**Introducción a la forénsica en sistemas Linux**

En las infraestructuras actuales basadas mayoritariamente en servidores Linux, cuando se produce un incidente de seguridad, éste debe de ser tratado de una forma adecuada, estudiando desde el comienzo todas las implicaciones que puede provocar en los activos de una compañía, en su reputación e incluso en su continuidad de negocio. En esta charla se muestra un incidente real de seguridad, como se planificó la respuesta a incidentes, como se actuó, investigó su alcance y se programó una herramienta personalizada para la obtención y análisis de evidencias.

****

**Herramienta de extracción forénsica en sistemas Linux *forensic\_get.sh***

Durante la realización del último proyecto de respuesta ante un incidente relativo al robo de datos de tarjetas de crédito en un sitio web comercial, decido programar mi propia herramienta de extracción de datos en vivo para servidores Linux. Si bien ya existen otras herramientas comerciales y abiertas, este script obtendrá unos datos más dirigidos a facilitar el posterior análisis de información en un caso como éste. Los servidores a analizar estaban virtualizados en VMware ESXi, por lo que la auditoría forénsica comenzó por la creación de la infraestructura de virtualización en mi laboratorio, la replicación de las máquinas virtuales, la creación de snapshots o imágenes de las máquinas a analizar y la modificación de las claves del usuario *root* para su arranque y puesta en funcionamiento.

Tal y como dicen los comentarios al principio del script ***forensic\_get.sh*** programado para Shell de tipo Bash, la realización de una auditoría forénsica en sistemas en vivo o live systems no resulta en la mayoría de los casos la mejor opción, pero tras la clonación del sistema puede resultar ideal como una solución híbrida para analizar un sistema en funcionamiento.

Este script realiza la extracción de todo tipo de información relevante durante una auditoría forénsica para sistemas Linux, Además se especializa en la búsqueda de malware específico para servidores Web (shells), de bases de datos MySQL, etc.

Al determinar que una auditoría de este tipo resulta un trabajo muy específico y arduo debido a la gran cantidad de información que debe de analizarse, decidí programar una herramienta para extraer aquella información relevante de cada máquina virtual hacia una unidad extraíble, de forma que pueda ser analizada posteriormente por un auditor especializado.

**Requisitos de la herramienta**

Este script ha sido programado para ejecutarse en un sistema virtual o físico basado en Debian Linux, Ubuntu Linux o cualquier derivado.

Se ejecutará desde un pendrive con gran capacidad de almacenamiento (mínimo 32GB) formateado con sistema de archivos EXT4 nativo de Linux. El script se ejecutará desde el directorio raíz del pendrive por lo que tiene que ser ejecutable (*chmod 755 forensic\_get.sh*).

En este pendrive debemos de haber copiado una serie de programas que sirven para realizar estudios de seguridad y forénsica detallados en la sección “Otro software a instalar”.

El sistema a auditar podrá tener conexión a Internet, aunque no es indispensable.

**Información y evidencias extraídas por el script**

Esta herramienta llamada “*forensic\_get.sh*” selecciona y extrae la información referida en esta sección y debidamente formateada para el análisis forénsico posterior (se aportan los ficheros generados en cada servidor en los directorios con las fechas de extracción en un archivo comprimido con el password definido en las variables del mismo script:

1. **Información del sistema auditado** (fecha, hora, zona geográfica, nombre de host, usuario utilizado en la auditoría, permisos del usuario utilizado, tiempo de funcionamiento del sistema, fecha introducida por el auditor como posible inicio del compromiso, fecha final del posible compromiso para el análisis, destino de la extracción, número de serie del dispositivo utilizado para la extracción, directorios relacionados con el análisis, usuario del servidor web, sistema operativo del servidor y kernel, conexión a internet del servidor).

**Ficheros:** *index.txt*

1. **Histórico de comandos ejecutados en el espacio de cada usuario del sistema** (historial de ejecución de cada usuario con su fecha y hora de ejecución).

**Ficheros:** *history\_root.txt / history\_nombredeusuario\_orig.txt*

1. **Variables del sistema durante la ejecución** (análisis de las variables del sistema en memoria durante la auditoría, para buscar indicadores de compromiso en aplicaciones corriendo en memoria).

**Ficheros:** *env.txt*

1. **Investigación de scripts de inicio de sesión de usuario** (búsqueda de scripts o aplicaciones que puedan arrancar de forma automática durante el inicio de sesiones de usuarios que indique “persistencia” del malware).

