Presente y futuro de los IDS

Alejandro Barrera García-Orea
Departamento de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
abarrera@neurosecurity.com

Resumen

La detección de intrusos en una red telemática sigue siendo, hoy por hoy, un problema de difícil resolución. Dada la naturaleza del problema, no existe un método probado que permita la detección infalible. No obstante, el presente documento intenta exponer un nuevo enfoque sobre la detección de intrusos, analizando cual es la problemática de estos sistemas (NIDS [1]) y señalando las carencias actuales de dichas infraestructuras. Se propone un nuevo modelo neuronal que permita realizar una detección heurística que disminuya la tasa media de falsos positivos de los sistemas actuales. Por último, se exponen razonadamente las fortalezas y debilidades del nuevo modelo así como una valoración de los resultados esperados de la aplicación del mismo.

1. Introducción

Cada vez son más comunes los ataques informáticos a grandes sistemas, sin embargo, aun hoy, muchos administradores siguen estando "ciegos" ante dichas intrusiones. Es por ello que desde hace más de 20 años [1],[2] se viene desarrollando un nuevo campo dentro de la seguridad informática, la detección de intrusos en los sistemas informáticos.

Los sistemas de detección de intrusos suelen ser infraestructuras, generalmente complejas, que permiten, mediante una serie de heurísticas, detectar cuando un sistema informático está siendo utilizado de forma no autorizada.

Estos sistemas, también conocidos por sus siglas inglesas **IDS** [2], permiten, no solo la detección de ataques cubiertos por otros componentes de seguridad, sino la detección de intrusiones que pasan desapercibidas a otros componentes del dispositivo de seguridad.

Todas esas detecciones son almacenadas en forma de registros de información que son de gran utilidad a la hora de reconstruir, mediante *técnicas de análisis forense*, qué es lo que le ha sucedido a un sistema informático.

[1] Network Intrusion Detection System
[2] Intrusion Detection Systems

Debido a lo anterior, podemos concluir que estamos ante sistemas tan complejos como delicados y que deben mantener altos niveles de detección minimizando las tasas de falsos positivos. Los falsos positivos son el mayor enemigo de los IDS puesto que elevados niveles de éstos, inhabilitan, desde el punto de vista forense, el valor de cualquier información que haya podido quedar registrada por el sistema tras un incidente.

2. Estructura de un Sistema de Detección de Intrusos

Dado que los IDS han sido estudiados desde hace bastantes años, existe gran diversidad de formatos y arquitecturas. Es por tanto que desde hace algun tiempo se está realizado un esfuerzo por unificar, en la medida de lo posible, la arquitectura y los formatos de los IDS. En el presente documento nos ceñiremos a la arquitectura definida por el *Common Intrusion Detection Framework* (**CIDF**) [3] en sus drafts [3].

Un sistema IDS consiste en una serie de componentes discretos que se comunican mediante el paso de mensajes. A grandes rasgos se pueden identificar cuatro componentes básicos:

- Generador de eventos (E-boxes)
- Motor de análisis (A-boxes)
- Unidades de almacenaje (D-boxes)
- Unidades de respuesta (R-boxes)

Los componentes son unidades lógicas que pueden producir y/o consumir los mensajes u eventos generados por los otros componentes. Estos pueden ser implementados de cualquier forma, bien como un solo proceso (o un solo thread) en una máquina, cluster, mainframe, etc, o bien como un conjunto de procesos distribuidos a través de varias máquinas o procesadores (Snortnet o DIDRA).

2.1. Generadores de eventos (E-boxes)

Los generadores de eventos, sensores o sondas, como suelen llamarse algunas veces, tienen como objetivo la obtención de datos del exterior del sistema de detección de intrusos. Son los "ojos" del IDS. Las entradas de los generadores de eventos serán los datos en bruto del entorno exterior al IDS. A su salida presentará esos datos procesados en forma de eventos comprensibles por el resto de los componentes. Los generadores pueden ser diversos, dependiendo del tipo de datos que recogen, aunque su funcionamiento a nivel conceptual suele ser muy similar. Reciben los datos de entrada, los preprocesan para pasarlos a un formato común al resto de los componentes [4] y proporcionan los eventos al resto de componentes prácticamente en tiempo real.

2.2. Motor de análisis (A-boxes)

El motor de análisis es el núcleo de los IDS. Es el motor de inferencia que, gracias a unos conocimientos, será capaz de discernir la relevancia de los eventos recibidos de las *E-boxes* y generar nuevos eventos como salidas. Estos motores de análisis pueden ser de muchos tipos, sistemas estadísticos de profiling, reconocedores de patrones, sistemas de correlación de eventos, etc.

[3] http://www.isi.edu/gost/cidf/

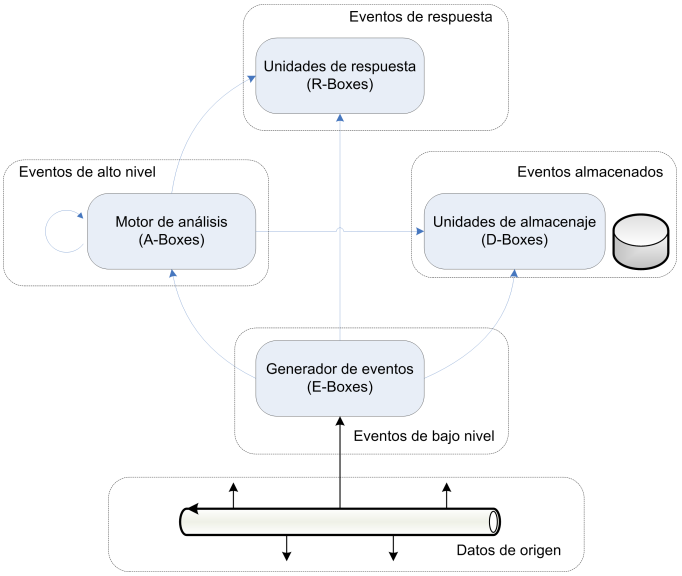

Figura 1: Diagrama de la arquitectura CIDF

Los motores de análisis han sido el componente que ha estado sujeto a mayor desarrollo puesto que sobre él recae la responsabilidad de analizar el flujo de eventos y de extraer información relevante. Es por tanto uno de los componentes más críticos y más complejos de un sistema de detección de intrusos.

