Centro de Investigación y de Estudios Avanzados del IPN
Departamento de Ingeniería Eléctrica
Sección Computación

Tolerancia a fallas para sistemas de detección de intrusos de red

Tesis que presenta

José Antonio Coria Fernández

Para obtener el grado de:
Maestro en Ciencias
en la Especialidad de
Ingeniería Eléctrica Opción Computación

Director de Tesis:
Dr. Jorge Buenabad Chávez

México, DF Junio del 2004.

ABSTRACT

Intrusion detection systems are complementary mechanisms to firewalls. Their purpose is to detect events caused either by attackers who access a system through the Internet, by authorized users of a system who attempt to gain additional privileges, or by authorized users who misuse the privileges given to them.

Currently, a drawback of intrusion detection systems occurs when a sensor of net traffic becomes non-operational due to a hardware or software fault. As a result the events occured, after the fault and while this remains, cannot be monitored. Part of the information of that sensor is lost, and cannot be correlated with the information provided by other sensors to detect intrusion patterns. Another drawback occurs when analytic server becomes non-operational since the analytic server is in charge of collecting and storing the alerts sent by the sensors, it is considered the most critical entity of an intrusion detection system.

This thesis presents the design of fault tolerance, based on redundant and cross-corroboration techniques, to solve both drawbacks mentioned above. Our design is also based on a free software intrusion detection system.

Our design introduce a minimal network overhead and is able to detect both, sensor and analytic server faults without loss of information.

Keywords Analitic server, intrusion detection systems, networks, security, sensor.

RESUMEN

Los sistemas de detección de intrusos (SDI's) son mecanismos complementarios a los mecanismos de protección perimetral (firewalls).

Su propósito es detectar eventos causados por personas no autorizadas que accesan un sistema desde Internet, por usuarios autorizados de un sistema que intentan obtener privilegios adicionales y por usuarios que abusan de los privilegios que se les han otorgado.

Una limitante en los sistemas de detección de intrusos se presenta cuando cualquiera de los sensores encargados de analizar el tráfico de la red deja estar disponible a consecuencia de una falla de hardware o software. El resultado es que los sucesos que ocurren en el momento en que se presenta la falla y mientras esta permanece, no se registran. La información relacionada a este sensor se pierde y por lo tanto no puede efectuarse la correlación con la información de otros sensores para detectar patrones de intrusión. Otra limitante existente se presenta cuando el servidor de análisis deja de estar disponible para realizar la recopilación y almacenamiento de las alertas enviadas por los sensores, actividad por la cual es considerado como la entidad más crítica en un SDI.

Esta tesis presenta el diseño de una arquitectura que incluye técnicas de redundancia y corroboración cruzada para dar una solución a las limitantes antes mencionadas. Nuestro diseño se apoya en un sistema de detección de intrusos basado en red de software libre para su desarrollo.

Nuestro diseño introduce una carga mínima en la red y es capaz de detectar las fallas de los sensores y del servidor de análisis sin perdida de información.

Palabras claves Redes, seguridad, sensores, servidor de análisis, sistemas de detección de intrusos

Agradecimientos

A *Dios* por permitirme una vez más concluir mis metas, en compañía de mis seres queridos y conocer a valiosos amigos.

A mis *Padres* por alentarme día a día a continuar superándome. Y darme todo su apoyo y confianza en los momentos difíciles.

A mis *Hermanos* por escuchar las historias que viví en el CINVESTAV y darme su apoyo en todo momento.

A mi *asesor* Dr. Jorge Buenabad Chávez por su dedicación y paciencia en el desarrollo y revisión de este trabajo.

A mis *sinodales* Dr. Sergio Chapa Vergara y Dr. Luis Gerardo de la Fraga por sus valiosos comentarios para la culminación de este trabajo.

A los *Doctores* de la Sección de Computación, quienes compartieron sus experiencias como sus conocimientos.

A *Sofia Reza* por todo el apoyo que me brindaste y ser paciente con todos los latosos como yo.

A mis *amigos* (sin tener un orden de preferencia) Juan Carlos Seck y Genaro Juárez por brindarme su amistad desde la licenciatura y por invitarme a incursionar y vivir esta experiencia en el CINVESTAV. Juan Carlos Medina, Jaime Arellanes, Jorge Lizárraga, José Guadalupe Ruiz, José de Jesús Trujillo, Paola Ramírez, Laura Molina por brindarme su amistad y compartir momentos de alegría y esparcimiento. Nancy Martínez por tus palabras de aliento y por permitirme compartirte mis vivencias. Jose Juan Rico por tu amistad e intéres en saber como iba la Maestría. Mauricio Rico por brindarme tu amistad y tu apoyo incondicional.

Al *CONACyt* por el apoyo económico brindado a través de la beca, lo que me permitió cursar las Maestría en el Departamento de Ingeniería Eléctrica Sección Computación.

Índice general

Índice de Figuras

Capítulo 1

Introducción

En la última década Internet ha brindado nuevas posibilidades en las formas de comunicarnos y de realizar negocios, pero también ha implicado un crecimiento en los riesgos de seguridad para los sistemas computacionales. Por esta situación, se han desarrollado mecanismos de seguridad de protección perimetral (firewalls), los cuales tratan de prevenir accesos sin autorización a los datos y recursos de un sistema.

Sin embargo estos mecanismos han sido insuficientes. Las actividades maliciosas o eventos de intrusión aún comprometen la confidencialidad, integridad y disponibilidad de los recursos computacionales. Usualmente estos eventos son causados por personas no autorizadas que acceden un sistema desde Internet, por usuarios autorizados de un sistema que intentan obtener privilegios adicionales, y por usuarios que abusan de los privilegios que se les han otorgado.

Por ello se ha propuesto un mecanismo complementario conocido como sistema de detección de intrusos (SDI). Un SDI registra y analiza información relacionada con las actividades de otro sistema con el propósito de detectar sus actividades maliciosas. También puede tomar una acción correctiva inmediata, o solo avisar a los administradores para que decidan sobre la solución posterior.

La información de las actividades de un sistema puede ser la bitácora de operaciones de una aplicación, la bitácora del sistema operativo de una computadora o bien los paquetes que circulan en una red.

Si la información a analizar son cualquiera de dichas bitácoras, un SDI consistirá de varios agentes (programas monitores), un servidor de análisis y una computadora monitor. En cada computadora de una red, un agente enviará periódicamente la(s) bitácora(s) o la(s) analizará y enviará su diagnóstico al servidor de análisis. Este servidor es generalmente otra computadora en la red, dedicada al análisis de la información, y sobre la cual se hacen consultas de la información almacenada a través de una computadora monitor. Esta configuración se conoce como SDI basado en computadora. Su ventaja es que tiene la habilidad de monitorear eventos locales a una computadora y puede detectar si un ataque fue exitoso o no, gracias a que utiliza los registros de la bitácora de una computadora, lo que permite determinar cuales procesos y usuarios están involucrados en el ataque. Presenta la desventaja de utilizar recursos de cómputo de la computadora que esta monitoreando, produciendo un costo en el desempeño de la misma.

Si la información a analizar son los paquetes de la red, un SDI consiste de un servidor de análisis, de una computadora monitor y de una o varias computadoras que actúan como sensores. Cada sensor monitorea y analiza los paquetes de un segmento de red y envía su diagnóstico al servidor de análisis. Esta configuración se conoce como SDI basado en red y en sensores. Su ventaja es que permite cubrir una mayor cantidad de computadoras al monitorear segmentos de red, además de tener un pequeño impacto sobre la red debido a que usualmente son dispositivos pasivos que escuchan el tráfico en una red alámbrica sin interferir en su operación. Tiene la desventaja de no poder determinar si un ataque fue exitoso o no, solo puede percibir que un ataque ha sido iniciado.

Otro inconveniente de esta configuración es que si un sensor deja de

funcionar por fallas de hardware o software, los sucesos que ocurren a partir del momento en que se presenta la falla y mientras ésta permanece, no se registran. Además al perder información correspondiente a un segmento de una red, se limita la visión global de los acontecimientos en la red, al no poder realizar la correlación con la información de otros sensores.

En el caso de que la falla se presentara en el servidor de análisis, se tendría un problema más crítico debido a que este tiene que recopilar y almacenar los mensajes de alertas enviados por todos los sensores para agrupar la información y correlacionarla a patrones de intrusión.

Sin embargo, pocos mecanismos de tolerancia a fallas se han considerado hasta ahora ante estas situaciones, aún cuando el tema de la tolerancia a fallas es bien conocido en el área de computación [14, 17, 18]. Si los componentes mencionados incurren en una falla de hardware o software, el sistema es incapaz de detectarlo o recuperarse del mismo.

En el modelo de Praririe Dog [34], se provee tolerancia a fallas a un SDI mediante otro SDI. Prairie dog busca comportamientos anormales en los procesos que se ejecutan en el SDI. Además propone emplear replicación activa de los procesos monitores del SDI. Sin embargo, este trabajo es solo un modelo conceptual; no se tienen referencias de su realización (el hardware y software requerido).

Esta tesis presenta el diseño de una arquitectura tolerante a fallas para SDIs basados en red y en sensores. Nuestra arquitectura provee de tolerancia a fallas al servidor de análisis por medio de la replicación pasiva del mismo. Se tienen dos copias, la primaria y la de respaldo. Las actualizaciones de información son procesadas por el servidor primario en primer lugar, y después el primario las envía al respaldo; si el primario falla, el secundario se convierte en primario automáticamente. Con este propósito el servidor de análisis primario envía mensajes periódicamente al secundario, la interrupción de estos mensajes es la señal de que el primario ha fallado.

No se consideró conveniente la replicación de los sensores para no incrementar los costos de implementación excesivamente, pues se requiere de un mayor número de equipos de cómputo. Consideramos justificada la replicación del servidor de análisis por ser el elemento más crítico.

Para los sensores se diseño e implementó la corroboración cruzada, es decir, la transmisión periódica de mensajes entre los sensores y hacia el servidor de análisis para comprobar que siguen activos y no presentan fallas. En el caso de presentarse alguna falla en cualquiera de los sensores se notifica a la computadora monitor del suceso, y la respuesta se deja a decisión del personal responsable de la seguridad de la red.

Además, si el servidor de análisis central falla, los sensores de manera automática se redireccionarán a la copia del mismo, dando así continuidad al flujo de la información recopilada por el SDI.

Nuestra arquitectura es simple, flexible y sin costo al emplear utilizar herramientas de software libre.

1.1. Esquema de la tesis

Hemos introducido el motivo y los alcances de nuestra investigación: la falta de un esquema de tolerancia a fallas en un SDI y una solución simple, flexible y sin costo a la misma. El resto de las tesis esta organizada de la siguiente manera.

El capítulo 2 presenta los elementos de los SDIs: fuentes de información, técnicas de análisis de información, la correlación, tipos de respuestas a las actividades de intrusión y las características de los diseños de los SDI.

En el capítulo 3 se describen algunos de los modelos de tolerancias a fallas para SDI propuestos, una alternativa a estos modelos y nuestro diseño de tolerancia a fallas para un SDI basado en red y nuestra propuesta.

El capítulo 4 presenta la implementación de nuestro diseño tolerante a fallas para un SDI basado en red.

El capítulo 5 presenta la evaluación de nuestro diseño tolerante a fallas para un SDI basado en red y sus resultados.

El capítulo 6 presenta nuestras conclusiones y sugerencias para trabajos futuros.

Capítulo 2

Sistemas de detección de intrusos

Hoy día las redes locales y el Internet son indispensables para que las personas se comuniquen y compartan recursos fácil y eficazmente. Sin embargo son también susceptibles a ataques. Estos ataques o intrusiones tienen el propósito de comprometer la confidencialidad, integridad o disponibilidad de los recursos computacionales. Para detectarlos se han desarrollado los sistemas de detección de intrusos (SDI).

Los SDI monitorean y registran los eventos que ocurren en una computadora o en una red de computadoras, para analizarlos y correlacionarlos en la búsqueda de señales de intrusión y dar así una respuesta que permita corregir esta situación [2]. Además, la información entregada por un SDI, permite vincular una intrusión dada a un posible responsable, y así tener bases para realizar cargos criminales contra el mismo. Debido a lo anterior los SDI han ganado aceptación como una herramienta adicional de seguridad en las organizaciones.

Un SDI puede ser descritos en términos de tres componentes. Las *fuentes de información*, o de donde se obtienen los datos que se analizan, pueden ser: la red (los paquetes que se transmiten), la computadora (los sucesos en el sistema operativo) o una aplicación (los eventos que se presentan en la ejecución de un programa en particular). El *análisis* son las

técnicas utilizadas para analizar los datos recopilados. Las técnicas más comunes son: la detección de usos incorrectos y la detección de anomalías. Las *respuestas* corresponden al conjunto de acciones que se ejecutan una vez que una intrusión es detectada. Pueden ser activas o pasivas. En este capítulo describimos con más detalle estos y otros conceptos relacionados.

2.1. Fuentes de información

Las fuentes de información de un SDI son de tres tipos. En la literatura [3] estos tipos se han utilizado para clasificar a los SDI.

Los SDI *basados en red* detectan ataques capturando y analizando paquetes de una red, escuchando en un segmento de esta pueden monitorear el tráfico correspondiente a múltiples computadoras que están conectadas al segmento. Generalmente utilizan un conjunto de sensores colocados en varios puntos de una red. Los sensores son computadoras dedicadas a ejecutar el SDI.

Los SDI *basados en computadora* operan con la información que se registra en la bitácora del sistema operativo. Esto les permite analizar actividades con gran confiabilidad y precisión, ya que es posible determinar exactamente cuales procesos y usuarios están envueltos en un ataque en particular. Además permiten *ver* las consecuencias de un ataque, ya que pueden acceder y monitorear los archivos de datos y la información de los procesos de un sistema utilizados en los ataques.

Finalmente, los SDI *basados en aplicación*, que son un subconjunto de los SDI basados en computadora, analizan los eventos que suceden únicamente dentro de los programas de aplicación. Operan con la información recopilada de las bitácoras de operación de una aplicación.

2.2. Técnicas de análisis de los SDI

La información de las actividades monitoreadas, de una red o computadora, es posible analizarla mediante dos técnicas principales conocidas como: *detección de usos incorrectos* y *detección de anomalías.*

La *detección de usos incorrectos* representa un evento o conjunto de eventos que describen un ataque en la forma de un patrón o *firma*, razón por la cual también es llamada *detección basada en firmas.*

Los métodos que utiliza esta técnica incluyen: sistemas expertos, monitoreo de lo que se digita en el teclado, detección de intrusión basado en el modelo de transición de estados y sistemas basados en la coincidencia de patrones [31].

Esta técnica es muy efectiva al construir descriptores o reglas específicas sobre una actividad de intrusión y no genera un abrumador número de falsas alarmas pues únicamente genera alarmas cuando encuentra una coincidencia con las reglas.

A continuación se muestra un ejemplo de un ataque aplicable al sistema operativo Sun versión 4.1.1 y su representación mediante diagramas de transición de estados [16].

Los pasos en este proceso se pueden explicar como sigue:

- Un atacante crea un enlace físico (liga dura) a un archivo script el cual pertenece al usuario root y tiene el permiso suid (permite realizar a usuarios sin privilegios actividades que requieren de estos, entre estas actividades se tiene el cambiar la contraseña) con el mecanismo #!/bin/sh. Y nombra al nuevo archivo comenzando con un guión, seguido de cualquier caracter.

- Ejecuta el archivo.

La característica del ataque antes mencionado, es que al crear un enlace físico de un archivo se crea nueva entrada de directorio con la información del

dueño y permisos del archivo original. Al ejecutar un script con el mecanismo #!/bin/sh se invoca un subshell. Además, si el nombre del subshell comienza con un guión se obtiene un shell interactivo con privilegios de root permitiendo hacer cualquier acción sobre el sistema. Como se muestra en la figura 2.1, se tiene un estado inicial A cuando se accede a una sesión en UNIX; se pasa al estado B cuando se crea un enlace físico al archivo con el permiso suid; y finalmente se pasa al estado C cuando se ejecuta el script obteniéndose los privilegios de root.

