Detección de anomalías HTTP trazando la sesión web de un usuario

Nombre Estudiante: Capraru Pons, Cristian Alexandre

Programa: Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)
Área: PFM ad-hoc
Consultor: Carles Garrigues Olivella
Profesor/a responsable de la asignatura: Helena Rifà Pous
Centro: Universitat Oberta de Catalunya
Fecha entrega: 6 de Junio de 2016

FICHA DEL TRABAJO FINAL

Título del trabajo:	Detección de anomalías HTTP trazando la sesión web de un usuario
Nombre del autor:	Cristian Alexandre Capraru Pons
Nombre del consultor/a:	*Carles Garrigues Olivella*
Nombre del PRA:	*Helena Rifà Pous*
Fecha de entrega (mm/aaaa):	06/2016
Titulación::	Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)
Àrea del Treball Final:	*PFM ad-hoc*
Idioma del trabajo:	*Castellano*
Palabras clave	IDS, Ciberseguridad, HTTP, URL

Resumen del Trabajo (máximo 250 palabras):

Desde hace años, la seguridad se ha convertido en una dimensión fundamental en los menesteres empresariales del día a día. A causa de esto, existen una gran diversidad de herramientas y catálogos con el objetivo único de evitar posibles agresiones maliciosas a sistemas y aplicaciones.

El uso de diversos IDS (Sistema de detección de intrusos) distribuidos y más aun siendo de diferentes tipos, permite localizar un rango más holgado de intrusiones; a parte, si varios IDS abarcan un mismo grupo de intrusiones, la certidumbre sobre las alertas emitidas desde estos IDS se incrementa, mermando los falsos positivos.

Después de analizar el estado actual de los IDS basados en reglas, se determinan preguntas como: ¿Se pueden optimizar las herramientas de seguridad?¿Se pueden reducir los flujos de filtrado de las aplicaciones basadas en reglas?

Esta tesis de fin de máster pretende responder a estas preguntas y dar una solución. Se analizarán las causas por la que se saturan estos programas y se propondrá una herramienta a modo de pre-filtro de contenido no relevante.

De esta manera, el objetivo de este trabajo de investigación y implementación finaliza con la creación de una herramienta que ayudará a optimizar un sistema de seguridad como es el del IDS .

Abstract (in English, 250 words or less):

In the last few years, security has become a fundamental dimension in the daily business. Because of this, there is a great diversity in tools and catalogues with the unique goal of preventing malicious aggressions to systems and applications.

The use of distributed and other types of IDS (Intrusion Detection Systems) allows a wider range of intrusions to be detected. Furthermore, if several IDS are looking for the same group of attacks, the certainty about emitted alerts of this IDS is incremented, lowering the false positives.

After analysing the current state of rule-based IDS, several questions arise: Can we optimize our security tools? Can we reduce the information flows by filtering them using rule-based applications?

This thesis pretends to answer this questions and to give a solution. It will analyse the causes why the programs that are filtering information are saturated and it will propose a tool for pre-filtering non relevant content.

Therefore, the final target of this investigation and implementation work is to create a tool that will optimize a security system like the IDS.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

En general, la seguridad es un termino que siempre envuelve a las personas o a las organizaciones. Cada vez más, vamos viendo como los sistemas de seguridad van incrementando su eficacia.

El presente trabajo se ha realizado en la empresa valenciana S2 Grupo, entidad con una larga experiencia y conocimientos en la seguridad informática. Esta Tesis de Fin de Máster pretende ser un estudio sobre la seguridad y un aporte a este ámbito, a través una aplicación para dicha empresa. Dicha tésis viene acompañada de un gran interés por adquirir la mayor cantidad de conocimientos posibles sobre ciber-seguridad.

La empresa S2 Grupo se dedica a proteger la integridad, confidencialidad y la disponibilidad de los servicios de otras empresas, entre otros servicios, y se hace necesario invertir en sistemas que vigilen las redes de estas, protegiéndolas de posibles ataques.

El uso de múltiples IDS (Sistemas de detección de intrusiones como CARMEN[1], Snort, Suricata) ayuda a detectar la mayoría de intrusiones que no detecten programas básicos como antivirus, aunque actualmente no consiguen analizar el 100% del tráfico que pasa a través de ellos. Sin entrar en detalles, el objeto de este proyecto es analizar el estado actual de los IDS y estudiar como mejorar los flujos de entrada a estos sistemas. La aplicación aportada detectará en una sesión de un usuario ya guardada, todas aquellas anomalías que se puedan encontrar.

Así pues se pretenderá hacer de los IDS, sistemas más eficaces, optimizando su eficiencia gracias a la implementación de esta herramienta de pre-filtrado.

Finalmente, mencionar que la herramienta propuesta es un prototipo que en un futuro tendrá una función muy importante a la hora de mejorar los rendimientos de los sistemas de seguridad desplegados en una organización.

[1] http://s2grupo.es/productos/carmen/

1.2 Objetivos del Trabajo

- Familiarizarse con los conceptos de IDS, IPS y sistemas de detección de anomalías.
- Conocer las diferentes etapas del transcurso de los paquetes que viajan por la red.
- Conocer patrones de peticiones realizadas por aplicaciones anómalas y no solicitadas dentro de la sesión de un usuario.
- Detectar pues, el malware que realiza consultas a través de las miles de peticiones que se realizan durante la sesión de un usuario.
- Desarrollar un “sniffer” que se ubique entre el exterior y los sistemas de detección (IDS, cortafuegos, etc) de la organización y que realice tareas de pre-filtrado de los paquetes HTTP.
- Facilitar, de esta manera, el trabajo al IDS a la hora de examinar grandes flujos de datos.
- El objetivo final pues, es eliminar parte del trabajo de los IDS, haciendo que estos sistemas sean más eficientes.

1.3 Enfoque y método seguido

I. **Identificación del problema, oportunidades y objetivos.**
Los sistemas de detección de intrusos de la empresa S2 Grupo son muy eficientes. Aun así, la dirección de la empresa comenta en diversas ocasiones, estos sistemas no llegan a recoger parte del contenido que pasa por la red. El alto coste del hardware, ralentiza la eficacia de dichos sistemas.

II. **Determinación de los requerimientos de información.**
Se necesitará desarrollar una herramienta que pueda ayudar a mejorar la eficacia de los sistemas IDS y que este dedique sus recursos solo a analizar contenido potencialmente dañino.

III. **Análisis de las necesidades del sistema.**
Para reducir la carga de trabajo de los sistemas de detección, se deberá reducir el contenido de entrada, buscando una manera de eliminar contenido no relevante.

IV. **Diseño del sistema recomendado.**
Se diseñará una aplicación que cumpla con lo estipulado anteriormente, con módulos que permitan coger un tráfico de paquetes determinado y los reduzca al máximo.

V. **Desarrollo y documentación del software.**
El script a desarrollar, se realizará en Python, ya que se dispone de una gran cantidad de librerías preparadas para ser usadas en este tipo de proyectos.

VI. **Pruebas y mantenimiento del sistema.**
Con el programa ya en ejecución, se realizará una batería de pruebas para ver si es capaz de sustraer contenido relevante y se comprobará si supone una gran eficiencia para el sistema.

VII. **Implantación y evaluación del sistema.**
Con las pruebas finalizadas, si son exitosas, dicho módulo será propuesto para ser anexionado a los sistemas de escudo de la empresa.