2.3. Unidades de almacenaje (D-boxes)

Este componente es el encargado de almacenar físicamente las inferencias del motor de análisis. Contendrá todos los eventos generados por las *A-boxes* y normalmente se organizan en forma de bases de datos. Es por tanto un componente esencial a la hora de aplicar técnicas de datamining y correlación de datos como fuentes de información forense [5].

2.4. Unidades de respuesta (R-boxes)

Las unidades de respuesta son los componentes encargados de realizar acciones en nombre de otros componentes del sistema. Este componente suele emplearse para desplegar unidades que ejecuten contramedidas ante una intrusión. Es decir, permiten al sistema reaccionar de forma activa ante las acciones procesadas por otros componentes.

Estas acciones pueden ser muy variadas, aunque en la gran mayoría de los casos están orientadas a prevenir ataques de fuentes maliciosas previamente detectadas o a cortar un ataque en curso. Cuando el sistema cumple lo anterior se dice que además de ser un sistema de detección de intrusos (lo cual implica pasividad), se le concede la denominación de sistema de prevención de intrusos o IPS [4]. En general, un IPS suele autocontener la noción de IDS, no obstante algunos puristas prefieren denominar a dichos sistemas como I(D\P)S.

[4] Intrusion Prevention System

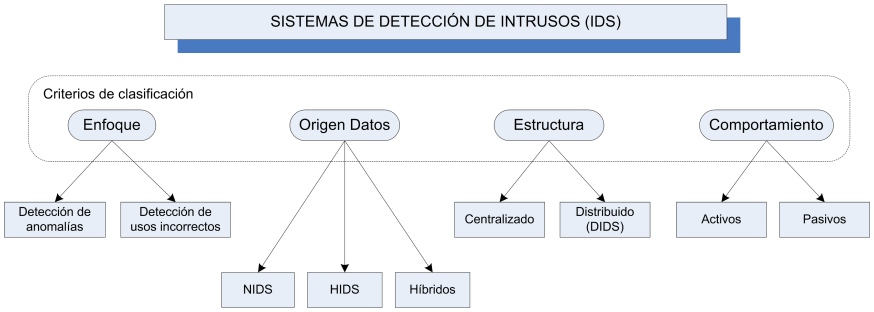

Figura 2: Clasificación de los IDS

3. Tipos de Sistema de Detección de Intrusos

Como hemos visto, un IDS contiene cuatro componentes básicos. Como en toda tecnología, existen distintos tipos de IDS. Existen distintas clasificaciones atendiendo a las características discriminatorias seleccionadas.

3.1. Tipos de IDS en función del enfoque

Si atendemos al enfoque del IDS podemos diferenciar dos grandes tipos: *anomaly detection* y *misuse detection*.

3.1.1. *Anomaly Detection*: Detección de anomalías

Este tipo de enfoque fue propuesto por Dorothy Denning [1] en su artículo: *An intrusion detection model*. En dicho artículo se proponía la creación de perfiles de uso de los sistemas durante un determinado período de tiempo. De esta forma se podían detectar desviaciones del perfil recopilado que posteriormente eran analizadas en búsca del por qué de la anomalía.

3.1.2. *Misuse Detection*: Detección de usos incorrectos

Este ha sido el enfoque más tradicional y sigue siendo el más extendido hoy en día. Estos sistemas tienen conocimientos específicos sobre determinados ataques y se basan en ellos para analizar los datos de entrada del sistema. Si alguno de los datos coincide con alguno de los patrones conocidos se emitirá una alerta.

3.2. Tipos de IDS en función del origen de los datos

3.2.1. HIDS: *Host-based Intrusion Detection Systems*

Los **HIDS** están diseñados para monitorizar, detectar y responder a los datos generados por un usuario o un sistema en un determinado host. Los primeros IDS que surgieron [1],[2] eran de este tipo y sentaron las bases de los IDS comerciales. Este tipo de IDS resultan útiles a la hora de identificar amenazas e intrusiones a nivel

del host local, es decir, solo maneja información del host donde reside. Sin embargo, cuando un sistema comprende cientos de hosts, los HIDS, por si solos, no son viables para realizar una monitorización adecuada.

3.2.2. NIDS: *Network Intrusion Detection Systems*

Los NIDS son muy parecidos a los HIDS en tanto en cuanto monitorizan, detectan y responden ante los datos generados, pero en vez de proteger un determinado host, reciben los datos de la red local donde estén instalados. Tienen la ventaja de ser, normalmente, pasivos, de tal forma que no interfieren con el correcto uso de la red. Adicionalmente permiten la detección de ataques a un mayor nivel de abstracción, puesto que ahora no solo tienen información de un solo host, sino de múltiples. No obstante, los NIDS solo perciben información que pasa por la red, siendo inutil ante los ataques locales de los usuarios de un determinado host.