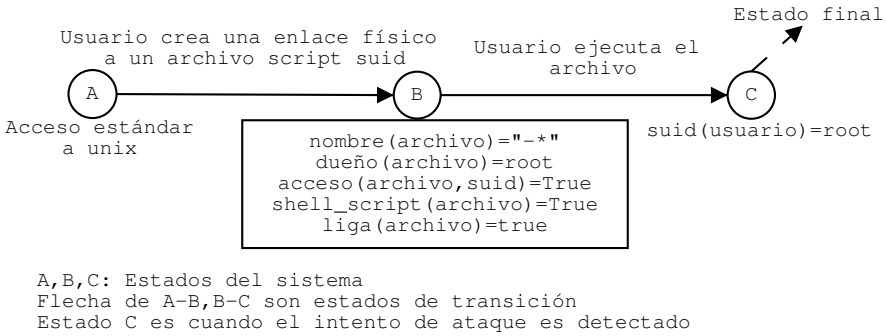

Figura 2.1: Diagrama de transición de estados para un ataque en SunOS 4.1.1.

La *detección de anomalías* construye un perfil de actividad normal para un usuario, computadora o conexión de red a partir de la recolección histórica de datos en un período de operación normal. Basándose en este perfil se monitorean las actividades posteriores en la búsqueda de comportamientos considerados inusuales (anómalos) que se desvían de la normal construida.

Desafortunadamente, esta técnica frecuentemente produce un gran número de falsas alarmas, debido a que los patrones normales de los usuarios y los comportamientos del sistema varían continuamente. Sin embargo, es la única manera de detectar nuevas formas de ataque, lo cual no es posible con la detección de usos incorrectos que confía en coincidencia de patrones de ataques ya conocidos.

Los métodos empleados en la detección de anomalías son: la generación de patrones predictivos, las redes neuronales y el enfoque estadístico [31].

A continuación se muestra un ejemplo de generación de patrones predictivos [31], la cual busca predecir el futuro basándose en eventos ocurridos. Así podríamos tener la regla representada en la figura 2.2:

```
E1 - E2 --> (E3 = 80%, E4 = 15%, E5 = 5 %)
```

Figura 2.2: Generación de un patrón de predicción.

E1,.....,E5 son eventos. Por ejemplo, E1 representa el iniciar una sesión en UNIX, E2 buscar un archivo, E3 leer ese archivo, E4 editar ese archivo y E5 eliminar ese archivo.

Los eventos E1 y E2 ya han ocurrido, con la restricción de que E2 debe ocurrir después de E1. La probabilidad de que cualquiera de los eventos E3, E4, y E5 ocurran después de E2 es la siguiente: con un 80 % E3, un 15 % E4 y un 5 % E5. Pero podría darse el caso que se realizará como evento alterno un enlace simbólico al archivo; este evento no esta contemplado en esta regla de predicción, pero su registro ayudaría a identificar variaciones del patrón inicial.

Se podría agregar esta nueva opción al patrón predictivo para su refinamiento y quizá pasar posteriormente a la construcción de una *firma* de este patrón cuando este se tenga ya bien identificado.

2.3. Correlación de información

Un SDI puede consistir de varias fuentes de información, y una vez que se ha hecho el análisis de información en cada una de ellas, se realiza una correlación de los diferentes análisis. El propósito es determinar si se trata de una intrusión aislada, si existe alguna relación entre distintas intrusiones o bien encontrar un nuevo tipo de intrusión. Por esta razón la correlación es un proceso fundamental en los SDIs, en la búsqueda de patrones para reconocer un ataque.

Algunos tipos de correlación de intrusión existentes son: correlación de red de una sesión, correlación de red de múltiples sesiones, correlación en tiempo real, correlación por intervalos y correlación de fuentes internas y externas de información [2]. A continuación las describimos y ejemplificamos solo algunas.

La *correlación de red de una sesión* analiza el flujo bidireccional de los paquetes únicamente de una conexión entre dos computadoras. En una conexión basada en TCP/IP se pueden analizar las direcciones IP fuente y destino, los puertos fuente y destino, el protocolo y el tiempo que se mantiene la conexión.

Por ejemplo, se envía un paquete P de una fuente F a un destino D. Al examinar la información de P (IP fuente-destino, puerto fuente-destino, protocolo), se guarda temporalmente. Cuando se recibe el paquete P' de respuesta de D a F, se verifica que la respuesta corresponde a la solicitud inicial de F a D. Este tipo de análisis se lleva a cabo en sistemas de protección perimetral o ruteadores para evitar que se lleven acabo ataques de falsificación de dirección fuente y hombre en el camino.

La representación de lo anterior se muestra en la figura 2.3:

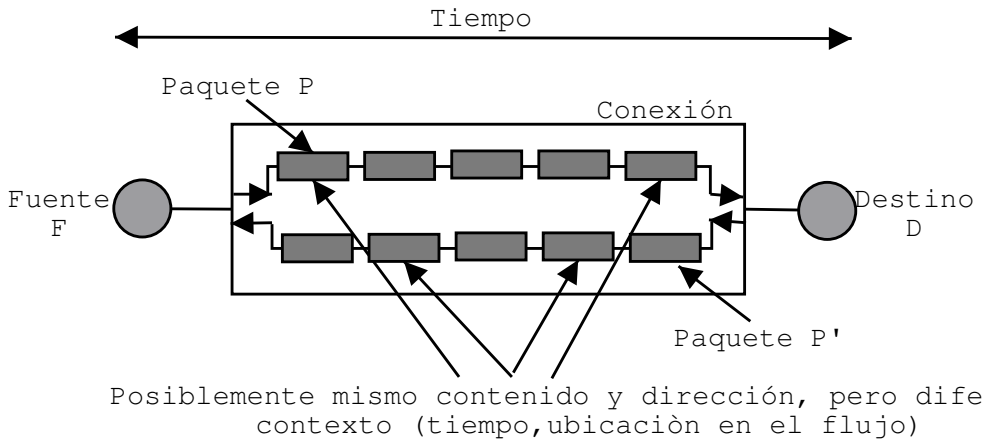

Figura 2.3: Correlación de una sesión individual.

La *correlación de múltiples sesiones* analiza la información de varias

conexiones, es el caso general del ante-rior. Busca patrones de actividad maliciosa entre múltiples sesiones independientes de una sola computadora o en varias.

Por ejemplo: se tienen cinco computadoras A, B, C, D y E. De la computadora A se comienza dos sesiones a las computadoras B y C. Si posteriormente no se inicia otra sesión de las computadora B o C durante un tiempo X, se podría descartar como un caso de actividad de intrusión. En caso contrario si de B se realizarán conexiones a D y E, esto podría considerarse como un incidente de seguridad, pues quizá se este realizando la propagación de código malicioso (gusano).

La figura 2.4 representa lo antes mencionado.

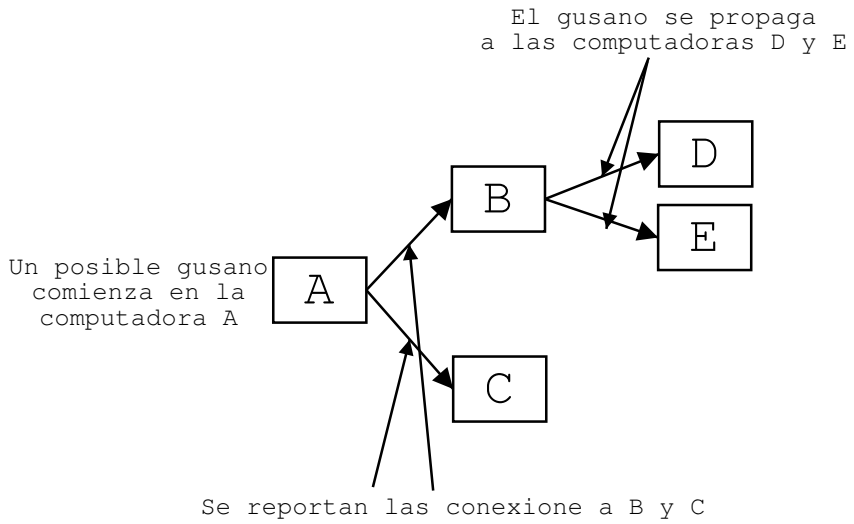

Figura 2.4: Correlación de múltiples sensores.

La *correlación en tiempo real* se efectúa en el momento en que se obtienen los datos de las fuentes de información, dando como beneficio una detección oportuna. Pero limita el conocer lo que sucede en el instante posterior a la captura de la información.

En la *correlación basada en intervalos* se puede tener toda la información acerca de una intrusión, pero quizá se desconozca si ésta prevalece o

se detectará después de un período largo desde que sucedió. Aún con este inconveniente es una de las herramientas de análisis de bitácoras.

La *correlación de fuentes internas* analiza información propia del sistema sin intervención o entrada de datos humana. Un ejemplo de este tipo de información son los encabezados que el protocolo TCP/IP concatena a los mensajes de una aplicación.

La *correlación de fuentes externas* incluye tanto a la información de fuentes internas como cualquier otra información que podría ser considerada en la correlación. Esto podría ser virtualmente cualquier cosa como: las noticias u opiniones de administradores de red. Un problema con este tipo de correlación es que es posible introducir más datos erróneos que influyan el análisis de un proceso de intrusión.

2.4. Respuestas de los SDI

Si el análisis de la información y/o su correlación detecta un problema se genera una respuesta. Las respuestas son de los siguientes tipos:

Las respuestas *activas* son acciones automatizadas tomadas cuando ciertos tipos de intrusiones son detectados. Se tienen tres categorías de respuestas activas:

La colección de información adicional de lo que se sospecha es un ataque, involucra incrementar el nivel de "sensitividad" de las fuentes de información. Un ejemplo de esto es incrementar el número de paquetes que debe capturar un SDI basado en red, y no restringirlo solamente a un puerto o computadora. Esto permite a una organización tener una mayor cantidad de información que puede utilizar para dar soporte a la investigación y aprehensión de un atacante.

El cambio del ambiente, consiste en detener un ataque en progreso vía reconfiguración de dispositivos como ruteadores o sistemas de protección

perimetral para bloquear el acceso del atacante.

Finalmente, tomar acciones contra el atacante, involucra lanzar ataques en contra del intruso o intentar activamente obtener información acerca de la computadora del atacante o el sitio donde se encuentra. Sin embargo, esta respuesta no es recomendable, debido a que muchos atacantes utilizan direcciones de red falsas cuando atacan sistemas, lo cual podría acarrear un gran riesgo el causar daños a sitios o usuarios inocentes de Internet y quizá se caiga en lo ilegal.

Las respuestas *pasivas* proveen información del sistema a los usuarios, dejando a los humanos tomar una acción "subsecuente" basada en esa información.

Se generan alarmas y notificaciones por parte del SDI para informar a los usuarios cuando un ataque es detectado. Las formas más comunes de notificar una alarma es presentando un mensaje de alerta en pantalla.

La información proporcionada en los mensajes de alarmas varía ampliamente. Puede ser una notificación en la que solo se muestre que una intrusión se ha llevado a cabo, hasta un mensaje detallado que contenga la dirección IP de la fuente y el objetivo de el ataque, la herramienta específica de ataque empleada para obtener acceso, y el resultado del ataque.

2.5. Diseño de SDIs

Los primeros diseños de SDIs se ejecutaban en la misma computadora que protegían, debido a que la mayoría de las computadoras eran mainframes, y el costo de ejecutar un SDI por separado involucraba un costo excesivo [5]. Esta situación presentó un problema de seguridad. Si un atacante tenia éxito en su ataque al sistema objetivo, podía simplemente deshabilitar el SDI como parte integral del ataque.

Con la llegada de las estaciones de trabajo y computadoras personales,

la mayoría de los diseños de SDIs movieron la ejecución del control y el análisis de información a una computadora por separado. Esto permite ocultar la existencia de un SDI de los atacantes, mejorando la seguridad. En esta sección presentamos los diseños de SDIs cuyo control y análisis de información se encuentran en una computadora dedicada a estas funciones. Recuérdese que un SDI incluye también las fuentes de información y las respuestas.

2.5.1. SDIs basados en computadora

Los SDIs basados en computadora, cuyas fuentes de información son la bitácora de una aplicación o la bitácora del sistema operativo (sección 2.1), se diseñan en base a agentes o programas monitores. En cada computadora se ejecuta un agente que monitorea los eventos de la misma. Ejemplo de eventos a monitorear incluyen la apertura de un archivo o la ejecución de un programa.

El análisis de información puede ser *centralizado* o *distribuido*. En el diseño *centralizado* de análisis de información, los agentes crean un registro por cada evento que monitorean y envían este registro a la computadora o servidor de análisis (al cual identificaremos como analizador) en un intervalo de tiempo.

La comunicación entre agentes y analizador usa un canal de comunicación seguro.

El analizador busca un patrón de intrusión con la información recibida y crea una bitácora. En el caso de detectar un patrón de intrusión, genera alertas que envía a varios subsistemas diferentes para notificación, respuesta y almacenamiento.

Lo anterior se muestra gráficamente en la figura 2.5.

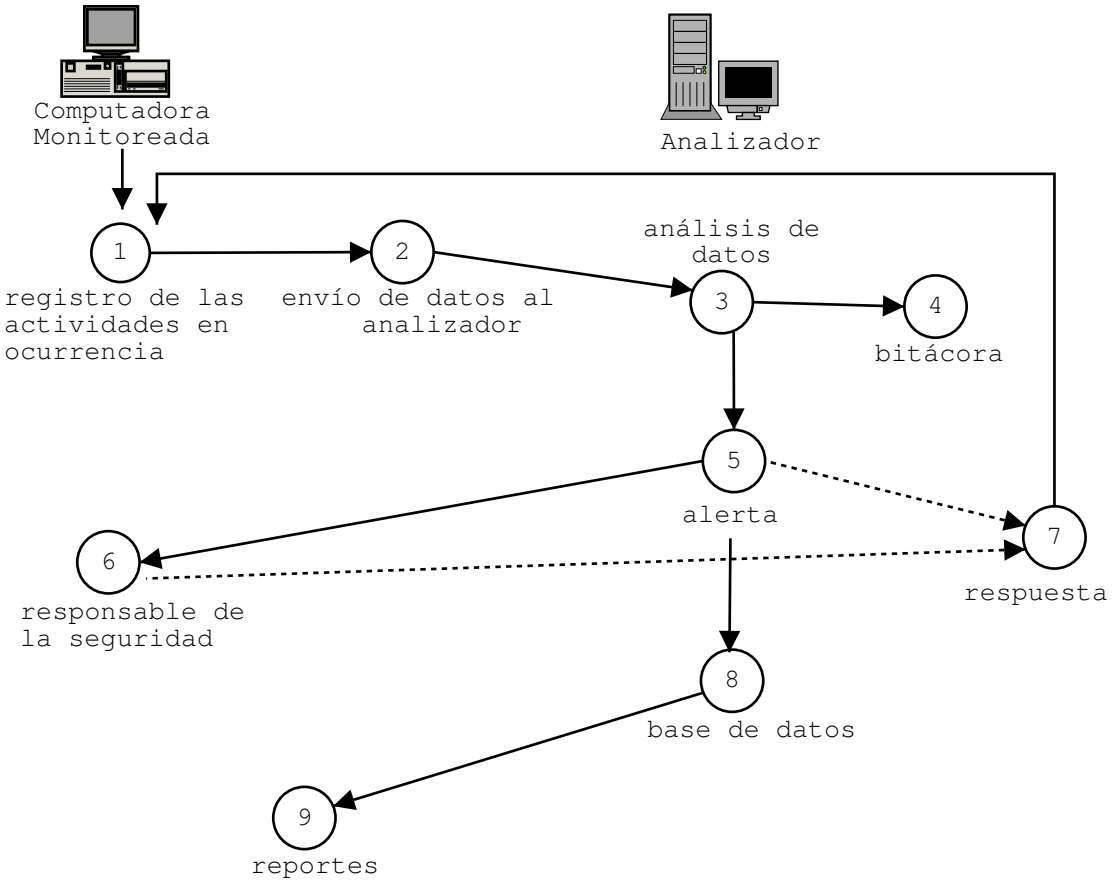

Figura 2.5: SDI basado en computadora y análisis de información centralizado.

En el diseño *distribuido* de análisis de información, cada agente analiza los registros de los eventos que monitorea. Así el análisis y una posible respuesta, ocurren en cada computadora. Pero las alertas se envían al analizador, en donde se almacenan para después correlacionarlas. La Figura 2.6 representa lo antes descrito.