1.4 Planificación del Trabajo

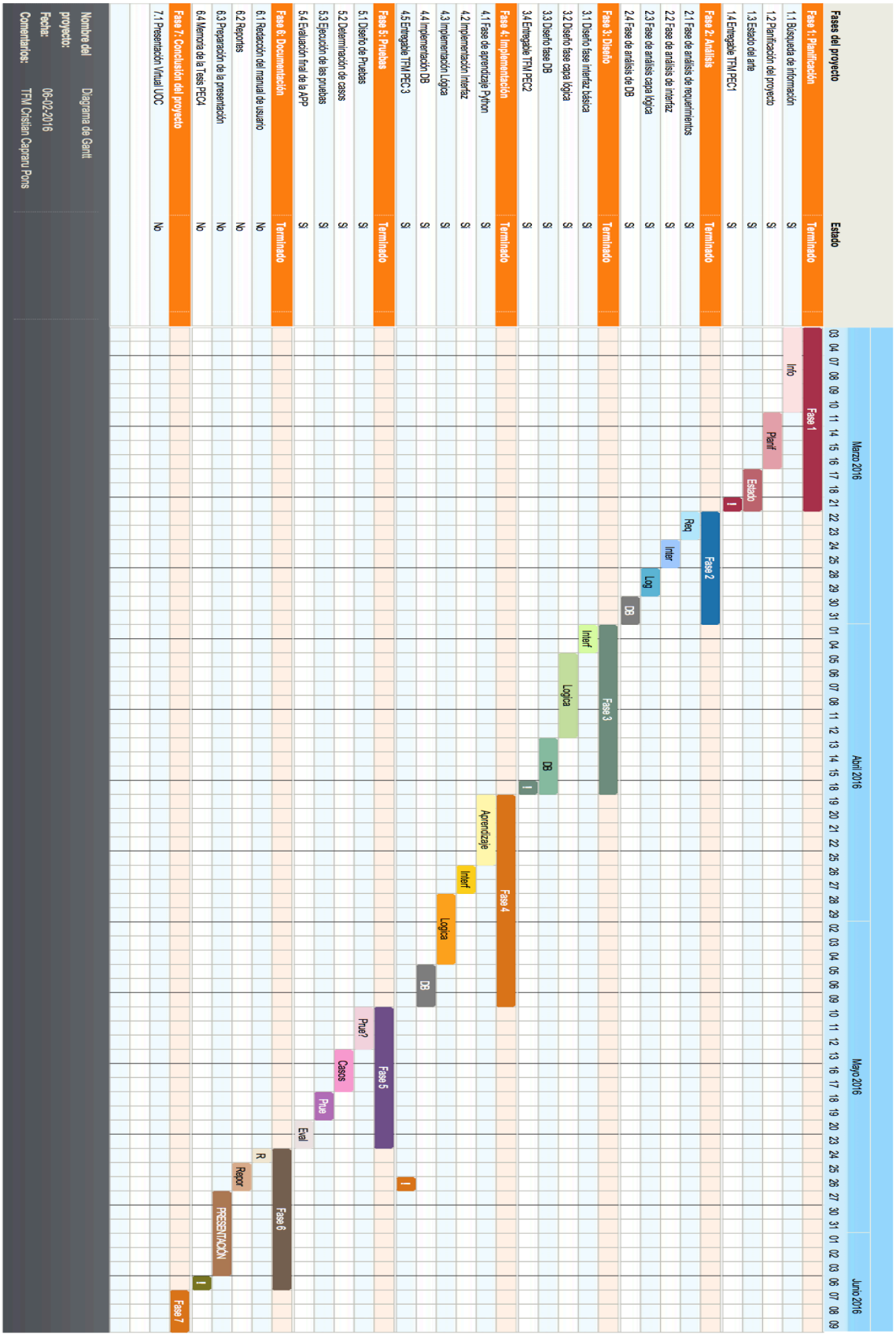

Ilustración 1. Diagrama de Gantt

1.5 Estado del arte

Toda empresa necesita disponer de programas de monitorización, ya sea en redes o en sistemas operativos. Se conoce que el pionero en hablar de NSM (Network Security Monitoring), fue Todd Heberlein[2], el cual creó el primer sistema de anti-intrusos en 1988. Este sistema, a partir de un flujo de tráfico, podía generar alertas mientras examinaba grandes cantidades de información. En 1993, el AFCERT (Air Force Computer Emergency Response Team) empezó a usar los sistemas de monitoreo proporcionados por Heberlein para realizar filtrados del contenido entrante y saliente de sus instalaciones a través de la red.

Para contrarrestar las amenazas digitales, aquellas organizaciones que tienen cierta preocupación y concienciación por la seguridad, constituyen equipos informáticos de respuesta contra incidentes (CIRT). Tanto si se dispone de una máquina como si se trata de un equipo de personas, los CIRTS deben manejar las intrusiones informáticas, para reducir las posibles brechas que se puedan generar en un futuro.

Disponer de un NSM no implica prevenir intrusiones al 100%, ya que la prevención a veces falla. Es más, una organización nunca podría estar protegida en su totalidad de ataques externos, a no ser que se desconecte por completo de la red externa. En esta tesis, las intrusiones de la red, concretamente las salientes del cliente hacia el servidor, serán el objetivo a tratar.

Los sistemas de detección de intrusos tienen una arquitectura similar a la siguiente figura:

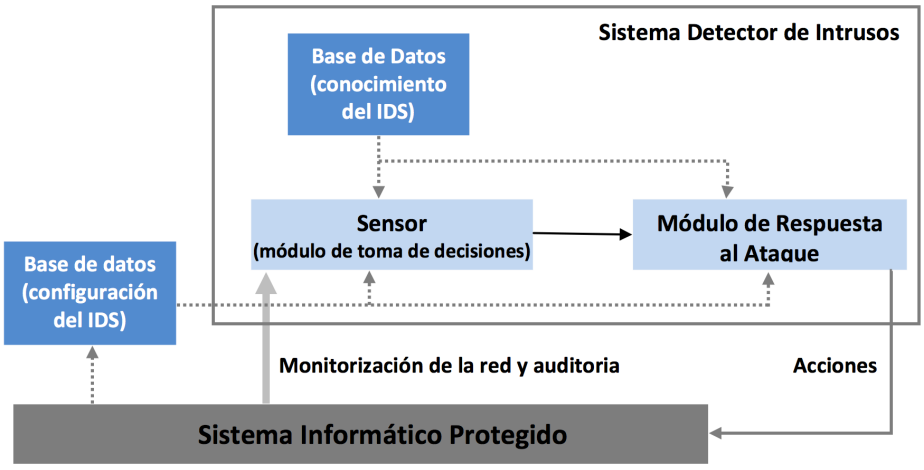

Ilustración 2. Funcionamiento IDS

[2] Developer of the original Network Security Monitor (NSM) http://www.toddheberlein.com/

Un IDS intenta detectar/monitorizar todo tipo de eventos ocurridos en un sistema informático o una red concreta, en busca de posibles intentos que puedan desequilibrar la armonía de dicho sistema o red.

Los IDS aportan, sobretodo, seguridad. No detienen ataques, en cuyo caso se llamarían IPS (Intrusion Prevention System), pero se pueden preparar para que generen ciertos tipos de respuesta ante posibles ataques. También aportan cierta capacidad de prevención. Estos programas están siempre alerta y intentan anticiparse a posibles actividades que puedan ser tildadas de sospechosas.

Para entender la arquitectura de un NSM en una organización, se propone la siguiente estructura donde se puede ver la función del IDS:

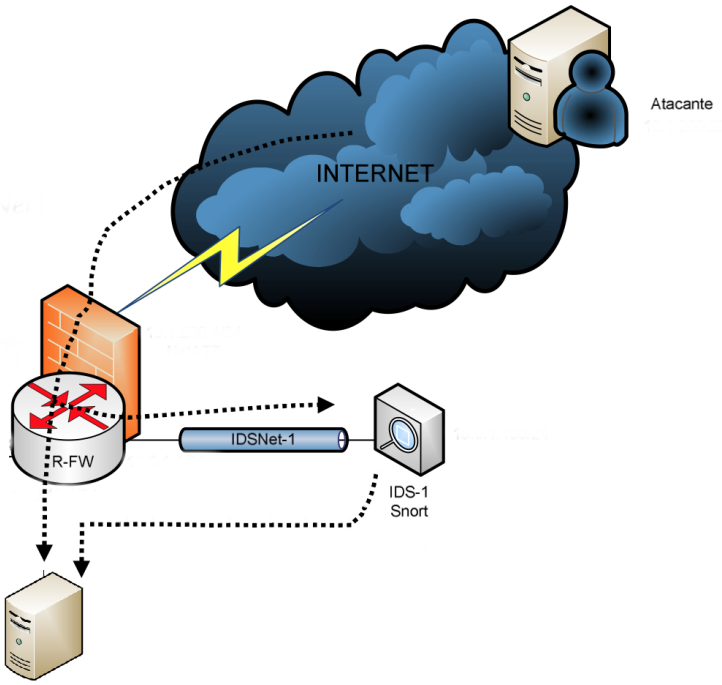

Ilustración 3. Estructura IDS

Como se puede comprobar, muchas organizaciones usan cortafuegos para evitar conexiones no autorizadas por el sistema. Un sistema IPS, suele contar con un cortafuegos y un IDS. Ambos reciben todo el contenido externo para analizarlo antes de que llegue a la organización.

El problema reside en los costes de *hardware*. Suelen ser altos ya que la carga de cómputo que soportan es elevada, y en la mayoría de las ocasiones, controlar todo el tráfico es casi imposible. En una organización media de unas doscientas personas, se generan casi 500 Mbps de media en un horario típico. En caso de una empresa de mayor envergadura, si una sede dispone de 1000 empleados, se puede llegar a los 8Gbps de salida y entrada de información. Hoy en día, los sistemas consiguen examinar casi todo el tráfico, pero no es suficiente. El tráfico capturado pasa a través de las reglas predefinidas por el responsable de seguridad informática y se generan alertas cuando se

encuentran paquetes que no cumplen con las políticas de seguridad de la empresa.

La vigilancia de la red es el principal objetivo de este centinela. Examina su red en busca de paquetes y datos sospechosos a la alerta de cualquier posible primera fase de un ataque, como un análisis de la red protegida, un barrido de puertos y/o servicios, etc.