**Ficheros:** *shell.txt*

1. **Investigación de scripts en el arranque del sistema** (búsqueda de scripts o aplicaciones que puedan arrancar de forma automática durante el inicio del sistema operativo que puedan indicar “persistencia” del malware. Se analizan los servicios de Linux “init”, “Sys-V” y “systemd”, además de analizar el actual runlevel de ejecución).

**Ficheros:** rc*sysinit.txt / initd.txt / system.txt*

1. **Análisis de trabajos programados mediante el servicio cron del sistema** (Otra forma habitual de obtención de persistencia en el sistema es el uso del servicio “cron” en Linux).

**Ficheros:** *cron.txt*

1. **Análisis de procesos en ejecución** (al realizarse la auditoría en un sistema “vivo” podemos estudiar aquellos procesos que estén en ejecución durante la misma para buscar procesos extraños, poco habituales o modificados. Sobre cada uno de estos procesos se analiza además la lista de ficheros y sockets relacionados y su historial de memoria y tiempo de procesamiento para ver comportamientos anómalos. Se vigilan especialmente procesos relacionados con usuarios y administradores).

**Ficheros:** *proc.txt*

1. **Estudio de drivers, controladores o módulos del sistema** (aunque no es algo demasiado habitual, el malware de tipo rootkit se suele ocultar tras módulos del kernel de Linux. Por ello analizaremos los módulos cargados durante la ejecución de la auditoría para buscar módulos desconocidos o extraños. Además se analizarán dispositivos USB o PCI conectados para relacionarlos con los módulos del kernel en búsqueda de inconcordancias. Se vigilarán errores en los módulos durante el arranque del sistema mediante mensajes del kernel “dmesg”).

**Ficheros:** *modules.txt*

1. **Usuarios presentes en el sistema y grupos de pertenencia** (en este tipo de auditorías forénsicas resulta siempre necesario analizar cada uno de los usuarios presentes y con que finalidad se han creado. Además se deben de analizar los permisos y pertenencia a grupos de cada uno de ellos en búsqueda de permisos excesivos. Principalmente aquellos usuarios que tengan algún tipo de login resultan sospechosos. Analizamos la fecha de modificación de los ficheros de configuración de usuarios).

**Ficheros:** *users.txt*

1. **Extracción de todo tipo de accesos o logins de usuarios en cada sistema** (se extraen uno por uno los login de cada usuario en el servidor, tanto los últimos como los históricos dentro del período a analizar. Esto es una labor lenta y ardua porque los ficheros históricos ya fueron archivados y comprimidos y hay que buscarlos y extraerlos dependiendo del sistema operativo y configuración de archivado. Se extraen de forma independiente los login del servicio remoto SSH con logins fallidos y exitosos).

**Ficheros:** *loginsnow.txt / loginsfail.txt / logins.txt / loginSSHok.txt / loginSSHfail.txt*

1. **Extracción de los intentos de login mediante el servicio fail2ban** (si este servicio se encuentra presente, se extraen los ficheros de reporte o logs del mismo para su posterior análisis de intentos de penetración).

**Ficheros:** *fail2banlogins.txt*

1. **Análisis de la configuración de red** (se investigan los ficheros de configuración de red, interfaces en modo promiscuo, servidores de nombres DNS y nombres de host asignados de forma manual, proxies, tablas de enrutamiento, etc. en busca de exfiltración de datos o ataques de tipo MiTM. Además se extraen las conexiones TCP y UDP abiertas en busca de conexiones e intentos de conexión a direcciones IP externas. Se revisa especialmente la configuración del servicio de conexión remota SSH).

**Ficheros:** *network.txt*

1. **Investigación del sistema de archivos y directorios** (esta investigación es una de las más importantes para la localización de trazas de archivos sospechosos o anómalos debido a sus características, análisis temporal, permisos, usuario y grupo de pertenencia, tipo de archivo, directorio de pertenencia, trazas en el contenido, etc). Debido a la importancia de este análisis lo desgranaremos en cada uno de los análisis realizados:
2. **Estudio sobre almacenamiento y puntos de** (en este punto se muestran los discos duros, particiones o unidades lógicas presentes en el sistema, además de aquellos montajes y puntos de montaje que se crean durante el arranque del sistema).

**Ficheros:** *fsinfo.txt*

1. **Generación de un fichero de tipo .csv (archivo separado por comas) de cada fichero del sistema de archivos** (cada archivo en una línea con los datos separados por comas: *Nombre, Fecha y hora de último acceso conocida en Linux como atime, fecha y hora de última modificación conocida en Linux como mtime, fecha y hora de último cambio de estado conocidad en Linux como mtime, Usuario o Identificador, Grupo o identificador, Permisos, tamaño en bytes, tipo de archivo analizado*. Esta estructura de .CSV se mantendrá en posteriores ficheros y directorios analizados).

