Una aproximación basada en Snort para el desarrollo e implantación de IDS híbridos

J.E. Díaz-Verdejo, P. García-Teodoro, P. Muñoz, G. Maciá-Fernández, F. De Toro
Departamento de Teoría de la Señal, Telemática y Comunicaciones. Universidad de Granada
ETSI de Ing. Informática. C/ Daniel Saucedo Aranda S/N
18071 - Granada
Teléfono: 958 24 23 04 Fax: 958 24 08 31
E-mail: jedv@ugr.es

Abstract *Apart from the modeling techniques, the development and deployment of anomaly-based intrusion detection systems still faces two main problems. The first one is related to the acquisition and handling of real traffic to be used for training purposes. The second one concerns the better performance of signature-based IDS for known attacks. In this paper the authors propose the use of a modified version of Snort which results in a hybrid detector/classifier. This version can be used both during the training phase of the anomaly-based system and as a deployed hybrid detector and traffic sniffer. Furthermore, it can be adjusted to work just as signature-based, anomaly-based or both (hybrid) detector. On the other hand, this version can be used to directly sniff, classify and split the network traffic according to its malicious nature, which eases the problems related to the acquisition and handling of training traffic.*

1. Introducción

Los sistemas de detección de intrusiones (IDS, del inglés *Intrusion Detection Systems*) constituyen un tema de investigación y desarrollo de enorme relevancia en el contexto actual, dada la gran penetración de las redes de ordenadores, en general, y de Internet, en particular, en las actividades cotidianas. Esta expansión de Internet ha venido aparejada a un fuerte incremento en el número de incidentes de seguridad [1], con el consiguiente impacto, tanto económico como tecnológico, en las actividades y servicios desempeñados. Resulta evidente, en consecuencia, la necesidad de disponer de técnicas y sistemas de protección, prevención y mitigación de fallos frente a actividades maliciosas. Entre estos sistemas se encuentran los ya mencionados sistemas de detección de intrusiones, cuya finalidad es detectar actividades maliciosas en un sistema o red de ordenadores y alertar a los administradores para que desencadenen los mecanismos de defensa que se consideren oportunos [2] [3]. Una evolución de estos sistemas son los denominados IPS (*Intrusion Prevention Systems*), que incluyen la capacidad de respuesta automática frente a incidentes sin necesidad de intervención humana. Aunque el presente trabajo se centra en los IDS, los problemas, técnicas y procedimientos que se abordarán también son de aplicación a los IPS.

El despliegue de los IDS requiere de la resolución previa de varios problemas de índole tecnológica y legal. Entre ellos, el más relevante está relacionado con los métodos de detección de actividad maliciosa a utilizar. Atendiendo a estos, los IDS se suelen clasificar en basados en firmas (*signature-based*, S-IDS) o basados en anomalías (*anomaly-based*, A-IDS) [4] [5]. Los primeros utilizan una base de datos de conocimiento (firmas o reglas) que incluye los patrones de actividad de comportamientos maliciosos conocidos, de tal forma que cualquier actividad detectada que siga algunos de los patrones contenidos en la base de datos es etiquetada como maliciosa, actuándose en consecuencia. Por el contrario, los sistemas basados en anomalías se fundamentan en la construcción de un modelo de la actividad normal (o anormal en algunos casos) del sistema o red monitorizado, de tal forma que cualquier desviación de dicha actividad es etiquetada como sospechosa o *anómala* y asimilada a un comportamiento malicioso (hipótesis de sospecha). Ambos tipos de sistemas presentan ventajas e inconvenientes que hacen que ninguna de las dos soluciones sea claramente superior a la otra. Así, los S-IDS resultan más fiables y proporcionan mejores rendimientos frente a ataques conocidos. Sin embargo, su capacidad para detectar nuevos ataques no incluidos en la base de datos de firmas es prácticamente inexistente. Por el contrario, los A-IDS presentan la capacidad de detectar ataques previamente desconocidos, aunque su rendimiento resulte, con la tecnología actualmente disponible, inferior. En cualquier caso, y sin desdeñar la necesidad de proteger los sistemas frente a ataques previamente observados, resulta de vital importancia disponer de sistemas que sean capaces de reaccionar ante el desarrollo de un nuevo ataque (*0-day attacks*), al resultar éstos los más dañinos

precisamente por la ausencia de defensas preestablecidas.

En este contexto, en el presente trabajo se presenta un IDS híbrido basado en red (NIDS, *network-based IDS*) desarrollado a partir de la incorporación de módulos adicionales a Snort [6] [7] que, adicionalmente, puede operar como clasificador del tráfico capturado en las fases de desarrollo del sistema. El sistema resultante puede operar, según la configuración utilizada, en modo S-IDS (operación normal de Snort), en modo A-IDS o en modo de detección híbrida (que denominaremos Hy-IDS)[1].