Los procesos de detección de intrusiones pasan por los siguientes puntos.

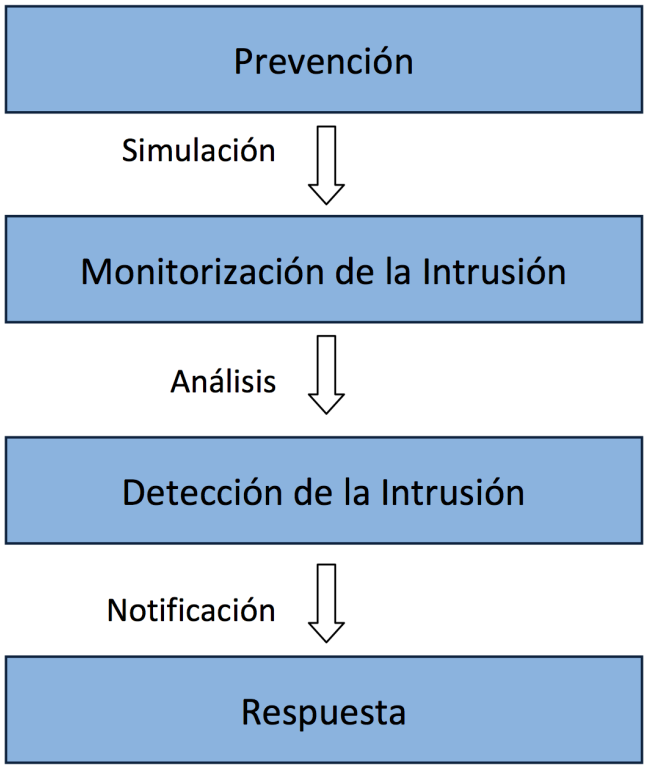

Ilustración 4. Procesos IDS

Este proyecto se centra en la parte de "detección de anomalías". Un IDS dispone de varios modos de detección, por ejemplo la detección por anomalías. En este caso, se trata de detectar cualquier intrusión examinando usos anómalos del sistema. Se pueden crear perfiles de conducta predefinidos o incluso desarrollarlos según interese. Si una conducta, se sale de este tipo de actuaciones, se debe considerar como un posible ataque.

Algunos IDS, como *BroIDS*[3] soportan una fase de entrenamiento en el que el sistema aprende a determinar lo que es un comportamiento "normal" o "típico". En esta fase, el sistema deberá tomar flujos de tráfico cuotidiano sin que reciba ningún tipo de ataques, de manera que se garantice un buen funcionamiento del IDS.

[3] https://www.bro.org/

El concepto de tráfico cuotidiano no significa que la red esté al 100% segura, y conseguir un tráfico sin anomalías es casi imposible en cualquier tramo del día en una organización. Se entendería como anomalía, el hecho de que, por ejemplo, una noche aleatoria, desde una sede, empezase a generarse una gran cantidad de tráfico a las 03:00 AM de la mañana cuando lo típico es que el tráfico en la red sea mínimo. Esto sería una anomalía y debería ser detectada por el IDS y notificada al responsable adecuado.

La ventaja más fundamental de estos sistemas es que se pueden detectar ataques nuevos gracias al análisis del comportamiento anómalos que ocasionen los ataques en el sistema.

Existen varios sitios en los que podemos colocar un IDS:

- Precediendo al cortafuegos: en este caso se cogerá todo el tráfico que entre y salga de la red. Existe una gran cantidad de que se generen falsos positivos.
- Posterior al cortafuegos: se controlará todo el tráfico que no sea detenido por el cortafuegos. Existe una posibilidad muy pequeña de que se generen falsos positivos.
- Colocando dos IDS, en cada una de las partes anteriormente mencionadas: la información obtenida pasa a ser de gran valor ya que si el firewall esta configurado de manera correcta, se pueden parar de manera más eficaz los posibles ataques. Se trata de una alternativa mucho más costosa que las anteriores, pero la dimensión de la seguridad, crece en gran medida.
- En situaciones muy concretas, el IDS puede ir colocado en el mismo mecanismo que el cortafuegos. Se consigue así tener una parte que detecte paquetes (cortafuegos) y otra que los analice (IDS).

El problema viene a la hora de analizar y destapar falsos positivos, ya que cualquier nueva actividad en el sistema, podría considerarse como un ataque. Las principales vulnerabilidades de los IDS que estén en modo “Detección de Anomalías”, es que se produzca un ataque durante la fase de entrenamiento, ya que éste podría entender el ataque como una conducta normal. En caso de que un intruso conozca la evolución de los perfiles del sistema, podría entrenar el IDS para que no actúe ante futuros ataques.

De ahí la necesitad de encontrar una herramienta que reduzca los falsos positivos y centre todos los recursos en encontrar anomalías de manera correcta. La herramienta que se presentará en los siguientes capítulos, actuará de filtro para el IDS, encargándose de descartar todo aquel contenido que no sea útil para su posterior análisis.

1.6 Breve sumario de productos obtenidos

Analizado el problema anterior, se ha procedido a desarrollar un prototipo de una herramienta mayor que permite reducir en gran medida el número de paquetes HTTP y con ello, grandes flujos de información que reciben los IDS. Se ha conseguido, pues, generar un programa en Python que tiene como

entrada una serie de paquetes en formato “.pcap” y que analiza aquellos que sean HTTP para ver cuales deben ser descartados por tratarse de contenido de relleno y que no aportan más que trabajo comparativo innecesario al sistema.

1.7 Breve descripción de los otros capítulos de la memoria

El resto de capítulos se divide en las siguientes partes:

- Descripción General: En este capítulo se pretende explicar el problema planteado por la dirección de la empresa en la que se realizan las prácticas de la UOC.
- Análisis: A raíz del problema planteado, se ha recogido información sobre como resolverlo y pensar en un posible diseño.
- Implementación: En esta sección se desarrolla el prototipo que actuará de analizador previo de peticiones HTTP.
- Conclusiones

2. Descripción General

2.1 Sistemas contra intrusiones

Los tres grandes sistemas de detección de intrusiones son los IDS, IPS y IRS. Estos sistemas de detección de intrusiones son sistemas basados en un hardware y software específico con el objetivo de alertar en caso de detección de anomalías a medida que van teniendo lugar. Estos ataques suponen acciones no autorizadas o en algunos casos, no deseadas por la organización y podrían suponer una brecha en la integridad, disponibilidad y confidencialidad de un determinado recurso.

Un IDS es capaz de detectar intentos de denegación de servicios al igual que intentos de acceso no autorizados. Si está bien configurado, se puede incluso detectar escaneo de puertos realizados con herramientas como por ejemplo “nmaps”. Dichas funciones de monitorización son posibles gracias a que los IDS, examinan una interfaz de red concretas o actividades ilícitas sobre un host.

Cuando un IDS emite una alerta, ésta contiene datos sobre la posible intrusión. Los sistemas de tipo HIDS OSSEC, que está asentado en el *host*, puede generar alertas sobre intentos de ataque con “Inyection SQL”, mostrando datos como la dirección IP fuente, la hora, el navegador utilizado y otros datos más.

Un NIDS Snort, basado en red, origina alertas que engloban: descripción de la alerta, nombre de la alerta, intruso, dirección IP, prioridad, etc. Con esto, se puede observar que diferentes IDS detectan diferentes tipos de información, aunque se refieran a una misma intrusión.

Todas las alertas son recibidas por el gestor, un elemento del IDS que se encarga de tomar decisiones y de generar una respuesta, según la política de seguridad de la organización. Las respuestas pueden ser tomadas por el administrador o ser ejecutadas automáticamente. La segunda opción es la deseable, ya que una vez detectada la intrusión, el sistema debe intentar resolverlo sin la necesidad de molestar a un operario.

Las respuestas de un IDS, pueden ser pasivas o activas, aunque normalmente suelen ser del primer tipo. Las respuestas pasivas no minimizan el daño causado por la intrusión, si no que notifican al administrador del IDS subministrándole información de lo ocurrido a través de:

- Logs
- SMS
- E-mails

Las respuestas activas, en cambio, tienen el propósito de minimizar la contusión realizada por parte del intruso (estaríamos hablando de IPS y IRS). Las medidas de contención pasarían por:

- Finalización de un servicio extraño
- Bloqueo de conexiones
- Revocar una cuenta de usuario comprometida

Por otra parte, se deben mencionar los NIDS, que son IDS basados en red. Estos, embeben tráfico que transita a través de un determinado segmento de red. Los NIDS monitorizan paquetes cacheándolos para encontrar actividades que puedan ser tildadas de intrusiones o ataques, según una serie de reglas preestablecidas. Para un correcto funcionamiento del NIDS, se requiere que la interfaz de red funcione en modo promiscuo. Se consigue con esto, que el tráfico sea redirigido hacia las capas más altas de la red para su estudio.

