
El asistente de instalación terminará mostrando un mensaje como el siguiente en la pantalla de salida:

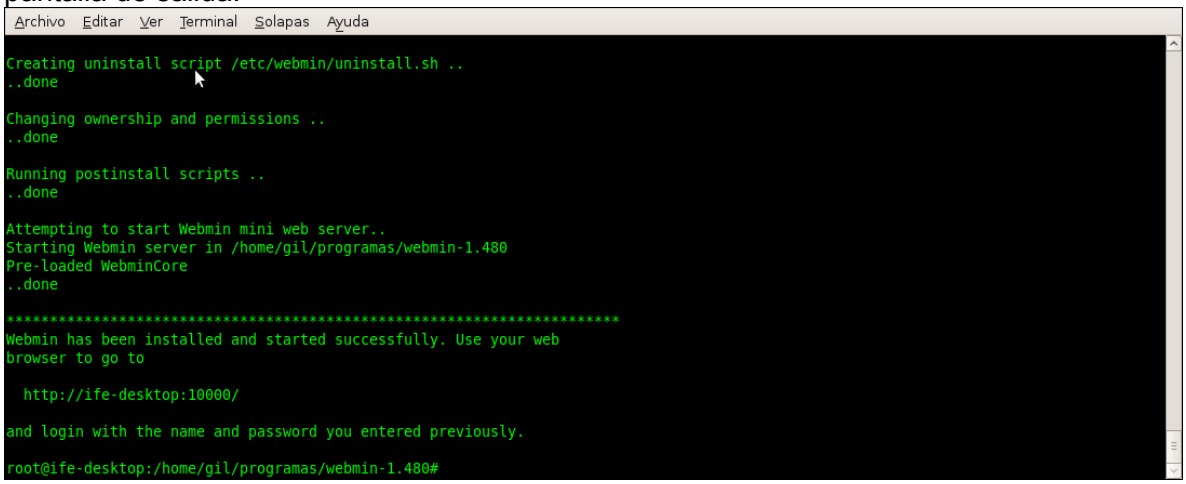

Figura C.7: Finalización de instalación de Webmin.

Con lo cual ratifica el final de la instalación y comenta que para ingresar a la aplicación debe de ser con el nombre de usuario y contraseña proporcionados anteriormente.

Para poder ingresar a Webmin y poder configurar un servidor Linux vía web únicamente es necesario introducir la dirección del servidor que tiene instalado así como el puerto por donde escucha y envía información Webmin.

Para este ejemplo se accederá al servidor 32 de pruebas con ip 10.49.129.16.

Para acceder a Webmin en este servidor introducir en la barra de navegación la dirección como se muestra a continuación:

https://10.49.129.16:10000/

Puede presentar una pantalla de advertencia como la siguiente:

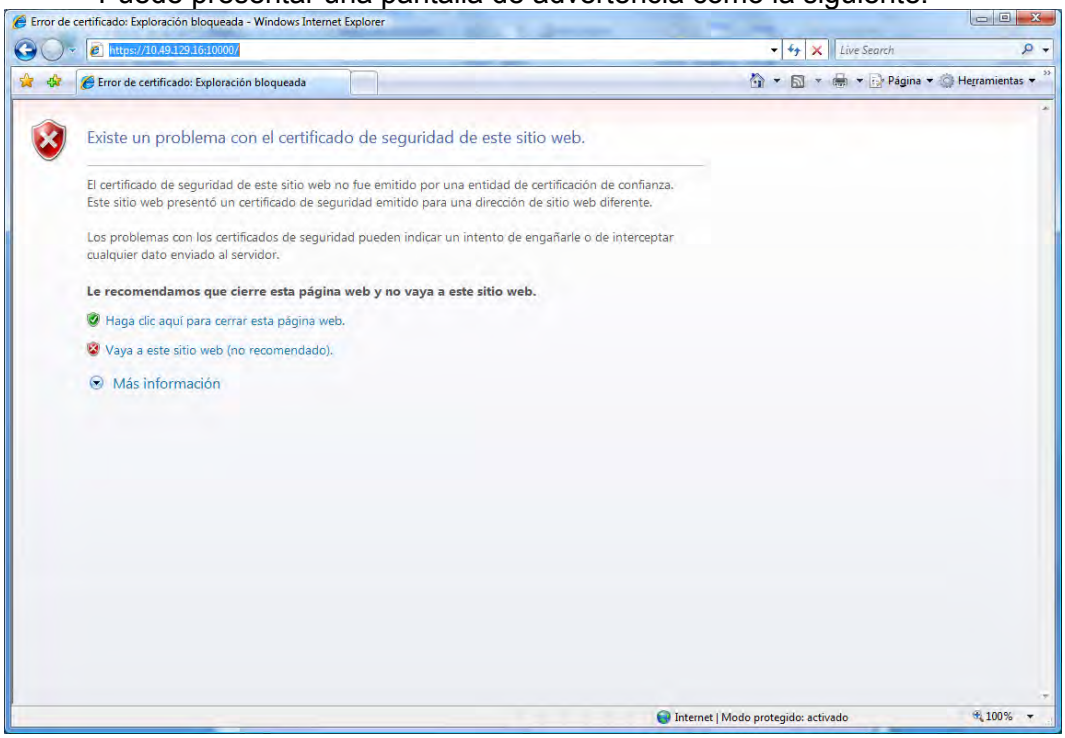

Figura C.8: Primer ingreso a Webmin.

Pero con hacer clic en la opción (Vaya a este sitio Web) permite el acceso a la aplicación Webmin tal y como se muestra a continuación:

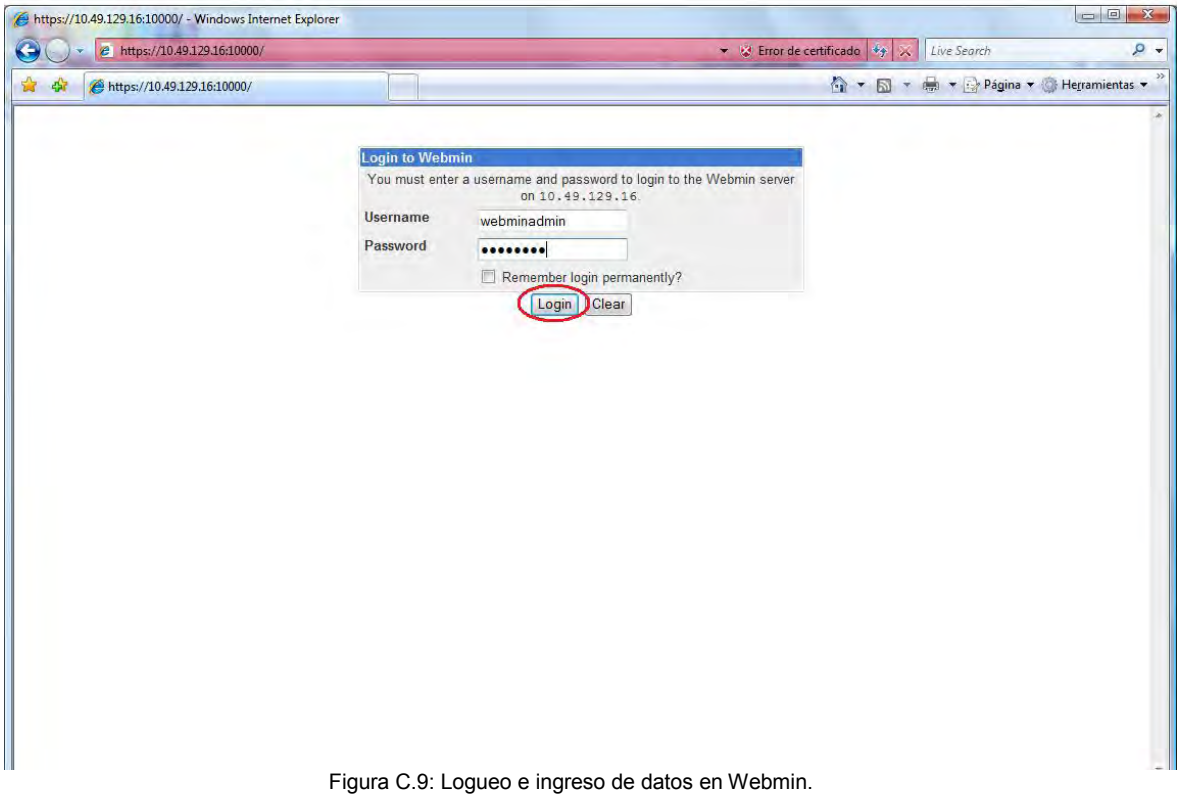

Figura C.9: Logueo e ingreso de datos en Webmin.

