Diseño de Sistemas Distribuidos de Detección de Anomalías de Red

Autor: Luis Campo Giralte
Tutor: Ricardo Jiménez Peris y Marta Patiño Martínez

Índice de Contenidos

Introducción 3
Estado del Arte 5
SmartDefense 6
Symantec Network Security 7100 7
Cisco Secure IDS 4230 9
Problemas Actuales 11
Ataques Distribuidos de Denegación de Servicio 11
Filtrado del acceso 13
Control de Flujo 15
Ataques de Fragmentación y Timeouts 18
NIDS Timeout < Host Victima 19
NIDS Timeout > Host Victima 20
TTL Fragmentación 21
Propagación de Virus 22
Escaneo Secuencial 23
Escaneo Binario 24
Preescaneo 25
Preescaneo con Preinfección 26
Arquitectura de Diseño 27
Diseño NIDS 27
Análisis de Protocolos 28
Reconocimiento de Patrones 30
Statefull Inspection 31
Motor de Detección de Snort 33
Reglas de Detección 35
Preprocesadores 36
Diseño Distribuido con Snort 40
Arquitectura 43
Conclusiones 48
Referencias 49

Introducción

Los sistemas de detección de intrusos basados en red(NIDS), en los cuales nosotros nos vamos a centrar, son aquellos capaces de detectar ataques contra diferentes sistemas de una misma red, aunque los hay que pueden detectarlos en diferentes redes. Para lograr su objetivo, al menos uno de los interfaces de red de esta maquina sensor trabaja en modo promiscuo, capturando y analizando todas las tramas que pasan por el en busca de patrones que indiquen se esta produciendo un ataque.

Los NIDS analizan el tráfico de la red completa, examinando los paquetes individualmente, comprendiendo todas las diferentes opciones que pueden coexistir dentro de un paquete de red y detectando paquetes armados maliciosamente y diseñados para no ser detectados por los cortafuegos. Pueden buscar cual es el programa en particular del servidor de Web al que se está accediendo y con que opciones y producir alertas cuando un atacante intenta explotar algún fallo en este programa. Los NIDS tienen dos componentes:

- Un sensor: situado en un segmento de la red, la monitoriza en busca de tráfico sospechoso.
- Una Consola: recibe las alarmas del sensor o sensores y dependiendo de la configuración reacciona a las alarmas recibidas.

Las principales ventajas del NIDS son:

- Detectan accesos no deseados a la red.
- No necesitan instalar software adicional en los servidores en producción.
- Fácil instalación y actualización por que se ejecutan en un sistema dedicado.

Los sistemas NIDS presentan también una serie de desventajas y generan serios cuestionamientos sobre su efectividad, algunos de ellos son los siguientes:

- No existen tecnologías nuevas; para la detección de eventos de seguridad (patrones, funciones estadísticas, algoritmos de inteligencia artificial, etc.)
- La concentración de elementos de seguridad en un solo punto genera los llamados “puntos únicos de fallo”, y hay que ir a sistemas distribuidos.
- El compartir recursos puede traer problemas; cuando se combina un IDS con un Firewall, un equipo con el doble de capacidad de procesamiento no es lo mismo que 2 equipos con la mitad de capacidad de procesamiento (ambos utilizan un mismo procesador, así como las mismas entradas y salidas de datos; esto genera cuellos de botella si la carga de trabajo es excesiva). Hoy en día muchos IDS tienen problemas de rendimiento debido al volumen de datos que tienen que analizar.

- Examinan el tráfico de la red en el segmento en el cual se conecta, pero no puede detectar un ataque en diferentes segmentos de la red. La solución más sencilla es colocar diversos sensores.
- Pueden generar tráfico en la red, aunque no deberían y si lo generan debería ser en otro segmento de la red.
- Ataques con sesiones cifradas son difíciles de detectar, por no decir que prácticamente imposible.
- 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 NIDS solo pueden analizar el trafico 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 NIDS actuales no son capaces de procesar a semejante ritmo y, por tanto, son propensos a perder paquetes con información relevante.
- Los NIDS deben ser statefull, 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.
- El componente que almacena los registros del IDS puede, bajo una alta carga, llegar a su lımite físico y, por tanto hacer que el IDS deje de funcionar, o bien que comience a perder registros significativos por falta de espacio.

Otro de los grandes problemas que existen es la falta de estandarización sobre el tema, hay dos grandes corrientes en la formalización de estos sistemas.

CIDF [1] (Common Intrusión Detection Framework), el cual estaba promovido por DARPA(Defense Advanced Research Proyects Agency) y esta orientado principalmente a la investigación de detección de intrusiones. Este tuvo muy poca aceptación comercial aunque sus conceptos si son usados en la actualidad. Este define un Lenguaje propio de comunicación entre elementos del Framework y también define un lenguaje para definir los datos, CISL (Common Intrusión Specification Languaje).

IDEF [2](Intrusión Detection Exchange Format), fue definido por el Intrusión Detection Working Group del IETF. Este estaba formado por empresas relacionadas con el mundo de las Intrusiones y en desacuerdo con la CIDF, define protocolos de comunicación como son el IDXP (Intrusión Detection Exchange Protocol) y para los datos el protocolo IDMEF (Intrusión Detection Message Exchange Format). Hay que hacer mención a un Framework de Detección de intrusos que sigue dicho estándar como es Prelude-IDS [3], que sigue el estándar de protocolo IDMEF.

Estado del Arte

A continuación vamos a ver las características mas relevantes de los NIDS que hay actualmente en el mercado, hay que comentar que ninguno de los fabricantes comentados dan información técnica interesante, solo que sus motores son en tiempo real, que si inteligencia artificial y características que quedan muy bien en un lenguaje comercial pero que no nos dan una idea lo suficientemente aproximada sobre su funcionamiento, de todas formas en la siguiente dirección, http://www.securitywizardry.com/N_ids.htm, tenemos una lista con los enlaces a muchos de estos Sistemas.

También hay que tener en cuenta que no por pagar un dineral por uno de estos sistemas vamos a estar protegidos, de echo los sistemas de seguridad cerrados la confianza que hay que tener en ellos es 100% y la realidad corresponde a que estos sistemas fallan, no detectan lo que deberían detectar, general falsas alarmas, su implantación con otros elementos depende del fabricante, no siguen ningún estándar y su integración con otras plataformas es prácticamente inexistente y por no hablar del coste que tiene un sistema de este tipo para una Pyme por ejemplo.

También hay que comentar que muchos de estos productos que hay en el mercado son realmente buenos, tienen muchas funcionalidades, tenemos el respaldo de una gran empresa detrás, nos abstraemos de funcionalidades que un ingeniero de seguridad debería conocer, se integran con otro tipo de productos de seguridad, etc…

SmartDefense

La solución SmartDefense[4] que inaugura las nuevas soluciones de defensa activa, ofrece protección avanzada en tiempo real contra todo tipo de ataques a red CheckPoint, líder mundial en soluciones de seguridad para Internet, presentó hace unos días SmartDefense, que introduce una nueva categoría de productos de seguridad de Internet: las soluciones de defensa activa. La solución protege activamente a las organizaciones contra ataques de red conocidos y desconocidos utilizando tecnología de seguridad inteligente. Bloquea los ataques por tipo y clase utilizando la tecnología Statefull Inspection patentada por el fabricante y ofrece una única consola centralizada que proporciona información en tiempo real sobre los ataques así como detección, bloqueo, logging, auditoria y alerta sobre los mismos.

Dentro de los distintos tipo de ataques que previene la herramienta se encuentran:

- Ataques Ip - en los que se incluye la fragmentación y el spoofing.
- Ataques de denegación de servicio (DoS) - incluso desbordamiento SYN y LANd.
- Vulnerabilidades de aplicaciones e Internet - incluyendo caballos de troya, ataques DNS y gusanos del tipo Nimda y Code Red.
- Sondeo de Red - entre los que se incluyen la exploración de puertos y servicios.

SmartDefense complementa los sistemas de detección de intrusos y soluciones anti-virus ya que proporciona un capa de protección adicional avanzada.

Adicionalmente, la arquitectura del software de CheckPoint permite a SmartDefende incorporar rápidamente inteligencia y protección contra nuevos y emergentes ataques - ofreciendo un nivel de flexibilidad que no puede ser alcanzado por hardware alternativo o arquitecturas basadas en ASIC.

Dentro de las principales características de SmartDefense se incluyen:

- Actualizaciones de seguridad on-line sobre ataques e inteligencia global para facilitar a los administradores el control de la actividad malévola.
- Detección mejorada de actividades sospechosas, para responder a posibles amenazas en tiempo real.
- Gestión flexible y sencilla mediante un interfaz intuitivo.
- Identificación de ataque basado en tipos o categorías, para una mejor protección contra ataques emergentes y desconocidos.

Symantec Network Security 7100

El personal de seguridad tiene la tarea de asegurar la disponibilidad de los datos críticos de una organización. La plantilla de las organizaciones puede incluir profesionales competentes, pero estos empleados a menudo están sobrecargados de tareas diarias de análisis y respuesta a incidentes, aplicación y prueba de parches y planes para impedir la próxima intrusión. Al mismo tiempo, se enfrentan también a un aumento de las amenazas y de las presiones para establecer prácticas y controles de seguridad sólidos. Las organizaciones necesitan no sólo mantener y procesar en paralelo las inversiones existentes, sino que también deben practicar proactivamente monitoreos, inteligencia y análisis constantes. Symantec Network Security 7100[5] proporciona una solución de seguridad de red con despliegue simplificado, administración centralizada y soporte completo.

Las funciones mas destacables son las siguientes:

- Aumenta la seguridad desplegada de Gateways y servidores existentes para impedir que las amenazas se propaguen a través de las redes.
- Combina múltiples tecnologías de detección, incluyendo detección de anomalías de protocolo e intercepción de ataques de vulnerabilidades en la arquitectura IMUNE™ para identificar y bloquear con exactitud tanto los ataques conocidos como los desconocidos y los gusanos.
- Ayuda a las organizaciones a establecer, medir e informar sobre las mejores prácticas de organización y las iniciativas de cumplimiento.
- Experiencia integrada de Symantec™ Security Response and Services que proporcionan información temprana de amenazas para ofrecer seguridad proactiva.
- No requiere volver a configurar la red para realizar una instalación de forma fácil.
- Los dispositivos son compatibles con ocho interfaces, para permitir a las organizaciones monitorear más segmentos de red.
- Tres modelos admiten ancho de banda de red agregado de 50 Mbps a 2Gbps, para satisfacer las necesidades de instalación en sucursales, sitios de distribución y la red central.
- Función AutoProtect que actualiza automáticamente las políticas de protección mediante la tecnología LiveUpdate para ayudar a las organizaciones a mantenerse por delante de las siempre cambiantes amenazas.
- Función Prevenir con un clic que realiza la transición de un dispositivo de detección a una herramienta de prevención con un solo clic del raton.

Prevención proactiva de intrusiones

Las funcionalidades de Symantec™ Network Security 7100 ofrecen prevención proactiva frente a las intrusiones de red en tiempo real para proteger las redes corporativas y atenuar las interrupciones de la actividad comercial ocasionadas por ataques conocidos y desconocidos (o de “día cero”) y por gusanos.

Arquitectura de atenuación de amenazas de red

Symantec Network Security 7100 emplea un innovador motor de red IMUNE (del inglés, Intrusion Mitigation Unified Network Engine = motor unificado de mitigación de intrusiones de red).

Despliegues empresariales flexibles

La serie 7100 ofrece tres modelos que proporcionan opciones flexibles de despliegues de prevención de intrusiones para satisfacer mejor las necesidades de despliegue de las organizaciones, tanto si se requiere seguridad de red para sucursales, sitios distribuidos o la red.

Contenidos y actualizaciones de seguridad inteligentes

La experiencia integrada de Symantec Security Response y los servicios Symantec DeepSight™ Early Warning, además de las directrices de seguridad fáciles de comprender, permiten respuestas aún más rápidas a los incidentes de seguridad.

Administración completa

Una administración exhaustiva que ayuda a las organizaciones a establecer, medir e informar sobre las mejores prácticas de organización.

Análisis de amenazas en tiempo real

Symantec Network Security 7100 reúne inteligencia de múltiples sensores en toda la empresa para identificar de forma rápida y automática las tendencias y detectar los sucesos relacionados, a media que ocurren.

Administración de intrusiones para múltiples productos

Los agentes inteligentes de seguridad de red de Symantec proporcionan administración de intrusiones de múltiples fuentes de toda la empresa recopilando, agregando y respondiendo a los sucesos de múltiples productos de seguridad de red y de Host de Symantec y de terceros. La rápida identificación de amenazas de múltiples fuentes de sucesos en toda la empresa permite a las organizaciones atenuar posibles daños a los activos de uso crítico.

Cisco Secure IDS 4230

El sensor Cisco Secure IDS 4230 [6] es un "dispositivo" de seguridad de red que detecta la actividad no autorizada que la atraviesa, como por ejemplo ataques por parte de hackers, mediante el análisis del tráfico en tiempo real, y permite a los usuarios responder con rapidez a las amenazas de seguridad. Cuando se detecta una actividad no autorizada, el sensor puede enviar alarmas a la consola de administración con detalles de la actividad y puede controlar otros sistemas, como los Routers, para terminar las sesiones no autorizadas.

El sensor Cisco Secure IDS 4230 se ha optimizado para el control de los entornos de 100 Mbps y resulta ideal para el control del tráfico de puertos SPAN (Switched Port Analyzer) y segmentos Fast Ethernet. También se recomienda para el control de múltiples entornos T3.
Los sensores pueden colocarse en lugares en los que otros dispositivos de seguridad no son útiles, por ejemplo:

- Segmentos internos de red.
- Delante de un Firewall.
- Detrás de un Firewall.
- Detrás de un servidor de Modems para acceso telefónico.
- En conexiones Extranet.

Los sensores pueden colocarse en casi todos los segmentos de la red de la empresa donde se requiera visibilidad de la seguridad.

Las Principales características son las siguientes:

- Forma parte del modelo SAFE para el comercio electrónico*:* el sensor es un componente necesario para una estrategia de defensa en profundidad y eficaz, y para complementar otros mecanismos de seguridad implantados (por ejemplo, Firewalls, cifrado y autenticación). Como componente dinámico de seguridad de la línea de productos de seguridad de Cisco, el IDS puede funcionar en entornos Internet e intranet para proteger toda la red de la empresa.
- Detección y respuesta a las intrusiones en tiempo real*:* el sensor proporciona control y detección del mal uso de la red en tiempo real, utilizando para ello a la propia red como fuente de datos (esto es, capturando paquetes directamente de la red). Responde de forma activa a la actividad no autorizada, bloqueando el acceso a la red o terminando las sesiones dañinas.
- Completa cobertura de reconocimiento de ataques/firmas*:* el sensor detecta una amplia variedad de ataques. También incluye un sofisticado reensamblaje de fragmentación IP y capacidades de detección anti-IDS "Whisker".
- Rendimiento de alta velocidad*: el* sensor es especialmente recomendable para el control de entornos de 100 Mbps.
- Dispositivo integrado de seguridad IDS*: el* sensor conforma una solución completa "llave en mano" y "plug-and-play". Todo el paquete hardware y software se fabrica, prueba y mantiene por un solo fabricante.