**Ficheros:** *allfilestimeline.csv*

**Acerca de la marcas de tiempo en Linux:**

La marca de tiempo de acceso *atime* es la última vez que se leyó un archivo. Esto significa que alguien usó un programa para mostrar el contenido del archivo o leer algunos valores de él. Nada fue editado o agregado al archivo. Los datos fueron referenciados pero sin cambios.

Una marca de tiempo modificada mtime significa la última vez que se modificó el contenido de un archivo. Un programa o proceso editó o manipuló el archivo. "Modificado" significa que algo dentro del archivo se modificó o eliminó, o se agregaron datos nuevos.

Las marcas de tiempo modificadas *ctime* no se refieren a los cambios realizados en el contenido de un archivo. Por el contrario, es el momento en que se modificaron los metadatos relacionados con el archivo. Los cambios en los permisos de archivos, por ejemplo, actualizarán la marca de tiempo modificada.

1. **Generación de un fichero de tipo .csv de cada directorio del sistema de archivos** (necesario para verificar fechas de creación, modificación o cambio de estado en cualquier directorio del sistema, así como analizar permisos y usuarios o grupos de pertenencia no habituales o incorrectos).

**Ficheros:** *alldirstimeline.csv*

1. **Árbol de directorios del sistema** (forma visual de analizar los directorios del sistema en busca de directorios extraños o no habituales, además de permitir hacerse una idea visual del contenido del sistema).

**Ficheros:** *dirtree.txt*

1. **Características de los directorios “tmp” y “mnt”** (algo que podría indicar intento de violación de permisos, ejecución de código aprovechando sus características o montaje de unidades remotas, entre otros).

**Ficheros:** *filestmp.txt / filesmnt.txt*

1. **Archivos ocultos en el sistema de archivos en formato .csv** (en ocasiones ocultar archivos en un listado simple, facilita la ocultación de scripts y malware, por lo que se debe estudiar la presencia de archivos ocultos en el sistema y especialmente en el directorio de la aplicación WEB).

**Ficheros:** *fileshidden.csv / fileshidden\_webdir.csv*

1. **Archivos cuya fecha de creación o modificación esté en el período del incidente** (búsqueda de archivos en el sistema y especialmente en la aplicación WEB que hayan sido creados o modificados en el período de estudio, algo que puede ser indicador de infección).

**Ficheros:** *filesperiod.csv / filesperiod\_webdir.csv*

1. **Archivos de tipo ejecutable o con permisos de ejecución** (se revisan y listan los archivos con permisos de ejecución, de tipo ejecutable que estén presentes en el sistema o principalmente en el directorio WEB o media).

**Ficheros:** *filesexecperm.csv / filesexecperm\_webdir.csv / filesexe\_media.csv*

1. **Archivos con extensiones ejecutables .sh, .exe y .bin** (se revisan y listan los archivos con extensiones de tipo .bin, .sh, .exe que estén en el sistema o principalmente presentes en el directorio WEB o media).

**Ficheros:** *filesbin.csv / filesbin\_webdir.csv*

1. **Archivos pertenecientes al usuario “root”** (algunos archivos que pertenecen al usuario o al grupo administrador del sistema puede indicar la presencia de un usuario administrador fuera de su contexto habitual).

**Ficheros:** *filesroot.csv / filesroot\_webdir.csv*

1. **Archivos que no pertenecen al usuario web dentro del directorio de la aplicación web** (dentro del directorio en el que reside la aplicación web, deberían figurar solamente archivos y directorios pertenecientes a este usuario. La presencia de archivos de otro usuario o grupo puede indicar la presencia de malware o simplemente mala administración, por lo que se deben vigilar estos archivos y directorios con precaución).

**Ficheros:** *filesbaduser\_webdir.csv*

1. **Revisar permisos de archivos y directorios del directorio de la aplicación web** (Folder permissions: 755, File permissions: 644, Mage permissions: 550, Configuration file permission: 440).

**Ficheros:** *filesbadperms\_webdir.csv / dirsbadperms\_webdir.csv*

1. **Búsqueda de scripts de instalación en el directorio de la aplicación web** (se buscan scripts en php de instalación de plataformas estándar que normalmente se nombran con nombres similares a install.php y por seguridad no deben estar presentes en el directorio web).