Y una vez introducidos el usuario y contraseña, hacer clic en “Login” para ingresar a la aplicación.

La cual mostrará una pantalla como la siguiente:

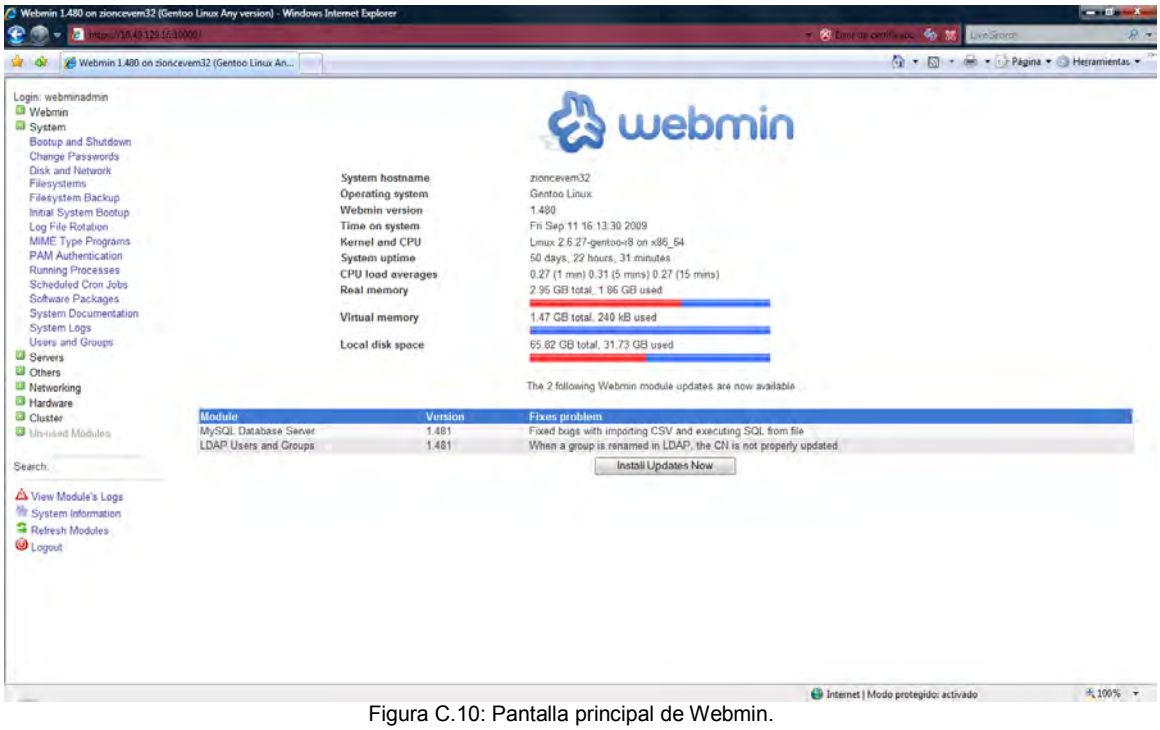

Figura C.10: Pantalla principal de Webmin.

En el panel de navegación izquierdo vienen las opciones para poder configurar el servidor, por poner un ejemplo se ejecutará un comando en el Shell de ese servidor.

Hacer clic en el panel izquierdo en la sección “Others” y posteriormente en la opción “Command Shell” con lo cual se ejecutará un comando del Shell en el servidor analizado y se mostrará en el navegador Web tal y como se muestra a continuación:

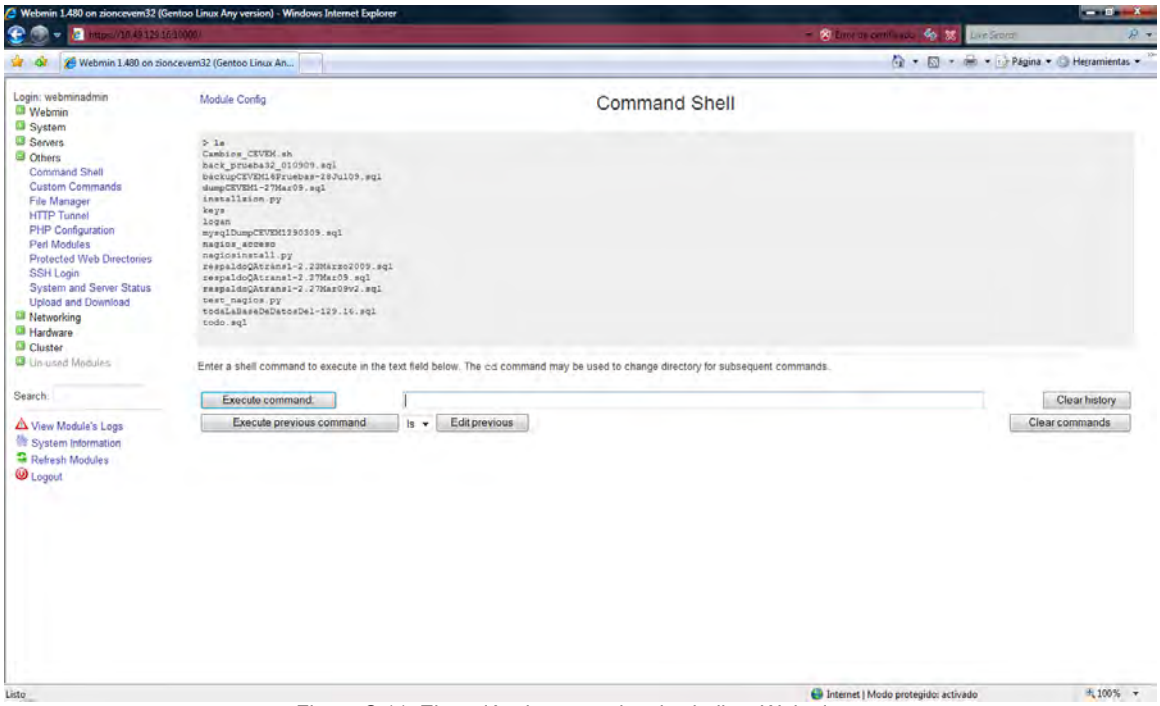

Figura C.11: Ejecución de comandos de shell en Webmin.

Otro ejemplo es la configuración de la rotación automática de los logs de diversos programas conocido como log rótate, el cual se encuentra en Webmin desde la sección "System" y bajo la opción "Log file rotation" y muestra la siguiente pantalla:

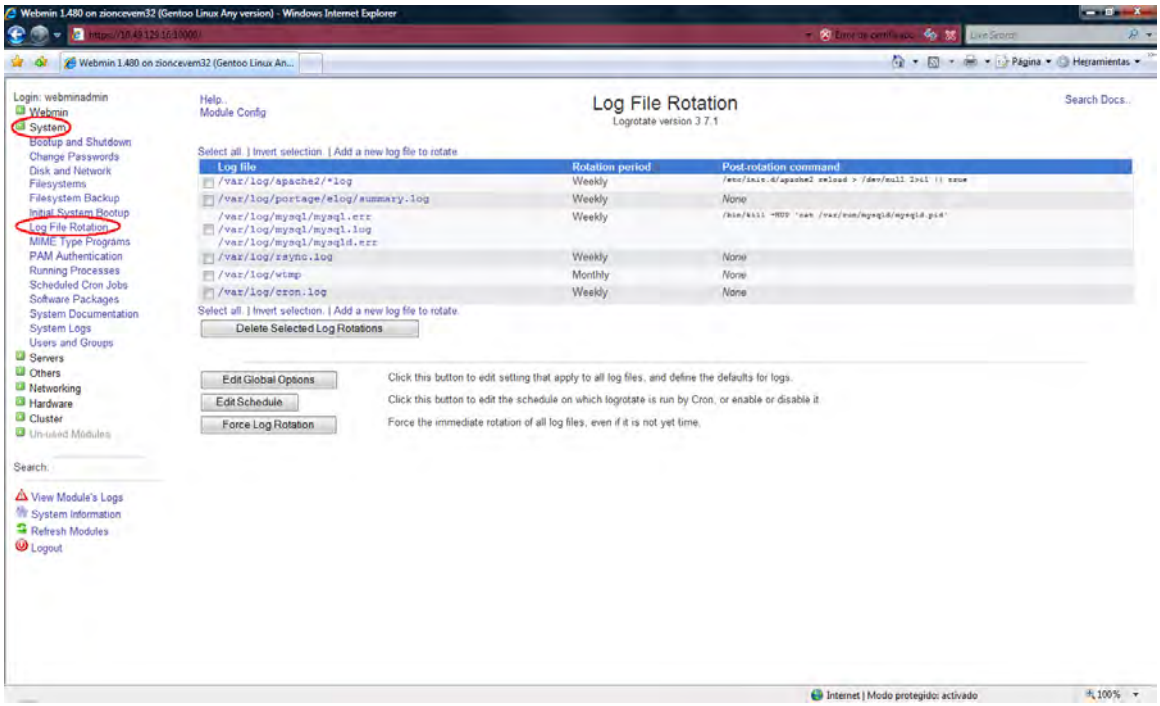

Figura C.12: Configuración de log rotate en Webmin.

Glosario de términos.

Algoritmo: Es la especificación detallada y libre de ambigüedad, de un proceso; es decir, un conjunto de pasos que hay que seguir para llegar a cierto fin medible o comprobable**.**

Archivos ejecutables: En informática, un ejecutable o archivo ejecutable, es un archivo binario cuyo contenido se interpreta por la maquina como un programa. Generalmente, contiene instrucciones en código máquina de un procesador en concreto, pero también puede contener bytecode que requiera un intérprete para ejecutarlo. Además suele contener llamadas a funciones específicas de un sistema operativo (llamadas al sistema).

AWK: AWK es un lenguaje de programación diseñado para procesar datos basados en texto, ya sean archivos o flujos de datos.

Benchmark: El benchmark es una técnica utilizada para medir el rendimiento de un sistema o componente de un sistema, frecuentemente en comparación con el cual se refiere específicamente a la acción de ejecutar un benchmark.

Bit: Es la cantidad mínima de información que un circuito electrónico puede representar, y es la base de operación binaria de las computadoras.

BIOS: El Sistema Básico de Entrada/Salida o BIOS (Basic Input-Output System) es un código de software que localiza y carga el sistema operativo en la RAM; es un software muy básico instalado en la placa base que permite que ésta cumpla su cometido.

Bug: Un defecto de software (computer bug en inglés), es el resultado de un fallo o deficiencia durante el proceso de creación de programas de computadora (software).

Byte: Es un conjunto de 8 bits, con esta estructura se pueden representar hasta 256 cosas diferentes.

Código fuente: El código fuente de un programa informático (o software) es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa.

Comandos: Es una instrucción o mandato que el usuario proporciona a un sistema informático, desde la línea de comandos (como una shell) o desde una llamada de programación.

Copias de respaldo: Un proceso que se utiliza para salvar toda la información de la que dispone en el PC hasta este momento.

Crackers: También conocidos como "crackers" muestran sus habilidades en informática rompiendo sistemas de seguridad de computadoras, colapsando servidores, entrando a zonas restringidas, infectando redes o apoderándose de ellas, entre otras muchas cosas utilizando sus destrezas en métodos hacking.

Dirección IP: Una dirección IP es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red del protocolo TCP/IP.

DNS: Un servidor DNS permite conectarse con la máquina sin necesidad de usar su dirección IP; basta con ingresar el dominio para que el servidor DNS resuelva y establezca una conexión.

DoS: (Denied of Service): es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos.

Ensamblador: El término ensamblador (del inglés assembler) se refiere a un tipo de programa informático que se encarga de traducir un archivo fuente escrito en un lenguaje ensamblador, a un fichero objeto que contiene código máquina, ejecutable directamente por la máquina para la que se ha generado.

Editor: Programa de uso general por medio del cual se crean textos o programas que le son alimentados a una computadora para su posterior procesamiento.

GNU: GNU is Not Unix, movimiento de software encabezado por Richard Stallman.

GUI: (graphical user interface) utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

HACKER SOMBRERO BLANCO: (White Hat) Persona con habilidades en la programación, electrónica, este tipo de hacker puede conocer mucho sobre seguridad pero usa sus conocimientos para ayudar y nunca para destruir las cosas ajenas.

HACKER SOMBRERO NEGRO: (Black Hat) Este es un hacker malicioso que usa sus conocimientos para hacer el mal, normalmente nos referimos a estos hacker como cracker ya que se introducen o atacan redes con el único fin de romper y destruir.

HTML: Siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web.