Es muy normal que un NIDS nunca influya en los flujos de trabajo que está analizando. Es por ello que se hace uso de herramientas como TAPS (Puertos de Accesos de Pruebas), que recogen tráfico en ambas partes y lo proporcionan al NIDS. En un lenguaje técnico, el sistema "puentea" el tráfico a modo de "mirroring" sin afectar al rendimiento de la red.

En caso de que se quiera frenar un ataque y que la versión centinela no sea suficiente, la solución que se plantea es el uso de IPS o IRS, que en algunas ocasiones, suele crear confusión según el modo de operar. Los dos sistemas necesitan de un IDS para que localicen intrusiones y notifiquen al gestor para que éste prepare una respuesta. Como se ha comentado antes, si la respuesta es activa, hablaremos de un sistema de respuesta a intrusiones o de un sistema de prevención de intrusiones.

El IPS (sistema de prevención de intrusiones) tiene como misión realizar respuestas de contención contra posibles intrusiones. A diferencia del IDS, que en algunas ocasiones no esta en la línea de flujo de transacciones, el IPS se sitúa entre el exterior y el interior de la organización. Por ejemplo, si el NIDS detecta intrusiones y genera alertas, se espera que el IPS esté ubicado sobre un firewall para mitigar el tránsito de datos en el cual se sitúa la intrusión sospechosa.

Por otra parte, un IRS realiza acciones idénticas que un IPS, pero no se ciñe a efectuar acciones de frenado sobre el dispositivo en línea si no que tiene preparadas una serie de respuestas que según la categoría del ataque, se seleccionará una o otra.

Las posibles respuestas activas podrían ser:

ATAQUE	RESPUESTA
Escalamiento de Privilegios	Inhabilitación del usuario afectado
Denegación de Servicios	Aislamiento de IP en el FW
RC no autorizado por Troyano	Finalización de proceso

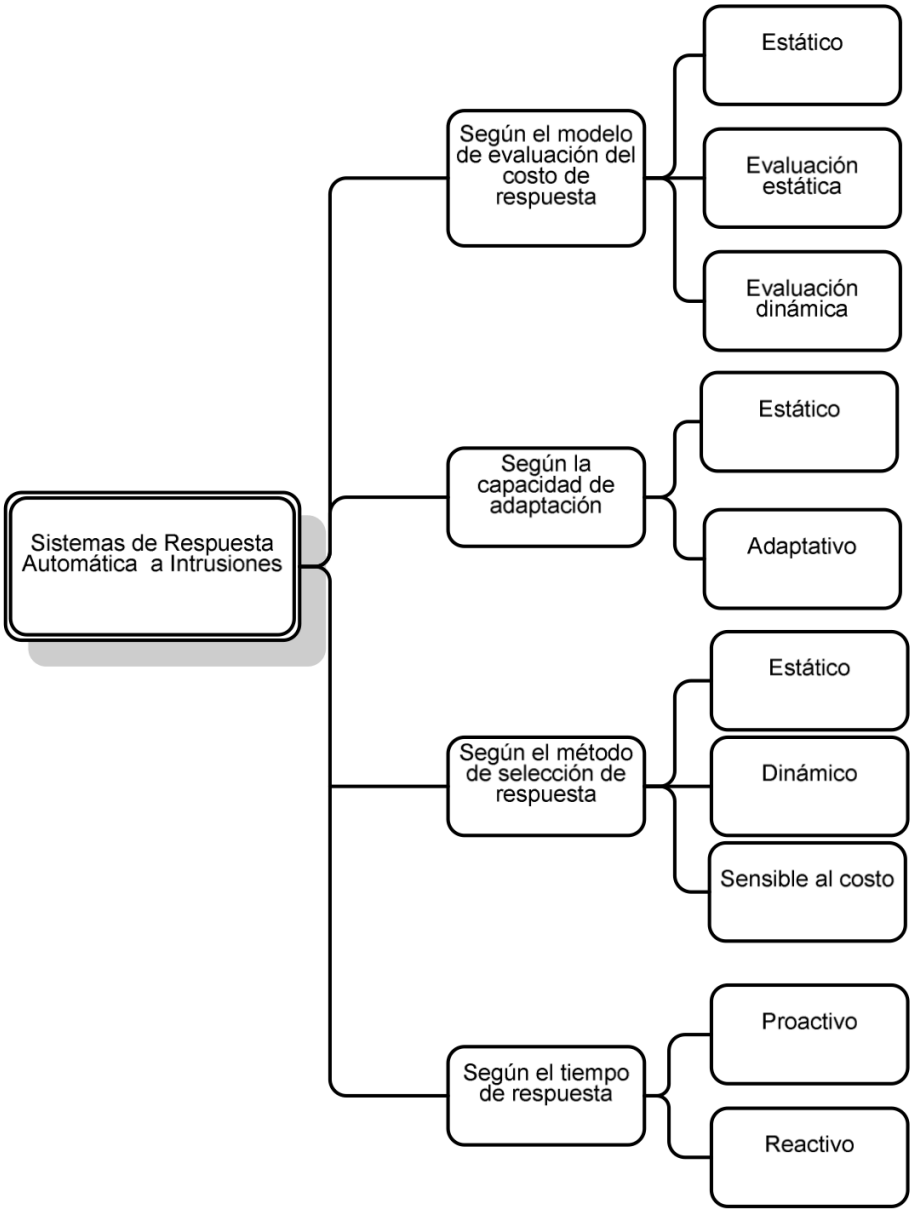

Ilustración 5. Estructura IRS

2.2 Amenazas Informáticas

Cuando hablamos de amenazas, hablamos de aquellos cambios del entorno por parte de un humano, una máquina o simplemente, un suceso que pueda comprometer la seguridad de la organización. Las amenazas pueden ser detectadas antes, después o durante el ataque. Para ello, las intervenciones posibles son:

- Prevención: se trata de mecanismos que ayudan a mejorar la seguridad en un funcionamiento rutinario.
- Detección: dispositivos encargados de revelar posibles violaciones de las políticas de seguridad
- Recuperación: mecanismos preparados para activarse cuando se produce un ataque y restaurarlo a su estado de funcionamiento normal.

Según las amenazas, podemos clasificarlas según:

Origen

- Amenazas Internas: pueden ser causadas por un uso incorrecto por parte de la plantilla o por personal técnico, que por motivos de necesidad en el trabajo, tienen acceso a partes críticas de la red. Los sistemas de prevención contra intrusiones o los cortafueos, no son eficientes contra amenazas internas, ya que no están guiados al tráfico interno.
- Amenazas Externas: son aquellas que proceden del exterior. No se tiene información evidente sobre la red y los atacantes deben primero entender como funciona para luego buscar una manera para asaltarla. El responsable informático de la red, tiene la oportunidad de prevenir de manera correcta este tipo de ataques.

Efecto

- Estafa, robo de dinero
- Suplantación de identidad
- Publicidad de datos personales
- DDOS de los sistemas
- Destrucción de información confidencial

Medio

- Virus: malware que intenta alterar la actividad normal del dispositivo, sin el consentimiento del usuario.
- Phising
- Ingeniería Social
- DDOS
- Spoofing

2.3 Nuevas formas de malware.

El nuevo malware, cuando se ejecuta en un equipo (o cuando tenemos un *Acces Point),* para poder comunicarse usa HTTP, ya que el servicio HTTP se encuentra abierto por defecto en todos los firewalls. El puerto 80 pues, es un objetivo perfecto para las intrusiones. El malware en si, no usa un navegador web, es decir, no se requiere de un navegador como Internet Explorer o Chrome para poder navegar. Lo que hace es usar librerías DLL especiales para realizar peticiones hacia fuera con peticiones en su mayoría GET y POST para comunicarse con el exterior.

Por ejemplo, una librería DLL maliciosa instalada en un PC, se podría comunicar a través del puerto 80 a un dominio C&C[4] y hacer una petición sospechosa a un *favicon*[5] de una web. Probablemente, este *favicon* realmente por dentro esté formado por un *.bat* [6] con código camuflado preparado para ser ejecutado. En caso de que un socket interno del .bat, genere una petición al puerto 80, esta nunca se realizaría a través del navegador y aquí reside la picaresca de la infección. Urllib[7] es un ejemplo de librería con la que se podría crear malware HTTP.

2.4 Soluciones posibles para contrarrestar el problema propuesto.

Una de las aplicaciones con mejor reputación y mas difundida en el entorno de la seguridad de la información es *Snort*, un famoso Sistema de Detección de Intrusiones que funciona tanto en UNIX como en Windows. Es una herramienta *Open Source*, gratuita, basada en firmas y que tiene como objetivo analizar el contenido de la red en busca de anomalías, comparando el tráfico real con las reglas guardadas en un repositorio de firmas.