3.2.3. Hybrid-IDS

Los sistemas híbridos reunen lo mejor de ambos tipos. Normalmente lo constituyen sensores en cada host que permiten una detección local de los sistemas y un sensor en cada segmento de red a vigilar. De esta forma cubrimos las necesidades del HIDS con las del NIDS.

3.3. Tipos de IDS en función de su estructura

3.3.1. Distribuidos (DIDS[5])

Los IDS distribuidos son aquellos sistemas en donde se implantan varios IDS que se comunican entre si o con un servidor central que permite centralizar y correlacionar todos los datos generados por ellos. El tener varios agentes distintos por toda la red permite ampliar la información de la que se dispone para la detección de un incidente en el sistema.

3.3.2. Centralizados

Son aquellos IDS que emplean sensores que transmiten información a un sistema central desde donde se controla todo. De esta forma se permite ahorrar en costoso equipamientos pero manteniendo un amplio abanico de sensores desde donde recoger información.

3.4. Tipos de IDS en función de su comportamiento

3.4.1. Pasivos

Los IDS pasivos son aquellos que solo se dedican a procesar la información en busca de intrusos. Una vez que se ha detectado una intrusión, se emité una alerta y se deja que el operador humano realice o no una acción en consecuencia. Si nos remitimos a la estructura básica de un IDS diríamos que el sistema carecería de las unidades de respuesta (*R-boxes*).

[5]Distributed Intrusion Detection Systems

3.4.2. Activos (IPS[6])

Por el contrario, existen IDS que si realizan acciones en base a los datos analizados. Muchos de ellos tienen unidades de respuesta que modifican las **ACLs**[7] del firewall corporativo para por ejemplo, bloquear ataques en curso, evitar el acceso a una IP de un intruso, etc. Estos sistemas reciben por tanto el nombre de *Intrusion Prevention Systems* o **IPS**. Aunque la idea parece muy sofisticada y util hay que tener presente que dichos motores de respuesta tienen un funcionamiento muy simplista y en absoluto inteligente. Es por ello peligroso emplear dichos sistemas, puesto que pueden bloquear o restringir el acceso a recursos del sistema informático debido a falsos positivos o a análisis erróneos de los datos de entrada.

4. Problemática de los Sistemas de Detección de Intrusos

Aunque los sistemas de detección de intrusos han proporcionado un gran avance en el mundo de la seguridad informática, estas soluciones no son la panacea. Todo IDS tiene numerosas carencias debido el tipo de problema que intenta solucionar.

La detección de intrusos es un problema de muy difícil resolución. Dada la naturaleza del ser humano, sus comportamientos son totalmente impredecibles y están sujetos a lo que a algunos les gusta llamar "libre albedrío". Es por tanto imposible poder diseñar un sistema que sea capaz de identificar al 100 % todos los intentos de intrusión por el mero hecho de que ni los propios humanos son capaces de discernirlos.

Sin embargo, en un gran número de ocasiones, las pautas de los atacantes suelen ser muy similares. Los pasos para conseguir acceso no autorizado a los sistemas informáticos suelen estar bastante bien definidos y por tanto pueden ser estudiados y clasificados de forma que se pueda llevar a cabo una detección con índices muy elevados de acierto.

Partiendo de la anterior base podemos decir que el concepto de seguridad lo conforma un conjunto de componentes que interactuando entre sí permiten minimizar el riesgo de una intrusión. A esto es lo que se conoce como "Seguridad en Profundidad". Tras este concepto existe la idea de apilar múltiples capas de protección entre un posible atacante y su objetivo. Es decir, introducir el mayor número de barreras que sean posible entre nuestros sistemas y los intrusos. Evidentemente, nunca conseguiremos eliminar la probabilidad de un acceso no autorizado, pero si podremos disminuir ésta a niveles muy bajos.

Al margen del problema intrínseco de la detección de intrusos, los IDS convencionales adolecen de numerosos defectos que los convierten en costosos sistemas con rendimientos muy bajos. A continuación se exponen una serie de defectos y problemas que se han ido identificando a lo largo de los años con respecto a los IDS.

4.1. Mantenimiento de los IDS

Los IDS no son sistemas que se instalan y luego se dejan en el olvido. Suelen ser sistemas complejos que requieren un mantenimiento casi diario. Suelen requerir a una persona que esté pendiente de las alertas que generan y que vaya ajustandolas, filtrándolas y realizando el *fine tunning* para minimizar los falsos positivos y el ruido.

[6] Intrusion Prevention Systems
[7] Access Control Lists

Esto es un gravísimo inconveniente porque aunque automatiza gran parte del trabajo de vigilancia, sigue requiriendo un ser humano que valide y mantenga el sistema. Esto no debería ser así, los IDS deberían ser sistemas inteligentes y completamente autónomos, es decir, capaces de autoconfigurarse, protegerse, aprender de su entorno, etc. Este concepto es el que desde hace algunos años persigue la división de **Autonomic Computing de IBM Research**[8].

4.2. Errores de diseño de los NIDS

Los NIDS son sin duda alguna el tipo de IDS más implementado en todas las organizaciones con sistemas informáticos. No obstante el actual diseño de la arquitectura de éstos contiene graves errores que, correctamente explotados, pueden inhabilitarlos como elementos de seguridad.