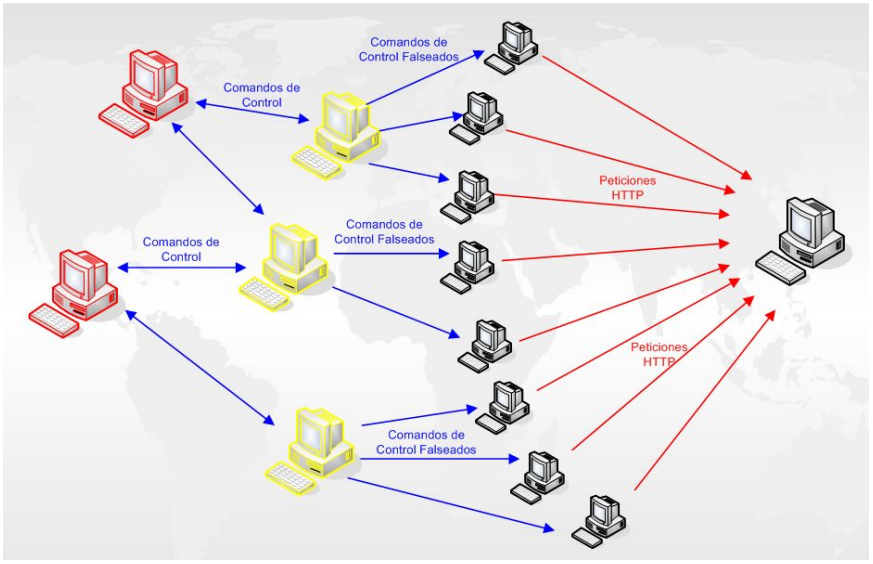
- Bajo coste de propiedad: el sensor es sencillo de instalar, configurar y mantener.
- Sencilla instalación: la instalación del sensor es rápida y sencilla, ya que requiere sólo siete parámetros de direccionamiento y no es necesaria una formación especial. Cuando el sensor se encuentra ya instalado, como un dispositivo autónomo con una potente configuración predeterminada, empieza a realizar su tarea de control inmediatamente.
- Funcionamiento transparente*:* el sensor no está diseñado para afectar al rendimiento de la red y es totalmente transparente al usuario final. Se incorpora a la red y es completamente invisible a los usuarios finales, lo que incrementa el nivel de seguridad sin afectar al rendimiento o la funcionalidad.

Problemas Actuales

Ataques Distribuidos de Denegación de Servicio

Hoy en día, los mecanismos de ataque por denegación de servicio resultan bastante más sofisticados, empleando debilidades de los protocolos Tcp/Ip para generar auténticas avalanchas de paquetes sobre un sistema concreto, o simples estados de inconsistencia que provocan que el proceso que atiende el servicio quede inoperante. En algunos casos se han llegado a detectar ataques que generaban más de 1Gbps hacia el sistema atacado. Es cierto que Tcp/Ip v.4 carece de mecanismos de seguridad que permitan atajar estas prácticas, pero cuando se diseñó la conectividad era lo importante, pues debía ser capaz de afrontar con éxito ataques externos en la infraestructura de comunicación, y en cualquier caso nadie podía imaginar en qué desembocaría aquella red.

Uno de los problemas en los cuales nos encontramos en la actualidad es la detección de una Denegación de Servicio Distribuida, hay métodos y técnicas los cuales nos indican que se esta produciendo como puede ser el análisis de trafico de ciertos puertos los cuales se usan para controlar los esclavos o Host zombies como llaman algunos autores, pero es muy difícil detectar lo que es el trafico que va dirigido a la organización victima y mas si este es un trafico cifrado como puede ser Https.



También existe los denominados Ataques de Amplificación, que son los últimos realizados actualmente, basados en el protocolo DNS.

El ataque [34] básicamente se basa en usar un registro de recurso (RR) de tipo TXT del protocolo DNS lo suficientemente largo como para que se dé la amplificación, de unos 4000 bytes podría ser más que suficiente. Así, para una

consulta que puede ser de unos 60 bytes podemos obtener un factor de amplificación de 66, suponiendo una respuesta de 4000 bytes.

Por ejemplo, con los servidores de hotmail.com se consigue una amplificación de 9.5, con los de aol.com un factor de 10 y con mci.com un 10.2. Aun así, la diferencia del tamaño de la respuesta no es lo suficientemente alarmante.

La limitación de este ataque se manifestaba por el límite de los 512 bytes del protocolo UDP. Sí una respuesta superara este límite se enviaría por el protocolo TCP, y no funcionaría.

Sin embargo en La Rfc 2671 [33] se sugiere que el cliente pueda cambiar el tamaño máximo del paquete UDP mediante un registro de recurso (RR) llamado OPT. Por lo tanto, basta con buscar un servidor DNS que soporte la extensión EDNS. Bind 9 lo soporta, con un límite de 4096 bytes de respuesta máximo. Djbdns, sin embargo, no lo soporta. De todas formas, en Bind se puede configurar este tamaño con la opción edns-udp-size.

Total, que con una lista de servidores que cumplan esas condiciones, y un registro TXT lo suficientemente grande, bastaría para poner en apuros a más de uno. Siendo bastante complicada la localización del verdadero atacante.

Las consecuencias del ataque las sufrirían tanto el servidor DNS como la víctima del ataque. Para reducir el ataque en el servidor autoritativo del dominio también convendría que el RR TXT en cuestión tuviera un TTL (Time To Live) alto, para que se cacheara la respuesta en los servidores recursivos utilizados para el ataque.

También se está discutiendo en las listas la idea de tratar a los "open resolvers" (los servidores DNS recursivos de acceso público) como a los "open relays".

Filtrado del acceso

Filtrando paquetes seleccionados se puede disminuir el impacto de algunos ataques DoS o también, en algunos casos, bloquear totalmente los ataques.

Por ejemplo, el filtrado de paquetes en el borde de la red debe prevenir:

- el flujo de paquetes falsos desde Internet para atacar un ordenador principal en una red de área local (LAN), o
- de un atacante en una LAN que ataca a un ordenador principal en Internet pretendiendo ser una persona diferente (engaño del origen).

Todos los paquetes entrantes deberían filtrarse por norma para descartar paquetes falsos tan rápido como sea posible. Este proceso, conocido como filtrado del acceso [7], se puede basar en los campos de la cabecera del Protocolo Internet (Ip), del protocolo de mensaje de control Internet (Icmp), del protocolo de datagramas de usuario (Udp) y del protocolo de control de transmisión (Tcp), a continuación vamos a ver como se realiza dicha técnica.

La Técnica de Filtrado de acceso, también llamada Unicast Reverse Path Forwarding(uRPF) es una técnica para implementar el filtrado de acuerdo con la rfc 2827 [8] . uRPF puede tirar paquetes con direcciones falseadas en los Routers fronteras, dependiendo del método usado, hay que recordar que uRPF solo funciona en el interfaz de entrada del Router frontera.

uRPF se puede implementar de dos formas, en modo estricto y en modo perdida , en modo estricto el Router toma la dirección origen del paquete y la valida contra la FIB(Forwarding Information Base) del interfaz de entrada del Router, y también valida que existe una ruta a la dirección origen en la tabla de adyacencia del interfaz, y si no existe entonces el paquete se tira.

El Modo perdida fue diseñado para entornos frontera entre ISP´s, y este solo realiza la comprobación del la FIB y la otra comprobación no la realiza Dicha técnica fue desarrollada por Cisco y necesita usar el Cisco Express Forwarding(CEF), una configuración básica podría ser la siguiente [9]:

```
ip cef
i
interface serial X/X/X
ip verify unicast reversepath <acl>
```

La opción de la acl(listas de acceso) es opcional, la lista de acceso de entrada es procesada con el primer paquete de entrada, solo después de que el Router halla comprobado la lista de acceso comprobara el uRPF, a continuación el modulo CEF termina su trabajo mirando el interfaz de salida del paquete, si la lista de acceso esta comprobada y ha pasado entonces el paquete se encamina, y si la comprobación del modulo CEF falla el paquete se desecha.

La sintaxis para soportar el modo Perdida es la siguiente:

ip verify unicast source reachablevia
(rx|any) [allowdefault]
[allowselfping]
[<list>]

La opción rx comprueba que el paquete tiene ruta de vuelta en el interfaz del que venia, como en modo estricto. La opción any es el modo Perdida que no comprueba el interfaz, el resto de opciones son opcionales, una posible configuración en un interfaz en modo perdida podría ser la siguiente:

ip verify unicast source reachable-via any

Cisco propone la siguiente Tabla [9] para implementar los diferentes métodos.

Deployment Situtation	**Type of uRPF to use**	**Config Notes**
Lease Line Customer	Strict Check	
Multihomed Lease Line Customer (same ISP)	Strict Check or Loose Check	Remember to use BGP Weights on Strict Check
Multihomed Lease Line Customer (different ISPs)	Strict Check or Loose Check	Remember to use BGP Weights on Strict Check
Dialup Customers	Strict Check	
DSL Customers	Strict Check	
Cable Modem Customers	Strict Check	
IXP Connection – noprivate peering	Strict Check	
IXP Connection w/private peering	Loose Check	
Private Peering – Dedicated Router	Strict Check	Strict Check Symmetry should be expected between the routes advertised and source addresses sent by the peering ISP.
Private Peering /w several ISPs on the same Router	Loose Check	
CoLocation Provider's Edge Routers	Loose Check	

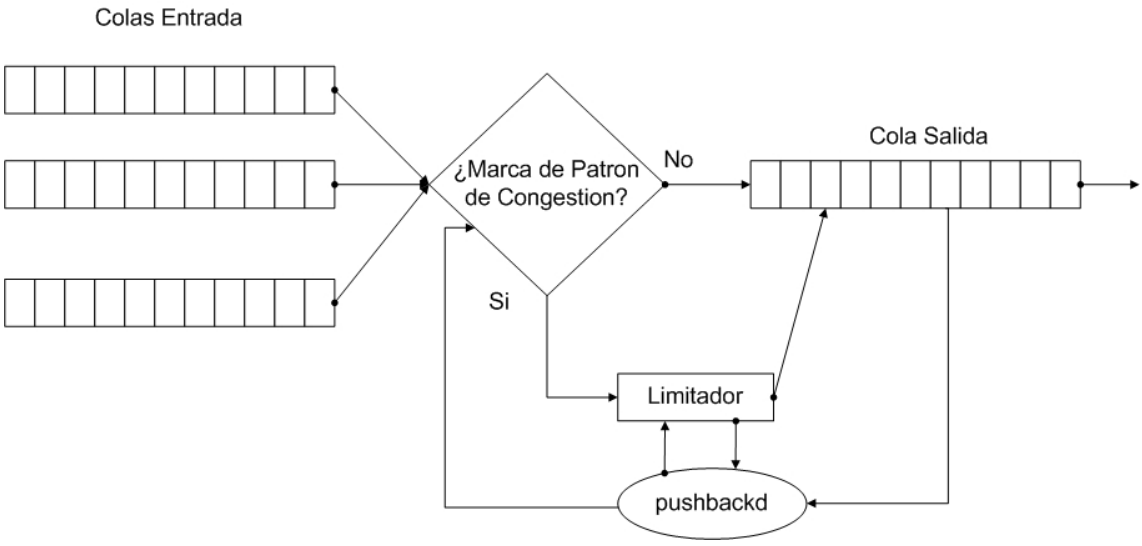
Control de Flujo

Uno de los mecanismos mas interesantes para mitigar y hasta detener el ataque son los sistemas basados en control de flujo, los cuales detectan y notifican una posible congestión de trafico “malo” y toman medidas para solucionarlo, una técnica usada es el método Pushback [10], que es una de las primeras propuestas para tratar este problema desde la perspectiva de la red.

Este método generaliza la propuesta de limitación de velocidad usada en los métodos de la QoS, que no diferencia entre flujos atacantes y no atacantes, si no que se basa en ciertos parámetros de congestión.

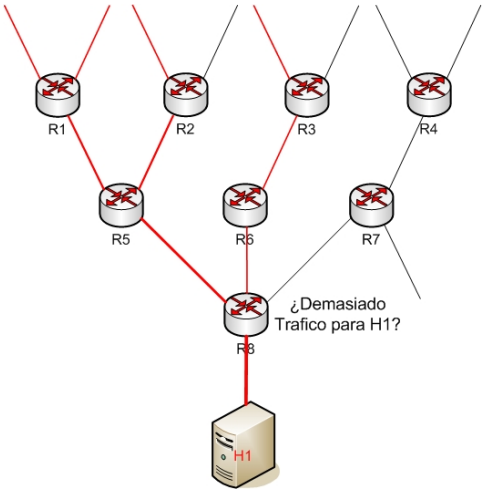
Cada Router de la red, implementado con OpenBsd , esta formado por los siguientes elementos:

- Una cola de salida gestionada con un mecanismo de descartes para evitar desbordamientos.
- Un agente añadido para analizar los paquetes eliminados por el mecanismo de descartes para identificar algunas firmas.
- Un módulo que limita la velocidad para limitar el número de paquete que se corresponden con las firmas atacantes que entrar en la cola de salida.

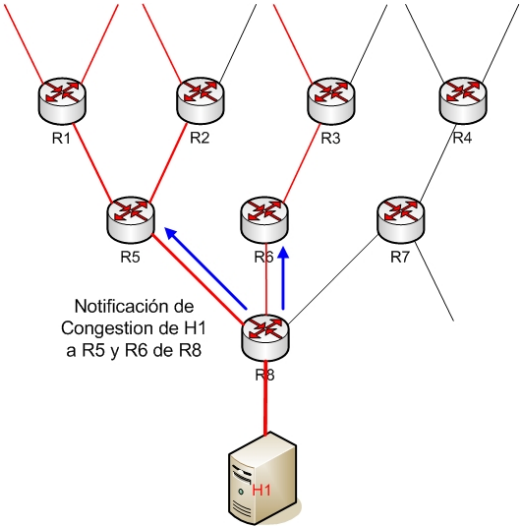


Si un elemento de red no detecta un ataque, los paquetes entrantes se encaminan directamente a la cola de salida. Tan pronto arranca el mecanismo de descartes de la cola para eliminar paquetes, el agente añadido intenta identificar secuencias. Si se identifica una secuencia, cualquiera de los siguientes paquetes entrantes que se ajusten a esta secuencia tienen que ser transmitidos al módulo que limita la velocidad antes de pasarlo a la adecuada cola de salida.

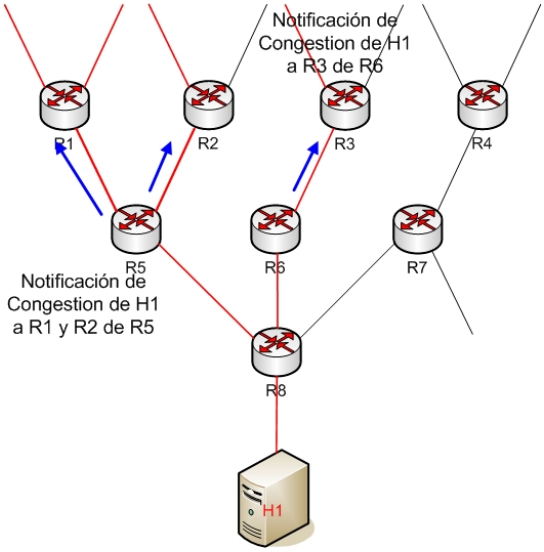
Una vez se ha identificado que un elemento de red ha identificado una secuencia atacante, reacciona localmente y después propaga la reacción a los Routers ascendentes de la siguiente forma.



Uno de los Routers detecta que hay demasiado trafico para el Host H1, en este caso lo detecta el Router 8 al recibir mas trafico del usual, en el dibujo marcamos el trafico malicioso por medio del color rojo el cual se va intensificando según llega a su destino.



El Router 8 envía mediante un datagrama UDP a los Routers 5 y 6 que el Host 1 esta recibiendo demasiado trafico y que puede entrar en saturación. En ese instante tanto el Router 5 como el 6 aplican latencia en el trafico dirigido al Host 1.