**Ficheros:** *installers\_webdir.txt*

1. **Listar para su revisión los ficheros .htaccess del servicio Web** (debido a sus especiales características en el servidor Web, los ficheros .htaccess ocultos pueden contener reglas, ejecuciones y filtros que los hacen muy peligrosos si son modificados por los atacantes).

**Ficheros:** *htaccess\_webdir.txt*

1. **Configuraciones de servicios Web** (aunque resulta una tarea ardua, se debe de revisar las configuraciones de servicios web como apache2 o en este caso nginx, así como las de servidores proxy y PHP en busca de configuraciones inseguras no intencionadas o intencionadas que permitan el acceso a directorios web, o la subida o ejecución de archivos maliciosos. Además se estudia la fecha de última modificación de estos ficheros y la versión de estos servicios).

**Ficheros:** *http.txt / HTTP\_SERVER\_DIR\_php.tar / HTTP\_SERVER\_DIR\_apache.tar.gz / HTTP\_SERVER\_DIR\_nginx.tgz*

1. **Versiones de otros interpretadores como Python y Perl** (se debe estudiar la presencia de runtimes de Python y Perl por si no debieran estar presentes y se lista su versión y librerías para buscar vulnerabilidades).

**Ficheros:** *python.txt / perl.txt*

1. **Compiladores y versiones presentes** (se debe estudiar la presencia de compiladores y lenguajes de compilación ya que lo normal y más recomendable es que no estén presentes en entornos de producción y se lista su versión para buscar vulnerabilidades).

**Ficheros:** *compilers.txt*

1. **Software criptográfico instalado** (se deberá revisar la versión de librerías y programas de cifrado como OpenSSL para buscar vulnerabilidades en el cifrado que puedan significar importantes fallos de seguridad, como ocurrió en ocasiones con TLS-SSL).

**Ficheros:** *crypto.txt*

1. **Búsqueda de certificados y claves privadas en el sistema** (los certificados y claves privadas se deben de manejar con suma precaución. Se busca la presencia de certificados y claves privadas relacionadas y se analizan sus propiedades. Se extrae el módulus de cada certificado y de cada clave para estudiar su coincidencia).

**Ficheros:** *certspub.txt / certskey.txt*

1. **Se buscan indicadores de compromiso y malware mediante análisis estático de código PHP, javascript, Ajax o Shell Script** (para ello se analizan posibles indicadores mediante comandos y código de programación habitualmente utilizado en la ejecución directa de código Shell o en la ofuscación de código).
2. **Análisis de Seguridad** (se descarga e instala la última versión compatible con el sistema operativo presente del escáner de seguridad lynis en búsqueda de configuraciones de seguridad débil o mejorable).

**Ficheros:** *lynis.log*

1. **Análisis de Malware** (se ejecutan las principales herramientas de detección de malware para Linux haciendo especial incapié en utilizar las últimas versiones en sus motores y las últimas actualizaciones de firmas. Se busca malware de todo tipo presente en ficheros web de tipo php, javascript, Shell o Ajax entre los que pueden figurar rootkits, shells, etc).

**Ficheros:** *chkrootkit.txt / rkhunter.txt / shelldetector.txt / neopi\_shelldetect.txt / php-malware-scanner.txt / php-malware-finder.txt*

1. **Análisis específico de Magento** (se localiza, instala y ejecutan las principales herramientas gratuitas o en versión de prueba relacionadas específicamente con la seguridad en aplicaciones Magento. Se ejecutan únicamente aquellas que permiten análisis offline o en sistemas live).

**Ficheros:** *magento\_malware.txt*

1. **Búsqueda de archivos de malware Magento conocidos** (con el mismo nombre que los utilizados en ataques a webs mediante las mismas o similares técnicas o que contengan keywords como “onepage | checkout | onestep | firecheckout”).

**Ficheros:** *magentomalware\_webdir.csv / magentoskimmer\_webdir.txt*

1. **Análisis de otros indicadores de compromiso IoC** (se buscan patrones presentes en archivos del sistema de archivo relacionados con malware, ofuscación de archivos en lenguajes PHP, o javascript como: passthru | shell\_exec | cmd| sh –c | system | phpinfo | base64\_decode | edoced\_46esab | chmod | mkdir | gzinflate | fopen | fclose | readfile | php\_uname | eval |atob).

**Ficheros:** *manual\_shelldetect.txt*

1. **Archivos o directorios extraños o dañados** (se analizan los archivos y directorios que puedan estar dañados o hayan sido dañados intencionadamente en búsqueda de indicadores de rootkits. Se listan los archivos sin usuario, sin grupo, con fechas en el futuro, o en un pasado muy lejano).