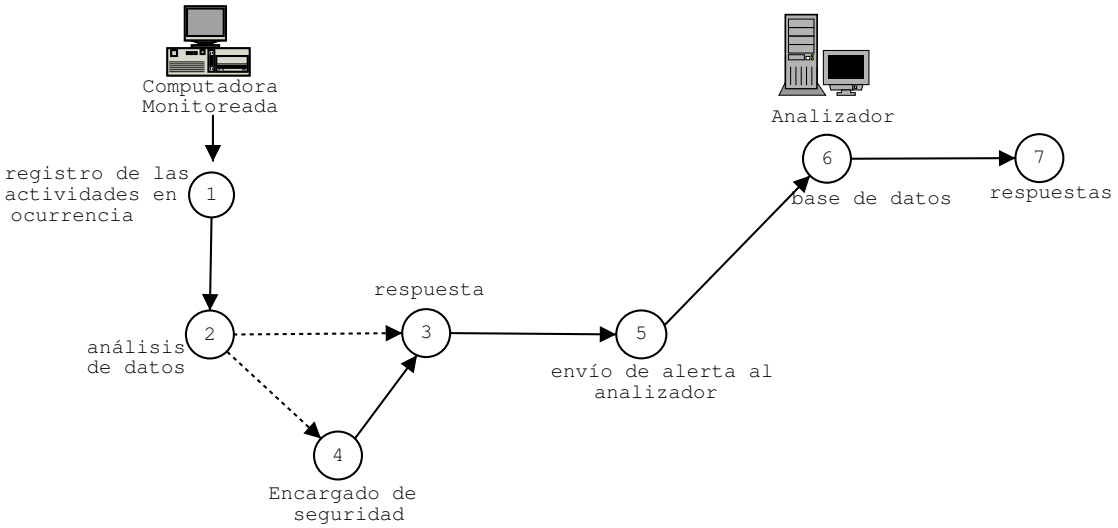

Figura 2.6: SDI basado en computadora y análisis de información distribuido.

2.5.2. SDIs basados en red

Los SDI basados en red, cuya fuente de información son los paquetes de red, se pueden diseñar con una recopilación de información *centralizada* o *distribuida* de tales paquetes.

Con recopilación *centralizada*, una computadora sensor monitorea los paquetes que transitan en un "segmento de una red". Cada computadora sensor analiza los paquetes en busca de intrusiones. Si detecta alguna intrusión envía la alerta al analizador y notifica al responsable de la seguridad del sistema o da una respuesta predefinida. Una respuesta particular de estos SDIs puede ser la reconfiguración de un ruteador o un sistema de protección perimetral para rechazar el tráfico generado desde una dirección fuente en particular.

La figura 2.7 ejemplifica lo anterior.

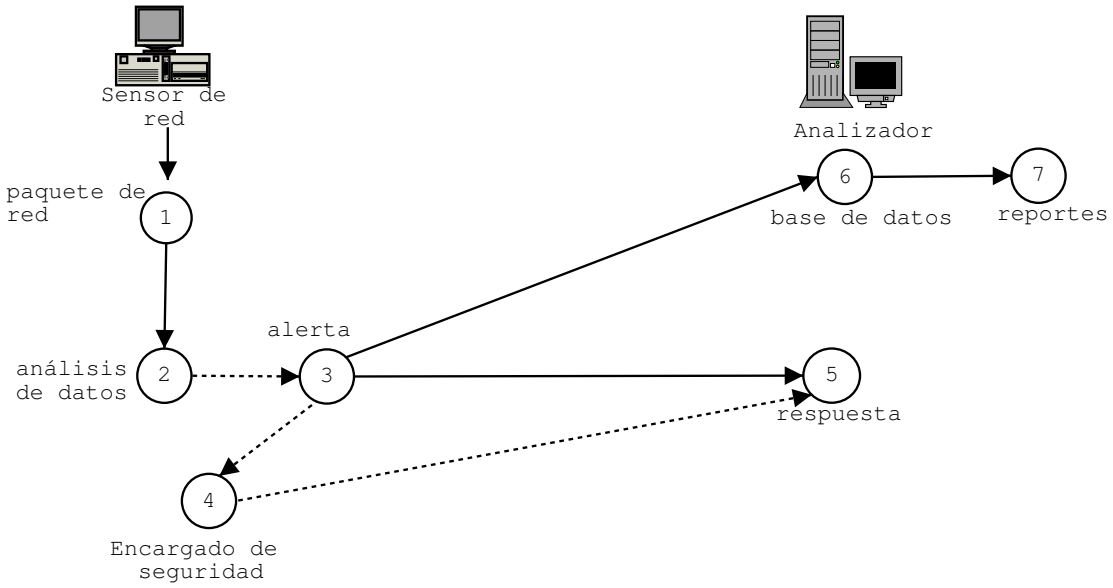

Figura 2.7: SDI basado en red y recopilación de información centralizada.

La configuración de SDIs basados en red y recopilación de información centralizada requiere que se determine la ubicación de los sensores.

Existen varias opciones para colocarlos, con diferentes ventajas asociadas a cada una, que a continuación mencionamos:

1. Ubicación detrás de cada sistema de protección perimetral externo ("Ubicación 1" en la figura 2.8). En esta ubicación se registran los ataques que se originan fuera de la red, y que traspasan el sistema de protección de perimetral, destacando los problemas con las políticas de seguridad o el desempeño de éste.

   Aún si el ataque de entrada no se reconoce el SDI puede algunas veces reconocer la intrusión en base al tráfico de salida proveniente del servidor comprometido.

2. Ubicación fuera del sistema de protección perimetral externo ("Ubicación 2" en la figura 2.8). Esta ubicación permite documentar el número y tipo de ataques originados en Internet que tienen como objetivo la red.

3. Ubicación en el canal de conexión principal de redes de Internet -backbone- ("Ubicación 3" en la figura 2.8). Esta ubicación permite monitorear una gran cantidad de tráfico de red, de tal forma que se incrementa la posibilidad de observar ataques. Es posible detectar actividades sin autorización por parte de usuarios autorizados dentro del perímetro de seguridad de la organización.

4. Ubicación en subredes críticas ("Ubicación 4" en la figura 2.8). Permite enfocarse a los recursos restringidos de la red considerados de gran valor, detectando ataques dirigidos a los sistemas y recursos críticos.

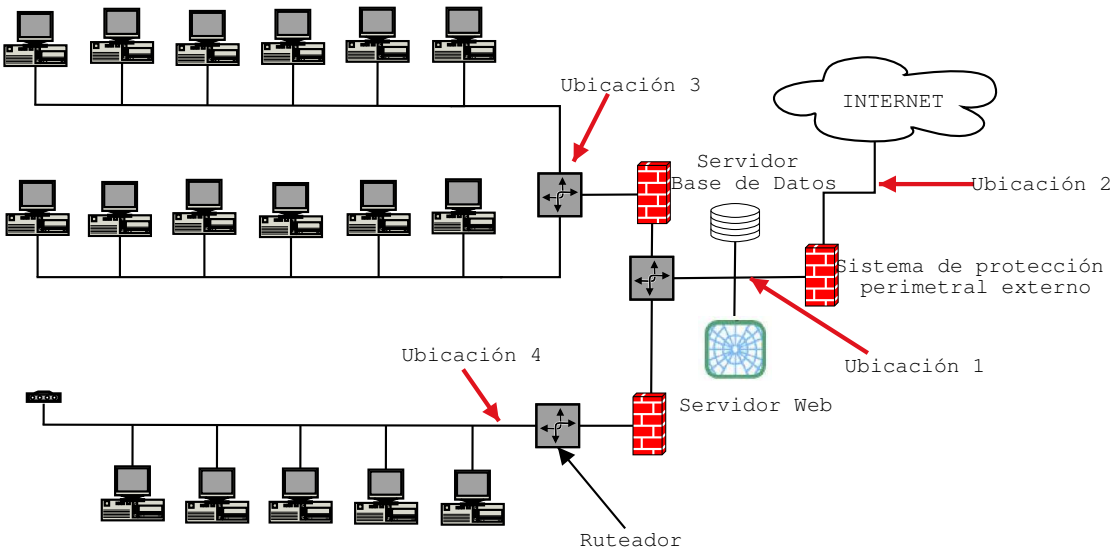

Figura 2.8: Ubicación de sensores de un SDI basado en red y recopilación de información centralizada.

En los SDIs basados en red y recopilación de información *distribuida* cada computadora de propósito general en un segmento de red, funciona también como sensor.

En cada computadora se ejecuta un agente que monitorea y analiza *únicamente* los paquetes enviados y transmitidos a esa computadora. Si un agente detecta una intrusión, envía la alerta al analizador para su almacenamiento y posterior correlación. Además, notifica al responsable de la seguridad del sistema, se da una respuesta y se generan reportes para resumir la actividad de las alertas. La figura 2.9 describe gráficamente lo anterior.

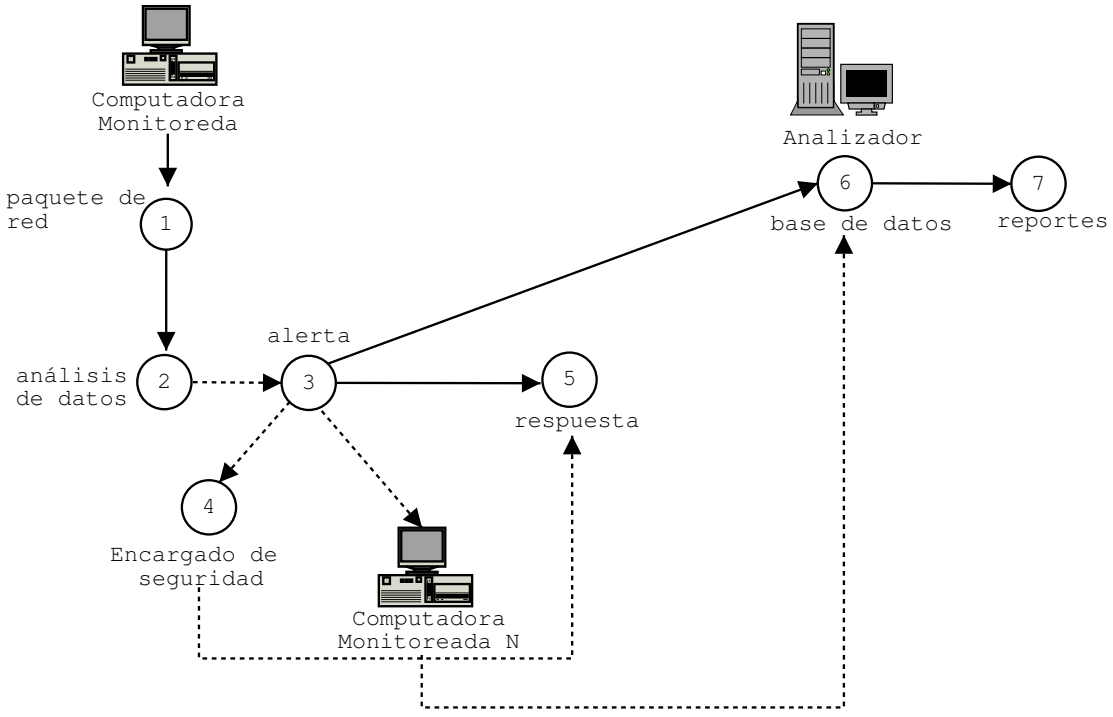

Figura 2.9: SDI basado en red y recopilación de información distribuida.

2.6. Resumen

En este capítulo se presentaron los elementos de los SDIs, los cuales son: las fuentes de información (computadora, aplicación, red) que proveen los datos, las técnicas de análisis (usos incorrectos y anomalías) para encontrar patrones de intrusión y las respuestas (activas y pasivas) que se generan una vez que se detecta una intrusión.

Otro proceso fundamental de los SDI es la correlación de la información. La cual tiene el propósito de interpretar, combinar y analizar información de todas las fuentes disponibles en una computadora o red de computadoras o cualquier otro medio de información que permita determinar si se trata de una intrusión aislada, si existe alguna relación entre distintas intrusiones o bien encontrar un nuevo tipo de intrusión. Algunos tipos de correlación de intrusión existentes son: la correlación de una sesión individual, la correlación de múltiples sesiones, la correlación en tiempo real, la correlación por intervalos, la correlación de fuentes internas y externas de información.

Finalmente presentamos algunos aspectos del diseño de los SDIs, es decir, la manera en que los elementos de este se organizan para el análisis, la recopilación, el almacenamiento y la generación de respuestas a actividades de intrusión. Entre los diseños de SDI tenemos: los SDI basados en computadora con análisis de información centralizado y distribuido y los SDI basados en red con recopilación de información centralizada y distribuida.

Capítulo 3

Tolerancia a fallas para SDIs

En el capítulo anterior se describieron los tipos de SDIs que se ha propuesto. Vimos sus componentes y la interacción entre los mismos. Actualmente, la mayoría de las investigaciones sobre SDIs tienen como principal enfoque la eficiencia para reducir el uso de recursos de cómputo, la precisión para mejorar las técnicas de análisis actuales y la cobertura para contener un mayor tipo de patrones de intrusión.

Estos aspectos son de suma importancia. Sin embargo si uno o varios de los elementos del SDI deja de funcionar, se pierden capacidades funcionales como la colección de datos, el análisis de información y el dar una respuesta, a partir del momento que se presenta una falla (software o hardware) y mientras ésta permanece. Es por esta razón que deben utilizarse técnicas de tolerancia a fallas como la *redundancia*, que es una técnica madura y ampliamente probada en el área de computación y la corroboración cruzada que permite la constante vigilancia del estado de los componentes de un sistema.

En este capítulo se presentan algunas propuestas de tolerancia a fallas en SDIs existentes y alternativas a las mismas. Finalmente se presenta nuestro diseño de tolerancia a fallas para SDIs basados en red.

3.1. Tolerancia a fallas de seguridad en SDIs

Algunos de los modelos de tolerancia a fallas para SDI, se enfocan a fallas correspondientes a la seguridad de una computadora. Por ejemplo, un error de programación puede producir una vulnerabilidad latente, la cual puede ser explotada por una atacante, y posteriormente propagarse a otros equipos.

Los modelos conceptuales de tolerancia a fallas Prairie Dog y de Agentes Móviles tienen este enfoque y se presentan a continuación.

3.1.1. Prairie Dog

El sistema Prairie Dog [34] es un meta-SDI, es decir, un SDI que protege a un SDI. Y está compuesto por tres elementos: el verificador de integridad (VI), el monitor del SDI (MDI) y el observador de la vecindad (OV).

El *monitor del SDI* verifica que el comportamiento y operación de los procesos del SDI se encuentren dentro de los limites considerados como normales. Para mantener la confiabilidad del monitor del SDI se tienen dos copias: el monitor del SDI-primario y el monitor del SDI-secundario, realizando las actualizaciones de información entre estas dos entidades mediante la replicación activa.

El *verificador de integridad* checa si se realizaron modificaciones y/o reemplazos sin autorización a archivos de configuración y ejecutables del monitor de SDI.

El *observador de la vecindad* monitorea al monitor del SDI-primario mediante el envío de mensajes encriptados periódicamente, utilizando tres copias de observadores del vecindario para realizar esta tarea. En el caso que el monitor del SDI primario y su respaldo se encuentren comprometidos, los observadores del vecindario generan mensajes de advertencia al personal de seguridad.

En la figura 3.1 las circunferencias externas representan computadoras,

las internas representan procesos o hilos en esas computadoras. Las flechas representan el flujo de los datos entre varios procesos o hilos. El monitor del SDI reside en la misma computadora donde se encuentra el SDI. Y cada una de las tres copias del observador del vecindario reside en una computadora remota diferente.