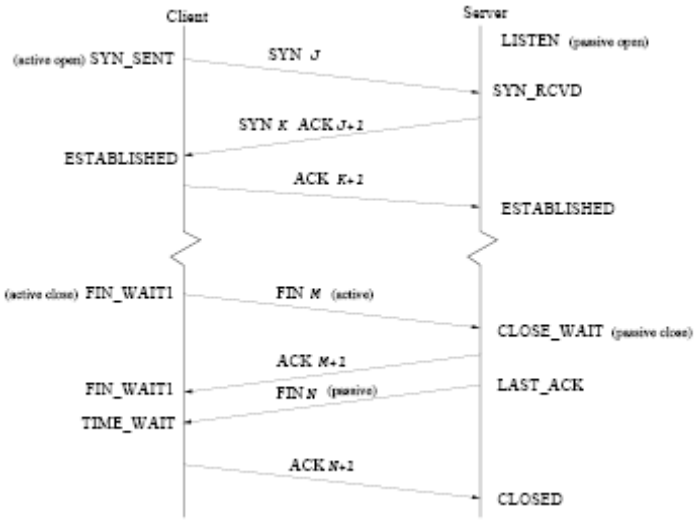


Tanto el Router 5 como el 6 envían a sus Routers adyacentes la notificación de Congestión del Host 1 y comienzan a reducir el flujo que va dirigido a este.

En cada uno de los Routers que reciben la información de congestión, aplican a los flujos que están implicados en H1, limitaciones de ancho de banda, con el fin de poder mitigar el posible ataque.

Lo interesante del método anterior es la necesidad de realizad distribución de elementos para intentar mitigar el ataca distribuido, vamos que si no puedes contra tu enemigo es mejor aliarse con el, además actualmente los clusters son muy baratos y por medio del Opensource se pueden realizar sistemas de bajo coste en los cuales los elementos hardware se abaratan considerablemente lo cual permite que dichos sistemas que eran proyectos de investigación se puedan llevar a cabo como el sistema anteriormente comentado.

Existe otro método de detección de DDos [32] , que se basa en la detección de los pares Syn-Fin de Tcp los cuales me indican que se ha producido un flujo Tcp, aunque la verdad en dicho método no comenta que se hayan producido en medio reconexiones las cuales tiran un poco dicho método abajo.



Ataques de Fragmentación y Timeouts

Vamos a empezar primero repasando unos ciertos términos los cuales luego nos ayudaran en la compresión de la problemática:

La **Fragmentación** es cuando un paquete de la capa de red es demasiado grande se fragmenta en otros mas pequeños, a esto se le llama fragmentación. El sistema destino es el encargado de realizar el reensamblado de dichos paquetes, si por la razón que sea alguno de estos paquetes se tiran por la red, por ejemplo por que su ttl sea mejor que uno, el sistema final será el encargado de tirar el resto de paquetes mediante un temporizador, ya que algo ha pasado por la red para que no llegue uno de los fragmentos. El Router que tiro el fragmento con el ttl menor que uno genera un paquete Icmp Time Exceded in Transit al sistema origen.

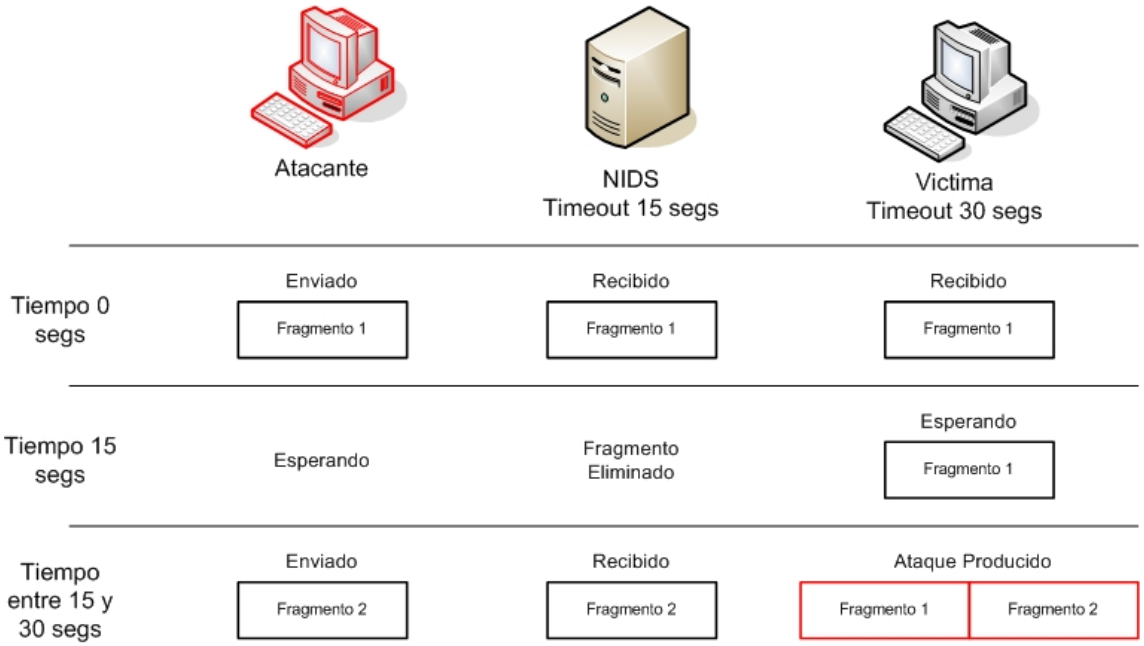
Con el **Tiempo de Reensamblaje** nos referimos al tiempo que da el sistema destino para descartar todos los paquetes Ip cuando uno de ellos no llega, esto se activa automáticamente en el destino, dentro de la pila Tcp/Ip, y es uno de los problemas con los que nos encontramos ya que los Sistemas operativos actuales tienen diferentes tiempos, por ejemplo los BDS tiene 60 segundos mientras que los sistemas Windows están en los 30 segundos aproximadamente.

El **Mensaje Icmp de Tiempo excedido para el reensamblaje** [11] es el mensaje Icmp que envía el sistema destino cuando uno de los datagramas Ip falta, el mensaje es un Time Exceded Message (type 11 y code 1).

A continuación vamos a explica mediante ejemplos los diferentes tipos de ataques de fragmentación que se producen y la necesidad de tener en los NIDS pilas de protocolos para sus posteriores análisis así como poder detectar dichos ataques los cuales juegan con la fragmentación de la información para que los NIDS no puedan detectar los ataques.

Los dos primeros tipos se diferencian en el tiempo de reensamblaje de los datagramas Ip, en el primer caso el NIDS tiene un tiempo menor de reensamblaje y en el segundo caso el tiempo es mayor, con respecto al tercer caso es un problema en el TTL de uno de los datagramas fragmentados.

NIDS Timeout < Host Victima



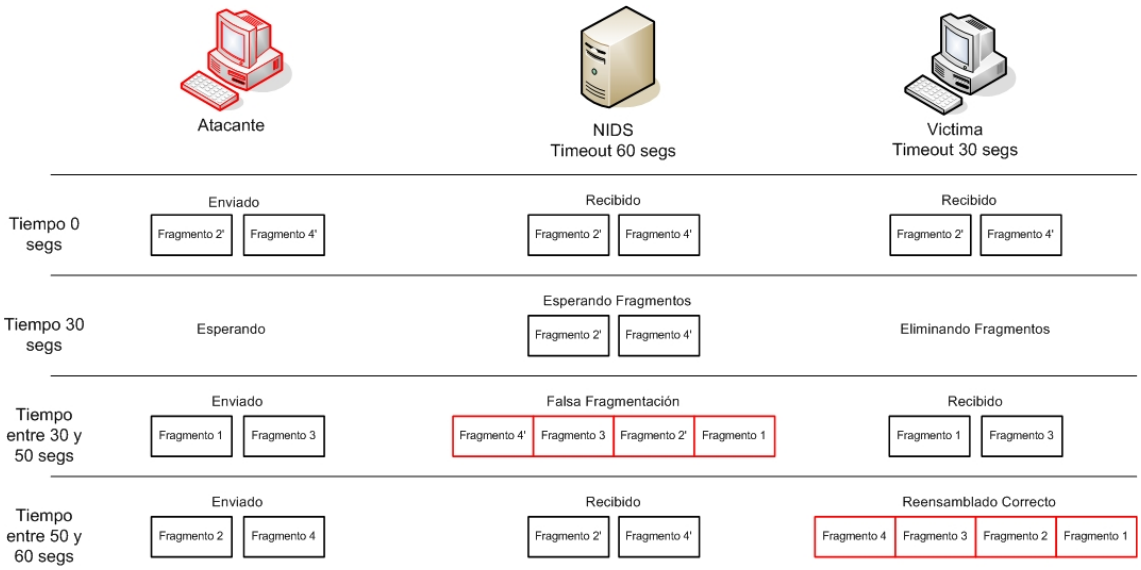
En esta primera figura tenemos el típico escenario en el cual tenemos el sistema de detección por ejemplo dentro de la subred de la victima.
En el tiempo 0 el atacante envía un exploit a la victima, pero dicho exploit lo envía en dos fragmentos, tanto el NIDS como la victima reciben un datagrama el cual el NIDS no puede realizar el análisis ya que no tiene el segmento Tcp al completo y simplemente tiene un trozo del exploit el cual todavía no es analizable en principio.

Cuando pasan 15 segundos el NIDS descarta el fragmento recibido y no genera ningún paquete ICMP ya que no es un Router y si lo generase indicaría que hay algún tipo de sistema por debajo y comprometeríamos la integridad del NIDS, en dicho tiempo el fragmento de la victima esta esperando ya que su timeout es de 30 segundos.

Entre los 15 y los 30 segundos el atacante envía el otro fragmento el cual es recibido por el NIDS el cual no se entera muy bien de que va la historia ya que el recibe un paquete fragmentado el cual no analizara y descartara ya que dicho paquete nunca se analizara. En este punto es donde el NIDS no puede generar la alerta o el evento que indique que se esta produciendo un ataque y el Host destino se vera comprometido, ya que este recibirá los dos fragmentos los reensamblara, los pasara al proceso correspondiente y se ejecutara el exploit correspondiente.

El siguiente caso es algo mas complejo en lo que se refiere al ataque pero en el vemos que seguimos con la misma problemática anterior, los temporizadores del reensamblado de los paquetes Ip.

NIDS Timeout > Host Victima

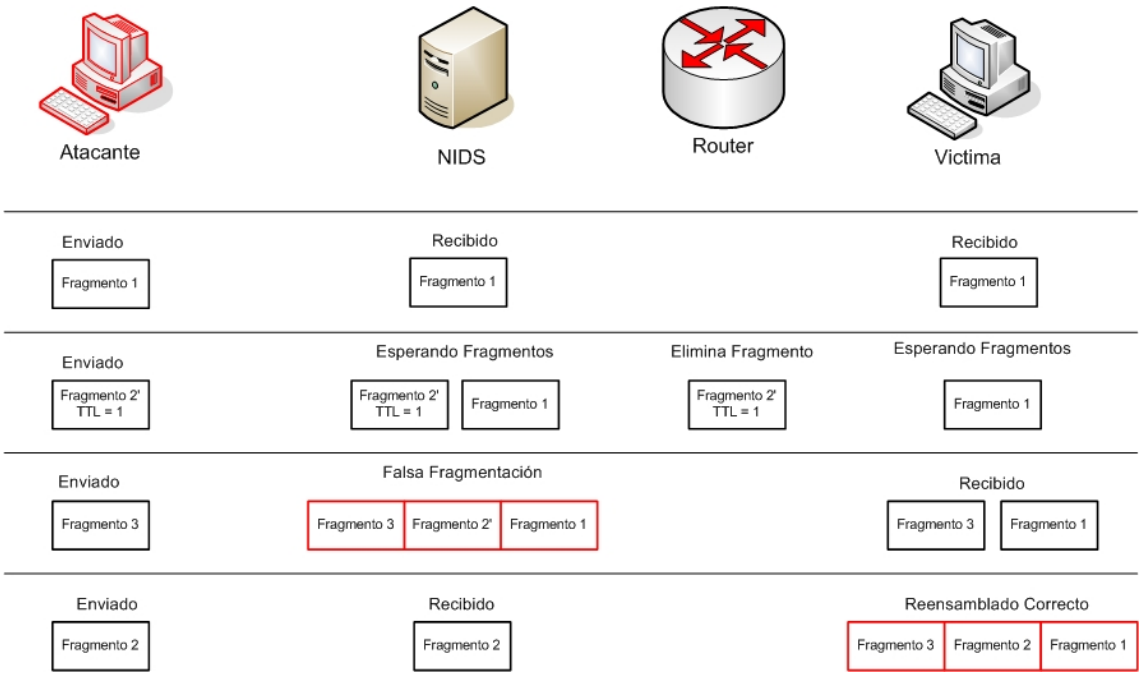


En este caso el NIDS tiene un timeout superior al Host destino.
El atacante envía dos paquetes fragmentados que nada tienen que ver con los originales los cuales los recibe el NIDS y el Host destino

En este caso el atacante envía lo primero dos paquetes fragmentados los cuales son falsos y ayudaran a que el NIDS realice el reensamble de una manera incorrecta.

En este punto donde el tiempo son 30 segundos podemos detectar un fallo de diseño de NIDS y es que el Host victima al tener un temporizador genera un paquete ICMP y el NIDS ignora y en este caso no debería ya que el se esta quedando con unos fragmentos de datagrama los cuales son incorrectos.

TTL Fragmentación



En este caso el ataque difiere de los anteriores, ya este no busca usar los temporizadores de reensamblaje del NIDS ni del Host destino.

El atacante necesita conocer previamente que tiene un Router entre el NIDS y la victima, hay técnicas para descubrirlo de echo en este caso el Router genera un paquete Icmp cuando uno de los datagramas le llega con ttl a 1, el cual descarta.

La técnica se basa en que el Router frontera descarta uno de los datagramas Ip y el NIDS no es capaza de realizarlo y este realiza un reensamblado de los datagramas incorrectos con lo que el posterior análisis será falso.

Con los problemas tanto de DDos y de Fragmentación nos encontramos actualmente y vamos a intentar realizar un diseño de nuestro sistema dando soluciones a dichos problemas.

Propagación de Virus

Desde la aparición en 2001 de “Code Red”, la comunidad Internet ha presenciado los terribles efectos de múltiples gusanos que han utilizado la Red como campo final de acción. Saphire/Slammer, Blaster y Sasser son algunas de las muestras más recientes y agresivas con las que hemos tenido que tratar en sistemas Microsoft. Si bien la mayoría de las empresas e instituciones que han confiado sus comunicaciones en Internet están cada día mejor preparadas para contrarrestar o evitar estas infecciones y los dispuestos en el código postinfección de estos programas maliciosos, todavía hoy nos encontramos a mucha distancia de poder decir que estamos totalmente a salvo de cualquier nuevo código vírico que pueda aparecer.

