4.ESTUDIO DE LAS SOLUCIONES PARA EL MOTOR DEL SISTEMA DE GESTIÓN DE MENSAJES.

4.1.Estudio de Foresight.

4.1.1.Introducción a Foresight.

Foresight es una combinación de elementos que proporcionan monitorización de seguridad de redes, consolidación de mensajes y correlación de eventos.

Combina además múltiples sistemas abiertos, una interfaz web y el servidor Prophet para correlación y motor de agregación.

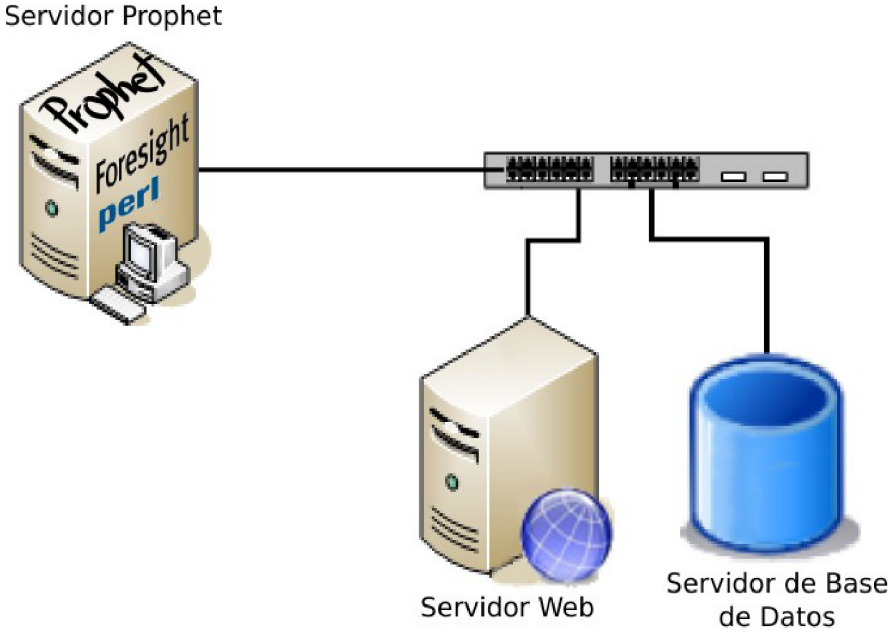

Figura 28.- Escenario simplificado de Foresight.

Los clientes envían mensajes referentes a eventos y datos sobre prevención y detección de intrusos, mensajes de cortafuegos, de evaluación pasiva y activa, mensajes de sistema Syslog y datos de sesión, los cuales los inserta el servidor Prophet en una base de datos para realizar posteriormente la correlación entre los de distinto tipo. Para ello, usa **p0f**, **snort**, **nessus**, **argus** y scripts Perl personalizados.

La comunicación con los clientes se hace por mensages XML sobre túneles SSL.

Todos los clientes tienen una característica de a prueba de fallos (fail-over) activa/pasiva: esto quiere decir que si el Prophet se configura como uno secundario (de copia de seguridad), con la caída de uno de los dos, los clientes caen sobre el secundario, hasta que vuelva el primario. A partir de que se restablezca el primario, los clientes volverán a enviar los datos al primario.

4.1.1.1. Cliente Snort IDS

La implementación del actual usa el el plugin([8]) de salida de la base de datos con su salida de la base de datos.

Ya que funciona con XML sobre SSL pueden registrarse los eventos desde el cliente Snort directamente al Prophet. Una vez que se ha hecho esto, la evaluación pasiva, el rastreo de sesión y el paquete, la caracteristica de registrado de paquetes será incorporada al demonio de Snort. Dicho demonio lee tramas de forma promiscua y compara con patrones de tramas de su base de datos. Esto proporciona un cliente que realiza múltiples funciones.

La salida Syslog también debería ser activada de modo que el registro del cliente pueda registrar eventos en el Prophet para permitir almacenar en caché alarmas.

4.1.2.Clientes de evaluación Pasiva/activa.

4.1.2.1. P0f + PNA.

La herramienta Foresight usa P0f junto con PNA (Análisis de redes pasivas) con guiones en Perl. P0f usando modo síncrono y con asentimiento, identifica activos (dispositivos y sistemas operativos) y escucha aplicaciones a través de la red, mientras que PNA lee el fichero de registros y avisa al servidor Prophet. Estos datos son mostrados como información secundaria para que Foresight determine los falsos positivos ([9]).