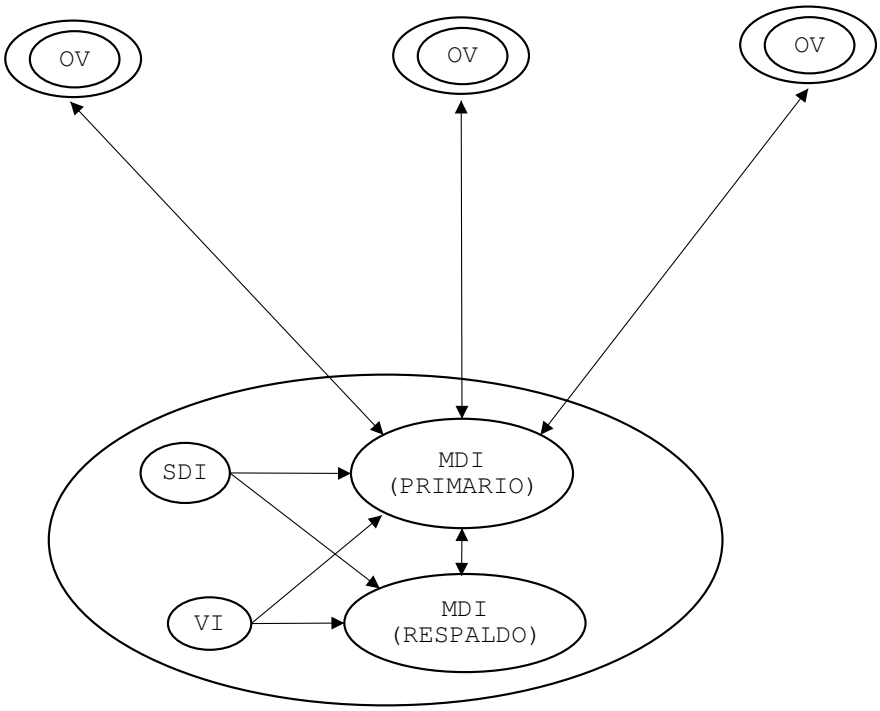

Figura 3.1: Sistema Prairie Dog. La circunferencias externas representan computadoras, las internas representan procesos o hilos.

El modelo Praire dog presenta similitud al diseño de SDI basado en computadora, ya que permite analizar las actividades del monitor del SDI y verificar si se realizaron modificaciones y/o reemplazos a archivos de configuración. Desafortunadamente tiene la desventaja de emplear gran cantidad de recursos de cómputo de la computadora donde reside, al ejecutar cuatro procesos simultáneos (monitor del SDI primario, monitor del SDI secundario, verificador de integridad y SDI). Situación que se puede reflejar en el desempeño de la computadora.

La característica de tolerancia a fallas utilizada entre el monitor del SDI primario y secundario, la replicación activa, presenta la desventaja de que ambos (primario y secundario) residen en la misma computadora. Al deshabilitar la computadora se perdería la información de lo ocurrido en el SDI.

Para mantener el monitoreo continuo del monitor del SDI, se envían periódicamente mensajes desde los observadores de vecindad, los cuales residen en diferentes computadoras. Esto implica que se debe contar con un mayor número de equipos de cómputo para realizar esta tarea, incrementando los costos.

3.1.2. Agentes móviles

En el modelo de agentes móviles [23], se consideran tres tipos de agentes todos capaces de migrar a diferentes nodos, estos son: agente crítico, agente representante y agente hijo. Cada uno de estos agentes reside en una computadora identificada análogamente con el nombre del agente.

En el agente/computadora crítico se realiza el análisis, control y almacenamiento de la información monitoreada. En el agente/computadoras hijo se obtienen y evalúan eventos. En la agente/computadora representante se provee la comunicación entre los agentes/computadoras críticos y los agentes/computadoras hijo.

La comunicación entre los agentes críticos, y los agentes hijo se lleva a cabo mediante los agentes representante con el objetivo de no revelar la ubicación en la red de los elementos críticos en una situación de intrusión.

Para la migración de agentes, se debe cumplir que: el tipo de computadora a donde el agente se va a mover pertenezca al mismo nivel. Un agente hijo en una computadora hijo solo puede moverse a otra computadora hijo con agentes hijo, y no podrá moverse a otro nivel.

Cada agente crítico cuenta con una o más copias de respaldo. El agente

crítico y sus copias residen en distintas computadoras críticas. Las copias del agente crítico mantienen información parcial o total del agente que están respaldando. Como se muestra en la figura 3.2 cuando un agente crítico se detiene por alguna otra razón, las copias de este agente se contactan para negociar cual de ellas se convertirá en el agente sucesor. El agente elegido como el sucesor asume las funciones del agente original. Las copias del agente que se descartaron terminan o pasan a ser copias de respaldo del agente sucesor. El agente sucesor elige quienes serán sus copias de respaldo, las cuales se encontraran en diferentes computadoras críticas.

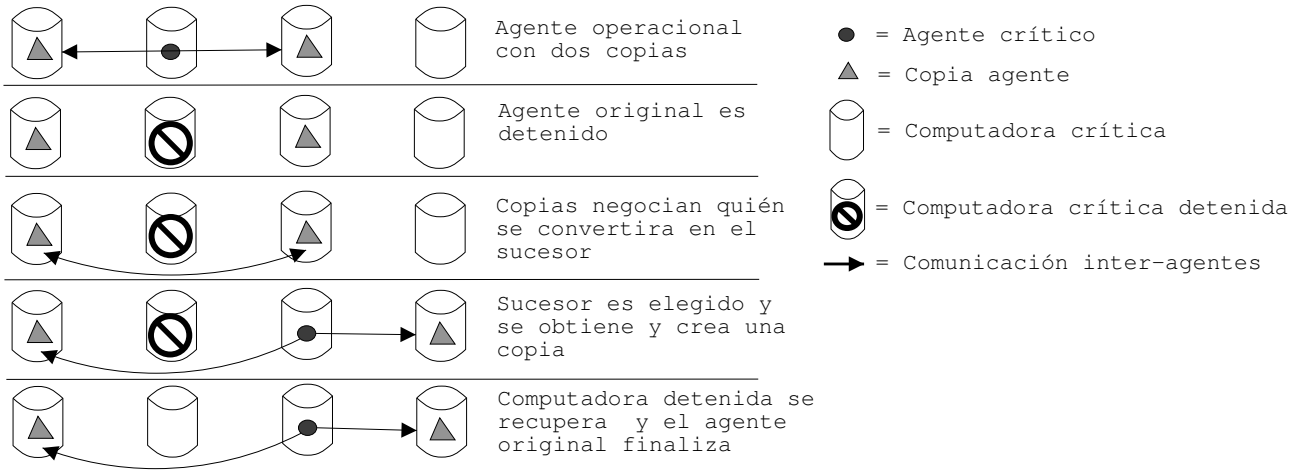

Figura 3.2: Agente móvil detenido y su respaldo mantiene su funcionalidad

Los agentes móviles tienen la característica de activarse en un tiempo determinado, cuando una computadora se da de baja por situaciones como un ataque o falla de software o hardware. El agente puede continuar sus operaciones mientras se mantiene en estado de ejecución, en otra computadora de una red. Esta característica permite eliminar un solo punto de falla y contar con tolerancia a fallas empleando varios agentes/computadoras. Cuando se presenta el caso en el que un agente/computadora primario se encuentre comprometido, se tienen al menos dos copias de este, de los cuales se hace una elección para reemplazar al primario. Una vez hecha esta elección, se decide si

la segunda copia, se termina o pasa a ser copia del nuevo agente/computadora primario elegido.

Aún cuando el enfoque anterior provee redundancia a los elementos críticos, implica un costo excesivo de implementación al emplear un recurso de cómputo por cada tipo de agente involucrado. Además presenta limitación en su desempeño, debido a que las herramientas de implementación suelen ser lenguajes interpretados, las cuales tienen un costo en el uso de memoria y CPU de la computadora.

Las herramientas de desarrollo de agentes móviles aún son inmaduras y no incorporan un modelo de seguridad para el desarrollo de soluciones basadas en estas, dando como resultado que se tengan vulnerabilidades latentes.

3.2. Alternativa a las propuestas existentes

Cada alternativa mencionada anteriormente es interesante en sí misma. Sin embargo tienen las siguientes desventajas: praire dog y agentes móviles son modelos enfocados a problemas de fallas de seguridad del SDI, la forma de trabajo que proponen implica modificar la funcionalidad o diseño actual de los SDI existentes, tienen un costo de desempeño al ejecutar varios procesos en una misma computadora o bien utilizar lenguajes interpretados para su implementación (en el caso de agentes móviles) y tienen un costo de implementación al utilizar recursos de cómputo adicionales.

Para reducir los costos de desempeño y de recursos de cómputo requeridos, podemos emplear el diseño de SDI basado en red y sensores. Aún cuando este no cuenta con la precisión que tiene el diseño de SDI basado en computadora. Los SDI basados en red son computadoras de propósito general dedicadas únicamente a la ejecución del SDI lo que tiene un beneficio en el desempeño de la computadora donde reside. Y su ubicación en distintos puntos de una red permite monitorear distintos segmentos de una red, cubriendo

así una mayor cantidad de computadoras.

De acuerdo al diseño de SDI basado en red y sensores descrito en la sección 2.5.2, los elementos que integran este diseño se representan en la figura 3.3.

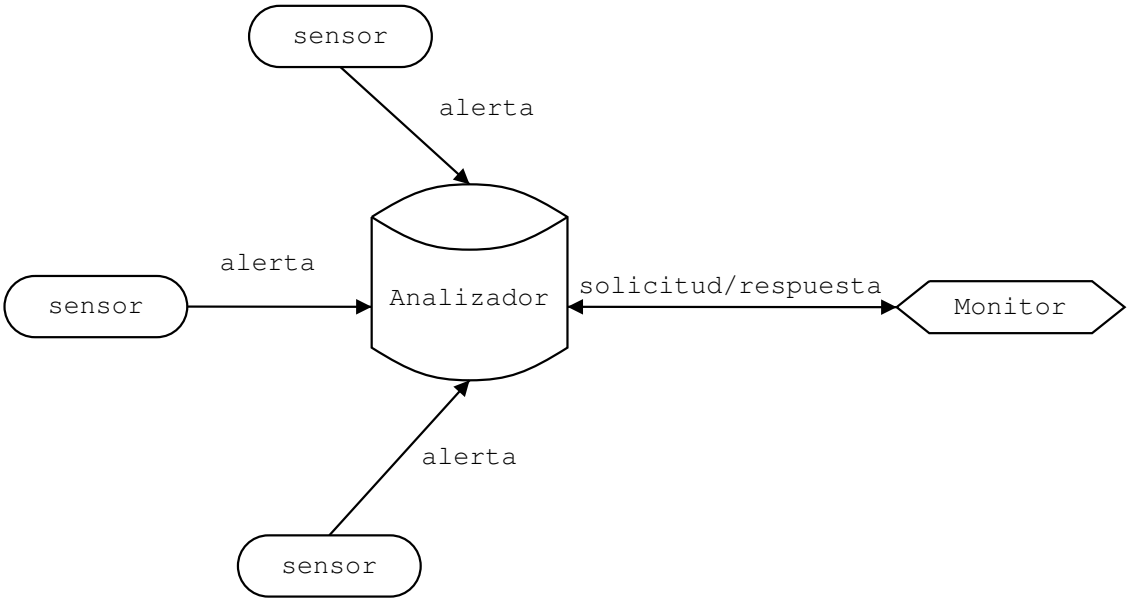

Figura 3.3: Elementos de un SDI basado en red basado en sensores básico.

Este diseño se considera semidistribuido, debido a que los sensores se encuentran repartidos en distintos segmentos de la red. Su mantenimiento es relativamente sencillo, ya que no se tienen que configurar y actualizar todas las computadoras en una red como en el caso de los SDI basados en computadora; únicamente se realizan estas tareas en las computadoras dedicadas a la ejecución en el SDI ubicadas en distintos segmentos de una red. Y da una vista global de lo que sucede en una red, al recopilar información desde distintos puntos de esta.

Idealmente, en este diseño se puede tener una replica de cada uno de los elementos (sensor, monitor y analizador) de este, para mantener la funcionalidad del SDI en caso de que cualquiera de ellos se encontrara comprometido. Adicionalmente se puede considerar un canal de comunicación seguro por

separado para el intercambio de mensajes entre los elementos del SDI, como se muestra en la Figura 3.4

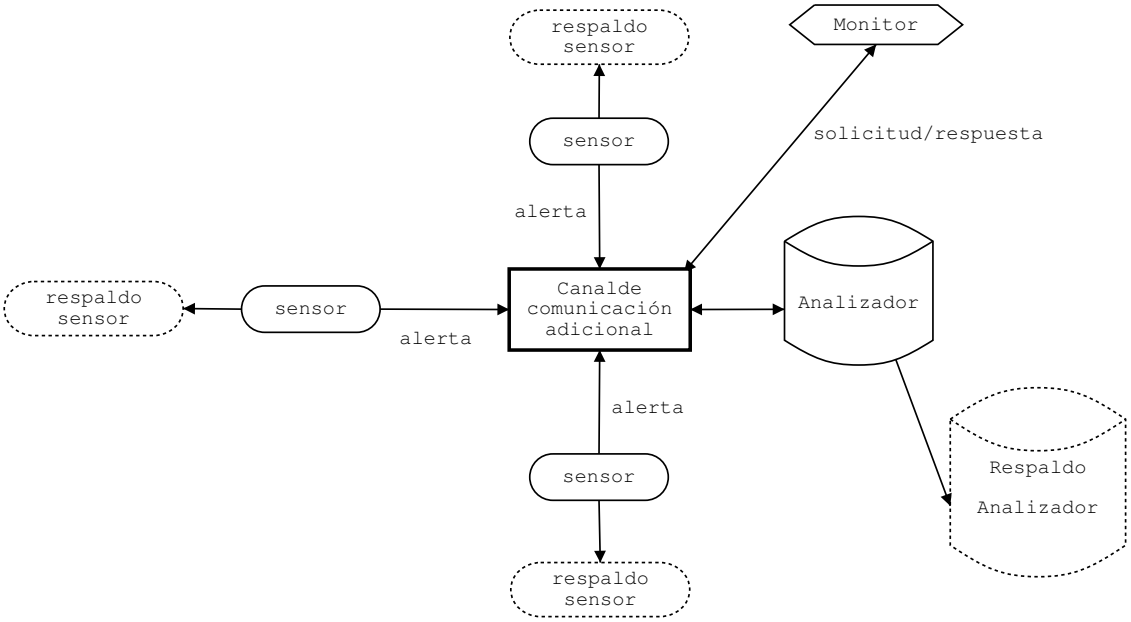

Figura 3.4: Diseño de SDI basado en red y sensores ideal.

Este diseño ofrece varias ventajas. Garantiza que cada elemento se mantenga funcional y que no se tenga problema si se presenta un ataque de inundación de información en el canal de comunicación principal, ya que el canal secundario permite enviar mensajes de alerta aún cuando el principal se encuentre saturado.

3.3. Nuestro diseño de tolerancia a fallas para un SDI

En el diseño ideal de SDI basado en red y sensores, no se modifica la funcionalidad o diseño actual de los SDI y disminuye los costos de desempeño al tener computadoras dedicadas a una sola tarea. Sin embargo, sus costos de implementación serían siendo elevados al utilizar un canal de comunicación secundario y un recurso de cómputo adicional de respaldo por cada elemento del diseño. Además su configuración y mantenimiento requiere de una mayor atención por parte del personal responsable.

Para diseñar nuestra propuesta consideramos los siguientes aspectos: no modificar la funcionalidad o diseño de los SDI existentes, mantener la funcionalidad del elemento más crítico, monitorear fallas de hardware y/o software generales, mantener información sobre el estado de elementos no considerados como críticos y bajo costo de implementación. Nuestro diseño toma como base el diseño ideal de SDI basado en red y se ajustó a nuestras consideraciones.

Vamos a tomar un SDI basado en red básico (figura 3.3). Se puede ver que en el diseño básico se pueden presentar los siguientes inconvenientes entre sus elementos, el primero de ellos representado en la figura 3.5 sucede cuando uno o varios de los sensores dejan de funcionar por fallas de hardware o software. Los sucesos que ocurren a partir del momento que se presenta la falla y mientras ésta permanece no se registran. Perdiéndose la información correspondiente al segmento de red monitoreado, lo cual limita la posibilidad de identificar una intrusión en ese segmento y consecuentemente restando datos que permitan realizar correlación de información para encontrar patrones de intrusión de manera global.