El gran problema al que nos enfrentamos es a la detección de este tipo de Virus que se expande por la red, obviamente dichas técnicas de detección, como detectar que un cliente de una red esta realizando intentos de conexión a Host con direcciones Ip secuenciales, paquetes mal formados para comprobar si existe una vulnerabilidad a explotar, etc…

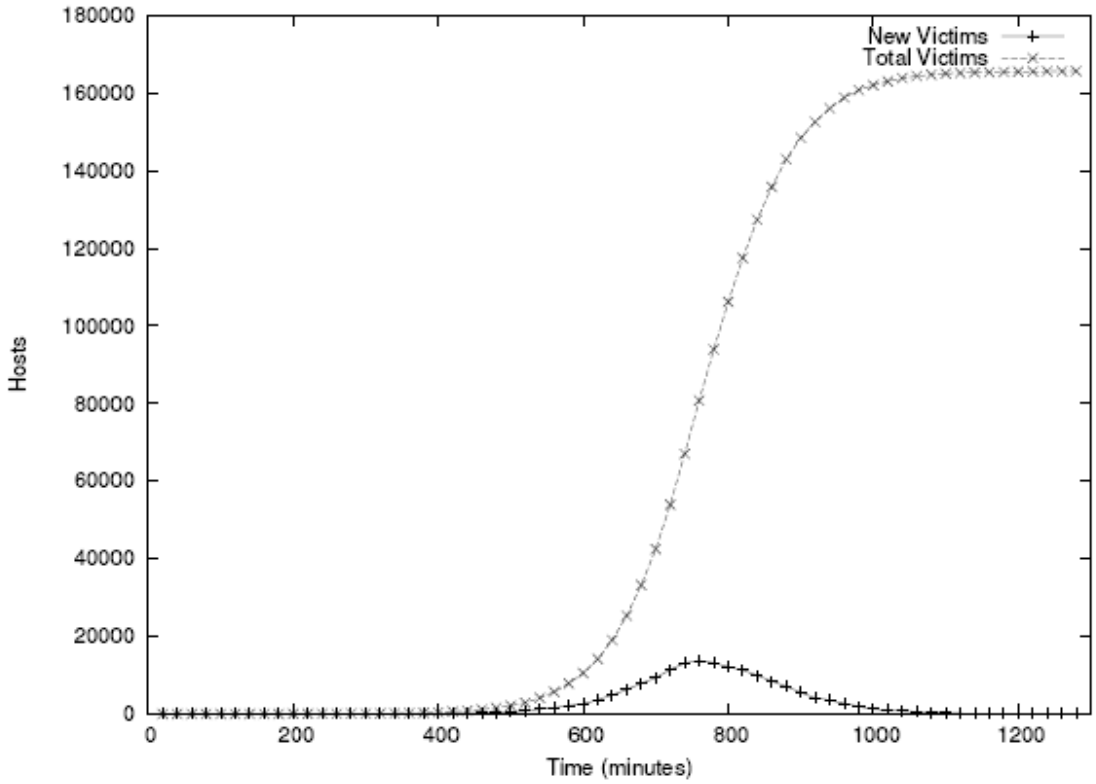
Dada la rápida actuación de las empresas proveedoras de servicios de detección de virus o código malicioso una vez este es reconocible, conseguir el mayor número de infecciones en el menor tiempo posible se convierte en uno de los principales objetivos de un gusano. Los factores que determinan la velocidad de propagación de un determinado gusano son parámetros de gran interés entre sus creadores para el desarrollo de futuros elementos más efectivos en este sentido. A día de hoy son destacables algunos de los avances realizados para tratar de identificar y optimizar los condicionantes directos de este ritmo de propagación.

Estos elementos de ataque se apoyan en la cuidada adopción de métodos [12] adecuados en la selección de direcciones, utilización de ejecución multi-hilo, realización de pre-escaneos para la detección de servicios y la utilización de rutinas eficientes en la búsqueda o la detección como en la posterior infección.

Actualmente las propagaciones de virus se pueden realizar de varias formas, vamos a comentar algunas de dichas técnicas pero sabiendo que hay muchas mas técnicas sobre el tema.

Escaneo Secuencial

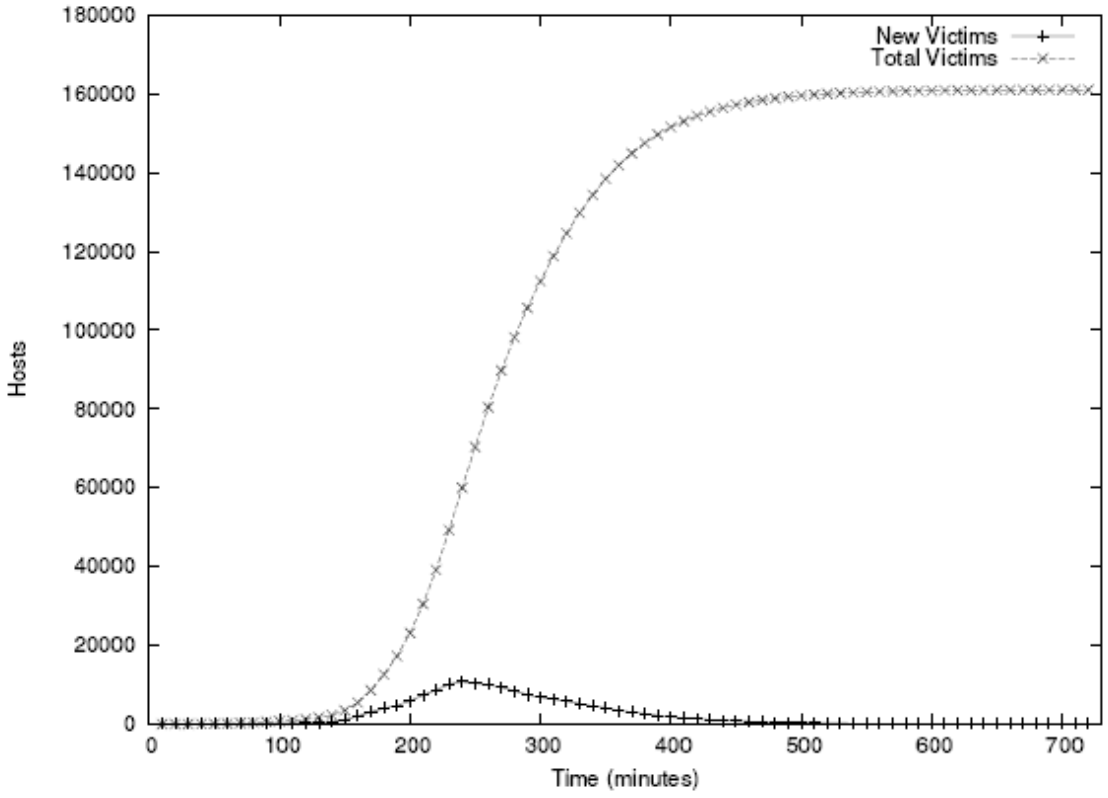
Se trata de ir secuencialmente escaneando el rango de direcciones Ip e intentando infectar los Host, el gusano Blaster usaba esta técnica para realizar su propagación.



La verdad que dicho gusano era muy escandaloso ya que enviaba muchos segmentos Tcp Syn y en los Logs se podía ver muy bien que era Blaster.

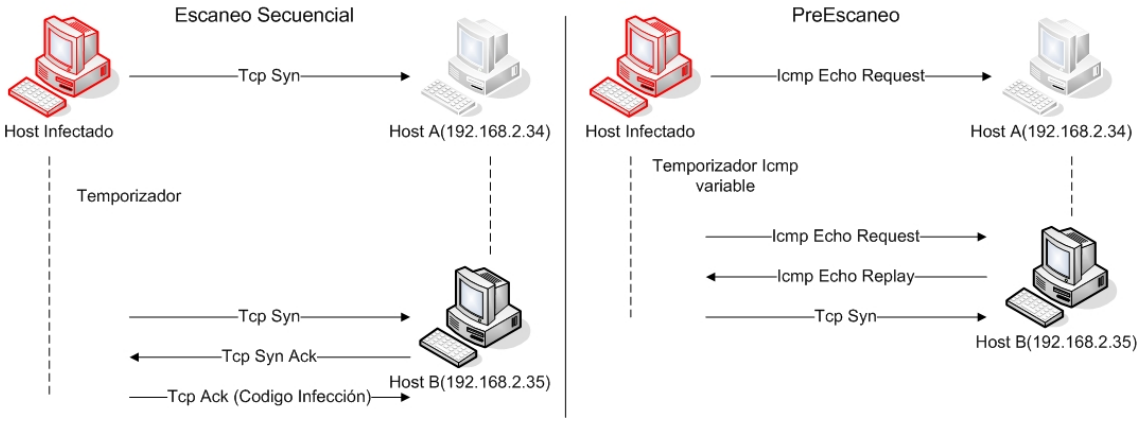
Escaneo Binario

Se trata de ir recorriendo el espacio de direcciones pero de una manera binaria, de esta forma se obtiene una mejor respecto al escaneo secuencial de direcciones y la propagación es mas rápida.

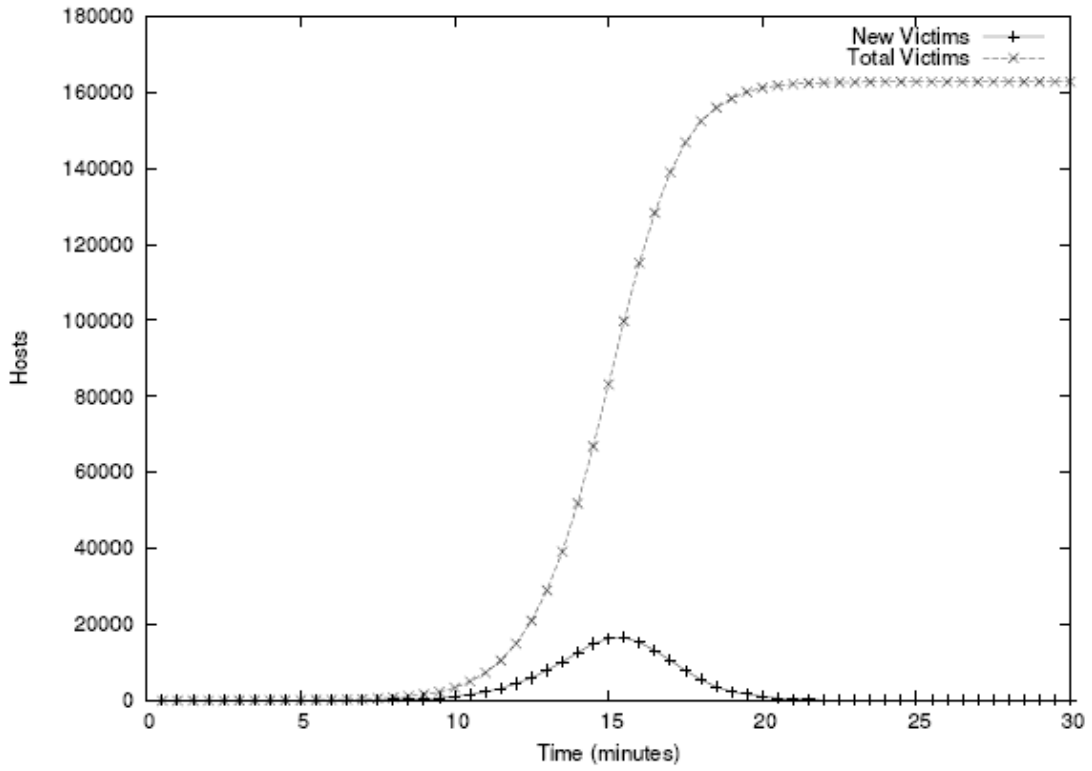


Preescaneo

Se trata básicamente de comprobar si al Host destino puede el virus realizar la infección, es una mejora de los anteriores métodos intentado eliminar el tiempo de respuesta de un Host que puede no esta encendido.

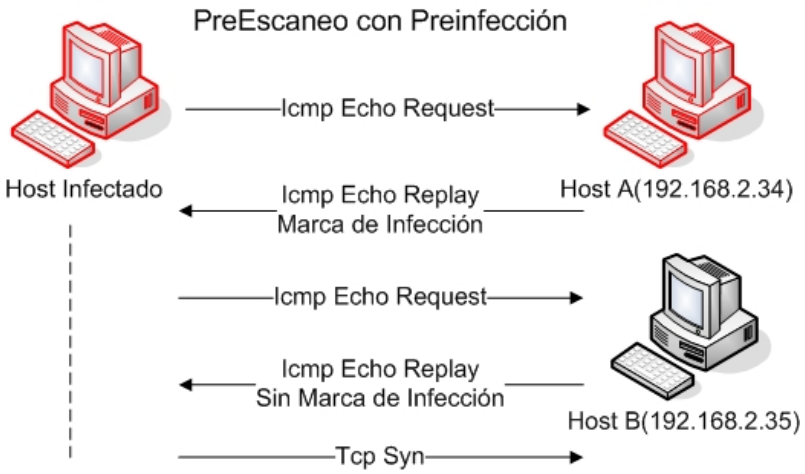


Con el preescaneo lo que se asegura el virus es que el Host destino esta accesible y así el tiempo de propagación se reduce considerablemente.

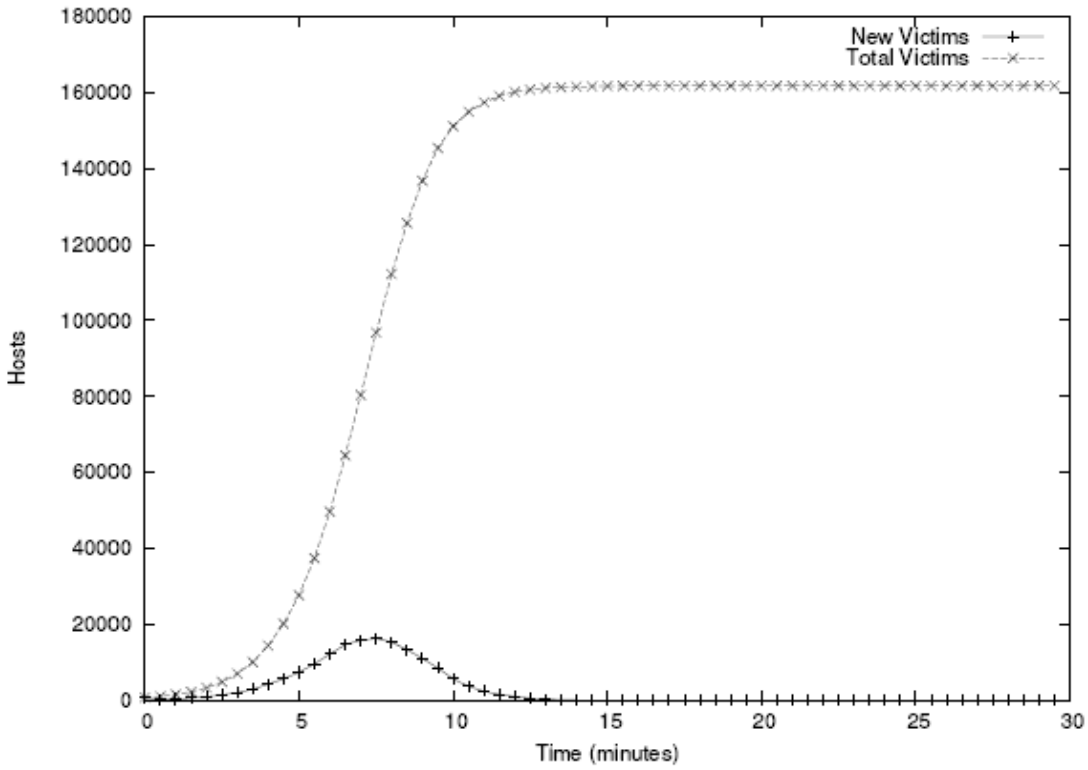


Preescaneo con Preinfección

Se trata básicamente de comprobar si al Host destino ya esta infectado y de esta manera eliminar el tiempo de infección de un Host ya infectado.



Normalmente este tipo de propagación se puede detectar con las marcas que necesita el virus para saber si dicho Host esta infectado o no, aunque tiene el inconveniente que hay que conocer dicha marca.