Normalmente, los sensores de los NIDS son elementos pasivos dentro de un segmento de red. Esto significa que lo único que hacen es analizar el tráfico de red que circula por su segmento de red. El posterior análisis de los datos recopilados suele ser en tiempo real y, por tanto, el IDS debe ser capaz de procesar los datos a la suficiente velocidad como para no perder información. No obstante ésto no siempre es así. Dada su naturaleza pasiva, los NIDS han de lidiar con los siguientes problemas:

- Las redes con *switchs* son uno de los grandes problemas de estos sistemas. Debido al uso, cada vez más común, de este tipo de redes, los sensores del IDS solo pueden analizar el tráfico que circula por el segmento de red en el que están instalados. Por tanto, es necesario desplegar una red de sensores para cubrir toda la infraestructura a vigilar.
- Los NIDS actuales tienen problemas para analizar segmentos de red con mucho tráfico. De media una página web genera alrededor de 600 bytes por paquete. Esto se traduce en aproximadamente 170,000 paquetes/segundo en una red Ethernet de 100 Mbps. Muchos IDS actuales no son capaces de procesar a semejante ritmo y, por tanto, son propensos a perder paquetes con información relevante.
- Los IDS deben ser *stateful*, esto es, deben mantener información sobre el estado de cada una de las conexiones TCP que tienen abiertas. Esto consume gran cantidad de memoria y, por tanto, puede suponer un problema de rendimiento cuando la carga de la red es alta.
- Los sensores pueden llegar a cegarse debido a una saturación del enlace en donde residen, siendo por tanto deshabilitados e inútiles frente a la detección.
- El componente que almacena los registros del IDS puede, bajo una alta carga, llegar a su límite físico y, por tanto hacer que el IDS deje de funcionar, o bien que comience a perder registros significativos por falta de espacio.

Todo los problemas expuestos anteriormente entran dentro de lo que se conoce como *resource exhaustion* o agotamiento de los recursos [9] (**DoS**). Los IDS suelen ser, por lo general, sistemas *fail-open*, esto es, el sistema deja de proporcionar protección cuando es deshabilitado forzosamente. Al ser dispositivos pasivos, si un atacante consigue que el IDS consuma todos sus recursos y quede deshabilitado, podrá atacar impunemente el resto de la red que éste protegía. En contrapartida podemos hablar de sistemas de seguridad que son *fail-safe* los cuales dejan al sistema protegido

[8] http://www.research.ibm.com/autonomic/

[9] http://online.securityfocus.com/infocus/1647

en el caso de que sean deshabilitados de forma forzosa. Un ejemplo de sistemas *fail-safe* son algunos *firewalls* que en caso de verse deshabilitados cierran todas las conexiones de la red que protegen, evitando así dejar el sistema abierto a sucesivos ataques.

4.3. Conocimiento del entorno

Los NIDS solo tienen conocimiento de la información que trasvasa su segmento de red, es decir, no tiene conocimientos acerca de la topología de la red, las máquinas que la constituyen, que sistemas operativos cohexisten, etc. Así mismo no tienen memoria que les permita recordar datos anómalos que hayan sucedido con anterioridad. Esto es un gran inconveniente a la hora de realizar una detección eficaz puesto que el IDS carece de información sobre su entorno inhabilitando la posibilidad de detectar ciertos ataques como los *slow scans*[10].

Este desconocimiento del entorno también favorece problemas como la inserción y la evasión[6].

En la inserción el IDS puede aceptar un paquete que el sistema final rechazaría. En su desconocimiento, el IDS comete un error en creer que el sistema final ha aceptado y procesado un paquete cuando realmente no ha sido así. Un atacante puede explotar dicha condición insertando así datos en el IDS que son descartados por el resto de sistemas de la red.

La evasión es el caso contrario, un sistema final acepta un paquete que el IDS rechaza. De esta forma un IDS rechaza erróneamente un paquete perdiendo así todo el contenido del mismo. Este problema puede ser explotado para evadir la detección del IDS dado que, al ser este más estricto que los sistemas finales, rechazará información que sí es procesada por los sistemas finales.

4.4. Evasión de la detección de patrones

Gran cantidad de IDS comerciales emplean bases de firmas como motor de análisis para la detección de patrones. Es decir, cuando se recibe un evento en el motor de análisis se compara contra toda la base de firmas. Si alguna de ellas coincide, se disparan las alarmas y se genera un evento que será almacenado para su posterior revisión.

La tecnología de firmas ha sido heredada de los motores antivirus que las llevan empleando desde sus inicios. No obstante ha quedado demostrada la facilidad que existe a la hora de burlar dichos sistemas de firmas [11], [12], [13]. Simples modificaciones en los contenidos de los paquetes o la fragmentación de los mismos sirven para que numerosos ataques pasen desapercibidos al motor de firmas.

Esto es lo que se conoce como “falsos negativos”, es decir cuando un evento pasa desapercibido al IDS aunque realmente haya sucedido.

Asimismo, cada vez es más común el uso de la criptografía para proteger las comunicaciones. Esto establece un nuevo problema para los IDS puesto que se imposibilita la detección de los paquetes cifrados y por tanto se abre un nuevo vector para ataques inadvertidos.

---

[10] http://www.insecure.org/nmap/data/nmap_manpage.html

[11] http://www.neurosecurity.com/presentations/dtws.pdf

[12] http://www.securityfocus.com/infocus/1577

[13] http://www.sans.org/resources/idfaq/rpc_evas.php

4.5. Exactitud del sistema: Falsos Positivos

Tal vez el mayor problema de estos sistemas son los **falsos positivos**. La exactitud de un IDS está comprometida cuando el sistema identifica como una intrusión algo que no lo es. Esto es lo que se conoce como falso positivo y es la pesadilla de todo administrador.

Como se comentó anteriormente, el problema de la detección es muy difícil y es normal que existan algunos falsos positivos. No obstante, si el procentaje de éstos es muy elevado el sistema se vuelve completamente inútil [7],[8].