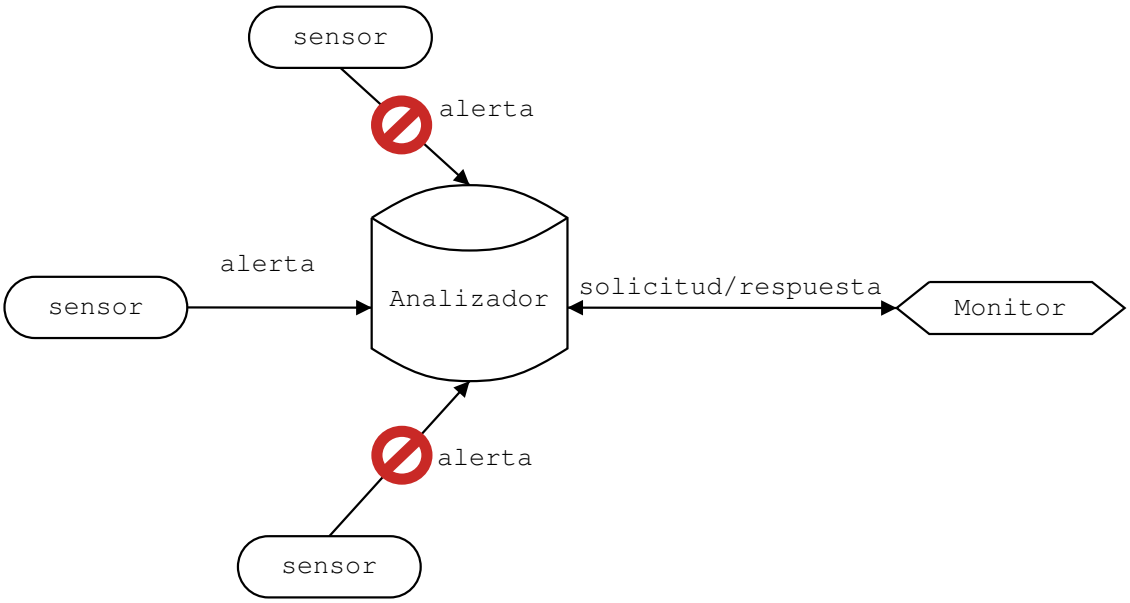

Figura 3.5: Falla de los sensores

Otro inconveniente representado en la figura 3.6, se presenta cuando el analizador falla de igual forma por razones de hardware o software. Este problema se considera más crítico que el anterior debido a que el analizador tiene, como se mencionó anteriormente la tarea de recopilar y almacenar todos los mensajes de alertas enviados por los sensores, para agrupar la información y correlacionarla de manera global a patrones de intrusión.

Para ofrecer tolerancia a las fallas mencionadas anteriormente, se planteó lo siguiente: implementar el enfoque de replicación pasiva para proveer tolerancia a fallas al analizador. Este enfoque consiste en procesar primero las actualizaciones de información en el servidor primario, para posteriormente enviarlas al secundario. Si el primario falla, el secundario se convierte en primario automáticamente.

Con este propósito el analizador primario enviará al secundario mensajes periódicamente. La interrupción de estos mensajes es señal de que el primario ha fallado.

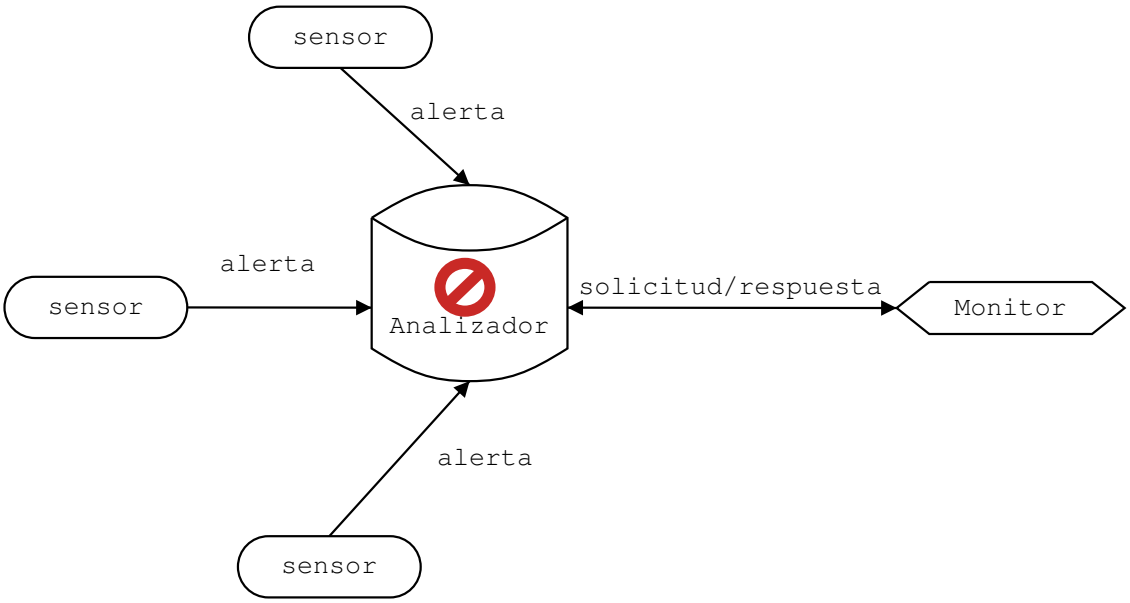

Figura 3.6: Falla del analizador

No consideramos conveniente la replicación de los sensores para no incrementar los costos de implementación excesivamente, pues se requiere de un mayor número de equipos de cómputo. Pero si consideramos justificada la replicación del analizador por ser el elemento más crítico.

Para los sensores se empleo la corroboración cruzada, es decir, la transmisión periódica de mensajes entre los sensores y hacia el analizador para verificar la disponibilidad de estos. En el caso de presentarse algún inconveniente en cualquiera de los sensores se notifica al monitor del suceso, y la respuesta se deja a decisión del personal responsable de la seguridad de la red.

No se estimaron las respuestas activas (sección 2.4) para no infringir en una negación de servicios, en caso de que la posible falla sea una falsa alarma.

Además, si el analizador falla, los sensores de manera automática se redireccionan al servidor secundario.

La figura 3.7 representa el diseño propuesto.

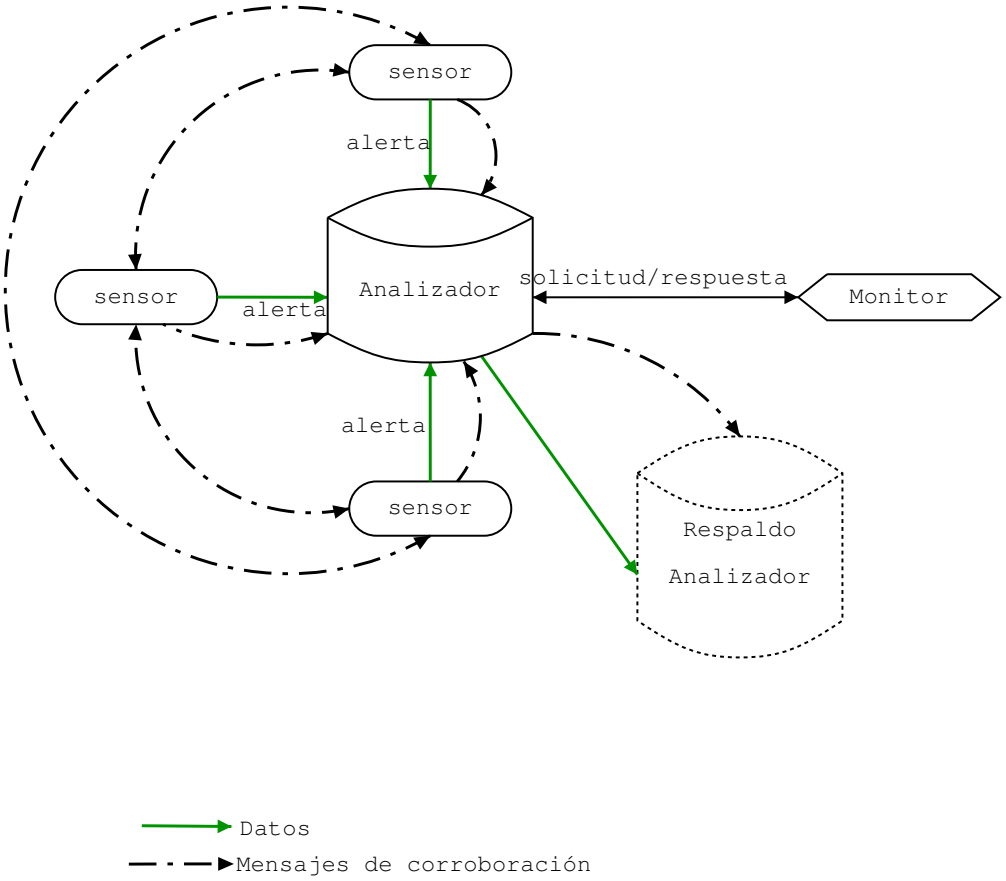

Figura 3.7: SDI basado en red y sensores básico con tolerancia a fallas

3.4. Resumen

En este capítulo presentamos los trabajos existentes acerca de tolerancia a fallas para SDIs: praire dog y agentes móviles. Ambos trabajos son solo modelos conceptuales cuyo objetivo es alertar a fallas de seguridad en un SDI. Este tipo de SDIs basados en computadora tienen la ventaja de verificar si se realizaron modificaciones y/o reemplazos a archivos de configuración del SDI o en los procesos de la computadora. Pero tienen la desventaja de utilizar gran cantidad de recursos de cómputo para mantener la funcionalidad de los elementos que lo integran.

También se presenta un diseño alternativo ideal en base a un SDI basado en red. En este diseño se tiene replicación para cada uno de los elementos que lo integran y un canal de comunicación adicional para evitar que se interrumpa la transmisión de mensajes entre los elementos de este SDI, al no depender en el canal de comunicación primario. Idealmente sería lo conveniente a emplear, pero al igual que en prairie dog y agentes móviles los costos de implementación y de mantenimiento son altos.

Finalmente planteamos nuestro diseño de tolerancia a fallas, que se deriva de un SDI basado en red y sensores ideal. Nuestro diseño cuenta con: replicación pasiva para el analizador, debido a que es el elemento más crítico por su tarea de recopilar y almacenar la información para realizar la correlación global de la información. Nuestro diseño cuenta con corroboración cruzada entre sensores, que consiste en la transmisión periódica de mensajes entre los mismos para verificar su estado. Si alguno de ellos es deshabilitado se alerta al responsable del sistema de seguridad.

Capítulo 4

Implementación de tolerancia a fallas para un SDI basado en red y sensores

En el capítulo anterior se describieron los modelos de prairie dog [34] y agentes móviles [23], los cuales proponen implementar tolerancia a fallas de seguridad en un SDI. Hasta donde hemos investigado, estos modelos son sólo modelos conceptuales y no se han implementado. También presentamos un diseño alternativo considerando un SDI basado en red agregándole características de tolerancia a fallas como: la replicación de los elementos que lo integran y un canal de comunicación adicional para la comunicación entre sus elementos. Este diseño es ideal para mantener la funcionalidad continua de un SDI, pero su costo de implementación sería relativamente alto debido a la replicación.

Por último se describió nuestro diseño de tolerancia a fallas para un SDI, el cual considera el diseño de SDI basado en red básico.

En el presente capítulo presentamos los requerimientos tanto de software como de hardware para la implementación de nuestro diseño.

Sus objetivos son: no modificar la funcionalidad o diseño de los SDI existentes, mantener la funcionalidad del elemento más crítico, monitorear fallas de hardware y/o software generales, mantener información sobre el estado de elementos no considerados como críticos y bajo costo de implementación.

Para mantener funcional al elemento más crítico, el servidor de análisis se utiliza la replicación pasiva. Para monitorear el elemento más crítico y mantener información sobre el estado de los elemento no críticos, los sensores, se utiliza la corroboración cruzada.

4.1. Aspectos de implementación de nuestro diseño tolerante a fallas

Nuestro diseño esta compuesto por los elementos de un SDI basado en red (sección 2.5.2): el servidor de análisis, los sensores y el monitor. Utiliza la replicación pasiva en el servidor de análisis por ser el elemento más crítico al recopilar y almacenar los mensajes de alertas enviados por los sensores, y la corroboración cruzada entre sensores y de los sensores hacia el servidor de análisis para verificar que su estado sigue siendo activo.

Considerando lo anterior no se modifica la funcionalidad o diseño de los SDI existentes, mantiene la funcionalidad del elemento más crítico, monitorea fallas de hardware y/o software generales y se mantiene información sobre el estado de elementos no considerados como críticos. La figura 4.1 muestra gráficamente nuestro diseño.

4.1.1. Software Propietario

Actualmente existen varios SDIs en el mercado. De los más destacados encontramos a: Shadow, Dragon, RealSecure y Network Fligth Recorder (NFR) [24]. Desafortunadamente sus costos son demasiado altos para ser adquiridos por organizaciones comerciales pequeñas, medianas o instituciones educativas. Como ejemplo, si se desea comprar una solución basada únicamente en software con diez sensores, un analizador, y el soporte por un año, tiene un costo aproximado de 77,800 dólares. A este costo hay que sumarle el costo del hardware, el cual por sus características (procesadores duales, memoria

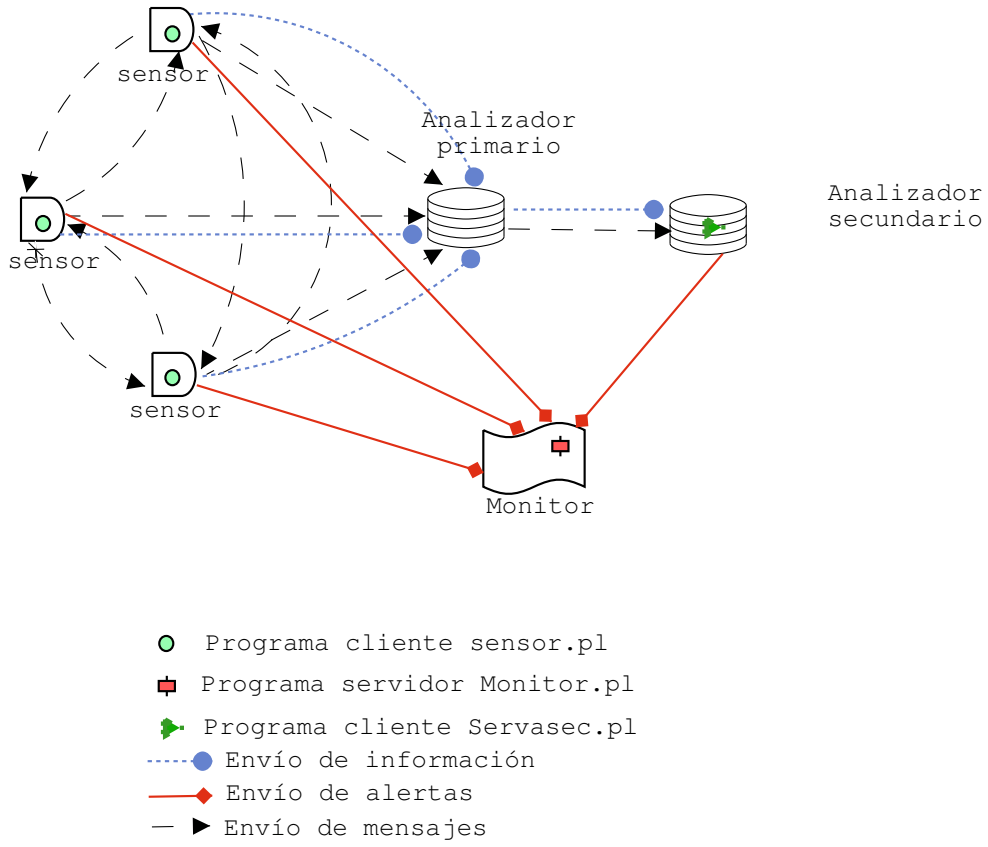

Figura 4.1: Nuestro diseño de un SDI basado en red con tolerancia a fallas.

RAM de 2GB, arreglos de discos SCSI) también resulta alto en sus costos.