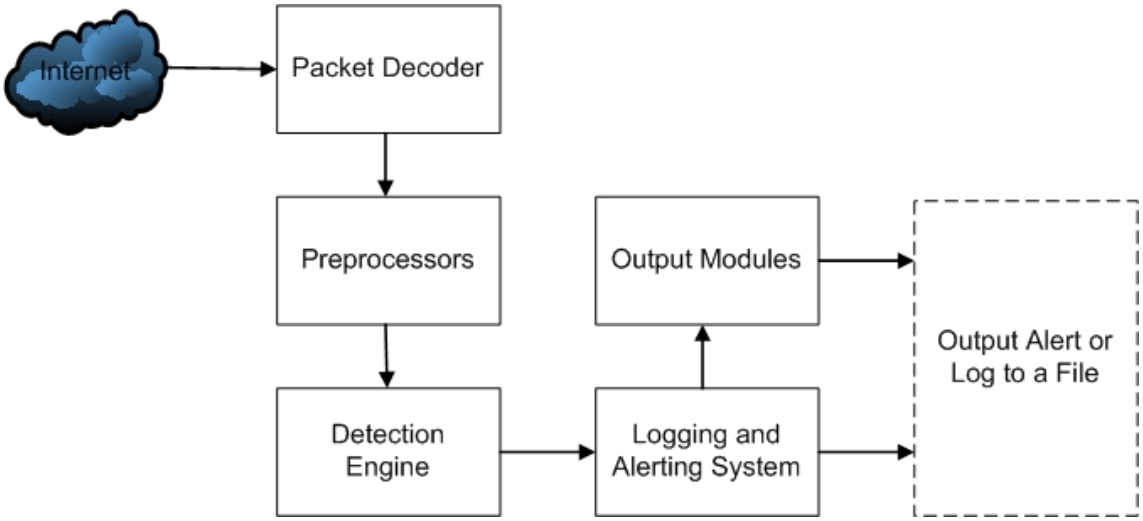
.

Arquitectura de Diseño

Diseño NIDS

Snort[13] es un IDS en tiempo real desarrollado por Martin Roesch y disponible bajo GPL. Se puede ejecutar en UNIX y Windows. Actualmente es el sistema de detección de intrusos más utilizado. La versión 1.9 dispone de unas 1.700 reglas y de multitud de aplicaciones para el análisis de sus alertas y las versiones 2.x disponen de unas 2500 reglas aproximadamente.

Un diagrama de los elementos mas significativos de Snort es el siguiente.



Snort se basa en la librería Libpcap[14] para la captura de paquetes.
La arquitectura de Snort está formada por cuatro capas:

- Decodificador de paquetes: se encarga de tomar los paquetes que recoge la Libpcap y almacenarlos en una estructura de datos en la que se apoyan el resto de capas.
- Preprocesadores: hacen un pretratado de los paquetes. Por ejemplo, el preprocesador ip_frag se encarga de reensamblar fragmentos Ip y el stream4 reconstruye el flujo Tcp a partir de los segmentos almacenados en memoria.
- Motor de detección: implementa el algoritmo Boyer-Moore para la búsqueda de firmas en las versiones 1.x. Este algoritmo es el más eficaz conocido para la búsqueda de un patrón en una cadena arbitrariamente larga. El problema es que en los IDSs no existe un único patrón sino varios. Se han diseñado ciertas mejoras añadiendo una estructura de datos del algoritmo Aho-Corasick [15] lo cual teóricamente mejora el rendimiento hasta en un 500% sobre la versión 2.x de Snort y también usa el algoritmo Wu-Manber.
- Etapas de salida: se utiliza cuando un ataque ha sido detectado. En este caso Snort dispone de plug-ins para el almacenamiento de la alerta en múltiples formatos: SQL (PostgreSQL, MySQL, Oracle, UnixOBDC), ASCII, XML, WinPopup, syslog, …

Análisis de Protocolos

Mucha gente que piensa en NIDS piensa en el reconocimiento de patrones y hay otra manera de implementar NIDS y es mediante el Análisis de Protocolos.

La arquitectura en el Análisis de Protocolos es diferente al reconocimiento de patrones, en el Análisis de Protocolos cada paquete es decodificado de acuerdo con la especificación sobre el estándar de dicho protocolo. Para los paquetes que conforman el estándar pero son un exploit, por ejemplo, el NIDS puede realizar reconocimiento de patrones en ciertos campos del paquete y de esta forma no realizar el reconocimiento de patrones en todo el paquete.

Por ejemplo se podría realizar primero un acceso directamente al campo URI y luego realizar un reconocimiento de patrón, así es como funciona Snort internamente, para buscar directamente “exploit.asp” o “SAMPLE.”

HTTP - Hyper Text Transfer Protocol
Command: GET
URI: /SAMPLE/exploit.asp
Version: HTTP/1.1..
Accept: */*..
Referer: http://www.victima.com/..
Accept-Language: en-us..
Accept-Encoding: gzip, deflate..
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)..
Host: www.victima.com..
Connection: Keep-Alive..
Cookie: RoxenUserID=0x673b30....

El rendimiento de esta metodología va ha depender de varios factores como puede ser el tamaño del paquete, el tamaño del campo a examinar, componentes a tener en cuenta, etc….

Una de las desventajas en el Análisis de Protocolos de los NIDS es que cada fabricante implementa su propio Análisis de Protocolos de acuerdo con su propia interpretación de la RFC, y esto causa muchas falsas alarmas sobre ataques que no se han producido o no produce las alarmas que debería al realizar una mala interpretación de la RFC en cuestión.

Otra desventaja es que si alguien se encuentra una manera de evadir el Análisis de Protocolos del NIDS todo el sistema de descodificación tiene que ser reescrito.

La tercera desventaja es que acción se realiza cuando no tenemos un decodificador de protocolo para un paquete con un protocolo desconocido para el NIDS. Esto se podría solucionar enviando una simple alerta al sistema indiciando dicho evento, pero también se podría realizar un análisis heurístico

para intentar determinar cual es el protocolo a analizar ya que actualmente hay cambios de puertos, redirecciones en estos, etc…
Dependiendo del nivel de complejidad de la red y de paranoia del administrador se pueden poner un NIDS o un sistema distribuido de NIDS en diferentes segmentos de red.

Hay dos cuestiones a tener en cuenta en el Análisis de Protocolos actualmente:

- Casi todas las empresas tiene internamente protocolos de enrutamiento para mantener sus tablas y sus VPNs, muchos de estos protocolos de enrutamiento se actualizan via Multicast, como pueden ser Ospf o Eigrp, o via Broadcast como pueden ser Rip o Igrp.
  En los últimos años las utilidades como son irpas [16], nemesis [17] o nmap [18] han sido rescritas con el fin de poder analizar y falsear dichos protocolos y con dichas herramientas cambiar rutas, redirigir trafico, falsear gateways, inyección de código, etc….

- No todos los profesionales que se dedican a la seguridad tienen en cuenta capa de transporte (Ethernet por ejemplo) y piensan que están libres de falseo de direcciones mac por que están en un entorno con switches, pero con dsniff [19] y ARP0c [20]. Con estas dos herramientas se puede engañar a los switches, pero hay que tener en cuenta que solo funciona en entornos LAN pero también hay que tener en cuenta dichos ataques y activar reglas en os NIDS para intentar detectarlos.

Reconocimiento de Patrones

El problema del reconocimiento de patrones es el detectar dentro de una cadena otra, hay muchos algoritmos de reconocimiento de patrones [21][22], pero nosotros nos vamos a centrar en tres de ellos, los cuales son los usados en la version mas reciente de Snort y que configuran el motor multipatrón de búsqueda, uno usado en Snort en las versiones 1.x y los otros en la versión 2.x que son el algoritmo de Boyer-Moore, el Aho-Corasick y el Wu-Manber.

Boyer-Moore

El algoritmo Boyer-Moore[22] consiste en alinear el patrón en una ventana de texto y comparar de derecha a izquierda los caracteres de la ventana con los correspondientes al patrón. Si ocurre una desigualdad se calcula un desplazamiento seguro el cual permitirá desplazar la ventana hacia delante del texto sin riesgo de omitir alguna coincidencia. Si se alcanza el inicio de la ventana y no ocurre ninguna desigualdad, entonces se reporta una coincidencia y la ventana se desplaza. Este algoritmo se considera el más eficiente algoritmo de búsqueda de cadenas de caracteres en aplicaciones comunes. Tanto es así que los comandos “buscar” y “reemplazar” de los editores de texto suelen utilizarlo aunque sea en una versión simplificada. Aunque, en peor caso, su complejidad pertenece a $O(n*m)$, por lo general su mejor caso no supera el $O(n/m)$.

Aho-Corasick

Algoritmo Aho-Corasick [22] diseñado por Margaret J. Corasick y Alfred V. Aho
Este algoritmo, usado en Snort 2.0, utiliza la idea de una máquina de estados (donde cada estado se representa con un número), que llama máquina de búsqueda de patrones (PMM). Una PMM de K es un programa que toma como entrada al texto X y genera, como salida, las posiciones de X donde comienzan patrones contenidos en K. Procesa el texto X leyendo sucesivamente sus símbolos, generando estados de transición y salidas ocasionales. La complejidad de este algoritmo es O(n), lineal

Wu-Manber

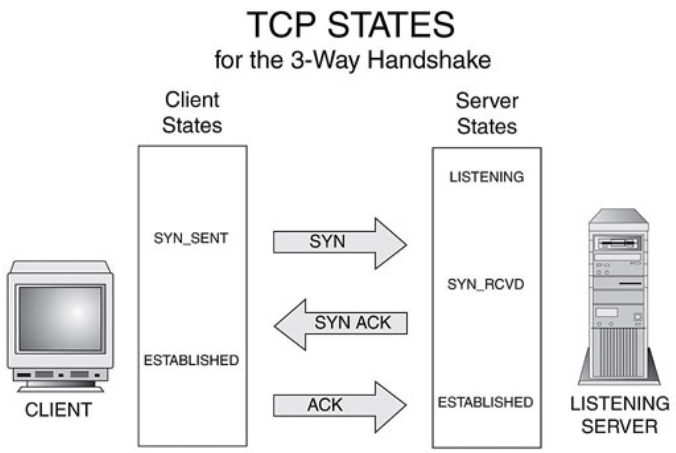
El Wu Manber[23][24] es el algoritmo que usa Snort 2.x por defecto en la búsqueda, es una variación sobre el algoritmo original, este utiliza una tabla hash con subgrupos para la búsqueda, gestión mejor la memoria que el Aho-Corasick aunque es un poco menos eficiente.
Es un algoritmo muy flexible y ha sido utilizado en el mundo Unix a lo largo de la historia. La complejidad de este algoritmo es de O(nm), con n el numero de caracteres de la entrada y m el numero de caracteres del patrón.

Statefull Inspection

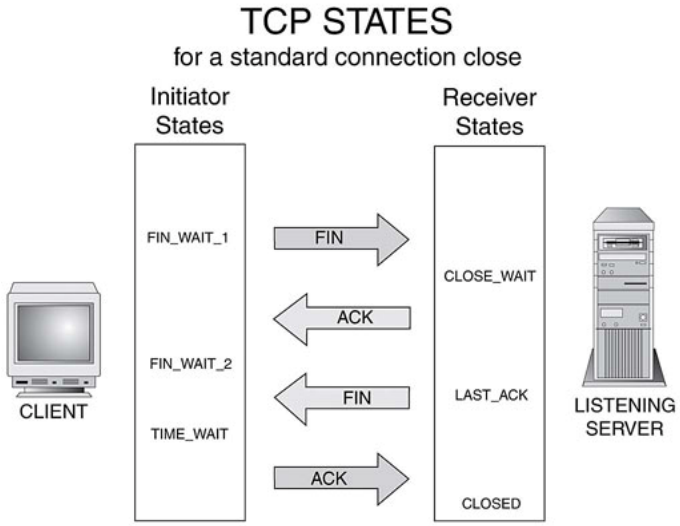
El termino Statefull inspection ha estado muy asociado a los Firewalls y a las capa 4(Tcp, udp). Los Firewalls han aumentado su complejidad a la vez que los nuevos protocolos han aparecido. Por ejemplo el protocolo FTP utiliza dos conexiones separadas, una para información de control en el puerto 21 y otra para la transferencia de datos en el puerto 20. Cuando un Firewall ve una conexión en el puerto 21 a un servidor Ftp este sabe que tiene que activar una regla dinámica para poder examinar el puerto 20, y esta regla dinámica va ha depender de la información de control que este pasando por el puerto 21.

Antiguamente los NIDS no se preocupaban del estado de la conexión hasta que se vieron la cantidad de problemas que ocurrían, como podemos ver anteriormente en temas de Fragmentación, y es por eso que los NIDS actuales realizan reensamblado de paquetes.

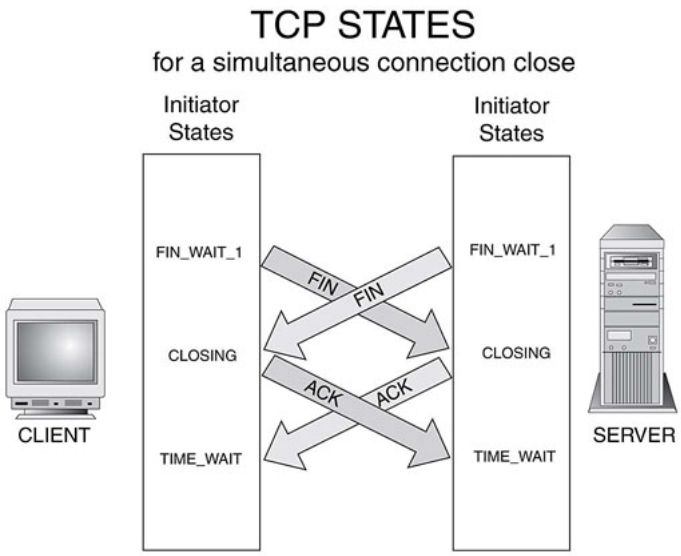
A continuación vamos a ver el tipico modelo de estados Tcp, para explicar que es un Firewall Statefull y llevar esta misma idea al nivel de los IDS.



Cuando al host destino le llega un paquete Syn, que el Firewall ha interceptado compraba si hay alguna regla predefinida para tratar dicha conexión, y aplica dicho criterio, como el Firewall no va ha estar mirando el otro Syn-Ack del host ni el otro Ack por que seria ejecutar continuamente búsquedas por el árbol de reglas para llegar a la misma conclusión, el Firewall lo que hace es en su tabla de estados dicha conexión para que cuando llegue el resto de paquetes con sus correspondientes flags mire si son de una conexión activa en su tabla de estados activos Tcp.



Las entradas para las sesiones de la comunicación del TCP en una tabla del estado se quitan cuando la conexión se cierra. Para prevenir las conexiones que están cerradas incorrectamente se utilizan los contadores de tiempo como esta definido para el protocolo Tcp. Mientras que en un inicio de conexión el valor del time out inicial usado es típicamente corto (bajo minuto), así que estas tablas de estado también llevan temporizadores de tiempo, aun así su rendimiento mejora y la administración es infinitamente mas cómoda que gestionar todas las reglas de los diferentes estados por los que pasa el protocolo Tcp..



Motor de Detección de Snort

El antiguo motor de detección de Snort, el del 1.x, era simple de implementar y añadir a este nueva funcionalidad era relativamente fácil, sin embargo este motor no era muy eficiente y era muy dependiente del numero de reglas que estuvieran activas, a mayor numero de reglas peor era la eficiencia de Snort.