JavaScript: Es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

IPSec: IPsec (abreviatura de Internet Protocol security) es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet (IP) autenticando y/o cifrando cada paquete IP en un flujo de datos.

IPtables: Es un framework disponible en el núcleo Linux que permite interceptar y manipular paquetes de red.

IRC: Es un protocolo de comunicación en tiempo real basado en texto, que permite debates en grupo o entre dos personas.

Keyloger: Es una herramienta de diagnóstico utilizada en el desarrollo de software que se encarga de registrar las pulsaciones que se realizan sobre el teclado.

Lilo: Lilo ("Linux Loader") es un gestor de arranque de Linux que permite iniciar este sistema operativo junto con otras plataformas en el mismo servidor.

Lisp: Lisp es el segundo lenguaje de programación más antiguo (después de Fortran) de alto nivel.

Lenguaje de máquina: Sistema de códigos directamente interpretable por un circuito microprogramable, como el microprocesador de una computadora o el microcontrolador de un autómata (un PLC).

Loader: Parte del sistema que tiene como propósito colocar en la memoria información codificada en lenguaje maquina, para que entonces la computadora pueda procesarla.

MBR: Es un sector en un disco duro, disquete, o cualquier otro dispositivo de almacenamiento de datos que contiene código de arranque.

Md5: En criptografía, MD5 (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de 128 bits.

NTFS: NTFS (New Technology File System) es un sistema de archivos diseñado específicamente para Windows NT.

Proceso: Conjunto de instrucciones escritas en lenguaje de máquina que ya están listas para ser ejecutadas por el procesador.

Programa fuente: Conjunto de instrucciones escritas en algún lenguaje de programación y sirven para modelar o solucionar un problema.

Programa objeto: Conjunto de instrucciones escritas en lenguaje maquina.

Proxy: El término proxy hace referencia a un programa o dispositivo que realiza una acción en representación de otro.

Registro del Sistema: Es una base de datos que almacena las configuraciones y opciones del sistema operativo Microsoft Windows en sus versiones de 32 bits, 64 bits y Windows Mobile.

Sed: Sed es un editor de flujo, una potente herramienta de tratamiento de texto para el sistema operativo Unix que acepta como entrada un archivo, lo lee y modifica línea a línea mostrando el resultado en pantalla.

Shell: Es un programa informático que actúa como interfaz de usuario para comunicar al usuario con el sistema operativo mediante una ventana que espera órdenes escritas por el usuario en el teclado.

Sniffers: Es un programa de captura de las tramas de red.

Socket: Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos.

Spam: Se llama spam, correo basura o sms basura a los mensajes no solicitados, habitualmente de tipo publicitario, enviados en grandes cantidades (incluso masivas) que perjudican de alguna o varias maneras al receptor.

Topología de red: Se define como la cadena de comunicación que los nodos conforman una red usan para comunicarse.

FUENTES DE INFORMACIÓN

BIBLIOGRAFÍA

"Análisis forense" Rubén Aquino Luna, Jesús Ramón Jiménez Rojas, Alejandro Núñez Sandoval Ed. DGSCA-UNAM 2006.

"Escribiendo Código Seguro" Alejandro Núñez Sandoval, Israel Becerril Sierra, Ed. DGSCA-UNAM 2006.

"*Ingeniería de software en entornos SL*" Marc Gilbert Ginesta, Álvaro Peña González Ed. UOC

"Apunte de Ingeniería de Software" Miguel Yepes Moyano Ed. Universidad de Córdoba.

"Introducción a la computación y a la programación estructurada" Guillermos Levigne Ed. McGrawHill.

"Seguridad en los sistemas de computo" Isaias Calderón www.lci.ulsa.mx/Material/pdf/seguridad_2002.pdf Ed. Universidad La Salle

"Análisis forense" Rubén Aquino Luna, Jesús Ramón Jiménez Rojas, Alejandro Núñez Sandoval Ed. DGSCA-UNAM 2006.

"Windows Server 2003 Administrator's Companion" Russel, Crawford, Gerend. Ed Microsoft Press.