Otro problema que es necesario resolver, especialmente en el caso de los A-IDS, es la disposición de trazas de tráfico capturadas en un entorno real que posibiliten el análisis de las actividades observadas, maliciosas o no, con la finalidad de extraer firmas de ataques y/o entrenar los modelos de normalidad, según el caso [8] [9]. Con independencia de los problemas legales que se planteen, relacionados con la privacidad de los datos, y centrándonos en el caso de los A-IDS, resulta de interés disponer de metodologías y técnicas que permitan una clasificación automática del tráfico en diversas categorías en función de sus características y las necesidades de desarrollo. Así, a modo de ejemplo, será interesante disponer de tráfico etiquetado como normal y tráfico etiquetado como ataque para el desarrollo de los modelos de normalidad (entrenamiento de los modelos). A este respecto, los autores han propuesto recientemente una metodología que permite abordar, con las garantías adecuadas, el entrenamiento y ajuste de los modelos de normalidad a partir de tráfico real [10]. Esta metodología requiere del uso de herramientas que permitan capturar y clasificar el tráfico obtenido de forma automática, para lo que se ha considerado incluir dicha funcionalidad en las modificaciones a realizar en Snort.

El presente trabajo se estructura en los apartados que se describen a continuación. En el Apartado 2 se aborda la problemática relacionada con el desarrollo de A-IDS, especialmente en lo que concierne a la gestión de tráfico real y entrenamiento de los modelos de normalidad. En el Apartado 3 se proponen y describen una serie de modificaciones y módulos adicionales para Snort que proporcionan las funcionalidades requeridas para el presente trabajo: clasificación de tráfico, operación como A-IDS y operación como IDS híbrido. El sistema resultante se aplica en el Apartado 4 a un caso de estudio, consistente en la implantación de una técnica de detección de anomalías desarrollada por el grupo de investigación de los autores, denominada SSM [11], que es aplicada y evaluada en un servicio HTTP. Finalmente, se presentan un resumen de las características más relevantes del sistema desarrollado y las conclusiones.

2. Desarrollo de IDS basados en anomalías

El desarrollo de IDS basados en anomalías presenta la peculiaridad, frente al basado en firmas, de necesitar un conjunto de datos a partir de los que obtener los modelos de normalidad en los que se fundamenta la detección. Este problema dista de ser de solución trivial, debido a las múltiples características que debe presentar dicho conjunto de datos. Estas características, que serán abordadas brevemente en el apartado siguiente, incluyen la recomendación de que los datos a utilizar correspondan a tráfico real capturado en redes en explotación [12] [9], lo que genera los problemas legales relacionados con la privacidad de los datos anteriormente mencionados. En consecuencia, aparecen dos efectos importantes relacionados con el desarrollo de este tipo de sistemas: la dificultad de obtención de datos adecuados y la imposibilidad práctica de comparar las diferentes técnicas y sistemas desarrollados, al no ser posible compartir los datos de entrenamiento y evaluación.

2.1. Metodología de desarrollo basada en tráfico capturado

Para paliar estos problemas, los autores han propuesto recientemente una metodología [10] para la captura y acondicionamiento de los datos que permite el entrenamiento, evaluación y validación de los sistemas de forma adecuada. Esta metodología se basa en el establecimiento de particiones en la base de datos de tráfico capturado de acuerdo a dos criterios principales: su carácter normal, anómalo o de ataque y su uso en el desarrollo del sistema, esto es, entrenamiento, test y validación. En la Fig. 1 se muestra un ejemplo de las particiones resultantes en la base de datos en el caso de entrenar un sistema híbrido (basado en firmas y en anomalías) sin contemplar la validación del sistema (particiones para entrenamiento y test), resultando en 6 particiones: NormE (tráfico normal para entrenamiento), NormT (tráfico normal para test), AnE (tráfico anómalo para entrenamiento), AnT (tráfico anómalo para test), AtE (tráfico de ataque para entrenamiento) y AtT (tráfico de ataque para test). Evidentemente, según el sistema a desarrollar, algunas de las particiones pueden resultar inútiles (e.g., AtE y AnE en este caso), aunque es necesario definirlas para preservar la representatividad de los resultados obtenidos (problema de sesgos debidos a falta de proporcionalidad [12] [13]). En cualquier caso, pueden usarse técnicas de tipo *Leaving-k-out* [14] para aumentar la representatividad de los datos

[1] Se propone el acrónimo *Hy-IDS*, del inglés *Hybrid IDS* en lugar de H-IDS debido a que en la bibliografía se reserva este término para los IDS basados en el ordenador (*host*)

y posibilitar un mayor aprovechamiento de los mismos, al poder usarse todos los datos obtenidos.