**Ficheros:** *filesnogroup.csv / filesnogroup\_webdir.csv / filesnouser.csv / filesnouser\_webdir.csv / filesfuture.csv / filesfuture\_webdir.csv / filesbefore2015.csv / filesbefore2015\_webdir.csv*

1. **Indicadores de “leaks o data breach” relacionados con robo de información en la empresa** (en muchos incidentes relacionados con el robo de datos, se manifiestan en el sistema archivos, habitualmente comprimidos de tamaño muy grande. Esto puede indicar además la presencia de backups por parte de los administradores que no deben de estar presentes en el sistema).

**Ficheros:** *filesgreater100MB.csv / filesgreater100MB\_webdir.csv / filesgreater10MB.csv / filesgreater10MB\_webdir.csv*

1. **Archivos que contengan el término “password” en el directorio Web** (debemos analizar principalmente aquellos archivos que contengan credenciales en el directorio de la aplicación web, ya que si incorporan credenciales de servicios o aplicaciones crean un potencial peligro en el sistema).

**Ficheros:** *passwords\_webdir.txt*

1. **Archivos que contengan términos relacionados con el hacking, exfiltración o servidores utilizados por cibercriminales** (podrán contener términos relacionados con sitios o direcciones habituales para la exfiltración de datos como: hack | malware | infected | compromis | protonmail | tormail | silentcircle | torguard | oneshar | pastebin | dropbox | drive.google).

**Ficheros:** *hack\_webdir.txt*

1. **Archivos que incluyan direcciones de correo, direcciones IP o enlaces URL dentro del directorio Web** (aunque esto genera muchos falsos positivos, la búsqueda de este tipo de endpoints dentro de archivos en el directorio Web podría indicar algún tipo de conexión o exfiltración de datos por parte del malware. Además se buscan scripts javascript externos al sistema en nodos remotos que puedan indicar compromiso o ejecución remota de código, presentes en malware de tipo minería mediante JS o keyloggers de datos).

**Ficheros:** *emails\_webdir.txt / urls\_webdir.txt / ips\_webdir.txt / javascript\_webdir.txt / jsendpoints\_webdir.txt*

1. **Análisis de PAN o números de tarjetas de crédito** (se utilizan al menos tres herramientas de búsqueda de patrones de tarjetas de crédito en ficheros presentes en el sistema de archivos. Se modifican algunas de ellas para generar reportes adecuados o por fallar en su ejecución).

**Ficheros:** *PANHunter.log / PANhunt.log*

1. **Análisis de logs de Nginx o de Apache2** (de forma externa, en el laboratorio se analizaran patrones de ataques web mediante el análisis de logs del servidor web nginx o apache2 y sus archivos de log estándar. Para ello se utilizan herramientas como Scalp para Linux).

**Ficheros:** *http.txt*

1. **Análisis de Bases de Datos** (se extraen las bases de datos MySQL presentes y se buscan patrones binarios, ofuscados o directamente en texto claro, como tarjetas de crédito, webshells, malware, etc. Búsqueda de patrones de tarjetas de crédito / débito en todo tipo de ficheros y en base de datos.

**Ficheros:** *mysql\_search.txt*

1. **Se archivan en formato tar comprimido mediante gzip los principales directorios** (se archivan y extraen los directorios “logs, root, etc, home y www” para el análisis posterior de otros datos de relevancia en el laboratorio forénsico).

**Ficheros:** *VAR\_LOG.tgz / ROOT\_HOME.tgz / ETC.tgz / HOME\_user.tgz / VAR\_WWW.tgz*

1. **Análisis dinámico mediante Google y otras webs de análisis de reputación** (aprovechamiento de los motores gratuitos existentes en el mercado para el análisis de reputación y de malware).

**Ficheros:** *loginsnow.txt*

1. **Errores durante la ejecución de comandos del script** (por motivos de depuración, este fichero muestra cualquier error que haya ocurrido durante la ejecución de cada uno de los comandos del script. Sólo es necesario en caso de errores para depuración).

**Ficheros:** *errors.txt*

**Información y evidencias extraídas por el script**

Esta herramienta llamada “*forensic\_get.sh*” selecciona y extrae la información referida en esta sección y debidamente formateada para el análisis forénsico posterior (se aportan los ficheros generados en cada servidor en los directorios con las fechas de extracción en un archivo comprimido con el password definido en las variables del mismo script:

**Configuración del script**

El script “*forensic\_get.sh*” debe de ser configurado antes de comenzar la extracción de datos forénsicos editando el archivo de configuración forensic\_get.conf que debe de acompañar al script. Los datos a introducir en este archivo son de vital importancia para la extracción forénsica así que deben de ser revisados a conciencia:

Nombre del investigador forénsico que realiza la extracción (para fines de archivado y registro de la información extraída):

**investigator**="Yago Hansen"

Número de serie de la unidad flash USB utilizada para la extracción forense, ejemplo: "2a65058b". Se puede obtener ejecutando el commando de Linux blkid una vez introducido el pendrive de archivado de datos forénsicos. Esta unidad debe de estar formateada idealmente en formato ext4 de Linux para mantener los propietarios y otras configuraciones.