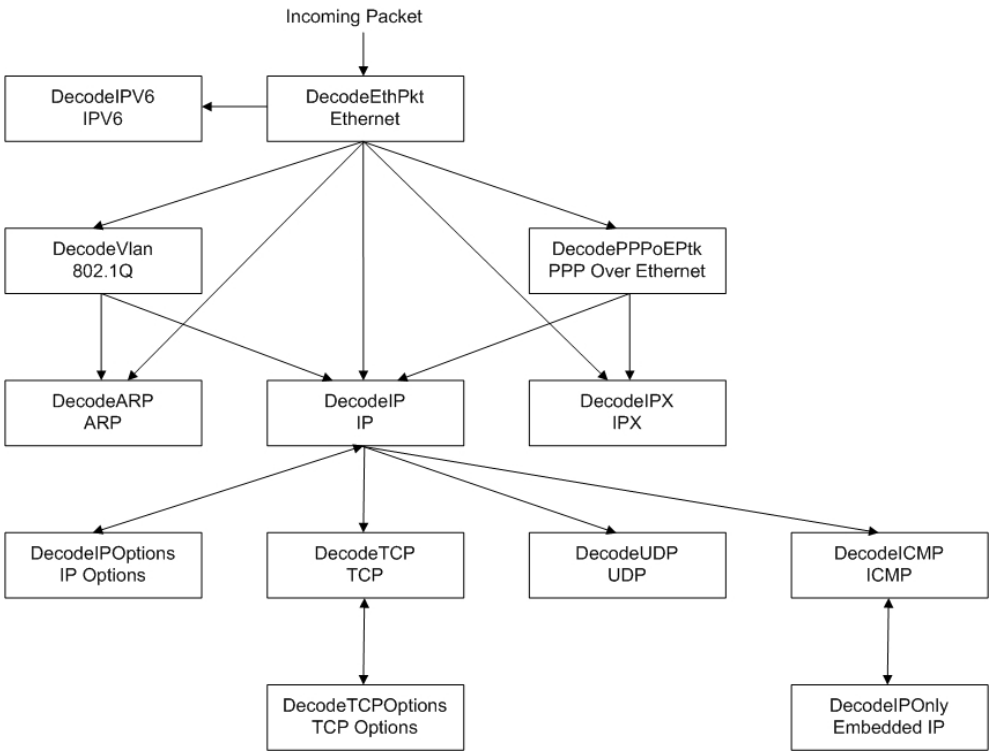
En los últimos años el numero de reglas se ha disparado en Snort, actualmente hay unas 2500 reglas activas, con el antiguo motor de detección la gestión de estas reglas era lo que hacia que Snort fuese lento y nada eficiente, y mas para un sistema de este tipo.

El nuevo motor de detección de Snort parte con un requisito inicial que es que Snort sea capaz de funcionar en redes Gigabit, y para que esto ocurra se reescribe totalmente el motor de Snort para que este sea capaz de gestionar trafico a giga.

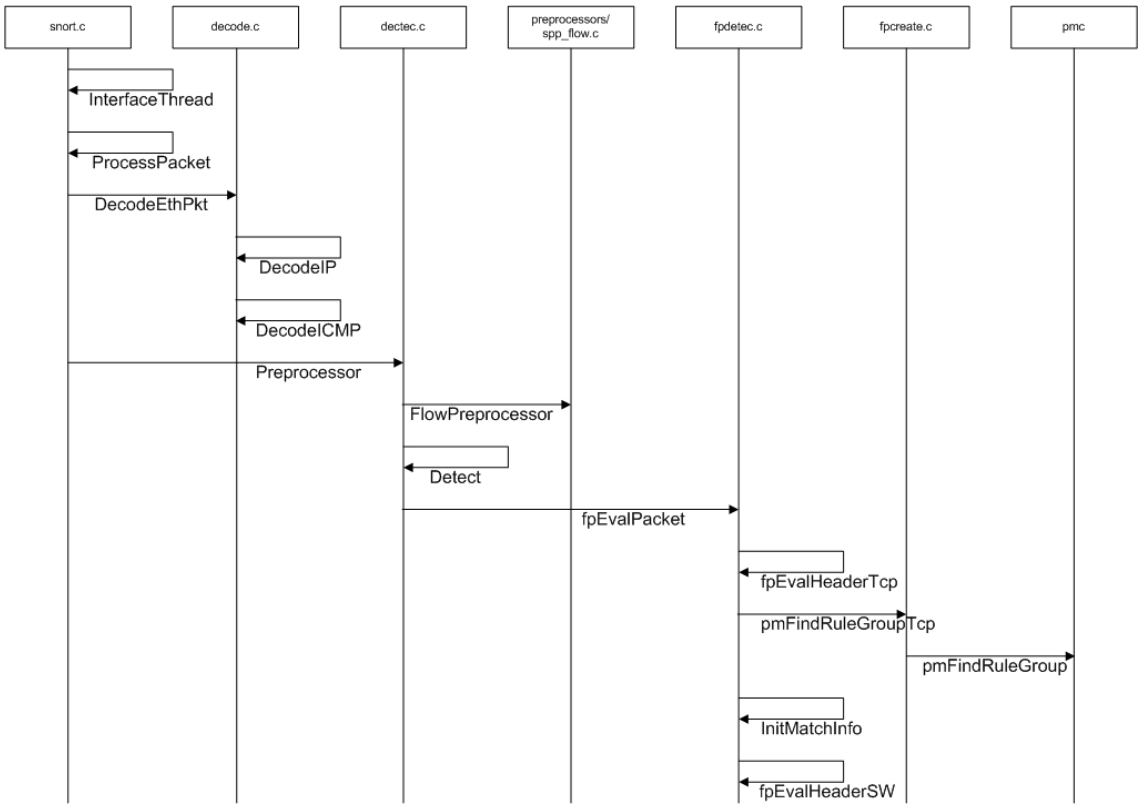
Los desarrolladores de Snort realizaron un nuevo motor de detección usando algoritmos multipatrón de búsqueda que es el núcleo del motor y que implementa múltiples reglas y que permite a Snort funcionar sobre redes giga.

El nuevo motor de detección construye cuatro grupos de reglas, una para el protocolo Tcp, otro para el Udp, otro para el Icmp y para Ip.

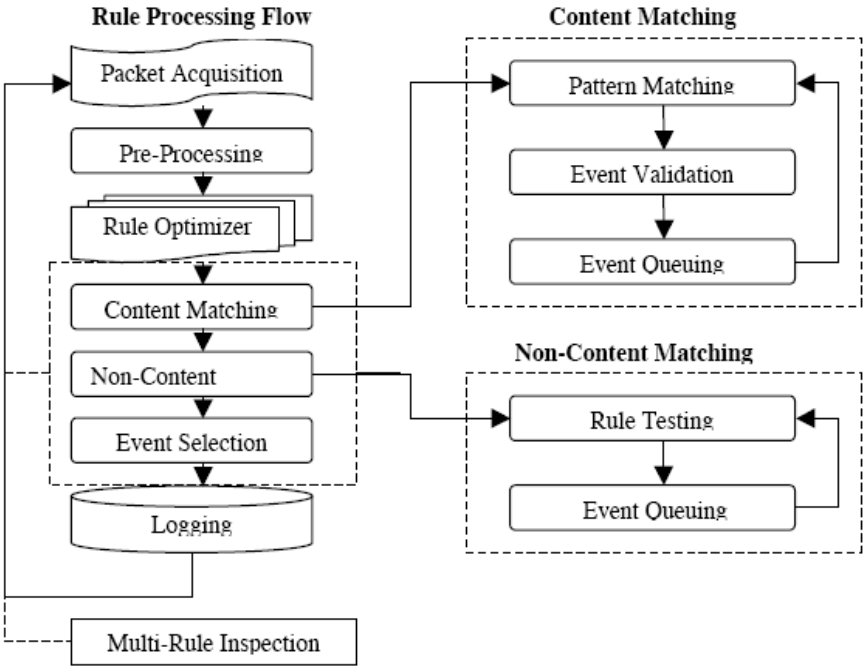
Cuando un paquete se captura mediante la librería Libpcap lo primer que se realiza es una descodificación de este para alinear cabeceras según el protocolo, como vemos en el siguiente diagrama.



Un posible diagrama de secuencia de la vida de un paquete en Snort seria el siguiente.



En dicho diagrama podemos ver primer como se realiza la decodificación de los paquetes, todo dependiente del protocolo analizado, y luego las llamadas a los preprocesadotes, por cada uno de los que estén instalados en Snort o que hallamos realizado nosotros con nuestros propios protocolos. A continuación un diagrama de flujo con todos los elementos mas significativos.



Reglas de Detección

Hay cuatro categorías de reglas par alas que un paquete es evaluado. Estas cuatro categorías esta divididas a su vez en dos grupos, las que tienen contenido y las que no tienen contenido. Hay reglas de protocolo, reglas de contenido genéricas, reglas de paquetes malformados y reglas Ip.

Reglas de Protocolo

Las reglas de protocolo son reglas las cuales son dependientes del protocolo que se esta analizando, por ejemplo en el protocolo Http esta la palabra reservada uricontent.

Reglas de Contenido Genéricas

Este tipo de reglas permite especificar patrones para busca en el campo de datos del paquete, los patrones de búsqueda pueden ser binarios o en modo ASCII, esto es muy útil para buscar exploits los cuales suelen terminar en cadenas de tipo “/bin/sh”.

Reglas de Paquetes Malformados

Este tipo de reglas especifica características sobre los paquetes, concretamente sobre sus cabeceras las cuales indican que se esta produciendo algún tipo de anomalía, este tipo de reglas no mira en el contenido ya que primero se comprueba las cabeceras en busca de incoherencias o otro tipo de anomalía.

Reglas Ip

Este tipo de reglas se aplican directamente sobre la capa Ip, y son comprobadas para cada datagrama Ip, si el datagrama luego es Tcp, Udp o Icmp se realizara un análisis del datagrama con su correspondiente capa de protocolo, este tipo de reglas analiza con contenido y sin el.

Preprocesadores

Los Preprocesadores son componentes de Snort, introducidos desde la versión 1.5, no dependen de reglas ya que el conocimiento sobre la intrusión depende del modulo Preprocesador. Se llaman siempre que llegue un paquete y se les puede aplicar reglas que estén cargadas en Snort.

La arquitectura de preprocesadotes de Snort consiste en pequeños programas C que toman decisiones sobre que hacer con el paquete. Estos pequeños programas C se compilan junto a Snort en forma de librería. Estos preprocesadotes son llamados justo después que Snort realice la Decodificación, ver diagrama de secuencia, y posteriormente se llama al Motor de Detección. Si el numero de preprocesadores es muy alto el rendimiento de Snort puede caer considerablemente y es por este punto donde mas adelante realizaremos la Distribución de componentes.

Aquí tenemos una lista de preprocesadores de los que incluye Snort:

portscan
: Preprocesador encargado de la detección de escaneos.

portscan-ingonoreHost
: Una modificación del anterior preprocesador con el cual se pueden ignorar escaneos básicos como el TCP Syn y el UDP.

sfportscan
: Detecta escaneos a nivel de herramienta, como puede ser un escaneo de puertos realizado con Nmap

frag3
: Reensambla paquetes fragmentados

stream4 y stream4_reassemble
: Realiza el Reensamblado de segmentos TCP

telnet_decode
: Examina los típicos comandos telnet.

rpc_decode
: Para el servicio de rpc, por defecto mira en los puertos 111 y 32771.

http_instpect y http_inspect_server
: preprocesadotes para el análisis de trafico http

asn1
: preprocesador encargado de codificaciones ASN1 erróneas.

clamav
: El Preprocesador clamav[25] que permite a Snort conectar con el antirivus ClamAv[26] para detectar virus.

bo
: Permite detectar la existencia del conocido Troyano Back Orifice.

arpspoopf
: Detecta falseo a nivel de dirección mac.

Estos preprocesadores se activan en Snort en el fichero de configuración snort.conf, simplemente con comentar una línea en dicho fichero el preprocesador se cargara o no.

```
#preprocessor arpspoof
#preprocessor arpspoof_detect_Host: 192.168.40.1 f0:0f:00:f0:0f:00
```

De esta manera el preprocesador de detección de ARP-Spoofing no estaría activado, y la carga de Snort seria menor.

A Continuación vamos a implementar un procesador básico el cual mas adelante realizaremos la Distribución de los procesadores en nuestro sistema.
El preprocesador se va ha incluir en el código fuente de Snort y posteriormente se realiza la compilación como en cualquier sistema Linux, los dos ficheros que vamos a comentar a continuación estarán en el directorio ./src/preprocessors de los fuentes de Snort.

Primero el fichero cabecera spp_ejemplo.h, el prefijo spp_ se pone por seguir un poco el estándar de Snort respecto al código fuente.

```
/* spp_ejemplo.h
/* fichero cabecera de ejemplo, en el incluimos cabeceras y
estructuras necesarias de nuestro preprocesador
*/
#ifndef __SPP_EJEMPLO_H__
#define __SPP_EJEMPLO_H__

typedef struct _ejemploStruct {
      int a,
      char buffer[100];
} ejemploStruct;

extern ejemploStruct miEjemploStruct;

/* cabecera de la funcion de inicializacion del Preprocesador */
void setupEjemplo(void);

/* cabecera de inicializacion de la funcion de tratamiento */
void ejemploFuncion(char *ptr);

#endif
```

Ahora codificamos el fichero spp_ejemplo.c

```
/* spp_ejemplo.c
/* fichero con la implementación del preprocesador
*/
/* fichero generado por el autoheader, definiciones, paths….. */
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

/* funciones AddPreprocessor y RegisterPreprocessor */
#include "plugbase.h"

/* definiciones de cabeceras de paquetes */
```

```
#include "decode.h"

/* nuestra cabecera */
#include "spp_ejemplo.h"

ejemploStruct miEjemploStruct;

/* Prototipos de funciones internas */
Void ejemploInit( char *argumentos);
Void ejemploFuncion();
Void ejemploExit();
Void ejemploRestart();

/* función que se llama desde plugbase.c */
void setupEjemplo(void)
{
      /* Aquí inicializamos los elementos propios de nuestro
      preprocesador, recursos de memoria, ficheros, buffers
      secundarios, etc…. */
      RegisterPreprocessor(“ejemplo”,ejemploInit);
}

void ejemploInit( char *argumentos)
{
      /* esta función se ejecuta una sola vez y en ella recibimos los
      parámetros de configuración del Preprocesador que incluiremos en
      el fichero snort.conf */

      /* añadimos la función que se tiene que ejecutar en cada paquete
      que llegue al sistema */
      AddFuncToPreproc(ejemploFuncion);

      /* ponemos el manejador para cuando se realice un exit de la
      aplicación */
      AddFuncToCleanExitList(ejemploExit,NULL);

/* manejador para los restart */
      AddFuncToRestarList(ejemploRestart, NULL);
}

void ejemploFuncion(Packet *pkt)
{
      /* esta es la función que se llamara con cada paquete */
      If (pkt->Tcph ¡= NULL)
            If (pkt->Tcph->syn == 1)
                  printf(“Nueva Conexión TCP\n”);
}

void ejemploRestart()
{
      /* código para restaurar el Preprocesador */
}

void ejemploExit()
{
      /* código para liberar los recursos del preprocesador, memoria,
      punteros, etc.. */
}
```

Una vez realizado el preprocesador hay que añadir las siguientes líneas de código en el fichero /src/plugbase.c y añadimos lo siguiente.

En la sección de includes

```
#include “preprocessors/spp_ejemplo.h”
```

Y en la función InitPreprocessors tenemos que hacer referencia a la función de inicialización que es setupEjemplo();

Para la compilación nos basta con añadir en el fichero Makefile en la sección libspp_a_SOURCES la inclusión de nuestro preprocesador, tanto el fichero .h como el .c

Respecto al fichero de configuración de Snort, snort.conf, tenemos que indicar que hay un nuevo preprocesador y sus parámetros, añadiremos a este fichero la siguiente entrada, en la sección de preprocesadores.

```
preprocessor ejemplo: argumento1
```

Diseño Distribuido con Snort

Hablar actualmente de un sistema de este tipo y que no realice interacción con otros sistemas de seguridad es impensable, tener un par de sensores con Snort en una red de forma distribuida que envíen los Logs a un servidor remoto y los ponga de una manera legible para que luego el Responsable pueda ver dicha información es muy pobre en la actualidad.