El problema con los IDS actuales es que los porcentajes de falsos positivos son muy elevados. A veces son tan altos que el sistema no tiene utilidad alguna. En numerosos casos dichos porcentajes se pueden disminuir mediante un árduo proceso de *fine-tunning* que requiere tiempo y personal cualificado.

5. Objetivos del proyecto

El siguiente apartado detalla todos los objetivos que se persiguen en este proyecto indicando detalladamente cuales son sus fundamentos, porqué se han tomado ciertas decisiones de diseño y cuáles son las desventajas del sistema propuesto.

5.1. Nuevo Concepto de motor de análisis

El objetivo del proyecto es desarrollar un nuevo concepto de motor de análisis (*A-box*) para disminuir los falsos positivos y falsos negativos durante la detección de intrusiones.

Como se ha visto en los apartados anteriores, la tecnología IDS todavía tiene numerosos problemas[14]. Existen soluciones para muchos de ellos (consumo de recursos, *fail-safe*, etc), sin embargo el problema de los falsos positivos y negativos sigue siendo uno de los más problemáticos. El proyecto propone un nuevo modelo de motor de análisis que permita disminuir hasta un nivel razonable los niveles de éstos.

5.2. IDS pasivo de detección de usos incorrectos

El sistema se basará en la detección de usos incorrectos (*misuse detection*) frente a lo que se conoce como detección de anomalías (*anomaly detection*). El sistema poseerá un comportamiento pasivo.

La detección de usos erróneos se basa en el uso de patrones de ataques conocidos que son comprobados contra los datos que recibe el sistema. Tiene la ventaja de que cuando se reconoce un ataque se sabe exactamente que ataque es indistintamente de si el ataque estaba incluido dentro del perfil de uso del sistema o no. Sin embargo, el mayor problema de este enfoque es la gestión y generación de las firmas o patrones empleados en la detección. Dada la diversidad de los ataques es prácticamente imposible tener un patrón para cada ataque, por tanto muchos sistemas solo tienen protección parcial frente a todos los ataques posibles. La creación de patrones es además un proceso “a posteriori”, es decir, el ataque debe conocerse para poder generar la firma o patrón. Es por ello que estos sistemas de firmas son incapaces de reaccionar ante ataques únicos o novedosos. Esto es un gravísima desventaja y más cuando tenemos en cuenta que una ligera variación de un

[14] http://www.mcafee.com/us/local_content/white_papers/wp_intruverttop10.pdf

determinado ataque pasará inadvertida al IDS si no actualizamos la firma para la nueva variante. Si además recordamos que un mismo ataque puede ocultarse mediante técnicas de ofuscación (fragmentación, inserción, cifrado, etc.) el número de variaciones aumenta y, en consecuencia, el problema se agrava.

La detección de anomalías es la otra cara de la moneda y tiene la ventaja de que permiten la detección de gran diversidad de ataques sin conocimiento previo de los mismos. Cada vez más, la diversidad de los ataques en las redes es mayor y por tanto es más costoso y complejo generar patrones de todos los ataques. Por ello la detección de anomalías es una de las soluciones que se han planteado para suplir este problema. Se genera un perfil de uso del sistema [1] y luego se analizan, normalmente mediante métodos estadísticos, las anomalías que se producen con los datos reales.

No obstante, este enfoque tiene también algunos problemas. Estos sistemas no son capaces de detectar ataques que coincidan con datos usuales y contenidos en el perfil de uso del sistema. Es más, adolecen de la inhabilidad de adaptarse a cambios bruscos en los comportamientos de los usuarios, lo que genera grandes tasas de falsos positivos [15]. Por otra parte pueden ser gradualmente entrenados hasta el punto en que un comportamiento, que una vez fue considerado anómalo, se considere normal. Es un enfoque que en general suele implementarse mediante análisis estadísticos. Esto tiene la ventaja de que es un campo muy estudiado, sin embargo adolece de otros problemas. Uno de ellos es la insensibilidad para detectar el orden de ocurrencia de un determinado evento. Al ser un proceso estadístico se pierde la interrelación secuencial entre eventos que pueden estar relacionados. Esto tambíen implica que son sistemas incapaces en algunos casos, de discernir a priori qué tipo exacto de ataque ha sido el que ha disparado las alarmas. Es importante mencionar que a estos IDS les resulta difícil diferenciar entre actividad anómala e intrusiva. La generación del perfil de uso es crítica y laboriosa. Es necesario ajustar muy finamente los parámetros y medidas empleadas para llevar a cabo dicha clasificación y, aun así, un determinado conjunto de medidas puede no resultar adecuada para capturar toda la diversidad de los ataques. Por último es necesario recordar que es probable que sea necesario desarrollar un perfil distinto atendiendo al sistema al que va dirigido. Adicionalmente, si el sistema no proporciona algoritmos de aprendizaje, una vez que se ha creado un perfil, si los usos del sistema cambian, es necesario volver a elaborar uno nuevo. Esto hace que la portabilidad de dichos sistemas sea bastante limitada [10].

Según lo expuesto anteriormente, aunque la detección de usos incorrectos tiene la gran lacra de la generación de patrones, se considera que un nuevo diseño (mediante redes neuronales) permitirá sobrepasar estas limitaciones en cuanto a la generación y aprendizaje de nuevas firmas. Cabe mencionar que lo ideal sería un sistema híbrido en donde se emplee lo mejor de ambos. Sin embargo, y en aras de la simplicidad del proyecto, se ha optado por desarrollar solo la detección de usos incorrectos.

Por último, dado que el objetivo principal del proyecto es probar la eficacia del modelo de análisis, el comportamiento del sistema será pasivo dado su menor coste de procesamiento y simplicidad de implementación. No se descarta, sin embargo, modificar en un futuro parte de la estructura del IDS para que implemente un comportamiento activo.