Usa un motor de detección de posibles ataques, barridos de puertos o análisis de protocolos que alertará, registrará y responderá (opcionalmente) en caso de encontrar alguna anomalía definida con anterioridad. Esto se consigue definiendo patrones ya conocidos que pretendan aprovechar o encontrar una vulnerabilidad del sistema víctima.

El sistema basado en reglas permite un uso potente y sencillo del programa, en el que se pueden generar filtros contra backdoors, ataques ddos, escaneos con herramientas como NMAP, ataques web, etc. En el momento que el programa detecta alguno de estos comportamientos, genera una alerta en tiempo real.

---

[4] Command & Control es una forma de referirse a los servidores con la función de dar instrucciones al malware.
[5] Imagen pequeña que está asociada a una web a modo de mini-logo.
[6] Archivo de texto en el que se pueden cargar instrucciones de forma secuencial.
[7] https://docs.python.org/2/library/urllib.html

Snort tiene diversos modos entre los cuales se encuentran:

- *Sniffer*. este modo permite ver a través de la consola del sistema todo nuestro tráfico en tiempo real analizando al interfaz de red configurada en Snort.
- *Packet Logger*: genera *logs* para que sean analizados a posteriori.
- *IDS*: monitoriza toda la actividad de la red a través del fichero de configuración en el que residen los patrones y la base de reglas que actuarán de filtro para evitar todo posible ataque.

Otra opción puede ser el uso de Suricata, otro sistema de detección de intrusiones muy interesante que detecta los protocolos, es multi-hilo y permite la visualización estadística de datos. Además, es compatible con los sistemas de firmas de Snort.

Snort y Suricata pueden aceptar como entrada un paquete .pcap generado a partir de esnifers como *Wireshark*. Wireshark es una herramienta Open Source para el análisis de redes, en la que a partir de una interfaz de red, el programa consigue generar un listado de los paquetes que pasan a través de ésta, diferenciando claramente entre peticiones del cliente GET/POST/PUT/HEAD/… y los códigos de respuesta que devuelve el servidor pertinentes (200 OK, 404 NOT FOUND, etc)

Algunas peticiones pueden no ser legítimas. Y estos programas no soportan grandes cantidades de entrada de datos. Para prevenir este tipo de peticiones “paja”, el programa aportado deberá filtrar todas aquellas peticiones que no sean relevantes y dejar pasar las que realmente tengan un contenido interesante para la siguiente fase de la cadena, que en este caso es el IDS de la organización.

De esta manera, de todos los paquetes analizados, podríamos descartar gran cantidad de información de relleno, pudiendo ver claramente, posibles *malwares* que gracias a las firmas, deberán ser detectados.

En conclusión, la herramienta que aquí se presenta tendrá una función muy importante como pre-filtrado de un IDS y debería reducir el flujo de trabajo a los sistemas de análisis de detección de intrusos y con lo cual, aumentar su efectividad.

3. Análisis

3.1 Análisis de requisitos

El presente capítulo tiene por objetivo definir los requerimientos de un script que actúe como un pre-IDS y que se focalice en las anomalías que puedan surgir y otras acciones sospechosas.

La herramienta deberá poder leer contenido de un .pcap es decir, a partir de una muestra o sesión de usuario ya guardada, y deberá captar todas aquellas peticiones que pasen por el puerto 80 del servidor (*dport*). Se detectarán pues, aquellas peticiones HTTP que no estén en el código fuente de la web consultada.

La salida de la herramienta debe presentarse en dos partes: por un lado, se debe exportar un archivo de tipo ".pcap" en el que se encuentren los paquetes sospechoso. Por otra parte, se debe mostrar un log donde se vea cómo el usuario visita una web X y línea a línea, se muestren las peticiones realizadas.

En caso de encontrar paquetes de los que no se pueda demostrar su procedencia, se deberán almacenar para su posterior análisis por parte del IDS. Es decir, toda aquella navegación que se realice de manera lógica, no deberá ser trascendental, pero aquella que demuestre ser peculiar, será retenida para ser enviada al IDS. De ahí el título del trabajo "Detección de anomalías en HTTP trazando la sesión web de un usuario".

La herramienta recibirá el nombre de Analizador de Sesiones de usuario. Esta herramienta debe ser capaz de identificar peticiones realizadas por un usuario e ir agrupándolas. Al final del análisis, en vez de por ejemplo, tener un millón de líneas, sería interesante haberlas reducido a unas cien mil. Con lo cual, el volumen de información sería bastante menor del inicial (un 90% de ahorro respecto a la cantidad inicial).

Se requerirá de la librería Scapy para que proporcione funcionalidad a la hora de monitorizar en un puerto determinado. En este caso, HTTP será el servicio a examinar, así que el puerto a observar es el 80. La aplicación deberá pues, detectar todo aquel *payload* que pueda aparecer en el puerto 80 y agruparlo por webs visitadas.

Finalmente, se ha determinado con la empresa que:

- Se ha de desarrollar una herramienta que no requiere de interfaz gráfica, con lo que toda ejecución será por consola.
- La herramienta se realizará con el lenguaje de programación Python
- La salida del programa debe mostrar visualmente los resultados obtenidos al analizar la sesión del usuario y las peticiones realizadas comparándolas con el código fuente de las webs consultadas.
- Los resultados, a ser posible, se devolverán en formato de lista, donde un dominio raíz genere unas URLs y el analizador/herramienta detecte qué peticiones no han sido solicitadas.

3.2 Diseño de la herramienta

La herramienta requerida será un proyecto I+D+I, centrado en la parte de innovación ya que se pretende implementar un tipo de herramienta nunca usada. Después de realizar un pequeño análisis de requisitos, se ha determinado que para ser funcional, la herramienta requerirá de 4 grandes módulos:

- [opcional] Módulo *sniffing*: encargado de realizar la captación de todo el contenido que pase por una interfaz de red determinada. Gracias a librerías de terceros y de programas de libre uso, se podrán realizar funciones de esnifado de paquetes en la red (Wireshark). El resultado debería ser un archivo *.pcap*.

- Módulo de adquisición: encargado de recoger el .pcap generado en el módulo anterior. Descartará aquellos paquetes que no sean HTTP a la par que descarga el código fuente de las webs consultadas por el usuario.

- Módulo de filtrado: preparado para realizar *matching* entre los recursos solicitados en el código fuente de la web y las peticiones HTTP del *.pcap*.

- Módulo de Resultados: que exportará un *.pcap* con los resultados de manera que se guarden los paquetes que no cumplan el *matching* anteriormente nombrado en un *.pcap*. Por otra parte, la salida del programa deberá mostrar los resultados de los paquetes sospechosos y los que no.

Para que se entienda esto, se propone la siguiente estructura:

- Paquetes de http://www.ebay.es/
- recurso solicitado 1
- recurso solicitado 2
- RECURSO NO SOLICITADO
- recurso solicitado 3
….
- Paquetes de http://www.rollingstones.com/
- recurso solicitado 1
- recurso solicitado 1
….

El recurso de color rojo, podría ser una petición realizada por un *malware* en un equipo y que la herramienta lo hubiese detectado. En ese caso, se deberá guardar ese paquete para su posterior análisis por parte del IDS.

Para poder llevarlo a cabo, se recomienda hacer uso de una arquitectura multicapa: una arquitectura multicapa se basa en un conjunto ordenado de sistemas y subsistemas donde cada parte tiene definida una función. Los objetos de cada una de las capas, normalmente son independientes, aunque pueden existir dependencias entre objetos de diferentes capas.

La separación por capas ofrece independencia entre niveles. Así, el mantenimiento y posibles ampliaciones de la aplicación que puedan ser necesarias en un futuro, se podrán realizar de manera limpia, rápida y sencilla.

El diseño de la aplicación se basa en una arquitectura multicapa, concretamente en una arquitectura de tres capas generales. Las tres capas generales son:

- Presentación
- Lógica
- Persistencia.

Ilustración 6. Arquitectura multicapa

Capa de presentación (GUI): También se la conoce como interfaz gráfica. Es la capa encargada de recoger los datos del usuario a la par que presenta resultados una vez los ha manipulado la capa lógica.

En nuestro módulo, no se dispone de una interfaz gráfica como tal, ya que la empresa, ha solicitado explícitamente que desea que la aplicación, por ahora, sea accesible por terminal.

Capa de negocio: Proporciona funcionalidad al módulo. Es la capa encargada de realizar operaciones a nivel de aplicación. Su funcionamiento hace posibles todas las operaciones que se han comentado en los apartados anteriores. Se encarga también de la conexión con la base de datos.

En nuestro caso, la capa lógica se encarga de recoger el conjunto de paquetes, de la comprobación del código fuente, del maching entre ambos y del posterior filtrado de malwares.