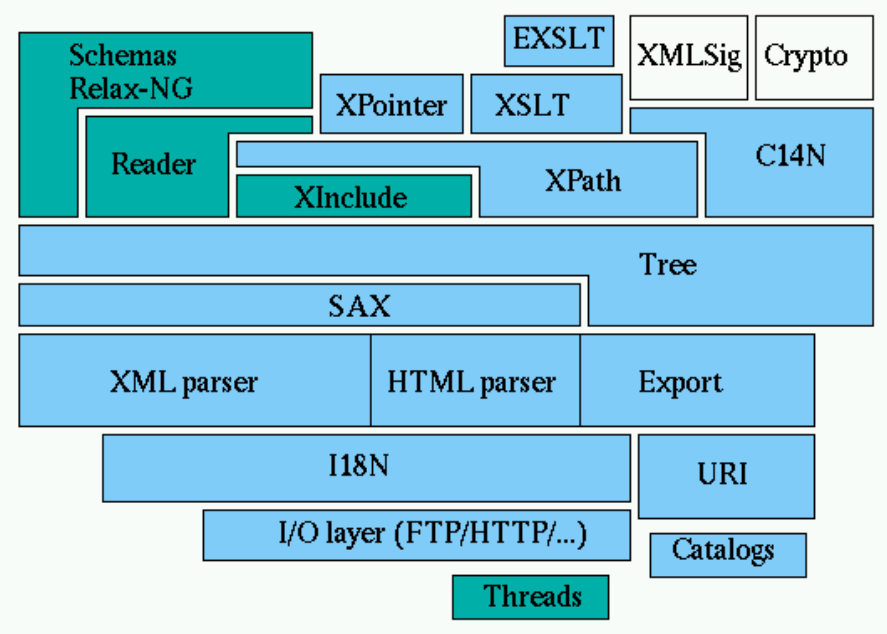
A continuación vamos a ir describiendo una arquitectura para la Detección de Intrusiones distribuida pero interactuando con los métodos actuales para combatir por ejemplo la denegación de servicio, expansión de virus, detección de intrusiones, minimizar impactos en servidores, etc….

En la actualidad hay un proyecto OSSEC HIDS [27] que es un sistema Hibrido con Snort y mas elementos con los que interactúa Snort, dicho sistema realiza análisis de Logs, integridad de archivos, detección de rootkits, sistema de respuestas activas y algunas funcionalidades mas, dicho sistema nos vale un poco de ejemplo sobre lo que vamos a diseñar, salvo que nosotros realizamos otro diseño totalmente distinto.

Para el diseño de nuestro sistema vamos a usar las siguientes Tecnologías, aparte de Snort.

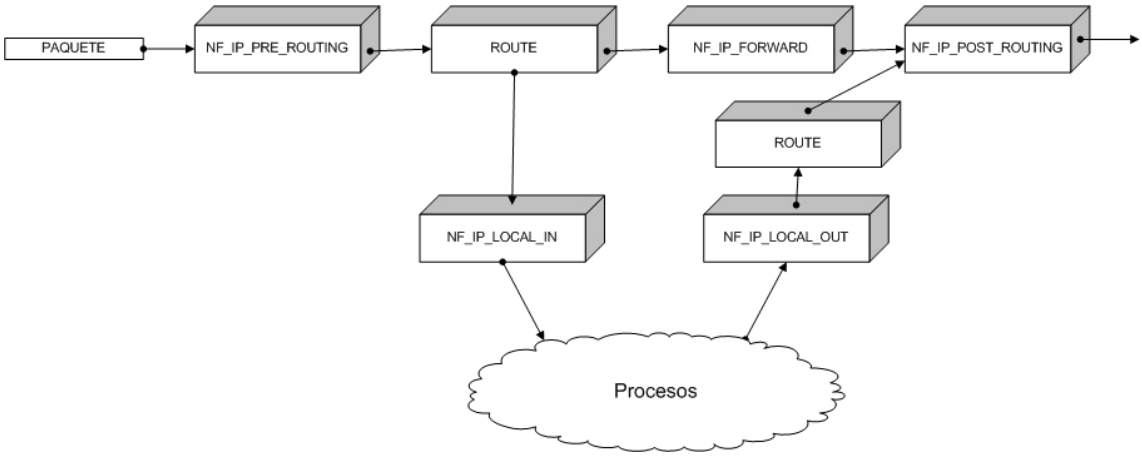
LibXML [28] : Librería para el manejo de ficheros XML. La usaremos principalmente para el manejo de los ficheros de configuración del sistema y por si mas adelante el sistema necesita interactuar con otro tipo de sistema y por realizar un estándar de nuestros datos.

Un diagrama de la arquitectura de LibXML es el siguiente.



Libnetfilter [29]: Librería para la captura de paquetes Ip desde el Kernel de Linux mediante el api de Netfilter, sustituye a la anterior API Libipq, dicha librería nos permite la captura de los paquetes antes de que lleguen al proceso destino y mediante snort realizar un análisis de este para tomar decisiones al respecto, de esta manera podemos enviar a procesos de usuario datagramas Ip o segmentos Tcp para su posterior análisis. Hay que comentar que actualmente Snort tiene un modulo añadido, Snort InLine [30], el cual es un Hibrido entre Firewall y IDS y que permite añadir reglas al Firewall de una forma automática.

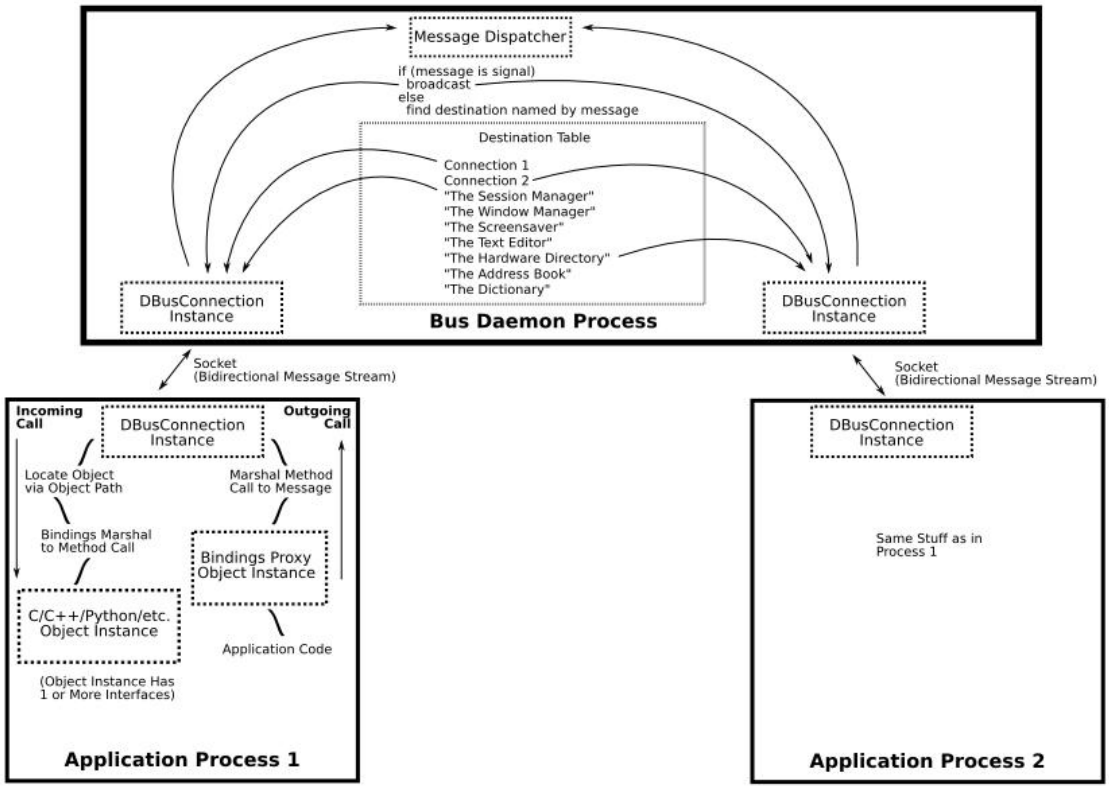
La arquitectura de Netfilter nos permite pasar a nuestro proceso Snort los paquetes de red y posteriormente tomar decisiones sobre estos.



También podríamos usar el proyecto Snort InLine para nuestro sistema ya que la implementación de captura de paquetes desde el Kernel de Linux esta implementada ya.

DBus [31] : D-BUS (Distributed Bus) es un bus del sistema de mensajes, un método sencillo para que las aplicaciones hablen entre sí.
D-BUS proporciona tanto un demonio (servicio) de sistema (para eventos como "nuevo dispositivo hardware añadido" o "la cola de impresión ha cambiado") y un demonio de sesión por usuario (para comunicaciones Ipc entre las aplicaciones del usuario). Además, el bus de mensajes se está construido sobre un entorno general de envío de mensajes uno-a-uno, que se puede utilizar para comunicar dos aplicaciones directamente (sin tener que utilizar el demonio del bus de mensajes).

En realidad, podemos tener todos los demonios que queramos iniciados, que se corresponderían con diferentes canales de DBUS, pero el escenario habitual será el tener un canal de sistema donde se informará a las aplicaciones interesadas de los eventos del sistema, y un canal de sesión de usuario que usarán las aplicaciones dentro de una sesión de usuario para comunicarse.

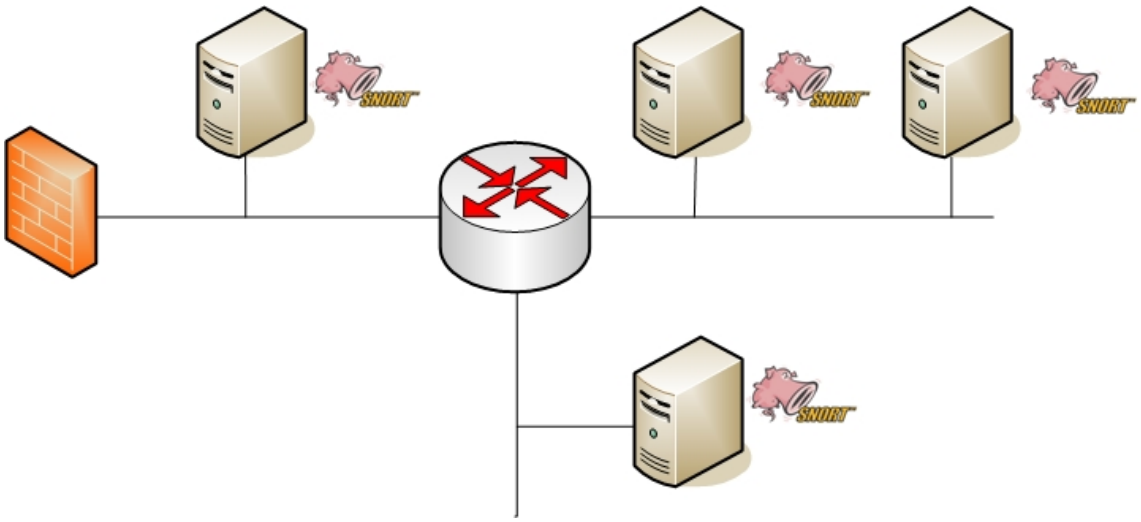


Este Bus es distribuido y podemos conectarlo entre diferentes maquinas o en la misma maquina.

Arquitectura

A continuación vamos a comentar los dos tipos de arquitecturas de distribución del sistema y vamos a ver sus pros y sus contras, obviamente nosotros nos vamos a centrar en una de ellas, pero si es conveniente comentar la otra distribución de componentes.

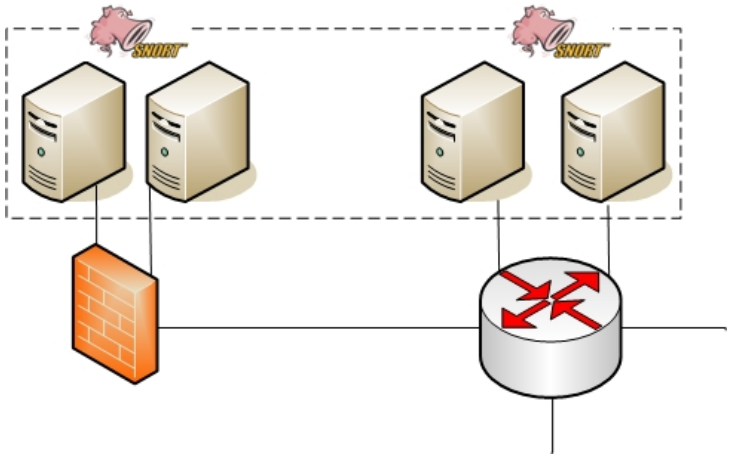
En la primera aproximación tenemos la típica distribución de sensores de Snort, los cuales están colocados en diferentes segmentos de red.



Dicha distribución tiene los siguientes inconvenientes:

- No hay interactividad con ninguno de los elementos de red, Routers, Firewalls, balanceadores de carga, etc…
- El sistema usa para el acceso a los sensores de la red la misma red, con lo que un acceso indebido al Firewall o al Router puede dar información al atacante respecto a la utilización de dichos sensores simplemente con mirar el trafico de la red.
- Como no hay separación de redes, la red corporativa y la red de seguridad es la misma, los Logs y la información que general los sensores va por la misma red y esta puede estar comprometida, una solución seria cifrar el trafico, pero si lo ciframos y un atacante accede a un elemento de red podría intuir que hay elementos extraños con comunicaciones cifradas desde un equipo, el sensor, a otro equipo al cual no puede acceder.
- En caso de una denegación de Servicio el Sistema como mucho puede informar, pero no mitigarlo, igualmente pasa con un ataque de virus.
- El manejo de las reglas de los sensores, aunque hay herramientas para distribuirlas, dependería mucho de la habilidad del administrador ya que según en que segmento de red están convendría activar unas y desactivar otras, lo que supondría un cierto caos en la gestión de estas reglas.

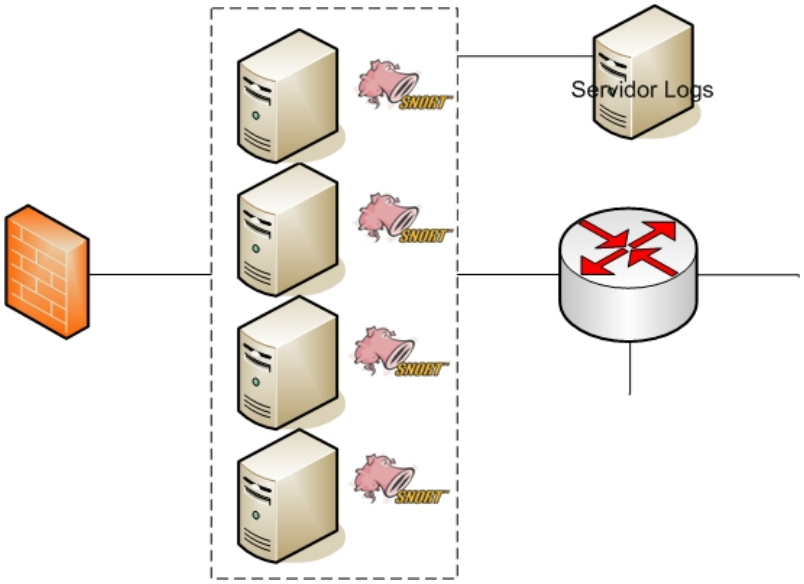
Por los inconvenientes anteriores comentados vamos a intentar mejorar el diseño.



Con el anterior diagrama, a modo de cluster y tomando el control el sistema sobre lo que es el Firewall y el Router, solucionamos problemas anteriormente comentados pero seguimos teniendo el problema que en caso de acceso al Firewall por ejemplo el sistema estaría comprometido, es por esta razón por la cual el sistema estará justo entre el Firewall y el Router y de una forma transporten, sin actualizar los TTLs de los datagramas, en modo invisible.