5.3. Desarrollo de un NIDS para protocolos TCP/UDP

El proyecto se centrará en el desarrollo de un NIDS, analizando solamente los protocolos TCP/UDP. En particular solo se tendrá en cuenta el contenido de los paquetes y no se analizarán las cabeceras en busca de anomalías o inconformidades con los RFCs.

Lo ideal sería una arquitectura en donde se emplearan distintos tipos de IDS simultánemente. HIDS que transmitan información sobre cada uno de los hosts e inline NIDS en lugares estratégicos que permitan proteger los segmentos de red críticos de la infraestructura. No obstante para no complicar en exceso los diseños se ha optado por centrarse en la estructura de los NIDS como punto de partida. Por la misma razón se ha decidido acotar los protocolos analizados a TCP y UDP dado que son los más comunes en casi todos los sistemas de la información. Solo se analizará el contenido o *payload* de los paquetes capturados puesto que es ahí donde se espera encontrar los *shellcodes* (ver siguiente objetivo). No obstante, lo óptimo sería realizar un análisis en profundidad de las cabeceras tanto TCP/UDP como IP en busca de anomalías y/o violaciones de los estándares RFC.

5.4. Detección de *Shellcodes*

El sistema se centrará en la detección de *shellcodes* tanto en sistemas operativos Windows como Linux. La arquitectura se supondrá siempre Intel x86.

Como se comentó previamente, existe una gran diversidad de ataques en Internet [9]. Sin embargo los incidentes más graves suelen venir de la mano de los temidos *exploits*. Estos programas aprovechan *bugs* en el software de los servicios de red (ftpd, httpd, sshd, etc.) que les permiten la ejecución de código arbitrario en la máquina remota. Estos tipos de ataques se conocen con el nombre de *Remote to Local* (R2L). No son estos los únicos vectores de ataque que existen (mala configuración, ingeniería social, fallos de validación, ataques por fuerza bruta, etc), sin embargo si suelen ser los más mortíferos. Estos *exploits* suelen contener lo que se conoce con el nombre de *shellcode*. Un *shellcode* es la representación hexadecimal de los opcodes de un conjunto de instrucciones en ensamblador. Se llaman así porque originariamente ese conjunto de instrucciones lo que hacía era ejecutar una shell. Esta *shellcode* es lo que se manda y se inyecta en el mapa de memoria del proceso remoto para que sea ejecutado con los privilegios del software vulnerable. Un *shellcode* puede ejecutar lo que se quiera y normalmente, cuando el bug es explotable remotamente, suelen ser *bindshells*, esto es, shells que se asocian a un determinado puerto de la máquina vulnerada para permitir la entrada a posteriori del atacante.

A grandes rasgos, la anatomía de un ataque mediante un *exploit* suele ser muy parecida en casi todos los casos. A continuación se muestra una figura con las etapas, generalmente comunes, que existe en un ataque mediante un *exploit*.

Dada la gran trascendencia de este tipo de ataques, el sistema propuesto se centrará en la detección de éstos, especialmente en la identificación de los *shellcodes* empleados por los *exploits*. Una vez verificado el sistema y su eficacia no se descarta su ampliación a la detección de otro tipo de ataques.

Mencionar, por último, que en la actualidad también existen ataques debido a fallos en las aplicaciones web que permiten la ejecución de código arbitrario mediante el protocolo *http*[15], [16]. Estos son casos particulares relacionados con las aplicaciones web y no se tendrán en cuenta en el IDS para no aumentar su complejidad.