En cualquier caso, uno de los problemas prácticos que se plantean, una vez capturado el tráfico real, es la categorización del mismo como normal, anómalo o ataque. A este fin, en [10] los autores proponen el uso de un detector basado en firmas con diferentes versiones de reglas o firmas (actualizadas y no actualizadas) de la siguiente forma (Fig. 2, bloque *captura/clasificación de tráfico*). En primer lugar, el trafico capturado por el programa de captura (*sniffer*) elegido, *DB.cap*, se clasifica utilizando un conjunto no actualizado de firmas, de tal forma que se obtienen los paquetes que generan alguna alerta, que serán los paquetes de ataque (*DB-ataque.cap* en la figura), y se vuelven a procesar los restantes (*DB-interm.cap*) con un conjunto actualizado de reglas. Este segundo filtrado separará los paquetes normales (los que no activan ninguna firma), *DB-normal.cap*, de los que se considerarán anómalos (los que activan alguna firma), *DB-anom.cap*. Posteriormente será necesario establecer las particiones de entrenamiento, test y validación (no todas mostradas en la figura) para el desarrollo global del sistema. Esta filosofía de operación está en consonancia con el desarrollo de detectores híbridos, al considerar que los ataques que corresponden con firmas conocidas son detectados por el módulo de firmas sin ninguna dificultad y centrar el desarrollo del módulo de anomalías en los ataques de reciente aparición que, es de suponer, serán los que marcarán la tendencia tecnológica de los nuevos ataques.

A partir de los bloques de datos capturados se procederá al entrenamiento y evaluación de los detectores.

La aplicación de esta metodología en un entorno real requiere de la disposición de un IDS basado en firmas que sea capaz de clasificar y separar el tráfico en dos bloques con suficiente fiabilidad, de acuerdo a las firmas de ataques, y de un IDS híbrido capaz de combinar ambos mecanismos de detección. Las herramientas disponibles carecen, en la actualidad, de la funcionalidad requerida, ya que los IDS existentes únicamente alertan de un ataque y, en algunos casos, lo filtran eliminándolo de la salida cuando operan a modo de filtro de entrada al sistema. Aunque cabría desarrollar un sistema específico, resultaría más adecuado adaptar un IDS basado en firmas ya disponible para que incorporase estas funcionalidades. De esta forma, la operación como detector basado en firmas estaría respaldada por la operación habitual de dicho detector (disponibilidad y fiabilidad de los conjuntos de reglas/firmas). Un sistema que reúne las características necesarias (disponibilidad del código fuente junto con la fiabilidad y disponibilidad de las reglas) es Snort [7].

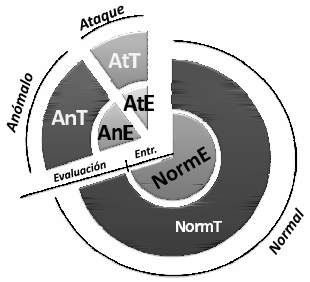


Figura 1: Particionado inicial del tráfico capturado para el desarrollo de IDS [10].

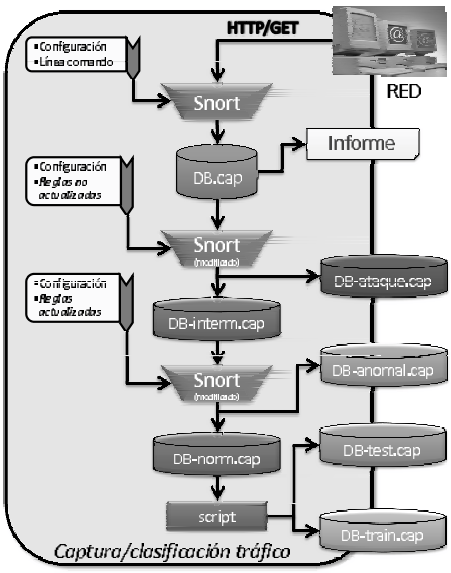


Figura 2: Metodología para la captura y clasificación de tráfico real basada en Snort.

3. Adaptaciones de Snort

Snort es un detector de intrusiones basado en red que utiliza el paradigma de detección mediante firmas (S-IDS). Es de dominio público, de código abierto y de amplia implantación, por lo que existen bases de datos de firmas de ataques que se actualizan con frecuencia y que presentan las garantías suficientes para su utilización. Las bases de datos de firmas presentan varias versiones, dependiendo de su procedencia, destacando las denominadas *VRT, Vulnerability Research Team*, de carácter cuasi-oficial, y las *community rules*, constituidas por aportaciones de los usuarios de Snort. Ambos conjuntos de reglas están disponibles en la página oficial de Snort (*http://www.snort.org*), existiendo numerosos conjuntos de reglas aportados por otros organismos o individuos. Las reglas

VRT son verificadas y comprobadas por un equipo de trabajo, garantizándose así un nivel de calidad adecuado.