**usbid**="2a65058b-e7c5"

Fecha inicial en la que se sospecha pueda haber comenzado el incidente de ciberseguridad (en formato AAAA-MM-DD). Esta fecha se introduce de forma **opcional** para delimitar la búsqueda de datos y registros indicando esta fecha de inicio. Cualquier dato anterior a esta fecha no será de forma general tenido en cuenta para la extracción:

**start\_date="2019-06-01"**

Fecha final en la que se sospecha pueda haber finalizado el incidente de ciberseguridad (en formato AAAA-MM-DD). Esta fecha se introduce de forma **opcional** para delimitar la búsqueda de datos y registros indicando esta fecha final. Cualquier dato posterior no será tenido en cuenta en la extracción:

**end\_date="2019-10-31"**

Archivado de información. Si se introduce el valor 1 en esta variable, se indica que además de recoger los valores importantes relacionados con el incidente, también se recogerán los ficheros originales relacionados. Estos ficheros y directorios (como: *var, etc, root, home, www*) serán comprimidos y almacenados en el pendrive de extracción forénsica:

**archive=0**

Extracción de base de datos MySQL. Si se introduce el valor 1 en esta variable, se indica que se desea recoger un volcado o dump de la/s bases de datos MySQL utilizadas en el servidor indicado para la extracción:

**extractsql=0**

Escaneo de seguridad. Si se introduce el valor 1 en esta variable, se indica que se desea realizar un escaneo de seguridad y búsqueda de malware tras el proceso de recogida de información. Se ejecutarán las herramientas recomendadas que deben de ser incluidas en el pendrive de extracción forense. Véase apartado de Herramientas de seguridad:

**secscan=0**

Obtención de Metadatos de imágenes. Si se introduce el valor 1 en esta variable, se indica que se desea obtener y archivar la información de metadatos que incluyen muchos formatos de imágenes como TIFF, JPG, PNG… Para extraer estos metadatos se utiliza la herramienta *exiftool* que debería estar instalada en el sistema destino. Si no estuviera instalada y hubiera conexión a Internet, se instalará mediante *apt-get*. Estos metadatos se archivarán en el fichero *imagesmetadata\_media.txt*:

**imgmetadata=0**

Web de tipo Magento. Si se introduce el valor 1 en esta variable, se indica que el servidor investigado aloja una aplicación web de tipo Magento, por lo que realizará ciertos tests de malware y tendrá en cuenta la estructura de este programa:

**magentosite=1**

Apagar el servidor al terminar de auditar. Si se introduce el valor 1 en esta variable, se indica que el script apagará el servidor Linux tras recopilar toda la información forénsica. Este proceso de recopilación de evidencias es largo y se recomienda atender la ejecución del mismo de forma activa por el investigador por si se produjera algún error durante la recopilación. Pero si se desea dejarlo corriendo y apagar el sistema tras la recopilación se puede indicar el valor 1 en esta variable:

**poweroffwhenfinish=1**

Contraseña de archivado. Al finalizar la recopilación de datos, el script archiva toda la información extraída hacia un archivo comprimido y cifrado mediante AES-256 CBC. Se debe de indicar una contraseña robusta para el archivado de información. Pero en caso de no definirla en esta variable de configuración, se utilizará por defecto esta contraseña: “D3f4ultPassWorD@@@####”. Pero recuerde que las contraseñas por defecto son peligrosas y que cada vez que se utiliza una, muere un gatito en algún lugar del mundo :-(

**packagepassword="MyF0rensicD47a!!!"**

Directorio de la investigación. En principio este script obtiene los datos a analizar de toda la estructura del sistema de archivos, pero en algunos casos es preferible obtenerlos de una unidad de red o punto de montaje en el sistema a analizar. Mediante esta variable se puede limitar el alcance de la búsqueda de evidencias a un directorio concreto:

**investigateonlydir="/"**

Directorio de almacenamiento Web. Normalmente este directorio, en el que el servidor Web aloja los archivos y aplicaciones suele ser “/var/www”, pero en algunos servidores sobre todo multihosting, este directorio corresponde a otra ruta. Aquí se define el directorio de almacenamiento Web.