4.1.2. Software Libre

Afortunadamente con el movimiento de software libre, es posible encontrar herramientas eficientes que, conjuntadas, permiten desarrollar un SDI accesible. Estas herramientas sirven como se verá, para extender la funcionalidad de SDIs y otros sistemas. A partir del año 2000 comenzó a crecer en importancia el SDI desarrollado bajo el proyecto de software libre conocido como SNORT [7]. El envío de más y más reportes de intrusiones detectados por SNORT a organizaciones como el CERT (Computer Emergency Response Team), le ha colocado como un competidor con los líderes en SDIs basados en red en

términos de efectividad. Y actualmente existen otros SDIs de software libre.

Es por esta razón que para desarrollar nuestro diseño se utilizó software libre, el cual tiene las siguientes ventajas: no se incurre en un costo para su adquisición, se tiene a disposición el código fuente de los programas (lo que da la flexibilidad de realizar modificaciones para su adaptación), se tiene soporte para distintas plataformas y se cuenta con gran número de fuentes de información a nivel mundial (páginas Web, foros de discusión, listas de suscripción, grupos de desarrollo, etc).

4.1.3. Selección Software Libre

Se empleó el sistema operativo LINUX, el cual cumple con las características mencionadas anteriormente, para realizar el desarrollo e instalación de las aplicaciones necesarias de nuestro diseño.

La aplicación principal que se debe tener para implementar nuestro diseño es un SDI. Afortunadamente actualmente existen varios SDIs de software libre. A continuación mencionamos algunos.

LIDS [15], es un SDI basado en computadora (sección 2.5.1) que reside dentro del kernel de LINUX con la finalidad de prevenir que el usuario root modifique partes importantes del sistema. Para llevar a cabo esto ofrece características de protección al sistema de archivos, protección contra acceso directo a puertos (TCP y UDP), protección contra acceso directo a memoria, protección contra acceso a disco y protección de archivos de bitácora.

PRELUDE [33], es un SDI híbrido, es decir, es un SDI que cuenta con características basado en computadora y basado en red. Esta compuesto de diferentes elementos. Un administrador recopila los mensajes y registros de los sensores y tiene soporte para manejo de bases de datos. Un SDI basado en red es un sensor que monitorea el tráfico de una red. El monitor de bitácoras el cual vigila la bitácora residente en una computadora y envía las alertas al administrador. La librería libprelude provee comunicación con el

administrador entre otras características y una interfaz en PHP para acceder los datos de almacenados en la base de datos.

SNORT [7], es un SDI basado en red (sección 2.5.2). Tiene tres usos primarios captura de paquetes, registro de paquetes, o como un SDI basado en red mismo. Cuenta con las siguientes características: análisis de tráfico en tiempo real y registro de paquetes en redes IP, análisis de protocolos en la búsqueda/coincidencia de contenidos para detectar una variedad de ataques y pruebas, por ejemplo ataques CGI (Common Gateway Interface), pruebas SMB (Secure Message Block), intentos de detección de SO, entre otros. Además utiliza un lenguaje flexible para la creación de reglas para describir el tráfico que debe colectarse o que debe pasar, y cuenta con un motor de detección que utiliza arquitectura modular.

De los SDI mencionados anteriormente, se eligió SNORT debido a que su diseño basado en red cumple con las características necesarias para desarrollar nuestro diseño.

Para almacenar la información de los sensores de SNORT en el analizador, se utilizó la base de datos MySQL [21]. En este mismo servidor también se instaló el servidor Web APACHE [4] para realizar consultas a la información almacenada desde el monitor. La aplicación ACID [10] (Analysis Console for Intrusion Database) desarrollada en PHP [36], a partir de la cual se generan reportes permitiendo realizar correlación de información de distintos sensores y finalmente el lenguaje PHP requerido por la herramienta ACID.

Se utilizaron canales de comunicación seguros entre sensores, analizador y monitor. Las aplicaciones para realizar esta tarea son: STUNNEL [32] y OPENSSL [35]. Este último también se utilizó para implementar el módulo de seguridad en el servidor Web APACHE del analizador.

Finalmente se empleó el lenguaje PERL [26] para desarrollar la aplicación que permite desempeñar el constante monitoreo de los sensores y del

analizador, además de automatizar la reconfiguración de los sensores y del servidor de respaldo. Esta aplicación se divide en tres programas:

- Programa cliente sensor.pl. Este programa se encarga del monitoreo entre sensores. Si cualquiera de ellos no responde envía una notificación al monitor. También tiene como tarea monitorear al analizador primario. Si se llegará a presentar un inconveniente en el analizador primario, se reconfiguren los sensores para que envíen los mensajes al analizador secundario.

- Programa servidor Monitor.pl. Este programa tiene la tarea de recibir todas las notificaciones tanto de los sensores y presentarlas en pantalla para notificarlo al personal encargado de la red.
  Este mensaje contiene, el nombre del sensor que lo envía, la máquina que presentó el inconveniente, y el día y la hora en que se dio el suceso.

- Programa servidor Servasec.pl. Este programa verifica que el analizador primario envíe mensajes al servidor secundario. En caso de no recibir estos mensajes, reconfigura al analizador secundario para que tome el lugar del primario.

Las características de hardware de las computadoras utilizadas para nuestro diseño son las siguientes. Para el analizador y su respaldo se utilizaron computadoras con procesador pentium IV a 2.4 Ghz con 120 Gb en disco duro y 768 Mb de RAM. En los sensores se emplearon tres computadoras: una computadora con procesador celeron a 800 Mhz con 20 Gb y 256 Mb de RAM, una computadora con procesador AMD Duron a 1.2 Ghz con disco duros de 20 Gb y 256 Mb de RAM y una computadora con procesador pentium III a 800 Mhz con disco duro de 20 Gb y 256 Mb de RAM.

4.2. Configuración de nuestro diseño de SDI con tolerancia a fallas

Antes de proveer las características de tolerancia a fallas, se requiere que estén en funcionamiento los componentes del SDI. Para que entren en funcionamiento se debe tener una ubicación de estos. Como se menciono en la Sección 2.5.2, en un diseño de SDI basado en red y sensores se tienen varias opciones donde colocar los sensores con diferentes ventajas. La ubicación de los sensores y del analizador de nuestro SDI se muestra en la figura 4.2.

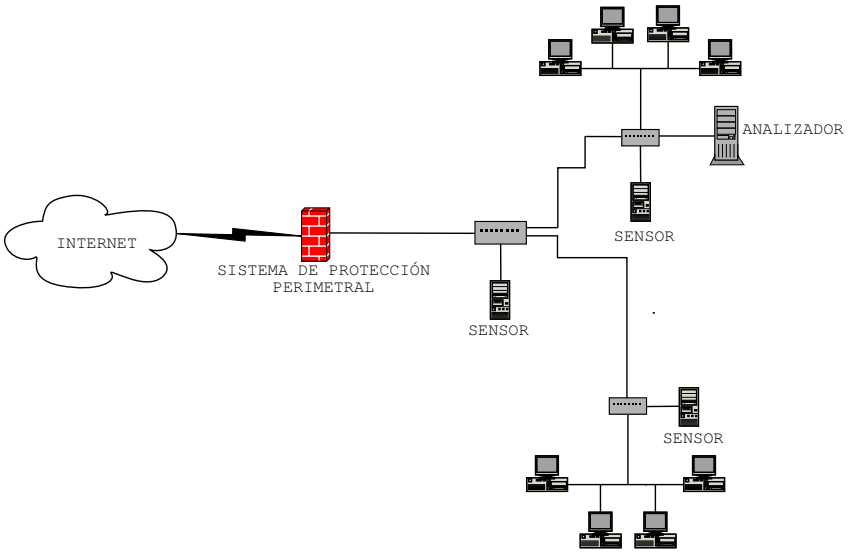

Figura 4.2: Ubicación del SDI con tolerancia a fallas propuesto.

La ubicación del sensor detrás del sistema de protección perimetral permite generar alertas de ataques legítimos y destacar los problemas que se tengan con las políticas de seguridad del sistema de protección perimetral. De esta forma se producen relativamente pocas falsas alertas o falsos positivos. Los sensores dentro de las dos subredes permite enfocarse a los recursos considerados de gran valor, detectando ataques a los recursos y sistemas críticos. También se provee en ambas ubicaciones de información valiosa para realizar forensia de datos en el caso de que un sistema se encuentre comprometido.

El crecimiento del SDI dependerá de si la política de seguridad contempla un mayor número de segmentos a monitorear y de la inversión disponible para esta tarea.

Una vez ubicados los componentes de nuestro SDI, se inicia la tarea de configuración de los elementos del SDI.

4.2.1. Analizador

El analizador requiere de una base de datos en MySQL que almacena la información transmitida por los sensores, el túnel de encriptación STUNNEL para la comunicación con los sensores, la configuración de TCP WRAPPER para restringir el acceso solamente a las computadoras asignadas, el servidor Web APACHE con soporte para conexiones seguras,el lenguaje PHP para utilizar la herramienta ACID que permite la correlación de la información en la base de datos, y PERL para ejecutar la aplicación de intercambio de mensaje con el analizador secundario.

A continuación se mencionan los pasos a seguir en este proceso.

Configuración de MySQL :

1. Instalación de las aplicaciones MySQL servidor y MySQL cliente versión 3.23.52.
2. Iniciar demonio de MySQL y actualización de la información de los niveles en los que se debe ejecutar esta aplicación.
3. Creación de la contraseña para el usuario con privilegios de administrador en la base de datos.
4. Conexión a la base de datos para verificar la contraseña de administrador.
5. Se crea un usuario de la base de datos sin privilegios de administrador, para envío de información de los sensores.

6. Se crea la base de datos.
7. Se crea la estructura de la base de datos, mediante el archivo por lotes proporcionado con la distribución de SNORT.
8. Verificación de la estructura de la base de datos.

Configuración STUNNEL servidor :

1. Instalación de OpenSSL y STUNNEL 3.4.
2. Generar archivo pem, que permite generar llaves y examinar certificados.
3. Editar archivo de configuración stunnel.conf para redireccionar el puerto de MySQL, al puerto con el túnel.
4. Se inicia el demonio.

Configuración TCPWRAPPER :

1. Editar archivo services para agregar el puerto y nombre de servicio de MySQL vía el túnel.
2. Editar archivo hosts.allow en el directorio etc para agregar únicamente las computadoras que van a utilizar este servicio.
3. Editar archivo hosts.deny en el directorio etc para restringir el acceso al servicio a todas las computadoras con excepción a las definidas en hosts.allow.

Configuración servidor Web APACHE :

1. Modificar el archivo httpd.conf (o archivo ssl.conf en caso de existir) para agregar el módulo para soporte de SSL, agregar el puerto en el cual va a escuchar las peticiones, en el caso que se vaya a utilizar un puerto diferente al 80 y se agregan las opciones de SSL como la llave privada, el certificado entre otras.

2. Se generan las llaves para generar el certificado que permita el emplear el mod_ssl.
3. Se inicia el demonio del servidor web.

Configuración de ACID :

1. Se descompacta el archivo acid-versión.tar.gz, ubicado en el directorio contrib del código fuente de snort.
2. Copiarlos al directorio /var/www/html.
3. Editar el archivo acid_conf.php, para agregar información referente al tipo de base de datos que se va a utilizar, el nombre de la base de datos, el nombre del usuario de la base de datos y su contraseña.

Configuración PERL :

1. Instalar perl
2. Verificar que se tiene el módulo IO, en caso de no ser así, adicionarlo.

4.2.2. Sensores

En los sensores se requiere de SNORT para realizar la captura y análisis de paquetes, el compilador de PERL para ejecutar el programa de monitoreo entre ellos y hacia el analizador, STUNNEL para el envío en forma segura de alertas que serán recopilas en la base de datos del analizador y de mensajes entre sensores y hacia el analizador y TCPWRAPPER para restringir a todas aquellas computadoras con las que no se debe mantener intercambio de información.

Configuración de SNORT :

1. Instalar el programa SNORT con soporte para MySQL
2. Editar el archivo de configuración snort.conf para configurar el nombre de la base de datos, el usuario, la contraseña y dirección donde se encuentra la base de datos a la que se enviaran las alertas.
3. Agregar en el archivo snort.conf el nombre del archivo de reglas que tiene como función el que se ignoren el intercambio de paquetes entre sensores, y hacia el analizador.
4. Se crea un usuario para ejecutar SNORT sin privilegios de root
5. Cambiar dueño y grupo al archivo snort.conf, con los atributos del usuario creado.
6. Crear archivo de inicio de SNORT.
7. Se iniciar después de iniciar el STUNNEL.

Configuración PERL :

1. Instalar PERL
2. Verificar que se tiene el módulo IO y NET, en caso de no ser así adicionarlos.

Configuración STUNNEL cliente :

1. Instalar OpenSSL y STUNNEL 3.2.
2. Crear usuario sin privilegios de root para ejecutar STUNNEL
3. Se crea el shell que inicie STUNNEL cada vez que se reinicie la máquina, y se ponen los parámetros para direccionar al analizador y el puerto de MySQL al puerto con el túnel.
4. Se inicia STUNNEL.

Configuración TCPWRAPPER :

1. Editar archivo hosts.allow en el directorio etc para agregar únicamente las computadoras con las que se tendrá comunicación y en que puerto.
2. Editar archivo hosts.deny en el directorio etc para restringir el acceso al servicio a todas las computadoras con excepción a las definidas en hosts.allow.

4.2.3. Monitor

En el monitor se requiere de PERL para recibir y mostrar las ventanas de notificación de falla y de un Navegador como: Mozilla, Netscape u Opera para acceder al analizador vía Web para realizar consultas a la base de datos a través de la herramienta de análisis de la base de datos.

Configuración PERL :

1. Instalar PERL.
2. Verificar que se tiene el módulo IO y Tk, en caso de no ser así adicionarlos.

Configuración Navegador :

1. Bajar el archivo binario del navegador de preferencia e instalarlo.

4.2.4. Replicación del analizador

Una vez configurados y en funcionamiento los componentes del SDI, se provee la característica de replicación pasiva al analizador primario, para mantener su funcionalidad. Este tipo de replicación consiste en enviar primero las actualizaciones al servidor primario y posteriormente enviarlas al secundario.

Para efectuar esto se requirió configurar otra computadora con las mismas particularidades que el analizador primario mencionadas en la sección 4.3.1. A esta otra computadora la identificaremos como analizador secundario. Los pasos para configurar los analizadores primario y secundario se describen a continuación. En nuestra implementación estos los realizamos automáticamente mediante un shell script cuyo código se muestra en el apéndice A en la página 71.

Configuración del analizador primario :

1. Crear un usuario con permisos file en MySQL.
2. Mostrar el estatus para tomar parámetros file y position que se utilizan en la configuración del servidor secundario.
3. Generar una imagen de la base de datos.
4. Verificar en el archivo de configuración my.cnf que se tenga el parámetro de bitácoras binarias y el parámetro que da un identificador al servidor.
5. Reiniciar MySQL después del analizador secundario para conexión y sincronización.

Configuración del analizador secundario :

1. Reproducir la imagen de la base de datos generada en el analizador primario.
2. Ingresar en MySQL los datos del analizador primario como IP, usuario (permisos file) y contraseña y los parámetros position y file tomados al mostrar el estatus del analizador primario para sincronización de información.
3. Modificar el archivo de configuración my.cnf con los parámetros del analizador primario.
4. Reiniciar MySQL.

4.2.5. Corroboración cruzada

La corroboración cruzada entre sensores tiene como finalidad verificar la disponibilidad de los sensores o del analizador primario. En caso de que no estén disponibles se envía una notificación al monitor. Periódicamente, cada diez segundos, se envían mensajes entre cada uno de los sensores y de estos hacia el analizador primario, esperando hasta treinta segundos por una respuesta. Si la respuesta de uno de ellos no se recibe después de treinta segundos, se decide que el equipo está inactivo. Posterior a este tiempo si un equipo llega a recuperarse, se ignora como equipo activo (siguiendo las política que manejan los ruteadores).

En el caso de que el analizador sea la entidad que no responde, se realiza el redireccionamiento del envío de mensajes de los sensores al analizador secundario. En caso contrario, si la entidad que no responde es un sensor, entonces se envían hasta tres notificaciones al monitor. El pseudocódigo de la corroboración de sensores se muestra en la página 51.

La corroboración cruzada del analizador primario al secundario tiene como finalidad verificar la disponibilidad del analizador primario. Periódicamente, cada diez segundos, se envían mensajes del analizador primario al secundario. La interrupción de estos mensajes en un lapso de tiempo de treinta segundos es señal de que el analizador primario ha fallado; es entonces cuando el servidor secundario activa los servicios Web y STUNNEL (para reenviar los mensajes a MySQL) y envía hasta tres notificaciones al monitor. El pseudocódigo de la corroboración de analizadores se muestra en la página 52.