La operación de Snort sobre los paquetes capturados es secuencial, aplicándose sucesivamente diversos módulos, de acuerdo al esquema mostrado en la Fig. 3.a). En primer lugar, tras su captura, los paquetes son analizados por diversos módulos decodificadores y preprocesadores cuya finalidad básica es preparar los paquetes para su análisis (normalización, detección de flujos, etc.). A continuación son procesados por los motores de detección que, en función de las reglas activas, van activando secuencialmente, en el orden establecido por las reglas, los diferentes módulos detectores asociados a las mismas. De esta forma, tras ser verificado el cumplimiento o no de alguna o varias de las reglas, lo que haría que el paquete fuese considerado como de ataque, se envía a los módulos posprocesadores indicándose los códigos de las reglas activadas. La finalidad de los posprocesadores es presentar los datos a la salida bien mediante la generación de los oportunos informes y estadísticas, bien mediante el filtrado de los paquetes de ataque, cuando opera en el denominado *modo inline* [15]. Es importante en este punto resaltar que, a la salida, sólo es posible obtener un informe de actividad o los paquetes que no han activado ninguna regla y, por tanto, se pueden considerar inocuos.

Las funcionalidades adicionales requeridas relativas a la detección de anomalías serán implementadas en forma de módulos de posprocesado, como se muestra en la Fig. 3.b), ya que deben operar sobre los resultados de detección obtenidos a partir de las firmas para no interferir en el proceso habitual. Como se comentará posteriormente, esta aproximación aporta ventajas adicionales al posibilitar un modo de operación híbrido. Por otra parte, la capacidad de generación de dos flujos de tráfico de salida para separar el tráfico normal y el de ataque debe considerarse, obviamente, en las etapas finales del procesamiento y en función de los resultados de los detectores, tanto basados en firmas como los adicionales basados en anomalías. Por tanto, para conseguir las funcionalidades adicionales requeridas es necesario modificar el flujo de procesamiento de Snort en dos aspectos. En primer lugar, se hace necesario incluir algún mecanismo que etiquete los paquetes que no activen ninguna regla y, por tanto, no sean considerados como de ataque. Este etiquetado actúa a modo de marcado para que, posteriormente, los posprocesadores adicionales que se establezcan puedan determinar qué paquetes corresponden a cada categoría (ataque/no ataque) de acuerdo al conjunto de reglas activas. En segundo lugar, es necesario añadir dos posprocesadores diferentes encargados de implementar las dos funciones de interés: analizar los paquetes no marcados como ataques, de acuerdo a la técnica de detección de anomalías deseada, y generar un flujo de salida sólo con los paquetes marcados como ataque y otro con los no marcados.

Adicionalmente, será necesario establecer las variables globales que requieran las funciones/módulos a implementar, así como los procedimientos de inicialización y/o finalización tanto de dichas variables como de los módulos añadidos. La filosofía de diseño de Snort [16] facilita todos estos procedimientos, teniendo previstos puntos de inserción de rutinas de inicialización, preprocesadores, detectores y posprocesadores.