4.1.2.2. Nessus.

Se usará Nessus junto con el guión en Perl *ness-to-xml*, es decir, para pasar de Nessus a XML. Otro guión de consola (shell script) es *Vulnscan*, el cual usa mensages *arp* para encontrar dispositivos activos en la red.

Todos los dispositivos que encuentra *Vulnscan* lo pasa a Nessus, y por medio de *ness-to-xml* se registra en el servidor Prophet cada resultado del escaneo, para proporcionar la capacidad de evaluación prevista.

4.1.3.Clientes de sesión y datos de paquetes.

Cada sensor ejecuta una copia de Argus([10]), mientras el servidor Prophet comienza el cliente **ra**, recogiendo información de sesión y registrando los datos en un fichero con un comando como el siguiente:

```
$rabin/ra -S <argus_src>:<argus_port> -nns startime, lasttime, proto,
 saddr, sport, spkts, sbytes, daddr, dport, dpkts, dbytes \ -tcp or udp
 or icmp >>
$arguslog/<argus_link>/<argus_log_file>&
```

El servidor Prophet ejecuta el cliente para el seguimiento de la sesión *sesstrack* en cada enlace monitorizado. *Sesstrack* lee el fichero de registro y comprueba su caché para determinar si la sesión se debería registrar en el Prophet.

---

[8] Plugin es un programa que interactúa con otro programa para aportarle una función o utilidad específica, generalmente muy específica. Este programa adicional es ejecutado por la aplicación principal

[9] Definiremos falso positivo como la generación de un mensaje de alerta o alarma que, teniendo alto riesgo de infección o intrusión, no ha tenido éxito al haber sido bloqueado. Por ejemplo, un IDS puede generar una alerta de intrusión que luego bloquea el cortafuegos. Se dice que el IDS ha generado un falso positivo.

[10] Argus es una aplicación para sistemas de utilización y generación generación del expediente de la auditoría de redes.

Pktcap se ejecuta en cada sensor de modo que monitoriza de forma pasiva cada tarjeta de interfaz de red (NIC) y registra eventos en el Prophet cuando encuentran datos de Caché.

4.1.4.Clientes de eventos Syslog y Procchck([11]).

El Perl-script llamado *logwatch* monitoriza un fichero de mensajes del sistema (syslog) específico para el cortafuegos, eventos de windows, y lo envía al Prophet.

En un principio estará disponible el mínimo soporte de registro, aunque se podrá añadir más en el futuro, como soportes de registros de cortafuegos IPTables, o varios tipos de eventos de Windows (enviados a syslog vía ntsyslog([12])), entre otros ([13]).

Procchck es un verificador de procesos que registra el estado del sistema de detección de intrusos (IDS), de p0f, nessus, demonios logwatch en el Prophet.

4.1.5.Otros clientes.

Oinkmaster([14]) necesita ser ejecutado en cada sensor, para crear las variables de entorno necesarias para el cliente de contexto. El cliente de contexto se ejecuta en cada sensor IDS, cada dos horas ([15]), y comprueba la información de la evaluación activa y pasiva. Esta información es usada para generar variables de entorno para Snort.

Cada sensor registra redes únicas, las cuales son añadidas a la variable de entorno HOME_NET, y además, cada sensor registra los servicios de escucha y los sistemas operativos usados para almacenarlos en sus respectivas variables de entorno: POP_SERVERS, WEB_SERVERS, etc.

4.1.6.Prophet Server.

4.1.6.1. Recopilación de eventos.

Recopila eventos de cada cliente, y cada evento es analizado. Una vez el tipo de registro es determinado y se usa el correcto *plugin* para mensajería, los eventos se insertan en la base de datos.

El cortafuegos y los eventos IDS son almacenados en una caché del sistema de ficheros para un usuario concreto un periodo definido. Dicha caché será compartida entre otros sensores y servidores Prophet, y es utilizada para determinar si deberían ser guardados posteriormente la sesión y los paquetes de datos.

Por ejemplo, un evento del cortafuegos llega sobre tcp con dirección origen 192.168.1.1 hacia la ip destino 192.168.1.2. Entonces se crea una entrada en la caché como **tcp-192.168.1.1-192.168.1.2.** Para los siguientes 15 minutos, todas las sesiones y paquetes con tcp-192.168.1.1-192.168.1.2 ó tcp-192.168.1.2-192.168.1.1 se registran en el Prophet y se insertan en la base de datos.