Corroboración de sensores :

```
Inicio
  Poner contador = 1
  Poner servidor = Dirección IP
  Poner aviso = 0
  Mientras sea verdadero hacer
    Poner p = Nuevo objeto ping (tcp)
    Abrir Archivo Ip_maquinas
      Mientras maquina = Lee archivo Ip_maquinas hacer
        Pausar 10 segundos
        Si p - > ping(maquina,expiración) entonces
          Imprime Máquina activa
        Si_no
          Si (maquina == servidor) y (aviso== 0) entonces
            aviso=1
            Copia stunnel2 a stunnel
            Reinicia stunnel
          Si_no
              Si contador < 4 entonces
                Incrementar contador
                Poner sock = Nuevo socket cliente
                Enviar a socket "Nombre_sensor: Alerta dir_ip: maquina"
                Limpiar sock
                Cerrar sock
              Si_no
                Imprime ya lo advirtió
                contador = 1
              Fin_si
          Fin_si
        Fin_si
        Cerrar p
      Fin_Mientras
  Fin_Mientras
Fin
```

Corroboración analizador primario-secundario :

Inicio
Poner Maxlen = 1024
Poner servidor = Dirección IP
Poner sock = Nuevo socket servidor
Mientras nuevo_sock = sock acepta conexión **hacer**
Poner pid = Crear proceso
Si pid == 0 **entonces**
Señal alarma = Expiración
Evaluar
Alarma 30 segundos
nuevo_sock -> recibe(msg,Maxlen)
Imprime Se recibió el Mensaje: msg
Fin_Evaluar
Si (señal alarma == Expiración) **entonces**
Imprime Expiro tiempo de espera
Inicia stunnel
Inicia httpd
Repetir desde contador=1 **hasta** contador < 4
Poner sock = Nuevo socket cliente
Envia a socket "Nombre_sensor: Alerta dir_ip: servidor"
Poner sock = Nuevo socket
Enviar a socket "Nombre_sensor: Alerta dir_ip: maquina"
Limpiar sock
Cerrar sock
Fin_repetir
Salir
Fin_si
Fin_si
Fin_Mientras
Cierra sock
Fin

4.3. Resumen

En este capítulo se presentaron los requerimientos de software y características de hardware para los componentes de nuestro diseño de un SDI basado en red con tolerancia a fallas.

Nuestro diseño está compuesto por un analizador, sensores y un monitor. El analizador almacena y recopila las alertas de los sensores en el manejador de base de datos MySQL. La consulta de esta información se realiza vía Web utilizando una aplicación conocida como ACID, lo que requiere de un servidor web APACHE y el lenguaje PHP. Al ser considerado el elemento más crítico se requiere de mantener su funcionalidad mediante la replicación de información a un servidor secundario.

Los sensores utilizan SNORT para analizar el tráfico de la red y encontrar coincidencia con reglas de intrusión predefinidas, cuando existen coincidencias se envían mensajes de alerta al analizador. El continuo análisis del tráfico de una red requiere que siempre estén activos. Para informar si continúan en este estado se efectúa la corroboración cruzada entre los sensores.

La comunicación entre el analizador y sensores requiere de canales de comunicación seguros. La implementación de estos se realizó mediante STUNNEL. La restricción de comunicación únicamente entre sensores y de estos al analizador se realiza mediante TCP WRAPPER

El monitor utiliza un Navegador como: Mozilla, Opera o Netscape, para acceder al analizador via web y realizar consultas a la base de datos del analizador.

Finalmente, se utilizo PERL para desarrollar la aplicación que permite efectuar la corroboración y notificación al monitor de las fallas que se presenten en los sensores o analizador.

Capítulo 5

Evaluación

En el capítulo anterior presentamos los requerimientos para implementar, con software libre nuestro diseño de un SDI basado en red con tolerancia a fallas. Este tipo de SDIs consta de tres componentes: analizador, sensor y monitor. El analizador recopila y almacena los mensajes de alerta enviados por los sensores, actividad por la cual es considerado el elemento más crítico. Utilizamos la replicación pasiva para mantener su funcionalidad.

En cada sensor se capturan y analizan los paquetes que transitan en un segmento de una red. La continuidad de esta actividad es importante para mantener la precisión en la correlación de información. Para verificar si los sensores continúan activos o no, se efectúa la corroboración cruzada.

En el monitor se realizan las consultas al analizador; además se reciben los mensajes de notificación en el caso de que en cualquiera de los sensores se detecte un inconveniente.

En este capítulo presentamos una evaluación de nuestro diseño, la cual se enfoca a dos aspectos: desempeño y correctitud, como se describe a continuación.

5.1. Método

Se recordará que nuestro diseño de tolerancia a fallas para un SDI basado en red (sección 4.2 y 4.3) realiza:

1. Corroboración cruzada entre sensores.
2. Notificación de falla de comunicación de un sensor.
3. Corroboración del analizador primario al secundario.
4. Transición del analizador primario al secundario.

Nuestro diseño trabaja de la siguiente manera. Los sensores emplean la corroboración cruzada, mediante la transmisión periódica de mensajes entre ellos para comprobar que siguen activos y no presentan fallas (punto 1). En caso de presentarse una contingencia en cualquiera de los sensores se notifica al monitor del suceso (punto 2), y la respuesta se deja a criterio del personal responsable de la seguridad de la red. Si el analizador primario falla, los sensores se redireccionan automáticamente al secundario.

En el analizador se empleó la replicación pasiva, en la cual se tienen dos servidores, el primario, y el secundario. Las actualizaciones son procesadas por el servidor primario en primer lugar, y después son enviadas hacia el secundario. El analizador primario además envía mensajes periódicamente al secundario; la interrupción de estos mensajes es señal de que el primario ha fallado (punto 3). Si el primario falla, el secundario se convierte en primario automáticamente (punto 4).

La evaluación de nuestro diseño la hemos enfocado a:

- La medición de tráfico de paquetes en la red, debido a que nuestro diseño de tolerancia a fallas genera mensajes en intervalos de tiempo para verificar que el estado tanto de los sensores como del analizador sigue activo.

- Comprobar que el mensaje de notificación de falla de comunicación llega al monitor en el momento que se desactiva cualquiera de los sensores o el analizador.

- Medir la perdida de información debido a la transición del analizador primario al secundario.

Como se recordará, las características de hardware de las computadora utilizadas para nuestro diseño son las siguientes: para analizador y su respaldo se utilizaron computadoras con procesador pentium IV a 2.4 Ghz con 120 Gb en disco duro y 768 Mb de RAM. En los sensores se emplearon tres computadoras: una computadora con procesador celeron a 800 Mhz con 20 Gb y 256 Mb de RAM, una computadora con procesador AMD Duron a 1.2 Ghz con disco duros de 20 Gb y 256 Mb de RAM y una computadora con procesador pentium III a 800 Mhz con disco duro de 20 Gb y 256 Mb de RAM.

5.2. Medición del tráfico de paquetes

En la medición del *tráfico de paquetes*, se capturó tráfico de una red durante dos semanas únicamente los días hábiles, una hora diaria entre las 13:00 horas y las 14:00 horas. Se utilizó el programa SNORT para efectuar esta tarea. Se capturaron dos tipos de paquetes por separado. Un tipo corresponde a los paquetes enviados por los sensores; el otro tipo corresponde al tráfico usual de una red. En la captura de tráfico usual de una red, se consideraron únicamente en paquetes del protocolo TCP tanto de entrada como de salida.

Se decidió utilizar el protocolo TCP para enviar los mensajes entre sensores por tener este protocolo menos restricción en los sistemas de protección perimetral.

La figura 5.1 muestra el porcentaje de sobrecarga al tráfico de una red.

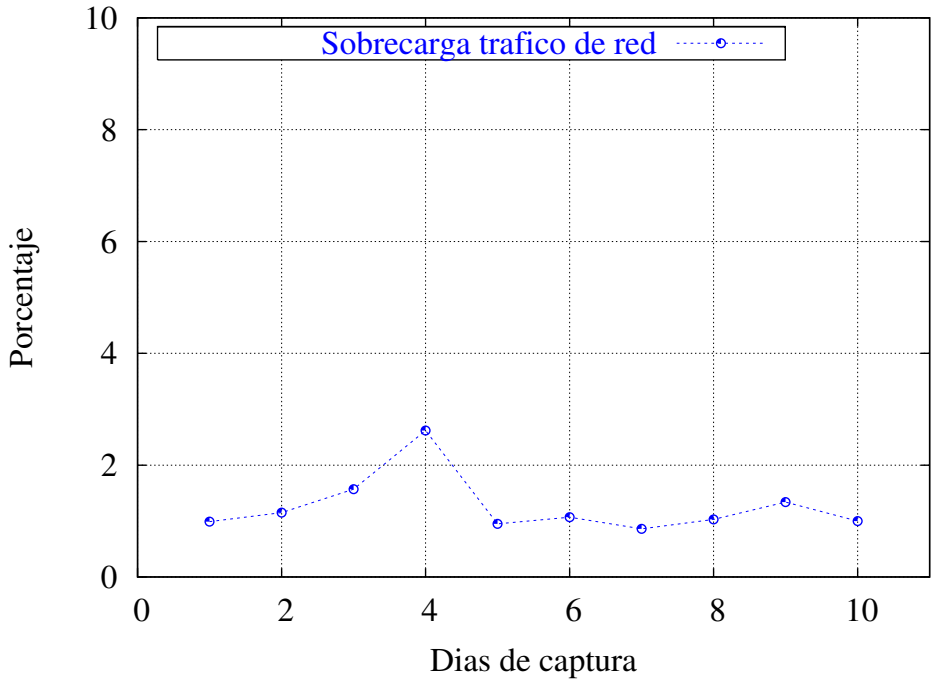

Figura 5.1: Porcentaje de sobrecarga de tráfico.

La gráfica anterior muestra que nuestro diseño incrementa el tráfico usual en una red en aproximadamente un 1.14 %. Este porcentaje resulta de sumar el número de paquetes generados por los sensores, un total de 29,900 paquetes; multiplicar esta suma por cien y dividirla entre el total de tráfico usual, un total de 2,619,322 paquetes.

Claro, el porcentaje anterior es relativo. Si la captura de paquetes se realizara en horarios no laborales, en los que no se tiene mucha actividad en una red, el número de paquetes generados por los sensores, representaría un porcentaje mayor del total del tráfico usual en horario no laboral.

Por lo anterior, se puede considerar que nuestro diseño, en general, no provocará un detrimento notable en el desempeño de una red.

5.3. Notificación de falla de comunicación

Para comprobar que el *mensaje de notificación de falla de comunicación* es recibido por el monitor en el momento que se presenta un inconveniente en alguno de los sensores, se desactivó la conexión de red en forma aleatoria e intencional a uno de los sensores, simulando así el caso en el que ya no se tiene ninguna respuesta por parte de este elemento.

Posteriormente se deshabilito la conexión del analizador para verificar que también se recibía la notificación en el monitor (además de realizarse la transición al servidor secundario).

En ambos casos se envío el mensaje de notificación al monitor. En la figura 5.2 se muestra el mensaje desplegado.

Esta actividad se repitió durante dos semana, realizándolo cinco veces en cada día y en cada ocasión se envío el mensaje de notificación.

Figura 5.2: Mensaje de notificación en el monitor.

5.4. Medición de la pérdida de información

Para medir la *perdida de información* durante la transición del analizador primario al secundario, Utilizamos otra computadora que también funciona como analizador primario. La llamaremos analizador primario B. Además, en cada sensor corrimos otro proceso SNORT (normalmente solo corre uno). El proceso SNORT adicional reporta al analizador primario B, el cual, en nuestras pruebas, nunca deja de funcionar, y por lo tanto tiene información completa de los mensajes de alerta de los sensores. Entonces desconectamos el analizador primario (real). De acuerdo con nuestro diseño, el analizador secundario se activo y los sensores (proceso SNORT normal) se direccionaron al mismo.

Una vez que finalizó este proceso, se realizo una consulta SQL, tanto a la base de datos del analizador primario B como a la del analizador secundario, a la tabla que registra los eventos y a la tabla que guarda la identificación de los sensores.

La finalidad de la consulta realizada en ambas bases de datos, es comparar los datos de los sensores y tiempo registrados. En la consulta se seleccionó el identificador del sensor, la firma y la fecha y hora de registro del mensaje, tomando como restricción el rango de tiempo a partir de un minuto antes de la hora que se desactivo el analizador primario hasta dos minutos después de que el servidor secundario entro en operación.

A continuación se muestran los resultados a las consultas efectuadas en ambos servidores.

En la consulta a la base de datos del analizador primario B se obtuvo lo siguiente:

hostname	cid	signature	timestamp
sensor 1	1	1	2004-02-17 17:19:34
sensor 3	2	1	2004-02-17 17:19:37
sensor 1	3	1	2004-02-17 17:19:40
sensor 2	5	2	2004-02-17 17:20:44
sensor 2	4	2	2004-02-17 17:20:44
sensor 1	7	2	2004-02-17 17:20:45
sensor 2	6	2	2004-02-17 17:20:45
sensor 1	9	3	2004-02-17 17:20:55
sensor 3	8	3	2004-02-17 17:20:55
sensor 1	11	2	2004-02-17 17:21:20
sensor 2	10	2	2004-02-17 17:21:20
sensor 2	16	2	2004-02-17 17:21:26
sensor 3	15	2	2004-02-17 17:21:26
sensor 2	14	2	2004-02-17 17:21:26
sensor 1	13	2	2004-02-17 17:21:26
sensor 3	12	4	2004-02-17 17:21:26
sensor 1	18	2	2004-02-17 17:21:27
sensor 1	17	2	2004-02-17 17:21:27
sensor 2	19	2	2004-02-17 17:21:37

En la consulta a la base de datos del analizador secundario se obtuvo lo siguiente:

```
+----------+-----+-----------+---------------------+
| hostname | cid | signature | timestamp           |
+----------+-----+-----------+---------------------+
| sensor 1 |   1 |         1 | 2004-02-17 17:19:34 |
| sensor 3 |   2 |         1 | 2004-02-17 17:19:37 |
| sensor 1 |   3 |         1 | 2004-02-17 17:19:40 |
| sensor 2 |   5 |         2 | 2004-02-17 17:20:44 |
| sensor 2 |   4 |         2 | 2004-02-17 17:20:44 |
| sensor 1 |   7 |         2 | 2004-02-17 17:21:16 |
| sensor 2 |   6 |         2 | 2004-02-17 17:21:16 |
| sensor 1 |   9 |         3 | 2004-02-17 17:21:26 |
| sensor 3 |   8 |         3 | 2004-02-17 17:21:26 |
| sensor 1 |  11 |         2 | 2004-02-17 17:21:51 |
| sensor 2 |  10 |         2 | 2004-02-17 17:21:51 |
| sensor 2 |  16 |         2 | 2004-02-17 17:21:57 |
| sensor 3 |  15 |         2 | 2004-02-17 17:21:57 |
| sensor 2 |  14 |         2 | 2004-02-17 17:21:57 |
| sensor 1 |  13 |         2 | 2004-02-17 17:21:57 |
| sensor 3 |  12 |         4 | 2004-02-17 17:21:57 |
| sensor 1 |  18 |         2 | 2004-02-17 17:21:58 |
| sensor 1 |  17 |         2 | 2004-02-17 17:21:58 |
| sensor 2 |  19 |         2 | 2004-02-17 17:22:08 |
+----------+-----+-----------+---------------------+
```

Los resultados de las consultas anteriores muestran en ambas consultas los mismos tiempos hasta las 17:20:44 (tiempo de inicio de la transición del servidor análisis primario al secundario). Después de ese tiempo, en ambas consultas se presentan los mismos identificadores de sensores y firmas, diferenciandose únicamente los tiempos de registro de estos eventos. Por lo que no existe perdida de información, solo un retraso en el registro de los eventos, correspondiente al tiempo que toma el redireccionamiento de los sensores y la transición del analizador primario al secundario. Los sensores

al no recibir reconocimiento del primario, correspondiente al último mensaje enviado, reenvían la información del mismo al secundario.