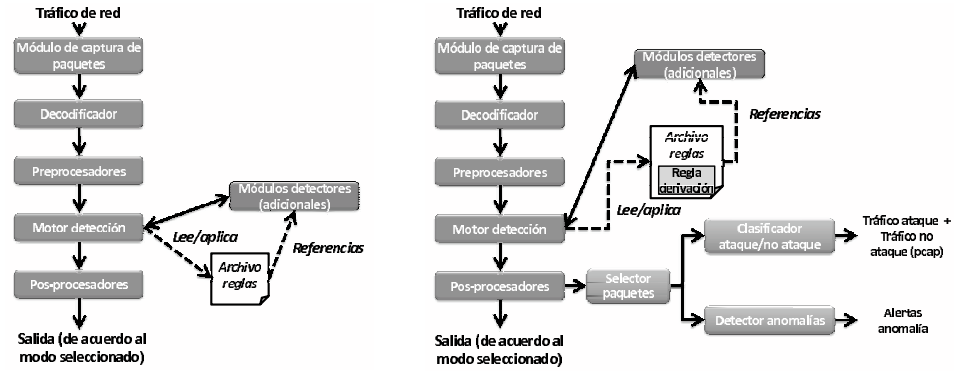
3.1. Marcado y selección de paquetes

El marcado de los paquetes se realiza mediante la inclusión de una regla específica –*regla de derivación* en la Fig. 3.b)– que incluye a todos los paquetes que se deseen procesar. De esta forma, al considerarse que el paquete es de ataque, se activan los módulos de posprocesado para el mismo. La regla, como cualquier otra regla de Snort, debe contener el nombre del módulo de posprocesado que se activará, las condiciones que debe cumplir el paquete para que se considere afectado por la misma y algunas opciones respecto de la salida que debe presentarse. Así, a modo de ejemplo, podría incluirse una regla de la forma:

```
ruletype selector {
   type log
   output log_selector: selector.log
} selector tcp any any ->any any
(msg:"Marca todo el  tráfico";sid:9999;)
```

que afectaría a todos los paquetes *tcp* cualesquiera que fuesen su origen o destino, marcándolos como una instanciación de un ataque asociado a la regla con número *sid=9999*. Evidentemente, no sería un ataque real, en el sentido de que no ha activado una firma de ataque, por lo que, en lo que sigue, diremos que es un *falso ataque*. En este ejemplo concreto se activaría un módulo de posprocesado denominado *selector* al que se enviaría el tráfico con la identificación de la regla activada, *sid*, con valor *9999* en este caso, almacenándose la información asociada a la alerta en el archivo *selector.log*.

El módulo *selector* asociado responde al esquema del código mostrado en la Fig. 4. En éste se puede comprobar que se verifica si la regla que activó la alerta que está siendo procesada es la de derivación (*sid=9999* en el ejemplo), en cuyo caso se envía el paquete asociado a dicha regla a los módulos de posprocesado adicionales y se actualizan los contadores de alertas de Snort y algunos auxiliares convenientemente. Para determinar el número de alertas generadas por el paquete que está siendo procesado se utiliza un contador global del número de alertas que se han activado, propio de Snort, junto con otros establecidos al



a) b)

Figura 3: Diagrama de operación de Snort, a) versión oficial, de acuerdo a [17], b) versión modificada.

efecto. Si únicamente se ha activado una regla y ésta corresponde a la de derivación, podremos concluir que corresponde a un *falso ataque* y actuar en consecuencia. En caso contrario, será un paquete de ataque, lo que también determinará el procesamiento posterior a realizar por los módulos adicionales, en su caso.

3.2. Clasificación del tráfico

La clasificación del tráfico mediante su separación en dos flujos de salida correspondientes a tráfico normal y a tráfico de ataque se puede realizar de forma sencilla, a partir de la estructura del módulo *selector* mostrada en la Fig. 4, sin más que programar adecuadamente las rutinas `ProcesaLimpio()` y `ProcesaAtaque()`. Estas rutinas pueden hacer uso de las funcionalidades propias de Snort para generar archivos de paquetes en formato *cap* y, simplemente, escribir los paquetes en dos archivos diferentes previamente establecidos.
Por tanto, a partir de las modificaciones propuestas, únicamente será necesario inicializar/cerrar los archivos de captura y establecer y gestionar las opciones de línea de comando asociadas.

3.3. Detección de anomalías

La inclusión de módulos de detección de anomalías se hace de forma análoga a la indicada en el apartado anterior, si bien en este caso puede resultar inadecuado procesar un paquete que ya ha sido clasificado como ataque por el sistema basado en firmas si lo que se pretende es simplemente su clasificación. En consecuencia, bastaría con sustituir la rutina `ProcesaLimpio()` del módulo *selector* con la que implemente la detección de anomalías. La información que es posible enviar a dicha rutina contiene el paquete completo, incluidas cabeceras y carga útil, por lo que son de aplicación los métodos de detección de anomalías que consideren un único paquete.
En este caso, las rutinas de inicialización y/o terminación pueden resultar de mayor complejidad que en el caso de clasificación del tráfico, ya que, obviamente, es necesario establecer la configuración y parámetros en los que se base la detección, así como generar informes con los resultados.
Es posible utilizar la versión de Snort modificada de acuerdo a lo indicado únicamente como detector de anomalías. Para ello bastaría con utilizar como única regla la de derivación, lo que haría que los paquetes se etiquetasen como *falsos ataques* únicamente.