**wwwdir="/var/www/vhost/www.example.com/htdocs"**

Directorio Media. Muchos servidores Web alojan el contenido de medios (imágenes, vídeos, logos, etc.) en la propia estructura de directorios, aunque otros servidores Web utilizan directorios remotos en otros servidores, normalmente basados en un punto de montaje NFS o SMB.

**wwwmedia=""**

Usuario Web. El usuario Linux con permisos limitados para poder navegar por el sitio web (habitualmente www-data). Se utiliza para comparar el propietario de los archivos en el directorio de la aplicación Web con el usuario utilizado para la misma, en busca de archivos con permisos indebidos.

**wwwuser="wwwuser"**

Grupo Web. El grupo Linux con permisos limitados para poder navegar por el sitio web (habitualmente www-data). Se utiliza para comparar el propietario de los archivos en el directorio de la aplicación Web con el usuario utilizado para la misma, en busca de archivos con permisos indebidos.

**wwwgroup="wwwgroup"**

Usuario MySQL. En esta variable se puede de forma opcional incluir el usuario que tiene acceso a la base de datos MySQL a analizar o volcar. En muchos ataques a aplicaciones Web, se archivan en registros SQL líneas de código malicioso utilizado como shells o ataques XSS, etc. Es conveniente volcar la base de datos para su análisis forénsico posterior.

**mysqluser="mysql\_usr"**

Contraseña del usuario MySQL. A fin de poder realizar un volcado o dump SQL de la base de datos o bases de datos MySQL se precisa de la contraseña del usuario anteriormente indicado. Debe de incluirse la contraseña aquí.

**mysqlpass="asdefaEFDA35454qdafas"**

Base de datos MySQL a volcar. Nombre de la base de datos MySQL a volcar mediante un dump SQL.

**mysqldb="db\_com"**

**Ejecución del script**

El script “*forensic\_get.sh*” debe de ser configurado antes de comenzar la extracción de datos forénsicos editando el archivo de configuración forensic\_get.conf que debe de acompañar al script en su directorio de ejecución.

Los datos a introducir en este archivo son de vital importancia para la extracción forénsica así que deben de ser revisados a conciencia:

################# Main configuration variables: #################

## forensic investigator name

investigator="Yago Hansen"

## USB flash disk serial number "2a65058b" (get with linux blkid command) formated in ext4 fs

usbid="2a65058b-e7c5"

## Suspicious hack time range (YYYY-MM-DD) optional from start\_date to end\_date

start\_date="2019-06-01"

end\_date="2019-10-31"

## if value 1 compress with tar main directories (var etc root home www)

archive=0

## if value 1 execute security scanners at the end

secscan=0

## if value 1 extract all the metadata from common image files to analyze it later

imgmetadata=0

## if value 1 run magento related tests

magentosite=1

## if value 1 power off system after forensic data extraction

poweroffwhenfinish=1

## (mandatory) create a secure password for encrypting forensica data gathered

packagepassword="MyF0rensicD47a!!!"

## dir to investigate.Sometimes it is necessary to investigate all (/) or just web root (/var/www)

investigateonlydir="/"

## directory where Web application is stored (usually: /var/www/)

wwwdir="/var/www/vhost/www.example.com/htdocs"

## Media, public or upload directory in web from $wwwdir (usually: media, pub, upload)

wwwmedia=""

## user and group name for Web application (usually: www-data, apache)

wwwuser="www-data"

wwwgroup="www-data"

## optional MySQL user, password and database to dump DB to ascii file

mysqluser="mysql\_usr"

mysqlpass="asdefeFDFADS344DFADFvczafDSFAds34"

mysqldb="db\_com\_db

#################################################################

**Herramientas de terceros a incluir**

El script “*forensic\_get.sh*” debe de ser configurado antes de comenzar la extracción de datos forénsicos editando el archivo de configuración forensic\_get.conf que debe de acompañar al script. Los datos a introducir en este archivo son de vital importancia para la extracción forénsica así que deben de ser revisados a conciencia:

**Neopi** (https://github.com/CiscoCXSecurity/NeoPI)

NeoPI is a Python script that uses a variety of statistical methods to detect obfuscated and encrypted content within text/script files. The intended purpose of NeoPI is to aid in the detection of hidden web shell code. The development focus of NeoPI was creating a tool that could be used in conjunction with other established detection methods such as Linux Malware Detect or traditional signature/keyword based searches.