MESOGRAFÍA

"Guía de operaciones de seguridad para Windows 2000 Server" http://www.microsoft.com/spain/technet/seguridad/2000server/chapters/ch01secops.aspx

"CSI/FBI Computer Crime and Security Survey" http://www.hispasec.com/corporate/noticias/110

"Programas troyanos (Caballos de Troya) " http://www.viruslist.com/sp/virusesdescribed?chapter=152540521

"¿Qué es el malware?" http://www.tech-faq.com/lang/es/malware.shtml

"Qué es un exploit" http://www.elhacker.net/exploits/

"Gusanos Informáticos" Arnoldo Moreno Pérez http://microasist.com.mx/noticias/nv/ampnv300902.shtml

"Crypto-Gram Newsletter" Bruce Schneier http://www.schneier.com/crypto-gram-0009.html#1

"Common Vulnerabilities and Exposures" http://cve.mitre.org/

"Microsoft Security Tool Kit:"
http://www.microsoft.com/technet/security/tools/stkintro.mspx

"Strategic Technology Protection Program"
http://www.microsoft.com/security/mstpp.asp

"Microsoft Security Notification Service:"
http://www.microsoft.com/technet/security/bulletin/notify.mspx

http://www.microsoft.com/spain/technet/seguridad/2000server/Job_aids/JA2-Top_Security_Blunders.aspx

www.internet-solutions.com.co/**encase**.php

"El portapapeles de windows una amenaza a nuestra privacidad"
http://www.wilkinsonpc.com.co/free/articulos/portapapeles-de-windows.html

"Win32::Clipboard – Interaction with the Windows clipboard"
http://www.xav.com/perl/site/lib/Win32/Clipboard.html

"CPAN" http://search.cpan.org/~jdb/libwin32-0.28/OLE/lib/Win32/OLE.pm

"*Netfilter"* http://www.netfilter.org/

"PromiscDetect" http://ntsecurity.nu/toolbox/promiscdetect/

http://www.diamondcs.com.au/index.php?page=console

http://www.diamondcs.com.au/index.php?page=console

"Iptables Manual Práctico" http://www.pello.info/filez/firewall/iptables.html

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/es/library/ServerHelp/820b0e5b-0371-404d-8a22-a255273d3289.mspx

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/es/library/ServerHelp/1d284efa-9d11-46c2-a8ef-87b297c68d17.mspx

http://es.tldp.org/Manuales-LuCAS/blfs-es/blfs-es-5.0/postlfs/firewall.html#postlfs-security-fw-kernel

"Monitoreando tu filesystem en linux – Integrit mini-howto"
http://www.malditainternet.com/integrit

"Sistema Operativo Unix" http://www.monografias.com/trabajos63/sistema-operativo-unix/sistema-operativo-unix.shtml

"Recuperación ante incidentes de seguridad" http://www.dimensis.com/num-128-recuperacion-ante-incidentes-de-seguridad

"Symantec"
http://shop.symantecstore.com/DRHM/servlet/ControllerServlet?Action=DisplayProductDetailsPage&SiteID=symanlam&Locale=en_PR&Env=BASE&productID=41496800

"Npot" http://www.foundstone.com/us/resources-free-tools.asp
"Open Port Check Tool" http://www.canyouseeme.org/
"Promiscdetect" http://www.ntsecurity.nu/toolbox/promiscdetect/
"Process Explorer" Mark Russinovich http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx

"PsList" Mark Russinovich http://technet.microsoft.com/en-us/sysinternals/bb896682.aspx

"AutoRuns for Windows" Mark Russinovich &Bryce Cogswell http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx

"PsInfo" Mark Russinovich http://technet.microsoft.com/en-us/sysinternals/bb897550.aspx

"Uptime" http://support.microsoft.com/kb/232243

"EnCase Edición Forense" http://www.internet-solutions.com.co/encase.php

"*SleutKit*" http://www.sleuthkit.org/

"*The Coroner's Toolkit (TCT)*" http://www.porcupine.org/forensics/tct.html

"Perl" http://www.perl.org/

"Iptables Manual Práctico" http://www.pello.info/filez/firewall/iptables.html

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/es/library/ServerHelp/820b0e5b-0371-404d-8a22-a255273d3289.mspx

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/es/library/ServerHelp/1d284efa-9d11-46c2-a8ef-87b297c68d17.mspx

http://es.tldp.org/Manuales-LuCAS/blfs-es/blfs-es-5.0/postlfs/firewall.html#postlfs-security-fw-kernel

http://www.malware.unam.mx/consultas.dsc?mal_id=374

http://www.malware.unam.mx/consultas.dsc?mal_id=314

http://www.malware.unam.mx/consultas.dsc?mal_id=476