3.4. Snort como IDS híbrido

La operación de la versión modificada de Snort como detector híbrido, es decir, que combine la detección de firmas y de anomalías, resulta trivial de acuerdo a la filosofía seguida en la introducción de las modificaciones. El flujo de procesamiento de los paquetes, en presencia de reglas a las que se añade al final la regla de derivación, responde al mostrado en la Fig. 5. Como se puede observar, los paquetes son analizados en primer lugar por los módulos ordinarios de Snort, que realizan la detección basada en firmas, y, posteriormente, por los detectores de anomalías que se establezcan. Debido a esta operación de forma natural como detector híbrido, denominaremos *Hy-Snort* a la versión modificada en lo sucesivo.
En cualquier caso, también resulta posible operar de forma inversa, realizando en primer lugar la detección de anomalías y, posteriormente, la basada en firmas. Para ello únicamente es necesario incluir en primer lugar la *regla de derivación*. En este caso, todos los paquetes serán considerados *falsos ataques*, independientemente de que puedan ser detectados posteriormente como ataques reales y, por tanto, sometidos al proceso de detección de anomalías. Este modo de operación puede resultar de interés para evaluar las capacidades de los detectores de anomalías incluidos, así como para

```
global int limpias; // Número de alertas "falsas"
global int alertas; // Número de alertas reales (firmas)

void Selector(Packet *p,char *msg,void *arg,Event *event) {
   unsigned long salert;
   int nsid;
   salert=(unsigned long)pc.alert_pkts;     // Número total de alertas generadas
   nsid=(int)event->sig_id;                 // El número SID que generó esta llamada
   if (nsid==9999) {                        // Actuar solo si regla de derivación
      limpias++;
      if (alertas==salert) {                // El paquete actual sólo generó una falsa alerta
         alertas++;
         if (p) {                           // Llamada a rutinas proc. (Paquete normal)
            if (p->packet_flags & PKT_REBUILT_STREAM)
               ProcesaLimpioStream(p, msg, arg, event);
            else
               ProcesaLimpioSingle(p, msg, arg, event);
            }
         }     } else {
         ProcesaAtaque(p, msg, arg, event); // Llamada a rutinas proc. (Paquete ataque)
         alertas=salert;                    //Se actualiza num. alertas
         alertas++;
      }
}
```

Figura 4: Esquema de la función de posprocesado para la selección de paquetes a analizar (módulo *selector*).

realizar la detección de forma conjunta a partir de la combinación de los informes generados tanto por los módulos de firmas como de anomalías de acuerdo a un análisis más complejo. Sin embargo, esta última posibilidad no se encuentra actualmente implementada.

4. Caso de estudio: implantación de la técnica SSM mediante Snort

Las modificaciones propuestas han sido implementadas y evaluadas durante el desarrollo de un sistema de detección de intrusiones que utiliza técnicas de detección desarrolladas por nuestro grupo de investigación. En particular, se ha implementado el sistema de detección de anomalías denominado SSM (*Segmental Stochastic Modelling*) [18], aplicable a las URI de las peticiones del protocolo HTTP.

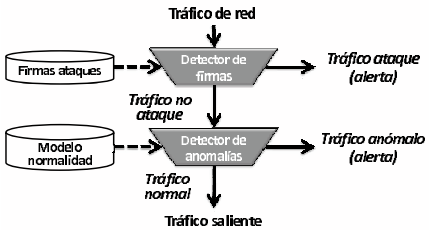


Figura 5: Flujo de procesamiento de paquetes del IDS híbrido propuesto.

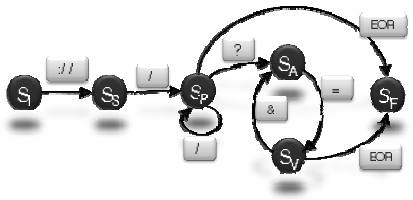


Figura 6: Autómata utilizado en el sistema SSM para el procesado de las URI [18].

De forma breve, el funcionamiento del sistema SSM se basa en la definición de un autómata de estados finitos estocástico capaz de evaluar la probabilidad de generación de una petición concreta. Dicho autómata, mostrado en la Fig. 6, es obtenido a partir de la especificación del propio protocolo, que define tipos de cadenas (campos) diferenciados, y de las probabilidades de aparición en cada uno de dichos campos de las diferentes cadenas que componen la petición. Cada estado del autómata corresponde a uno de los posibles campos: inicial (S_I), servidor (S_S), camino/recurso (S_P), atributo (S_V), valor (S_V) y final (S_F). Cada petición es segmentada, de acuerdo a un conjunto de delimitadores determinados por el protocolo, en los campos que la constituyen. Por otra parte, para cada estado existirá un *diccionario* con el conjunto de posibles valores del campo y la probabilidad de cada uno de dichos valores. El autómata permitirá, por tanto, dada una petición, evaluar si dicha petición es legítima (corresponde al modelo) y