[15] http://www.milw0rm.com/id.php?id=1247

[16] http://www.milw0rm.com/id.php?id=1113

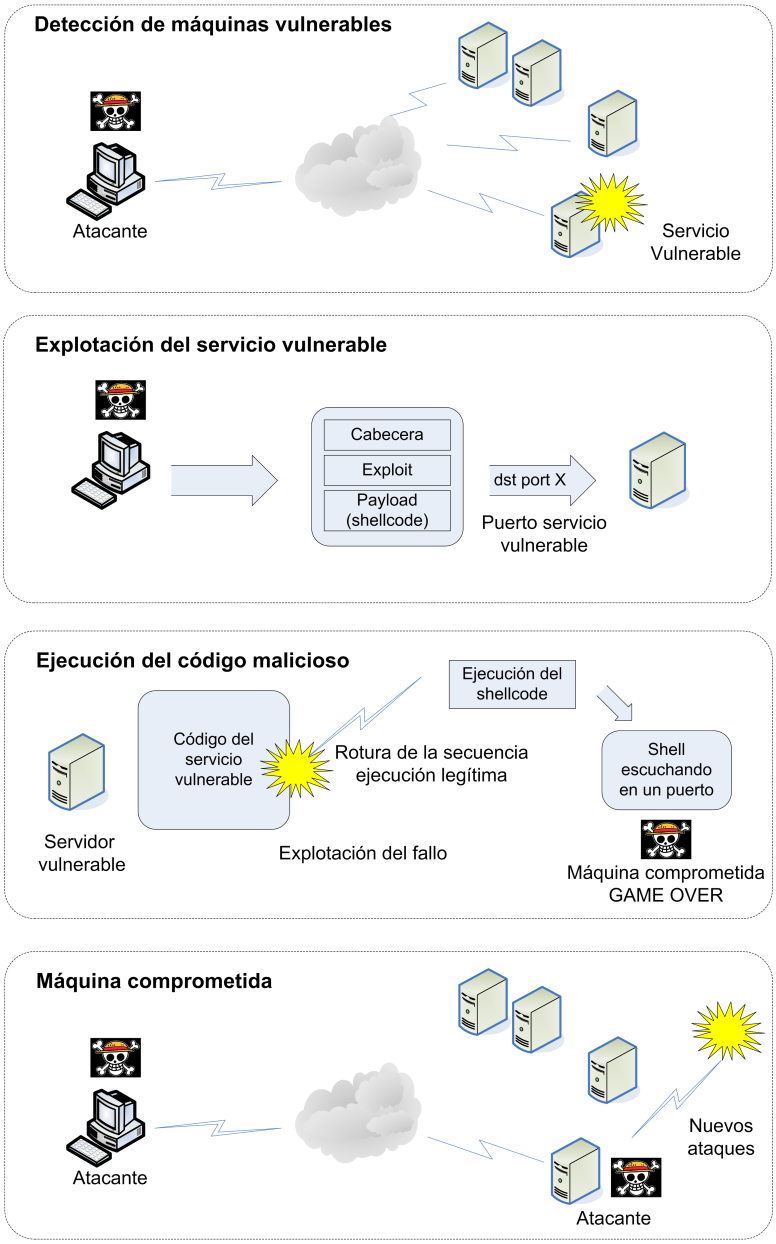

Figura 3: Esquema de un ataque R2L

5.5. Uso de redes neuronales artificiales para detectar y clasificar los ataques

Uso de una red neuronal para almacenar los patrones de interés mediante el aprendizaje de sus invariantes.

Gran parte de los sistemas que implementan *misuse detection* almacenan sus patrones en forma de firmas. Es evidente que es la forma más fácil de guardar dicho conocimiento; sin embargo, es muy ineficaz. Es por ello que se propone el uso de *redes neuronales artificiales* para contener todos esos patrones de ataques. Este tipo de sistemas permiten el almacenamiento de una cantidad casi ilimitada de patrones. Son sistemas distribuidos de computación con altas tasas de tolerancia a fallos, esto es, siguen funcionando relativamente bien aunque parte de la red neuronal esté dañada. Estos modelos siguen un enfoque biológico ya que intentan reflejar el modelo de procesamiento del **cortex** humano. Una de las ventajas de estos sistemas es que permiten identificar desviaciones de los patrones aprendidos, es decir, la red neuronal aprende un determinado ataque pero es capaz de inferir que ciertos ataques, aunque no iguales al aprendido, deben pertenecer a esa determinada clase debido a que sus invariantes son idénticos [11].

Un **invariante** es aquella característica que distingue de forma inequívocamente un objeto. En el campo del reconocimiento de patrones también se conoce como característica discriminante. Por ejemplo, el **neocortex** humano es capaz de diferenciar una cara humana de una silla porque las caras contienen dos ojos, una nariz, dos orejas, una boca, etc. Sin embargo una silla no tiene ojos, ni boca, ni orejas, sino un número definido de patas (normalmente cuatro) y una superficie sobre la que descansar el peso. Si el neocortex humano detecta una cara sin nariz es probable que la siga clasificando como una cara aunque se extrañaría y sabría que eso no es lo normal, que está ante una cara anómala.

Los modelos neuronales son muy plásticos y potentes, sin embargo tienen varios inconvenientes. El más importante tal vez sea el hecho de que son computacionalmente muy costosos. El aprendizaje en los modelos actuales requiere una potencia de computación bastante elevada. No obstante la potencia de cálculo disponible hoy en día esta haciendo que el despligue de estos sistema sea cada vez más accesible. Otro de los grandes problemas de las redes neuronales es que no proporcionan **trazabilidad**, esto significa que no se puede trazar el por qué de un valor a la salida de la red. No se puede obtener los criterios que ha empleado el modelo para llegar a esa conclusión. Esto es un gran problema, especialmente de cara a la legalidad de las detecciones en un IDS. Si no podemos demostrar por qué nuestro IDS ha clasificado dicho ataque como tal podemos tener problemas legales para realizar una acusación formal contra el atacante.

5.6. Modelo neuronal basado en *Spiking Neural Networks*

El motor de análisis del IDS será una modelización de una *Spiking Neural Network* en donde se tenga en cuenta parámetros biológicos de gran importancia como la dimensión temporal y la predicción.

En general, los modelos actuales suelen ser muy simplistas y esto es un problema que está, por el momento, limitando la potencia de estos modelos. Los sistemas de redes de neuronas artificiales se basan en modelos excesivamente simples en relación con el funcionamiento del cerebro. Hasta la fecha se han obviado gran cantidad de parámetros como la **bidireccionalidad** de la información que trasvasa la red neuronal y el factor **tiempo**. Muchos modelos no tienen en cuenta la

dimensión temporal ni en el aprendizaje, ni cuándo, se comprueban los patrones siendo realmente ésto una de las claves del funcionamiento neuronal en el cerebro. Evidentemente esto son afirmaciones genéricas puesto que sí existen modelos que contemplan tanto la bidireccionalidad como la dimensión temporal, sin embargo son muy pocos, no completos y desde luego sólo se estudian en el ámbito científico.

Por otro lado, los modelos empleados hasta la fecha (véase Mapas Autoasociados o perceptrón multicapa con *backpropagation*) no terminan de capturar la esencia de los invariantes de los patrones aprendidos. En general, estos modelos suelen trabajar con datos normalizados de forma que una variación lo suficientemente grande (por ejemplo un cambio de escala o un objeto rotado en un sistema de reconocimiento visual) ocasionará problemas al clasificador y es muy probable que lo clasifique incorrectamente.