4.1.6.2. Capacidades del flujo de subida.

Cada servidor Prophet puede ser configurado para operar en modo flujo de subida.

---

[11] Procchck es la abreviatura de Process Check o verificación de procesos.

[12] NTSyslog es una herramienta Windows para dar soporte de envío de mensajes al demonio syslog de Unix.

[13] Actualmente sólo soporta registros de cortafuegos IPTables.

[14] Gestor de reglas para el Snort IDS.

[15] Crea una entrada en la tabla de procesos que ejecuta el sistema de forma periódica, por medio del demonio *cron*.

Esto habilita el sensor para registrar en una base de datos local, pero también seguido de forma remota por un servidor central.

Cada servidor funcionando en modo de flujo de subida puede tener un flujo de subida primario y secundario, lo cual permite a grandes empresas tener sus propias bases de datos de forma remota, mientras todos los eventos de seguridad son enviados simultáneamente a un almacén de registros centralizados.

4.1.6.3. Prophet en modo de seguridad gestionada.

Los servidores Prophet operando en modo gestor de seguridad permiten a los usuarios configurar un ID específico y personalizado. Cada evento es encolado con un ID especial. Sobre el servidor de seguridad gestionada del flujo de subida, estos identificadores y eventos son registrados y almacenados en diferentes bases de datos. Esto permite a los proveedores de servicio de seguridad gestionada utilizar el Prophet para separar y agregar eventos desde múltiples clientes, extendiéndose a muchas bases de datos.

Se permite redundancia de equipo con varios Prophet, varios servidores de bases de datos y varios Web o de otro tipo simultáneamente.

4.1.7.Tablas de ejecución periódica (Crontab) para la interfaz Web.

Guión para ejecutar la verificación de remedio ante errores (Se ejecuta una vez al día a las 4:40):

```
Crontab: 40 4 * * * /usr/bin/run-parts /etc/foresight/remediate.sh 1> /dev/null
#!/bin/sh
/usr/bin/logger -t "Foresight Threat Manager" "Running Remediation Checker"
cd /usr/local/apache2/htdocs/foresight/reports
./remediator.pl -c ./perl/config/remediate.conf
```

Guión para ejecutar la verificación de informes (Se ejecuta una vez al día a las 4:40):

```
Crontab: 40 4 * * * /usr/bin/run-parts /etc/foresight/reports.sh 1> /dev/null
#!/bin/sh
/usr/bin/logger -t "Foresight Threat Manager" "Running Reports"
cd /usr/local/apache2/htdocs/foresight/reports

./reporter.pl -c ./perl/config/report.conf
sleep 2
rm -f /usr/local/apache2/htdocs/foresight/reports/*.pdf
```

Tabla de ejecución periódica para escaneo de intervenciones (Se ejecuta cada 3 minutos):

```
*/3 * * * * /usr/local/apache2/htdocs/secmanage/audit/audit.pl
```

4.1.8.Desarrollo actual y tendencia.

Se están integrando funcionalidades para clientes múltiples dentro de Snort como un plugin, ya que Snort ya está monitorizando todos los paquetes a lo largo del cable, y puede almacenar los datos de sesión tcp. Se necesita hacer algunas modificaciones a un preprocesador de p0f que se ha desarrollado para Snort. También es necesario finalizar con XML sobre el plugin de salida cifrada SSL de Snort.

En un futuro a corto plazo se pretende crear un esquema de base de datos escalable para soportar cientos de miles de eventos, y optimizar el código PHP para más seguridad y velocidad.

4.2.Estudio de Prelude.

4.2.1.Introducción.

El esquema general será un sistema centralizado de logs en torno al gestor *prelude-manager*, el cual rellenará la base de datos. Dichos sensores podrán tener o no soporte para Prelude, es decir, que podrán generar mensajes en IDMEF. Los que no tengan dicho soporte, podrán adaptarse al estándar IDMEF por medio del LML, el cual rellenará los campos IDMEF con la información que reciba de los distintos sensores. Prelude-LML además actuará como sensor de los mensajes de sistema y del núcleo (syslog) de la máquina en la que esté instalado.

Por último, la Interfaz de monitorización, desarrollada en Java, operará directamente con la base de datos y de forma independiente con el resto de componentes.