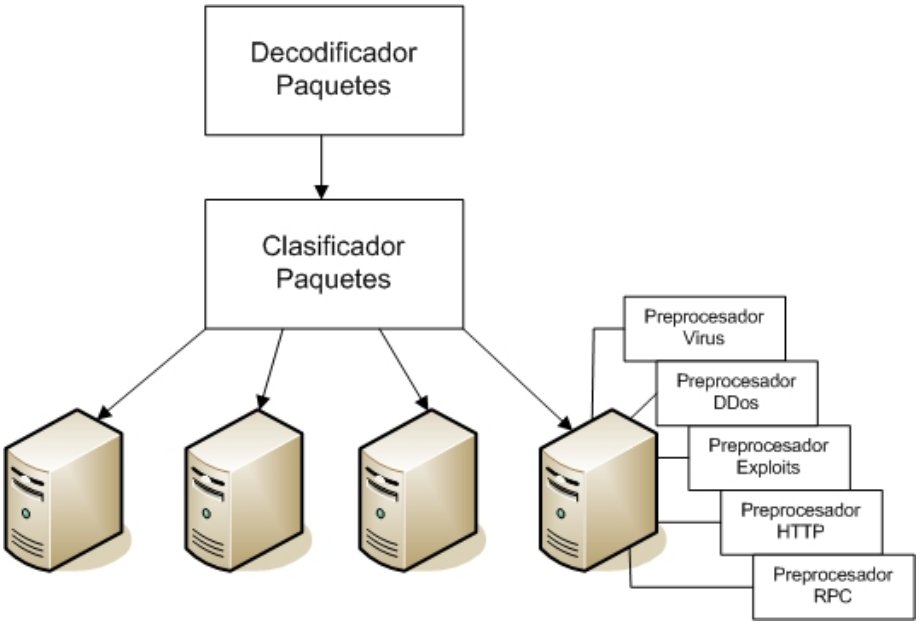
Con el sistema a modo de Proxy transparente conseguimos que ningún atacante pueda detectar la existencia de este, solo mediante cálculos de latencia y con un estudio minucioso por parte del atacante se podría darse cuenta de la existencia del sistema.

Respecto a la información de Logs nunca usaremos la red corporativa para enviar información por ella, ya que estaríamos delatando a la invisibilidad del sistema, y actualmente un interfaz de red y un servidor de Logs tiene un coste despreciable.



Actualmente las redes poseen una latencia la cual nosotros vamos a aprovechar para realizar el análisis de los flujos de información, nuestro sistema estará puesto entre dos redes a modo de paso entre las dos pero de una forma transparenté, nuestros Sistemas no realizaran modificaciones sobre los TTLs de los datagramas Ip, actuaran de forma transparente.

La primera Distribución de componentes a realizar es la distribución de los Preprocesadores en diferentes maquinas, las razones son que cada preprocesador es dependiente del protocolo que tiene que analizar y dependen mucho de la anomalía que este analizando, también otra razón para realizar la distribución es el alto numero de direcciones Ip que hay que analizar, no es lo mismo una red Lan de 20 Ordenadores que la red de un ISP que puede llegar a los 2000 o mas ordenadores conectados.

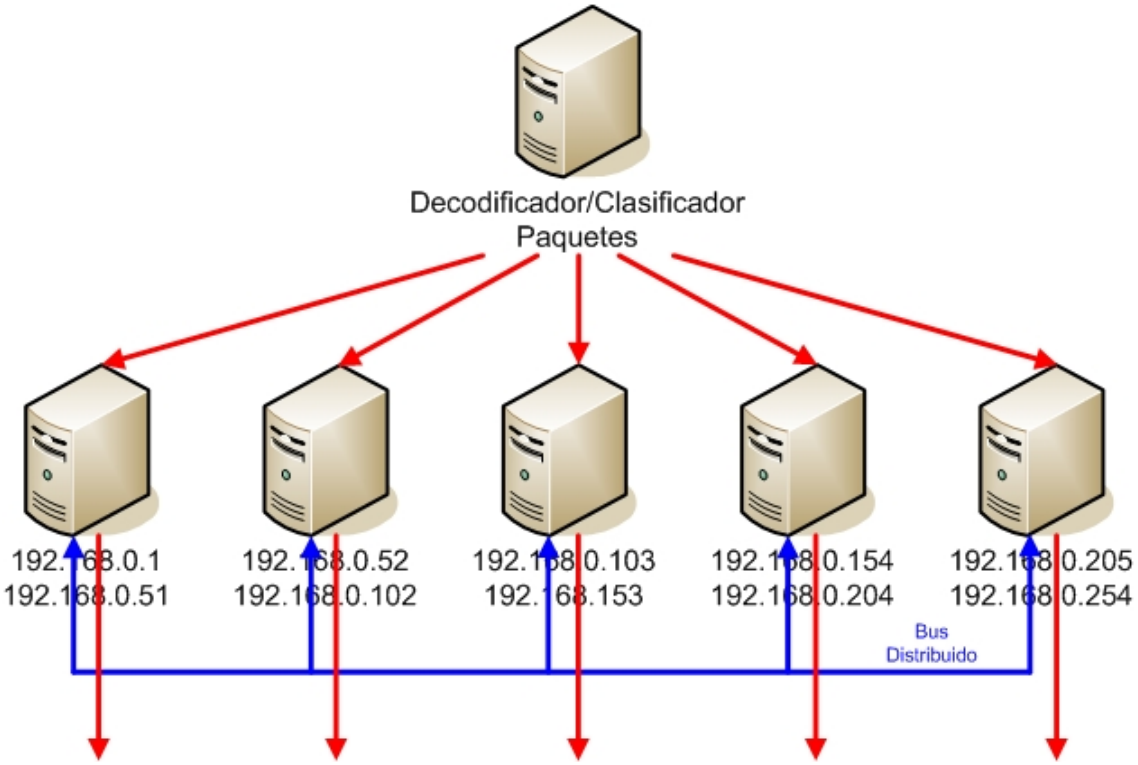


Es por esta razón por la que se necesita un Clasificador de paquetes, que va ha ir enviando a cada Host secundario del cluster el paquete correspondiente, en este tipo de clasificadores. Actualmente cualquier sistema operativo basado en Opensource como pueden ser OpenBSD o Linux dispone de sistemas de gestión de colas para cuando están activadas las opciones de enrutamiento, filtrado, etc….

Una vez dimensionado el sistema, en cuestión de flujos de información a analizar, podríamos tener el siguiente esquema a modo de ejemplo, con las diferentes tecnologías antes comentadas, el D-Bus para interconectar los diferentes elementos del cluster, cuando uno de ellos entra en saturación puede notificarlo al resto por ejemplo.

En un ataque distribuido de denegación de servicio o bien la detección de un gran numero de conexiones la lleva gestionada el decodificador/clasificador o bien es cada uno de los elementos del cluster el encargado de la detección, en caso que se cada uno de los elementos del cluster este podría entrar en saturación rápidamente o bien comunicarlo a los demás elementos del cluster y

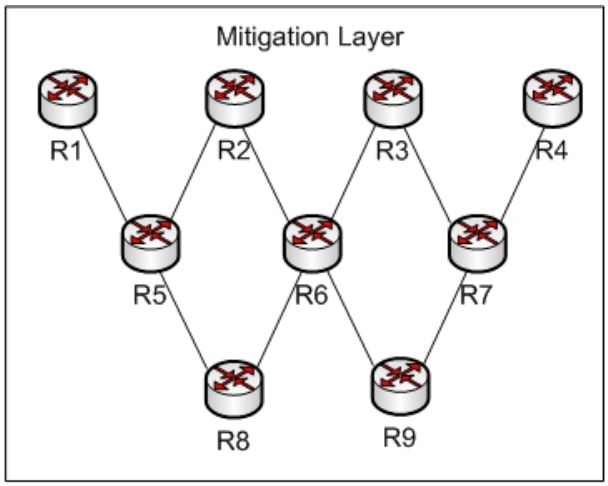
al decodificador/clasificador y que este lograse realizar el balanceo entre los demás elementos del cluster, para de esta forma mitigar el impacto. En caso que el ataque distribuido fuese de unas dimensiones considerables de podrían descargar conexiones entrantes al sistema afectado pero manteniendo las que están activas por si realmente no es un ataque de denegación de servicio.

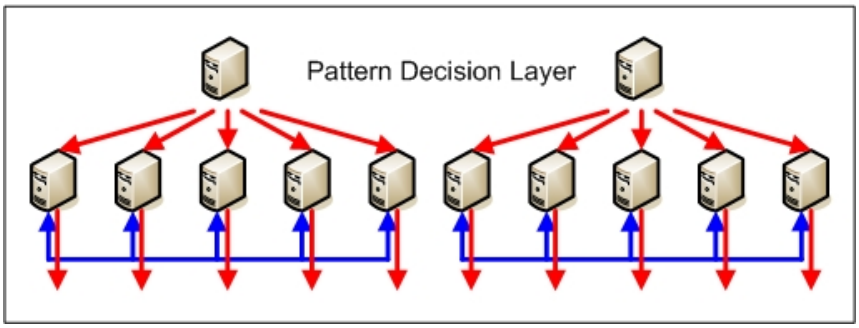


En este esquema nuestro sistema ya toma decisiones, pero seguimos con una problemática básica, el acceso al sistema, este sistema tiene un punto de acceso el cual es el Decodificador/Clasificador de Paquetes que realmente el código que tiene que ejecutar es poco y rápido, pero ante un gran flujo de información este sistema es vulnerable.

A Continuación vamos a unir dos ideas comentadas para realizar nuestro Framework de actuación ante los problemas comentados anteriormente, denegación de servicio, expansión de virus, vulnerabilidades, etc…

En nuestro Framework tenemos dos partes muy bien diferenciadas que son una parte dedicada a mitigar la Denegación de Servicio por medio de la técnica de PushBack, y que llamaremos Mitigation Layer(ML), y otra capa la cual esta dedicada al análisis de vulnerabilidades que estaría formada por Snort y un sistema de distribución de Preprocesadores anteriormente comentada que llamaremos Pattern Decision Layer(PDL).





En esta primera capa definida en el dibujo de la parte superior, tenemos definido lo que es el framework de detección con Snort, maquinas funcionando en paralelo con los preprocesadores separados en función de la carga y dicha capa conectada entre ellos por medio de un bus distribuido como puede ser dbus., y como capa superior tendríamos un sistema para mitiga la denegación de servicio como un posible subsistema de detección de esta, justo por encima del sistema distribuido de detección de Snort, esto lo realizamos de esta manera por que podemos abstraer lo que son ataques de firmas, exploits, etc... de los que son de flujo, ddos, aunque hay ciertos como pueden ser las expansiones de virus las cuales pueden estar coordinadas entre el subsistema ddos y el subsistema de detección de firmas.

El PDL envía decisiones de Firewalls, bloqueo de direcciones a los Routers R8 y R9, los cuales en función de la carga del ML propagan o no hacia sus nodos superiores, quitamos de esta forma el procesamiento de análisis de las cabeceras Ip y Tcp del PDL y se la enviamos al ML, de forma que al PDL solo le llegan flujos de información que el decide.

Conclusiones

Las soluciones Opensource como pueden ser Snort o Prelude-IDS están actualmente muy valoradas y suponen un ahorro de costes considerablemente respecto a las soluciones comerciales como pueden ser las de Cisco.

Según los ataques o las nuevas técnicas de evasión avanzan, la complejidad de dichos sistemas crece, por lo que se necesita constantemente revisar la optimización de los sistemas y prácticamente esta a la ultima en técnicas de intrusión, evasión, Ddos, virus, etc….

"Si no puedes con tu enemigo alíate con el", si hay ataques distribuidos de denegación de servicio o virus las únicas soluciones que mitigan dichos ataques son los sistemas distribuidos, un solo Firewall, Router o NIDS entraría en saturación rápidamente.

Actualmente Snort es el mejor sistema de Detección de Intrusos, no solo por su motor de detección, sino por su versatilidad en diferentes entornos de red, por la cantidad de software que hay de complemento, análisis de Logs, interacción con Firewalls y por estas y muchas mas razones Snort fue comprada por CheckPoint , por algo será.

Hay que destacar que lo que se tiende actualmente son a Frameworks de Defección o entornos especializados, principalmente distribuidos, ya que una sola maquina no puede hacer frente a 1000, y es por eso por lo que las soluciones que adoptemos tienen que ser soluciones abiertas y no cerradas lo que nos limitan solo a un fabricante.

Referencias

[1] Common Intrusión Detection Framework http://gost.isi.edu/cidf/

[2] Intrusión Detection Exchange Format http://www.ietf.org/html.charters/idwg-charter.html

[3] Prelude-IDS, The Hybrid IDS Framework, http://www.prelude-ids.org

[4] CheckPoint http://www.checkpoint.com/

[5] Symantec http://symantec.be/region/la/product/appliance/7100/DS00153-SL_NetSec_7100_fs.pdf

[6] Cisco www.cisco.com

[7] P. Ferguson, D. Senie: «Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing», IETF RFC2827, 2000.

[8] Greene, Barry Raveendran, Neil Jarvis. “Unicast Reverse Path Forwarding (uRPF) Enhancements for the ISP ISP Edge, Version 1.5.” 18 Feb. 2001. 31 July 2003 <ftp://ftpeng.cisco.com/cons/isp/security/URPFISP.pdf>.

[9] Greene, Barry. “Loose Mode Ingress Prefix Filter Template, Version 4.” 8 Apr. 2003. 18 Aug. 2003. <ftp://ftpeng.cisco.com/cons/isp/security/IngressPrefixFilterTemplates/TipprefixfilteringressloosecheckV4.txt>.

[10] J. Ioannidis, S. M. Bellovin: «Implementing Pushback: RouterBased Defense Against DDoS Attacks», Proceedings of the 2002 Network and Distributed Systems Security Symposium, 2002.

[11] RFC-792 http://www.faqs.org/rfcs/rfc792.html

[12] Simulating and optimising worm propagation algorithms. Tom Vogt <tom@lemuria.org>.29th September 2003. (updated 16th February 2004)

[13] Snort www.snort.org

[14] Libpcap http://www.Tcpdump.org/

[15] M. Roesch. “Snort Presentation in The Black Hat Conference. Las Vegas’01”,http://www.blackhat.com/presentations/bh-usa-01/MartyRoesch/bh-usa-01-Marty-Roesch.ppt

[16] irpas http://www.phenoelit.de/fr/tools.html

[17] Nemesis http://www.packetfactory.net/projects/nemesis/

[18] Nmap http://www.insecure.org/

[19] Dsniff http://www.monkey.org/~dugsong/dsniff/

[20] Arp0c http://www.phenoelit.de/arpoc/

[21] Exact String Matching Algorithms http://www-igm.univ-mlv.fr/%7Elecroq/string

[22] Applying Fast String Matching to Intrusion Detection http://woozle.org/~mfisk/papers/setmatch-raid.pdf

[23] Implementation of the Wu-manber Algorithm http://www2.dcc.ufmg.br/~ghuiban/paa/tp3/node19.html

[24] A FAST ALGORITHM FOR MULTI-PATTERN SEARCHING Sun Wu, Udi Manber, May 1994

[25] http://www.bleedingsnort.com/

[26] ClamAV http://www.clamav.net/

[27] Ossec Hids http://www.ossec.net/hids/

[28] LibXML http://www.xmlsoft.org/

[29] Netfilter http://www.netfilter.org/

[30] Snort InLine http://snort-inline.sourceforge.net/

[31] DBus http://www.freedesktop.org/wiki/Software_2fdbus

[32] Detecting Syn Flooding Attacks, http://www.cs.wm.edu/~hnw/paper/attack.pdf

[33] Rfc 2671 , http://www.faqs.org/rfcs/rfc2671.html

[34] Dns Amplification Attacks, http://www.isotf.org/news/DNS-Amplification-Attacks.pdf