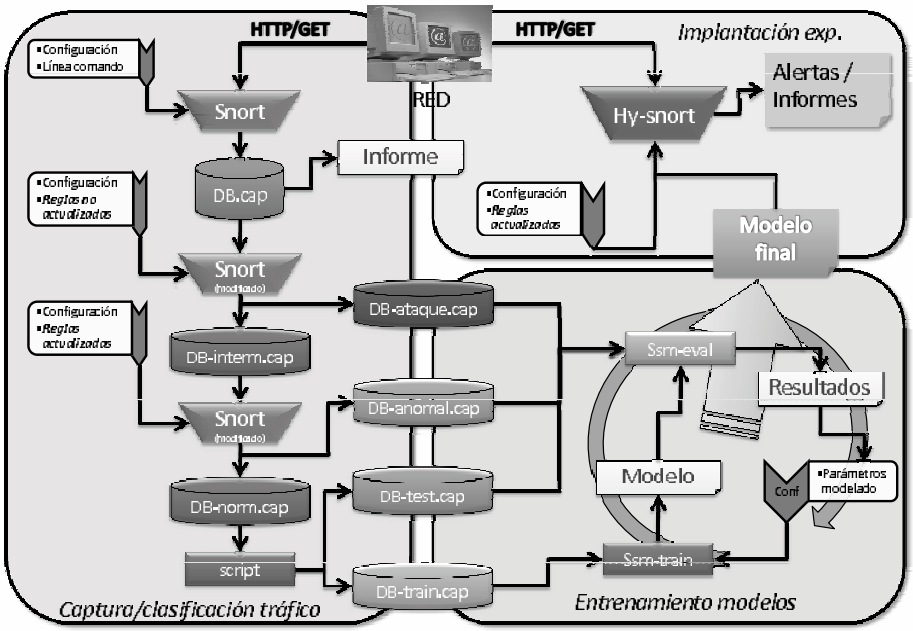


Figura 7: Metodología utilizada para el desarrollo e implantación de un IDS híbrido basado en Snort que implementa la técnica SSM.

su probabilidad. En función de la probabilidad y de un umbral se clasificarán las peticiones como normales o anómalas. Por otra parte, aquellas peticiones que no puedan ser evaluadas debido a la aparición de transiciones no contempladas serán etiquetadas como *fuera de especificación*, pudiéndose considerar como incorrectas.

El modelo propuesto presenta, pues, dos componentes: uno basado en especificación y, por tanto, establecido a partir del protocolo; y otro probabilístico, que depende del servidor concreto considerado, que debe ser estimado a partir de la observación de peticiones lícitas a dicho servidor. Se requiere, por tanto, de una fase de entrenamiento a partir de tráfico libre de ataques. En consecuencia, para el desarrollo de este sistema resulta necesaria la aplicación de la metodología descrita en el Apartado 2.

Para la implantación de la técnica SSM se han utilizado las dos funcionalidades añadidas a Snort, junto con las ya incluidas en la versión original. En primer lugar, se ha procedido a capturar tráfico real mediante el propio Hy-Snort operando como *sniffer*. A continuación se utilizó la capacidad para agrupar en dos archivos diferentes el tráfico capturado separando el tráfico libre de ataques y el tráfico de ataque, de acuerdo a la metodología propuesta (Fig. 7, bloque *Captura/clasificación tráfico*). A partir de los bloques de datos capturados se procedió al entrenamiento y evaluación del módulo de detección de anomalías, que fue convenientemente ajustado para optimizar su rendimiento (bloque *Entrenamiento modelos* en Fig. 7). El sistema fue entrenado con una parte del tráfico libre de ataques y evaluado con el resto y los ataques. La evaluación se realizó mediante el uso del detector de anomalías SSM incluido en Hy-Snort. Una vez ajustado el modelo, se ha procedido a su puesta en explotación en un entorno real a partir de los modelos desarrollados y las firmas disponibles (bloque *Implantación exp.*).

Adicionalmente, una vez en explotación, se analizaron las capacidades del detector mediante la instanciación de 1500 ataques sintéticos diferentes generados a partir de la información disponible en [19], con resultados satisfactorios. Es decir, se alcanzó un 100 % de detección operando en modo híbrido, así como en modo AIDS (operación sin las firmas de los ataques).

El sistema resultante ha mostrado unas prestaciones acordes a lo esperado, según el sistema SSM y la detección basada en firmas. Por otra parte, la inclusión del módulo selector no produce ningún incremento apreciable en el tiempo de procesamiento de cada uno de los paquetes operando en modo híbrido.