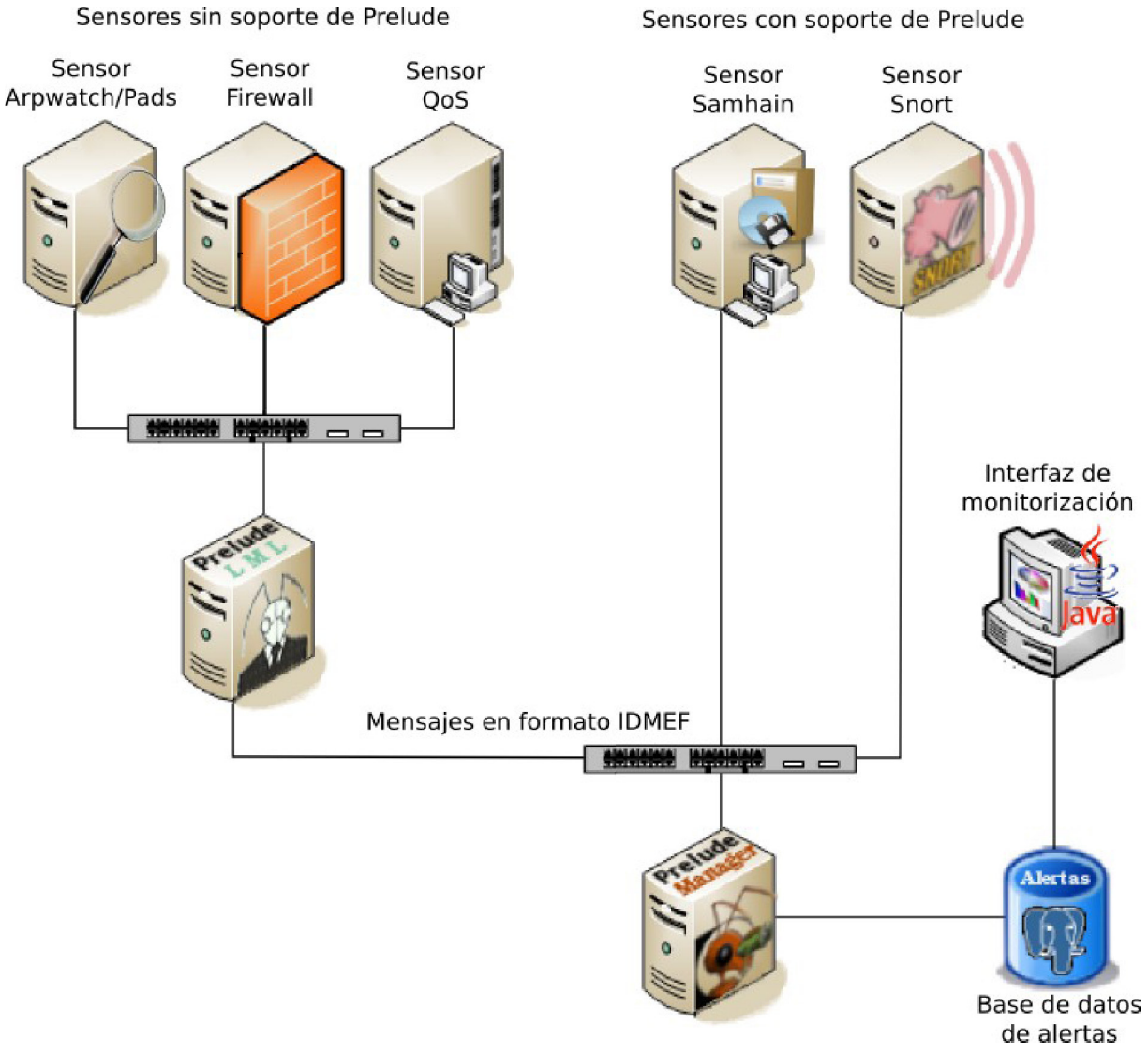

Figura 29.- Escenario genérico con Prelude-IDS.

4.2.2.Librería Libprelude.

Libprelude posee las funciones necesarias para prelude. Proporciona el marco de métodos y utilidades para que el gestor de prelude Prelude-Manager. Dicho marco es provisto en C, Python y Perl de modo que se puedan usar aplicaciones de seguridad en el sistema Prelude. Estas funciones serán usadas tanto por el Prelude-Manager como por un sensor/analizador de mensajes cuya salida es IDMEF como el PreludeLML. Nessus es un escáner de seguridad remoto para linux.