Capa persistencia o de datos: Capa dedicada al almacenamiento de toda la información de nuestra aplicación. Se asegura también, el acceso a la información de una manera segura y lo más importante: controlada.

En nuestro caso, la capa de persistencia deberá almacenar el archivo *.pcap* que contendrá aquellos paquetes interesantes para su revisión posterior.

3.3 Herramientas necesarias:

- **Python**: Python es un lenguaje de programación interpretado usa tipado dinámico y es multiplataforma. Se ha usado este lenguaje por su simplicidad de código y la facilidad a la hora de crear potentes herramientas con poco código.

- **Pycharm**: se trata de un IDE preparado para programar con Python. Proporciona análisis de código, un depurador gráfico, un medidor de unidad integrada, integración con sistemas de control de versiones,y soporta desarrollo web con Django. PyCharm es desarrollado por la empresa checa “JetBrains”.

- **Libnids**: librería preparada para ser usada en C y en Python que permite re-ensamblar los paquetes TCP, presentando toda la comunicación que ha habido entre un *host* **A** y un *host* **B**, junto con sus *payloads*. Permite realizar capturas como conocidos programas como *Wireshark* o *TCPDump*.

- **Pynids**: librería IDS especian de python, llamada como *wrapper* de libnids, preparada para esnifar paquetes, desfragmentar contenido IP y para streaming TCP.

- Beautiful Soup: biblioteca de Python ideal para parsear documentos HTML. Se puede sacar información

- Scapy: se trata de un gestionador de paquetes en red con el que podremos controlar las interfaces de red. Gracias a esta herramienta, se puede realizar un testeo de la seguridad de una red.

- Termcolor: gracias a esta librería, podremos sacar por pantalla resultados en colores para poder diferenciar mejor visualmente, entre paquetes permitidos y paquetes ilícitos.

4. Implementación

4.1 Entorno de desarrollo

Para la implementación de la aplicación, se ha procedido a generar una herramienta con las siguientes características:

Característica	Entorno de desarrollo
Sistema operativo	Ubuntu 14.04 LTS
Lenguaje de programación	- Python 2.7 - Scapy 2.3.1 - Beautiful Soap - Termcolor
Máquina	MacBook Pro 2014

La primera versión de este prototipo que más adelante se podría ampliar, consigue recoger una sesión de usuario (en formato .PCAP), en la que éste haya navegado por páginas *index* de diferentes dominios. Para ello, y como se ha comentado anteriormente, se ha usado una arquitectura multicapa muy básica separada en tres fases: interfaz, lógica y base de datos.

La primera capa, interfaz, es la encargada de dar la bienvenida al usuario, guiándolo a través de una serie de opciones:

1. **Seleccionar un .pcap por defecto**
2. **Seleccionar un .pcap personal**
3. **Salir**

La segunda capa, la lógica, tiene como entrada el *.pcap* elegido, ya sea el de por defecto o el personalizado. A través de esta capa, la herramienta realiza las siguientes fases:

1. Recibe con parámetro un .pcap por parte de la interfaz.
2. Gracias a un filtrado, selecciona solo aquellos paquetes HTTP.
3. Busca los hosts de todas las peticiones GET
4. Se realiza una serie de operaciones en las que se separan los “hrefs” y “src” del código fuente, que contienen peticiones al servidor de datos que necesitan.
5. Se hace un matching de los paquetes que hay en el .pcap con los “hrefs“ en busca de coincidencias.

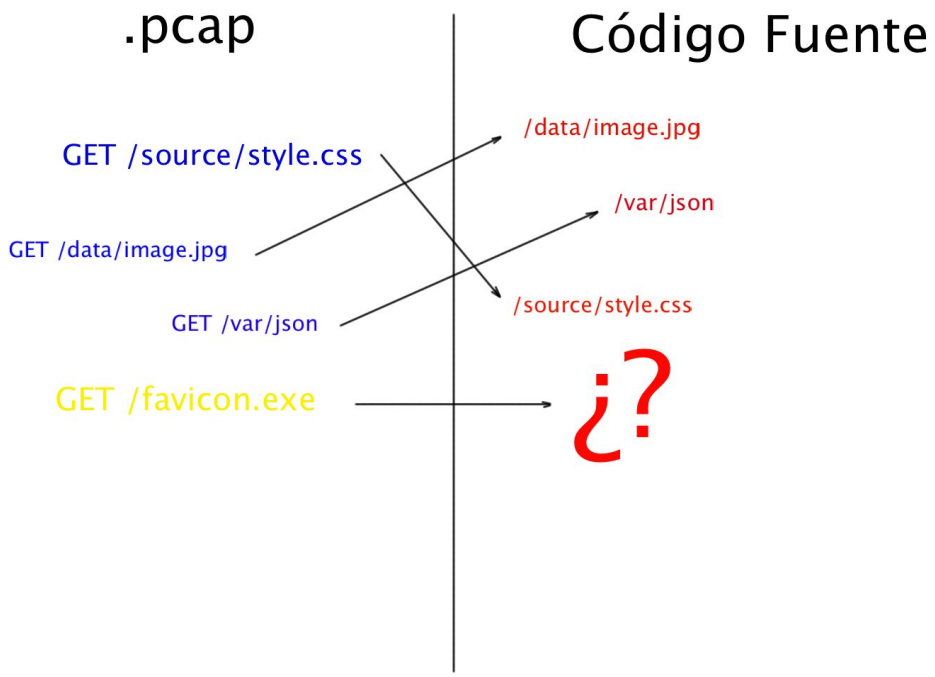

Ilustración 7. Proceso de Matching

6. Todas aquellas coincidencias que no se consigan parear, se considerarán como anomalías.

La última capa, la de base de datos, se ha hecho sin el uso de SQL ya que lo que se pretende obtener son archivos .pcap preparados para ser transmitidos al IDS.

4.2 Test de la herramienta (Pruebas)

Todos los pasos que se muestran a continuación están explicados también en un README adjunto al proyecto.

Para iniciar el programa, se debe disponer de las librerías antes mencionadas en el punto de “Herramientas necesarias”. El comando necesario para iniciar la aplicación es:

$ python interfaz.py

Una vez iniciada la aplicación, se cargará una pequeña interfaz de bienvenida:

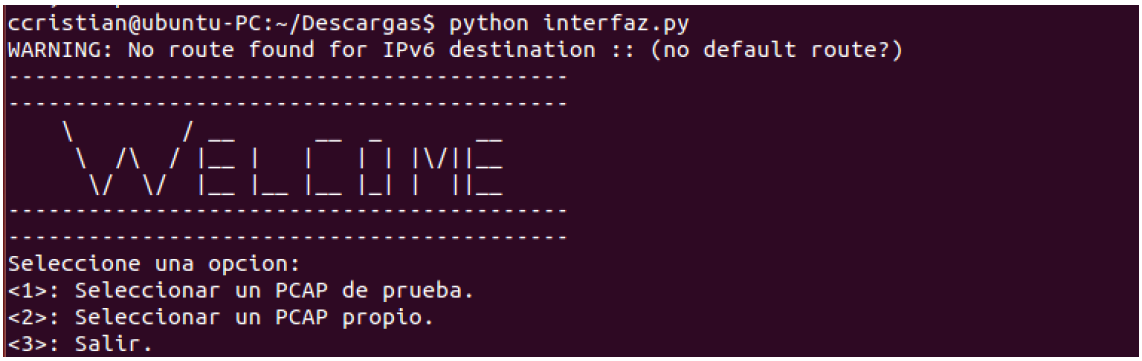

Ilustración 8. Interfaz Herramienta - Inicio

Seleccionamos la opción por defecto (1) para que muestre los resultados de un .pcap que ha guardado la sesión de un usuario, en la que ha accedido a un blog [androasia.es] y luego ha hecho una petición GET a una IP aleatoria a modo de prueba como si de un virus se tratase [IP: 111.111.111.111]:

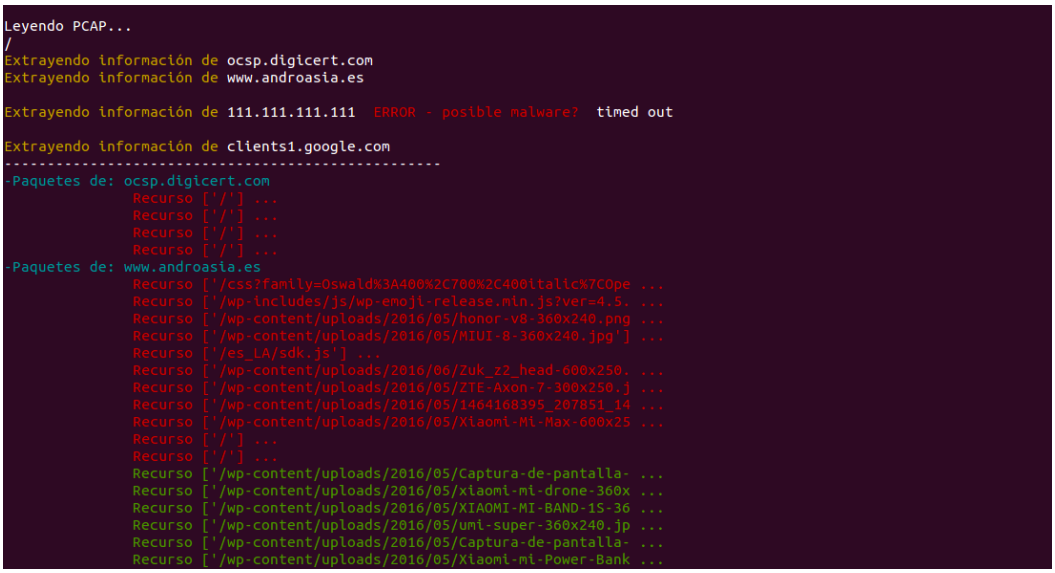

Ilustración 9. Interfaz Herramienta - Selección de PCAP

Si se desea analizar un .pcap alternativo, existe para ello la opción 2 del programa. En este caso se le pasa como parámetro al programa la ruta en la que se encuentre el .pcap:

```
WARNING: No route found for IPv6 destination :: (no default route?)
----------------------------------------
----------------------------------------
    \        / __        __  _       __
     \  /\  / |__ |   |   |  | |\/||__
      \/  \/  |__ |__ |__ |_| |  ||__
----------------------------------------
----------------------------------------
Seleccione una opcion:
<1>: Seleccionar un PCAP de prueba.
<2>: Seleccionar un PCAP propio.
<3>: Salir.
>2
Introduzca la ruta del pcap de la siguiente manera:
Usage -> /ruta/*.pcap
>>/home/ccristian/Documentos/pcap2.pcap
```

Ilustración 10. Interfaz Herramienta - PCAP Alternativo

4.3 Generación de resultados

Los resultados obtenidos como salida del programa, se guardan en un archivo .pcap [suspicius_requests.pcap] que puede ser leído por Wireshark, como se puede comprobar en la siguiente imagen:

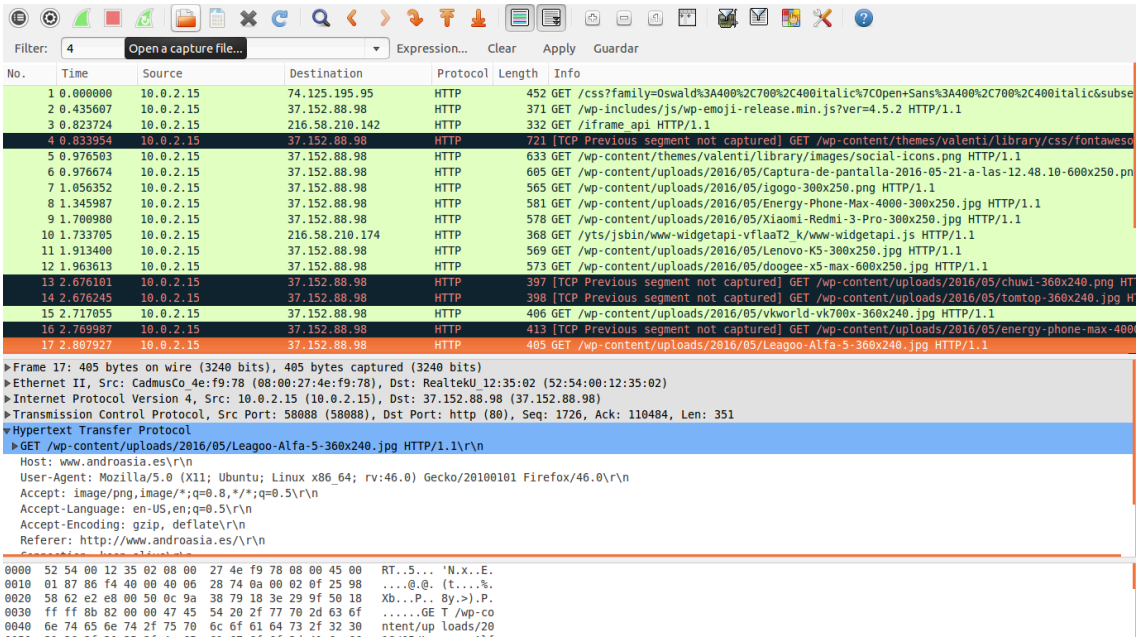

Ilustración 11. PCAP resultante

Si buscamos, podemos ver que en el paquete 16 se encuentra nuestra petición que será la “anomalía” que buscábamos:

Ilustración 12. Paquete anómalo detectado

Como se puede comprobar, de 125 paquetes HTTP que habían en el .pcap por defecto, nos encontramos con tan solo 23 paquetes. Esto significa una reducción del 50% en el número de *request* y del 82% en el número total de paquetes HTTP.

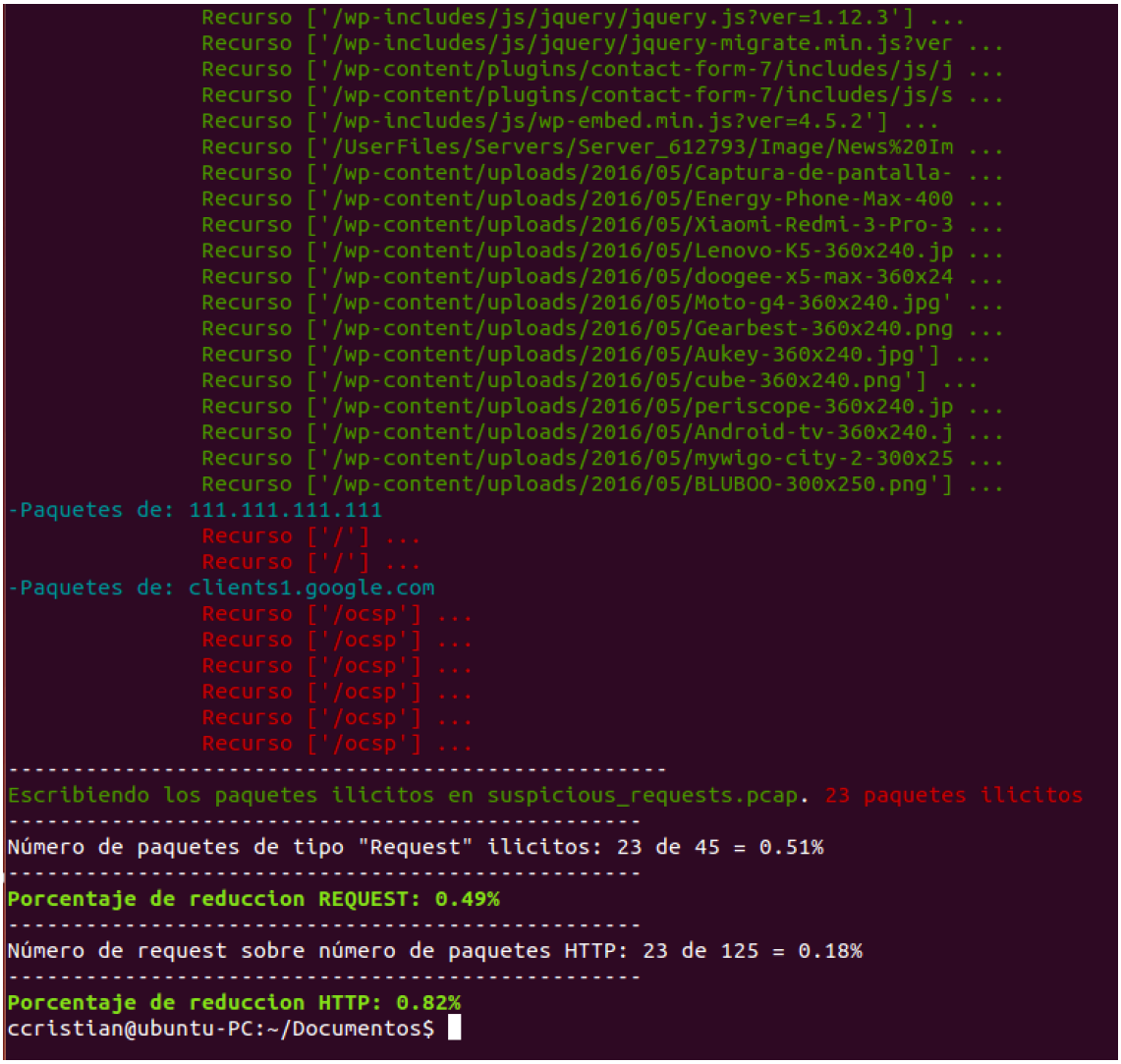

Ilustración 13. Salida por pantalla del programa

Con estos números en la mano, se puede decir que con este programa “filtro”, reducimos la carga de trabajo de los paquetes HTTP que recibe el IDS.