5. Conclusiones

En el presente trabajo se han presentado un conjunto de modificaciones de Snort destinadas a facilitar su utilización en el desarrollo e implantación de sistemas IDS híbridos. Las modifica-

ciones presentan una doble motivación: posibilitar la implantación de detectores híbridos que combinen la detección basada en firmas con la basada en anomalías con las garantías adecuadas respecto al rendimiento y permitir la clasificación automática del tráfico capturado en las fases iniciales del desarrollo de IDS. Las modificaciones han sido implementadas satisfactoriamente, habiéndose comprobado su utilidad durante el desarrollo de un IDS híbrido para la detección de ataques en las cargas útiles de las peticiones HTTP. La incorporación de estas características no degrada de forma apreciable el rendimiento, en número de paquetes procesados, y garantiza un nivel de detección superior o, en el peor caso, igual al proporcionado por el sistema basado en firmas.
Si bien el resultado es satisfactorio, resulta conveniente explorar la posibilidad de incluir otras características adicionales relacionadas con la detección en flujos de tráfico, en lugar de en paquetes aislados. También resulta de interés la inclusión de mecanismos que decidan qué paquetes son de ataque a partir de la combinación de la información procedente de ambos tipos de detección.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Programa Nacional de I+D+I (2004-2007) del MEC (proyecto TSI2005-08145-C02-02, 70% fondos FEDER).

Referencias

[1] CERT coordination Center statistics, http://www.cert.org/stats/cert_stats.html, 2006.

[2] Kabiri P., Ghorbani A.; *Research on Intrusion detection and response: A survey*, International Journal on Network Security, Vol. 1, N. 2, pp84-102, 2005.

[3] McHugh, J.; *Intrusion and Intrusion Detection*, Internation Journal on Information Security, Vol. 1., No. 1., pp.14-35, 2001.

[4] Allen, J. y otros; *State of the Practice of Intrusion Detection Technologies*. Technical Report CMU/SEI-99-TR-028, Software Engineering Institute, Carnegie Mellon Univ., 2000.

[5] Estévez-Tapiador, J.M.; García-Teodoro, P.; Díaz-Verdejo, J.E.; *Anomaly Detection Methods in Wired Networks: A Survey And Taxonomy*, Computer Communications 0140-3664; Vol 27, pp. 1569- 1584, 2004.

[6] Sturges, S.; *Snort users manual*, available at www.snort.org, 2006.

[7] Roesch, M.; *Snort-Lightweight Intrusion Detection for Networks*. Proc. USENIX, Lisa, 1999.

[8] McHugh, J.; *The 1998 Lincoln Laboratory IDS Evaluation. A critique*, In RAID 2000, LNCS 1907, pp 145-161, 2000.

[9] Antonatos, S., Anagnostakis, K., and Markatos, E.; *Generating Realistic Workloads for Network Intrusion Detection Systems*, Proc. of the 4th International Workshop on Software Performance (WOSP), 2004.

[10] M. Bermúdez-Edo, R. Salazar-Hernández, J. Díaz-Verdejo, and P. García-Teodoro, *Proposals on Assessment Environments for Anomaly-Based Network Intrusion Detection Systems*, J. Lopez (Ed.): CRITIS 2006, LNCS 4347, pp. 210-221, Springer-Verlag, 2006.

[11] P. García Teodoro, J M Estévez Tapiador, J. E. Díaz Verdejo; *Detection of Web-Based Attacks Through Markovian Protocol Parsing*, 10th Symposium on Computers and Communications; Cartagena 2005.

[12] McHugh, J.; *Testing Intrusion Detection Systems: A Critique to the 1998 and 1999 DARPA Intrusion Detection Evaluations as Performed by Lincoln Laboratory*, ACM Transactions on Information and Systems Security, Vol. 3. No. 4, pp. 262-294, 2000.

[13] Athanasiades, N. y otros; *Intrusion Detection Testing and Benchmarking Methodologies*, Proc. 1st IEEE International Workshop on Information Assurance (IWIA'03), Darmstadt (Germany), 2003, pp. 63-72.

[14] Duda, R., and Hart, P.; *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[15] Vossen, J.P.; *Snort Technical Guide*, disponible en http://www.snort.org/docs/.

[16] HNS Staff, *The Story of Snort: Past, Present and Future*, disponible en http://www.net-security.org/article.php?id=860.

[17] Arboleda, A. Bedón, A.; *SnortTM diagrams for developers*, disponible en http://afrodita.unicauca.edu.co/ cbedon-/snort/snort.html, 2005.

[18] Estevez-Tapiador, J. M. y otros; *Stochastic Protocol Modeling for Anomaly-Based Network Intrusion Detection*, Proc. 1st IEEE International Workshop on Information Assurance (IWIA'03), pp. 3-12, Darmstadt, 2003.

[19] *Arachnids: Advanced Reference Archive of Current Heuristics for Network Intrusion Detection Systems*. http://www.whitehats.com/ids, 2003.