5.5. Resumen

En este capítulo se evalúo el desempeño de nuestro diseño de tolerancia a fallas en un SDI basado en red.

Las características evaluadas fueron las siguientes:

1. La medición de tráfico de paquetes en la red, debido a que se generan mensajes en intervalos de tiempo para verificar el estado activo tanto de los sensores como del analizador. Se observó un incremento mínimo en el tráfico de la red, lo cual no disminuye su operabilidad.

2. La notificación de falla de comunicación enviada al monitor en el momento en que se presenta un inconveniente en cualquiera de los sensores o en el analizador primario. Para ambos casos se presento la notificación sin percance.

3. Finalmente se midió la pérdida de información en la transición del analizador primario al secundario. Efectuando para esta evaluación consultas a las bases de datos tanto del analizador primario B como a la del analizador secundario. No se tiene pérdida de información, solo un retraso en el registro de los eventos.

Capítulo 6

Conclusiones y Trabajo Futuro

Esta tesis presentó el diseño de tolerancia a fallas para un SDI basado en red.

Estos SDIs analizan los paquetes que transitan en una red y se componen de un servidor de análisis, una computadora monitor y una o varias computadoras que actúan como sensores.

Las propuestas de tolerancia a fallas para SDIs, como prairie dog y agentes móviles, se enfocan a fallas correspondientes a la seguridad de la computadora y son solo modelos que no se han implementado.

Nuestro diseño de tolerancia a fallas para un SDI se caracteriza por ser fácil de integrar a los SDIs existentes, por estar basado en software libre y portable a las plataformas más usadas (LINUX, WINDOWS). Duplica la funcionalidad del elemento más crítico, el analizador, sin perdida de información en caso de que uno de los dos servidores falle. Monitorea fallas de hardware y/o software generales, manteniendo información sobre el estado de los elementos considerados como no críticos mediante la corroboración cruzada. Además incurre un bajo costo de implementación.

Nuestro diseño utiliza la corroboración cruzada y la replicación pasiva. La corroboración cruzada tiene como objetivo transmitir periódicamente mensajes para verificar la disponibilidad de cualquiera de los sensores y del servidor de análisis primario y notificar, en caso de presentarse una falla, al

monitor mediante un mensaje de alerta.

La replicación pasiva del servidor de análisis tiene como función principal el mantener disponible un repositorio para los mensajes de los sensores. Se efectúa procesando las actualizaciones primero en el servidor primario y posteriormente enviándolas al servidor secundario.

Nuestro diseño se caracteriza además por utilizar software libre lo que ofrece flexibilidad de modificación y adaptación al estar disponible el código fuente. Otras ventajas importantes de utilizar software libre es que: tiene soporte para distintas plataformas, no se incurre en un costo para su adquisición, cuenta con distintas fuentes de documentación como: páginas web, foros de discusión, listas de suscripción, etc.

Nuestro diseño está compuesto por un servidor de análisis primario, un servidor de análisis secundario, varios sensores y un monitor.

Tanto el servidor de análisis primario como el servidor de análisis secundario requieren de las siguientes aplicaciones: la base de datos en MySQL que almacena la información transmitida por los sensores, el túnel de encriptación STUNNEL para la comunicación con los sensores, la configuración de TCP WRAPPER para restringir el acceso solamente a las computadoras asignadas, el servidor Web APACHE con soporte para conexiones seguras, el lenguaje PHP para utilizar la herramienta ACID que permite la correlación de la información en la base de datos, y PERL para ejecutar la aplicación de intercambio de mensaje con el servidor de análisis secundario.

Los sensores requieren de las siguientes aplicaciones: SNORT para realizar la captura y análisis de paquetes, PERL para ejecutar el programa de monitoreo entre sensores y hacia el servidor de análisis, STUNNEL para el envío en forma segura de alertas que serán recopilas en la base de datos del servidor de análisis y de mensajes entre sensores y hacia el servidor de análisis. Finalmente se requiere TCPWRAPPER para restringir a todas aquellas computadoras con las que no se debe mantener intercambio de información.

El monitor requiere de PERL para recibir y mostrar las ventanas de notificación de falla y de un Navegador como: Mozilla, Netscape u Opera para accesar al servidor de análisis vía Web para realizar consultas a la base de datos mediante la herramienta de análisis ACID.

La corroboración cruzada se anexa mediante una aplicación en PERL (lo cual ofrece portabilidad), que ejecuta el constante monitoreo de la disponibilidad de los sensores y del servidor de análisis; además de automatizar la reconfiguración de los sensores y del servidor de respaldo. Esta aplicación se divide en tres partes con la siguiente finalidad:

- El programa ubicado en el sensor se encarga del monitoreo entre sensores; si cualquiera de estos no responde, entonces envía una notificación al monitor. También monitorea al servidor de análisis primario, y en el caso que se presentara un inconveniente en este, automáticamente se reconfiguran los sensores para enviar los mensajes al servidor de análisis secundario.

- El programa ubicado en el monitor tiene la tarea de recibir todas las notificaciones tanto de los sensores como del servidor de análisis secundario y desplegarlas en pantalla para notificación del personal encargado de la red.
  En este mensaje se incluye, el nombre del sensor que lo envía, la máquina que presentó la falta de disponibilidad, y el día y la hora que sucedió la falla.

- El programa en el servidor secundario ejecuta el monitoreo recibiendo mensajes del servidor primario. En el caso de que no se reciban más mensajes del servidor primario, se reconfigura al servidor secundario activando los procesos de servidor Web y Stunnel para la comunicación y recopilación de mensajes de los sensores.

Utilizando la corroboración cruzada se tiene una incremento mínimo

tráfico de la red, lo cual no produce un detrimento en el desempeño de una red.

En la replicación pasiva entre el servidor de análisis primario y secundario, se tiene una imagen de la base de datos del servidor primario en el secundario y las actualizaciones de los mensajes enviados por los sensores son procesadas primero por el servidor de análisis primario y posteriormente enviadas al servidor de análisis secundario.

Al establecerse la transición del servidor de análisis primario al secundario, no se tiene pérdida de información, solo se tiene un retraso en el registro de los eventos.

6.1. Limitaciones y Trabajo Futuro

Nuestro diseño de tolerancia a fallas se puede mejorar:

- Para proveer de mayor tolerancia a fallas al servidor de análisis, se podrían utilizar varios de tales servidores, fraccionar la información por segmentos de red y desarrollar una aplicación que permita colectar la información de distintas fuentes como si fuera una unidad. La tolerancia a fallas se obtendría al particionar la información de cada segmento sobre varios servidores de análisis, pero más de una vez con diferentes funciones de mapeo. Así si un servidor falla, su información se podría recuperar de los otros servidores, utilizando funciones de mapeo alternas. Sin embargo, esto lleva consigo un costo de implementación adicional al requerir más equipo de cómputo.
- Actualmente la notificación enviada al monitor, cuando uno de los sensores falla, es enviada por algún otro sensor y solo contiene el identificador del sensor que falló, su dirección IP y la fecha y hora en que falló. Para contar con mayor información se podría adicionar en cada sensor, un SDI basado en computadora (los cuales operan con la información de la bitácora de un sistema). Esta información adicional podría ayudar a determinar la causa por la que un sensor ya no se encuentra disponible. Un agente móvil en cada sensor, tomaría la información generada por el SDI basado en computadora, y la enviaría al monitor antes de que el sensor falle. Lo anterior tiene un costo en el desempeño de la computadora en donde se ejecutan. Sin embargo, ofrece contar con información de mayor precisión de lo que ocurrió.

La implementación de nuestro diseño actual de tolerancia a fallas se puede mejorar:

- El uso del lenguaje PERL en las funciones de corroboración cruzada, no

es el óptimo, ya que PERL es interpretado. Su función puede trasladarse posteriormente a código compilado para reducir el uso de memoria y CPU.

- Actualmente las aplicaciones de corroboración cruzada en los componentes del SDI se activan manualmente. Si el equipo se reiniciara se tendrían que efectuar las activaciones correspondientes. Estas aplicaciones se deberían activar automáticamente, mediante un script que las llame desde el arranque de la computadora.

Apéndice A

Scripts de configuración para la replicación del analizador primario

```
#!/bin/bash

### Configuración Replicación Servidor primario ###
#####  Usuario, contraseña y Base de Datos de mysql ######
usubd="root"
passbd="cornan1517"
BD="snortdata"
echo "####  Configuracion Servidor primario ####"
sleep 2
## Creación de usuario con permisos de archivo en mysql ##
echo
echo "Usuario y contraseña para usuario para conexion del servidor secundario"
echo -n "Usuario: "
read user
echo -n "Contraseña:"
read pass
mysql --user=$usubd --password=$passbd --execute="grant file on *.* to $user@'%' \
identified by '$pass';"
mysql --user=$usubd --password=$passbd  --execute="flush tables with read lock;"
mysql --user=$usubd --password=$passbd --execute="show master status;" > temp
## Parámetro necesario para el servidor secundario ##
```

```
echo "NOTA: Los siguientes valores se requieren para "
echo "      la configuracion del servidor secundario"
gawk '{print NR, $1 $2}' temp
rm temp
mysql --user=$usubd --password=$passbd --execute="unlock tables;" > temp
## Creación de imagen de la base de datos ##
echo "Creando imagen de la base de datos"
mysqldump --user=$usubd --password=$passbd --opt $BD > $BD.sql
echo "El archivo con la imagen de la Base de Datos es : $BD.sql"

## Archivo my.cnf con parámetros necesario ##
cat << DONE > /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-bin
server-id=1

[mysql.server]
user=mysql
basedir=/var/lib

[safe_mysqld]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

DONE
## Reinicio del demonio de mysql ##
/etc/init.d/mysqld  restart

echo "#### Termino de configuracion servidor primario ####"
```

```
#!/bin/bash

### Configuración Replicación Servidor secundario ###
#### Usuario, contraseña y Base de Datos de mysql ####
usubd="root"
passbd="cornan1517"
BD="snortdata"
#### Datos del servidor primario ####
IpSerPri="192.168.1.3"
usurepl="replica"
passrepl="replicacion"
Archreg="mysql-bin.003"
Posarch="174"
echo "####  Configuracion Servidor secundario ####"
if ! test -e $BD.sql
then
echo "Se requiere del archivo $BD.sql generado en el servidor primario"
else
## Creacion base de datos ##
echo "Creación imagen de la base de datos"
mysql --user=$usubd --password=$passbd --execute="create database $BD;"
mysql --user=$usubd --password=$passbd $BD < $BD.sql

## Parámetro necesario para el servidor secundario ##
mysql --user=$usubd --password=$passbd --execute="change master to"
mysql --user=$usubd --password=$passbd --execute="master_host='$IpSerPri'"
mysql --user=$usubd --password=$passbd --execute="master_user='$usurepl'"
mysql --user=$usubd --password=$passbd --execute="master_password='$passrepl'"
mysql --user=$usubd --password=$passbd --execute="master_log_file='$Archreg'"
mysql --user=$usubd --password=$passbd --execute="master_log_pos='$Posarch';"

## Configuración my.cnf secundario ##
cat << DONE > /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
master-host=$IpSerPri
master-user=$usurepl
master-password=$passrepl
```

```
master-port=3306
server-id=2

[mysql.server]
user=mysql
basedir=/var/lib

[safe_mysqld]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

DONE

## Reinicio del demonio de mysql ##
/etc/init.d/mysqld  restart
fi
echo "#### Termino de configuracion servidor secundario ####"
```

Bibliografía

[1] Allen Julie, et al, "State of the Practice of Intrusion Detection Technologies", Carnegie Mellon Software Engineering Institute, January 2000.

[2] Amoroso Edward, *Intrusion Detection*, AT&T Laboratories, New Jersey, January 1999.

[3] Bace Rebecca, *Intrusion Detection*, Macmillan Technical publishing, 2000.

[4] Behlendford Brian, et. al., APACHE Http Server Project, 2004. http://httpd.apache.org

[5] Bace Rebecca, Mell Peter, "Special Publications on Intrusion Detection Systems", National Institute of Standards and Technology, 2000.

[6] Caswell Brian et al, *Snort 2.0 Intrusion Detection*, Syngress Publishing Inc., 2003.

[7] Caswell Brian, Roesh Marty, Snort The Open Source Network Intrusion Detection System, 2004. http://www.snort.org

[8] Coolen R., et al, "Intrusion Detection: Generics and State of the Art", RTO/NATO Technical Report 49, January 2002.

[9] Cozens Simons, Wainwright Peter, *Beginning Perl*, Wrox Press, 2000.

[10] Danyliw Roman, Analysis Console for Intrusion Detection, 2003. http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html

[11] Einwechter Nathan, An introduction to Distributed Intrusion Systems, Security Focus online, January 2001. http://online.securityfocus.com/idsn

[12] Escamilla Terry, *Intrusion Detection: Network Security Beyond the Firewall*, John Wiley & Sons, Inc., 1998.

[13] Garfinkel Simson,Spafford Gene, *Seguridad practica en unix e internet*, O'Relly Segunda edición, 1999.

[14] H. Garcia-Molina,et. al., "Overview of Disaster Recovery for Transactions Processing Systems", Proceedings of IEEE 10th International Conference on Distributed Computers Systems, May 1990, pp. 286-93.

[15] Huagang Xie, LIDS Linux Intrusion Detection System, 2004. http://www.lids.org

[16] Ilgun koral, USTAT - A Real-time Intrusion Detection System for Unix. Master's thesis, University of California at Santa Barbara, November 1992.

[17] J. Gray and A. Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmmann,San Mateo, CA. 1993

[18] K. Hu, S. Mehrotra,S. Kaplan, "An Optimized Two-Safe Approach to Maintaining Remote Back Systems", Proceedings of 8th International Conference on Management of Data, December 1997, pp. 51-68

[19] Kemmer A.Richard, Giovanni Vigna "Intrusion Detection: A brief history and overview", IEEE Computer Volume 35 issue 4 (suplement), April 2002, pages 27-32

[20] Kent Stephen, "On the trail of intrusion into information systems", IEEE Spectrum volume 37 issue 12, December 2000, pages 52-56.

[21] Lentz Arjen, et. al., Mysql Reference Manual, 2004. http://dev.mysql.com/doc

[22] Mchugh John, Christie Alan, Allen Julia, "Defending yourself: The role of intrusion detection systems", IEEE Software volume 17 issue 5, Sep-Oct 2000, pages 42-51.

[23] Mell Peter, Marks Donald, Mclarnon Mark, "A denial of service resistant intrusion detection architecture", National Institute of Standards and Technology, Computer security division, 24 May 2000.

[24] Northcutt Stephen, Novak Judy, *Network Intrusion Detection (an analyst's handbook)*, New Riders second edition, 2001.

[25] Northcutt Stephen, Cooper Mark, et. al., *Intrusion Signatures and Analysis*, New Riders second edition, 2001.

[26] Perl Foundation, PERL Directory, 2004. http://www.perl.org

[27] Proctore Paul E., *The practical Intrusion detection Handbook*, Prentice Hall, New Jersey 2001.

[28] R. Zuver, A thousand heads are better than one? The present and future of distributed intrusion detection, SANS Institute, April 2002. http://www.sans.org

[29] Sherif S. Joseph, Dearmond G. Tommy, "Intrusion Detection: Systems and Models", IEEE International Workshops on Enabling Technologies, 2002.

[30] Sriam Srinivasan, *Advanced Perl Programming*, O'Reilly, 1997.

[31] Sundaram Aurobindo, An introduction to Intrusion Detection, ACM Crossroad Student Magazine, January 2001, http://www.acm.org/crossroads/xrds2-4/intrus.html

[32] Trojnara Michael, et. al., STUNNEL.ORG Universal SSL Wrapper, 2004. http://www.stunnel.org

[33] Vandoorselaere Yoan, Gomez Gene, et. al., PRELUDE Hybrid IDS Project,
2004. http://www.prelude-ids.org

[34] Y. Peggy Shen, T. Liu, W. Tai, "Attack tolerant enhancement of intrusion detection systems", Department of Computer Science and Engineering, Arizona State University, 2000.

[35] Young Erick A., Hudson J. Tim, OPENSSL Open Source Implementation Secure Socket Layer Project,
2002. http://www.openssl.org

[36] Zeev Suraski, Gutmans Andi, et. al., PHP General Purpose Scripting Language Project,
2004. http://www.php.net