También se aporta un script que se encarga de realizar el análisis a través de esta herramienta y después usa el .pcap saliente para pasárselo al Snort.

4.4 Propuestas de ampliación, líneas de trabajo

Entre las posibles mejoras que podría recibir el proyecto que proporciona este proyecto, estaría la de recoger el tráfico y analizarlo al momento en vez de realizar análisis sobre archivos *.pcap.* Por otra parte sería interesante tener el programa distribuido en diferentes células (Raspberry Zero). Se van a explicar a continuación ambas propuestas unidas:

Hasta ahora se había hablado de realizar análisis sobre archivos que contenían paquetes en la red (.pcap), en el que el contenido ya ha sido extraído de la red y se pretende analizar a posteriori para ver qué contenido era malicioso. Esto muestra una parte negativa. Si una empresa decidiese realizar un guardado de toda la información que pasa por la red durante por ejemplo, 30 días, las estimaciones de almacenamiento se calcularían según la siguiente regla:

Almacenamiento en disco duro por un día = Media de uso de la red x 1 byte x 60 segundos x 60 minutos x 24 horas

En caso de querer hacer un cálculo de una empresa mediana, la media estaría en 100 Mbps, con lo cual = 100 Mbps x 1 byte x 60 segundos x 60 minutos x 24 horas = 1,080,000 MB al día => 1,08 TB diarios => **33,5 TB al mes**.

Como se ha comentado anteriormente, los sistemas de detección de intrusiones no están preparados para realizar análisis de flujos tan elevados como aquí se muestran. Es por ello que muchas veces hay que buscar otras opciones para conseguir capturar todo el tráfico de una organización.

Por último, la opción que queda, aunque bastante más cara, sería la del uso de módulos de red como los PF_RING[8], capaces de capturar paquetes con un nuevo sistema de rotación en el cual se adquiere un paquete en red y se pasa a ejecución a través del kernel (usando CPU) a través de la entrada NIC. Los datos marcan que son capaces de analizar un mínimo de 750 kpps para un Core2Duo de 1.86 GHz. Teniendo en cuenta que un ordenador normal genera entre 3 y 6 kpps y que una mediana empresa tiene entre 200 y 300 ordenadores, se puede comprobar que es la mejor opción para poder controlar todo el tráfico en red.

Como se ha visto, si se desea implantar un sistema IDS para detección de anomalías en organizaciones de tipo PYMEs, los costes de Hardware pasan por diferentes precios. En el primer caso, los costes de usar Raspberrys Zero ejecutando scripts distribuidos en python para reducir la carga sería una opción. Esto tendría un coste muy barato en contraposición de los precios altos que tendría un sistema PF-RING

[8] http://www.ntop.org/products/packet-capture/pf_ring/

6. Conclusiones

Una vez finalizado el trabajo, las conclusiones han sido:

- Hoy en día, el mundo empresarial requiere de comunicación entre organizaciones: gestiones en internet, emitir y recibir correos electrónicos.
- La evolución en la red y las tecnologías emergentes implica un estudio paralelo sobre las necesidades en seguridad de los clientes con la intención de disponer de la mayor protección posible.
- Las redes necesitan ser examinadas. Requieren de un estudio muy meticuloso y continuo en lo que a sistemas de detección y prevención de intrusiones se refiere.
- La seguridad no es solo lógica, sino también física, ya que el objetivo de ambos es el mismo: defender y ser defendido.
- Se ha entendido la importancia de los sistemas de detección de intrusiones en las organizaciones y la función tan fundamental que tienen a la hora de prevenir posibles ataques.
- Luego de realizar una revisión del estado del arte y de ciertos conceptos relacionados con la materia, se ha realizado una herramienta que reduce la saturación de dichos sistemas.
- Con la herramienta finalizada, las pruebas han demostrado que realmente era un apoyo a los IDS, ya que elimina una gran cantidad de paquetes a analizar y optimiza el trabajo de éste.
- Se ha seguido la planificación del trabajo con alguna semana de retraso.
- En un futuro, sería interesante la idea de modificar el programa para que en vez de coger archivos guardados en .pcap, lo hiciese en “raw” es decir, en tiempo real, filtrando toda la información al tiempo que le llega al IDS.
- Para comprobar la efectividad del producto final, se han realizado una serie de pruebas con diferentes .pcaps, y se ha obtenido que el programa, de X paquetes de tipo HTTP, conseguía reducir la información sustancial de un 20 a un 50% de su valor total. Se trataría pues de una reducción bastante importante y que sería ideal en caso de querer mejorar la eficiencia del IDS.
- Implementar pues, soluciones de seguridad de este tipo, asegura comunicaciones más fiables, mayor tranquilidad para el cliente hacia amenazas externas y prevención contra el hurto o perdida de información.

Agradecimientos:

Las expectativas han ido mejorando poco a poco conforme iban saliendo resultados y creo que toda la inversión de tiempo ha valido la pena y que este año en la UOC me ha servido para aprender y adquirir conocimientos. Espero que dichos conocimientos me sirvan en el entorno laboral en el que me hallo y que día a día, aprenda cada vez más sobre Seguridad en las TIC.

7. Glosario

ANOMALÍA: cualquier acción que desentone en un comportamiento típico o normal.

CARMEN: software desarrollado por la empresa S2 Grupo que hace frente a Amenazas Persistentes Avanzadas.

ESNIFAR: capturar

HTTP: protocolo de transferencia de hipertexto

HREF: referencia de hipertexto. Usado en el código fuente HTML para solicitar recursos de otras webs.

IDS: sistema de detección de intrusiones

IP: Internet Protocol

IPS: sistema de prevención de intrusos

NIDS: sistema de detección de intrusos en una red

PCAP: formato de los paquetes que han sido capturados con la herramienta Wireshark.

SNORT: IDS preparado para ser usado con reglas

SURICATA: IDS preparado para ser usado con reglas

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

URL: Uniform Resource Locator

WIRESHARK: herramienta especial preparada para esnifar paquetes en una determinada interfaz.

8. Bibliografía

- The Practice of Network Security Monitoring, Richard Bejtlich, San Francisco, 2013

- Auditoría Informática: Un enfoque práctico, 2a edición, Mario G Piattini, Emilio del Peso, Ra-Ma, Madrid, 2001

- Hackers 2: Secretos y soluciones para la Seguridad de redes, Joel Scambray, Stuart McClure, George Kurts, Mc Graw Hill, Madrid, 2001

- Network taps, D. Gonzales-Gómez. http://www.dgonzalez.net/papers/roc_es/node4.html, 2-05-16

- IDS or IPS: what is best?, M. Papadaki and S. Furnell, Network Security, vol. 2004,

- Implantación de un Sistema de Detección de Intrusos en la Universidad de Valencia. Ingeniería Informática. Emilio José Mira Alfaro. Universidad de Valencia 2002. http://www.rediris.es/cert/doc/pdf/ids-uv.pdf, 8-05-16

- Sistemas de Detección de Intrusos. Revista del Instituto Tecnológico de Informática. http://www.iti.es/media/about/docs/tic/06/2005-02-intrusos.pdf, 15-05-16

- Seguridad en Redes y Telecomunicaciones. Tipos de ataques en la red http://www.slideshare.net/alexpolanco1/tipos-de-ataques-en-la-red-alex-anny-dilannia-sixta-y-virtudes , 16-05-16

- Sistemas de detección de intrusos. Antonio Villalón Huerta. Universidad Politécnica de Valencia. Mayo 2005.

- Snort. SNORT users manual 2.9.8 https://s3.amazonaws.com/snort-org-site/production/document_files/files/000/000/100/original/snort_manual.pdf?AWSAccessKeyId=AKIAIXACIED2SPMSC7GA&Expires=1464728322&Signature=47eSfG0yCURkZLpBxaz8fLkWWfY%3D , 25-05-16

9. Anexos

9.1 Manual de instalación

Para instalar las librerías pertinentes, se debe proceder a realizar las siguientes indicaciones:

1. Instalación de Scapy:

   wget scapy.net
   unzip scapy-[version].zip
   cd scapy-[version]
   sudo python setup.py install

2. Instalación de Termcolor:

   https://pypi.python.org/pypi/termcolor

3. Instalación de Beautiful Soup:

   pip install beautifulsoup4
   ó
   apt-get install python-bs4

4. Instalación de Snort:

   wget https://www.snort.org/downloads/snort/snort-2.9.8.2.tar.gz
   tar xvfz snort-2.9.8.2.tar.gz
   cd snort-2.9.8.2.tar.gz
   ./configure --enable-sourcefire && make && sudo make install
   ó
   sudo apt-get install snort