NeoPI recursively scans through the file system from a base directory and will rank files based on the results of a number of tests. It also presents a “general” score derived from file rankings within the individual tests.

**Rootkit Hunter** (https://github.com/youngunix/rkhunter)

Rootkit Hunter (rkhunter) es una herramienta de Michael Boelen para encontrar evidencia de software malicioso en sistemas que ejecutan Linux, Mac OS X y UNIX. Como autor original de esta herramienta, lancé la primera versión en 2003. En 2006, el proyecto fue entregado a un nuevo equipo, para asegurar que su desarrollo continuara.

**PHP malware scanner** (https://github.com/scr34m/php-malware-scanner)

Traversing directories for files with php extensions and testing files against text or regexp rules, the rules based on self gathered samples and publicly available malwares/webshells. The goal is to find infected files and fight against kiddies, because to easy to bypass rules.

**Lynis security scanner** (https://github.com/CISOfy/lynis)

Lynis is a security auditing tool for systems based on UNIX like Linux, macOS, BSD, and others. It performs an in-depth security scan and runs on the system itself. The primary goal is to test security defenses and provide tips for further system hardening. It will also scan for general system information, vulnerable software packages, and possible configuration issues. Lynis was commonly used by system administrators and auditors to assess the security defenses of their systems. Besides the "blue team," nowadays penetration testers also have Lynis in their toolkit.

**chkrootkit** (https://github.com/Magentron/chkrootkit)

Chkrootkit o Check Rootkit es un programa famoso de código abierto, es una herramienta que se utiliza para la digitalización de rootkits, botnets, malwares, etc en tu servidor o sistema Unix/Linux.

**chkrootkit** (https://github.com/Magentron/chkrootkit)

Chkrootkit o Check Rootkit es un programa famoso de código abierto, es una herramienta que se utiliza para la digitalización de rootkits, botnets, malwares, etc en tu servidor o sistema Unix/Linux.

**magescan** (https://github.com/steverobbins/magescan)

The idea behind this is to evaluate the quality and security of a Magento site you don't have access to. The scenario when you're interviewing a potential developer or vetting a new client and want to have an idea of what you're getting into.

**PANhunter** (https://github.com/dbohannon/PANHunter)

Command line tool used to search files for credit card numbers (PAN). Card numbers are verified with regular expression and Luhn (i.e. mod10) checks. Results are written to a spreadsheet containing the file, line number, full card number, and masked card number.

**PANhunt** (https://github.com/Dionach/PANhunt)

PANhunt is a tool that can be used to search drives for credit card numbers (PANs). This is useful for checking PCI DSS scope accuracy. It's designed to be a simple, standalone tool that can be run from a USB stick. PANhunt includes a python PST file parser.

**PHP-malware-finder** (https://github.com/nbs-system/php-malware-finder)

PHP-malware-finder does its very best to detect obfuscated/dodgy code as well as files using PHP functions often used in malwares/webshells. Detection is performed by crawling the filesystem and testing files against a set of YARA rules. Yes, it's that simple!

**Apache-scalp** (https://github.com/neuroo/apache-scalp)

Scalp! is a log analyzer for the Apache web server that aims to look for security problems. The main idea is to look through huge log files and extract the possible attacks that have been sent through HTTP/GET (By default, Apache does not log the HTTP/POST variable). Scalp is basically using the regular expression from the PHP-IDS project and matches the lines from the Apache access log file. These regexp has been chosen because of their quality and the top activity of the team maintaining that project. You will then need this file default\_filter.xml in order to run Scalp.

**Shell-Detector** (https://github.com/emposha/Shell-Detector)

Shell Detector – is a application that helps you find and identify php/cgi(perl)/asp/aspx shells. Shell Detector has a “web shells” signature database that helps to identify “web shell” up to 99%. Shell Detector is released under the MIT License http://www.opensource.org/licenses/mit-license.php

**maldetect** (http://www.rfxn.com/downloads/maldetect-current.tar.gz)

Linux Malware Detect (LMD) is a malware scanner for Linux released under the GNU GPLv2 license, that is designed around the threats faced in shared hosted environments. It uses threat data from network edge intrusion detection systems to extract malware that is actively being used in attacks and generates signatures for detection. In addition, threat data is also derived from user submissions with the LMD checkout feature and from malware community resources. The signatures that LMD uses are MD5 file hashes and HEX pattern matches, they are also easily exported to any number of detection tools such as ClamAV.

**Other dependencies**

*Exiftool linux package*

*Clamav antivirus signatures*

*Prelink linux package*

*autoconf*

*libssl-dev*

*python-colorama*

*python-progressbarcd*