Es importante incorporar mecanismos de **predicción** que permitan al modelo operar más eficientemente, permitiendo predecir el comportamiento del sistema de forma similar a como el neocortex humano predice que una cara debe tener una nariz. De esta forma se dota al sistema de la capacidad de relacionar eventos secuencialmente. Cuando se encuentra frente a una inconsistencia, esto es, lo que se ha detectado difiere de lo que se esperaba, el sistema analizará en profundidad dicho evento y realizará las acciones oportunas.

5.7. Modelización de métodos biológicos de aprendizaje

Se adoptará un modelo de aprendizaje biológico en donde se modelen los estadios de memoria a largo plazo o *Long-term memory* (**LTM**) y memoria a corto plazo o *Short-term memory* (**STM**).

Las redes neuronales además son capaces de realizar aprendizajes tanto supervisados, como no supervisados. Sin embargo, los modelos que se han empleado hasta el momento en los IDS (perceptrón multicapa con *backpropagation*, mapas autoasociados de Kohonen, etc.), generalmente, suelen limitarse a un solo tipo de aprendizaje. Existen, no obstante, modelos híbridos que aplican ambos aprendizajes (por ej. SOM[17] + LVQ [18])[11],[12] aunque son todavía pocos.

Por otro lado cabe mencionar que prácticamente ninguno [13],[14] de los modelos empleados hasta la fecha realiza un aprendizaje "*on-line*". Es decir, las redes neuronales tienen dos fases. La fase de aprendizaje ("*off-line*") en donde van aprendiendo los patrones hasta que consiguen reconocerlos con una márgen muy pequeño de error y la fase clasificativa ("*on-line*") en donde emplean los conocimientos aprendidos para la clasificación de patrones nunca observados. Cuando la red neuronal pasa a la fase de clasificación, solo clasifica, no aprende nuevos patrones. Esto es un gran problema para los modelos actuales dado que es necesario para que el sistema reconozca nuevos patrones, llevarlo "*off-line*" y volver a realizar todo el aprendizaje para incorporar los nuevos ataques.

Por ello se propone un nuevo modelo que contemple ambos tipos de aprendizaje de forma similar a como aprendemos los seres humanos, dotándo al sistema de una mayor versatilidad a la hora de actualizar sus conocimientos. Se propone la codificación de los procesos que rigen la memoria a corto y largo plazo en la medida de lo posible.

---

[17] Self Organized Maps

[18] Learning Vector Quantization

6. Conclusiones

La propuesta expuesta en el presente documento es, hasta cierto punto, una propuesta arriesgada y ambiciosa. Se pretende emplear un nuevo modelo que mezcle conceptos propios de la detección de intrusos con conceptos neurobiológicos. Se espera que el modelo que se propone alcance tasas de detección y falsos positivos aceptables, pero manteniendo la capacidad de proceso necesaria para implementarlo dentro de unos límites adecuados. No se espera alcanzar las mejores tasas, sin embargo si se pretende sentar las bases para futuros desarrollos de sistemas realmente autónomos. Esto es, sistemas capaces de aprender de forma automática (supervisados y no supervisados) y de extraer razonamientos y juicios respecto a los datos recibidos de su entorno.

Referencias

[1] Dennin, D. E. (1986). An Intrusion-Detection Model. IEEE Computer Society Symposium on Research in Security and Privacy. 118-131.

[2] Anderson, J. P. (1980). Computer security thread monitoring and surveillance. Fort Washington, PA. April.

[3] Porras, P., Schnackenberg, D., Davis, Stillman, M. & Wu, F. (1998). The Common Intrusion Detection Framework Architecture.

[4] Feiertag, R., Porras, P., Kahn, C., Schnackenberg, D., Staniford-Chen, S. & Tung, B. (1999). A Common Intrusion Specification Language (CISL)

[5] Nolan, R., O'Sullivan, C., Branson, J. & Waits, C. (2005). First Responders Guide to Computer Forensics. CERT Training and Education. Carnegie Mellon, Software Engineering Institute. Pittsburgh, PA.

[6] Ptacek, T. H. & Newsham, T. N. (1998). Insertion, Evasion, and Denial of Service: Eluding Network Instrusion Detection. Secure Networks Inc.

[7] Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschogrod, D., Cunningham, R. K., & Zissman, M. A. (1999). Evaluating Intrusion Detection Systems: The 1998 DARPA Offline Intrusion Detection Evaluation. Lincoln Laboratory MIT. RAID'99.

[8] McHuge, J. (2000). Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory Carnegie Mellon University. ACM Transactions on Information and System Security, Vol. 3, No. 4, November. 262-294.

[9] Bellovin, S. M. (1993). Packets found on an Internet. August 1993. ACM SIGCOMM Computer Communication Review. 26-31.

[10] Kumar, S. (1995). Classification and detection of computer intrusions. Ph.D Thesis. Purdue University.

[11] Haykin, S. (1999). *Neural Networks: A comprehensive foundation*. London, Prentince Hall International. 2nd Edition.

[12] Carrascal, A., Couchet, J., Ferreira, E. & Manrique, D. (2005). Arquitectura neural jerárquica para la detección de anomalías en el tráfico TCP/IP.

[13] Albus, J. S. (1975). A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC). *Journal Dyn. Syst. Meas. Control, Trans.* ASME 97. 220-227.

[14] Cannady, J. (2000). Applying CMAC-Based On-Line Learning to Intrusion Detection. Proceedings of the 2000 IEEE/INNS Joint International Conference on Neural Networks.

[15] Cannady, J. & Harrel, J. (1996). A comparative Analysis of Current Intrusion Detection Technologies. Proceedings of Technology in Information Security Conference (TISC). 212-218.