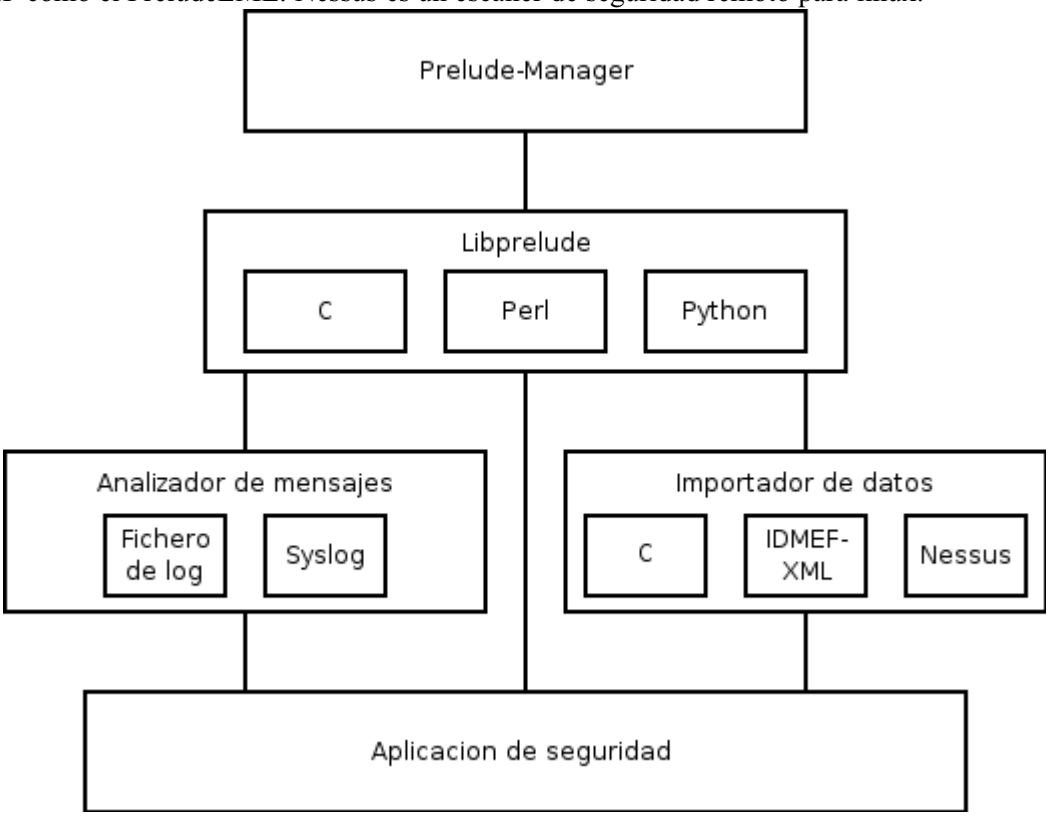

Figura 30.- Marco de prelude.

Libprelude permite al gestor/colector de mensajes recopilar la información desde los ficheros de logs o desde sensores que emitan en IDMEF.

También provee de importantes características como *failover* (guardando la información crítica en un fichero local para una retransmisión posterior) en caso de que se caiga uno de los servidores con prelude-manager. Además, da la posibilidad de crear sensores que rean los eventos recibidos por uno o un conjunto de prelude-managers.

4.2.3.Librería Libpreludedb y la Base de Datos de Alertas.

Librería para el acceso fácil a la base de datos de Prelude. Provee de una capa de abstracción basada en SQL que proporciona las funciones para que el gestor Prelude-Manager almacene las alertas en la base de datos (PostgreSQL y/o MySQL). Se proporcionan los ficheros **.sql* necesarios para la generación de las tablas, secuencias e índices. A continuación los índices y el árbol de índices de Prelude.

`/usr/share/libpreludedb/classic/pgsql.sql` (fragmento)

```
[...]
CREATE INDEX prelude_analyzer_index_model
```

```
      ON Prelude_Analyzer (_parent_type,_index,model);
CREATE INDEX prelude_classification_index_text ON Prelude_Classification (text);
CREATE INDEX prelude_reference_index_name on Prelude_Reference (name);
CREATE INDEX prelude_impact_index_severity ON Prelude_Impact (severity);
CREATE INDEX prelude_impact_index_completion ON Prelude_Impact (completion);
CREATE INDEX prelude_impact_index_type ON Prelude_Impact (type);
CREATE INDEX prelude_createtime_index ON Prelude_CreateTime (_parent_type,time);
CREATE INDEX prelude_detecttime_index ON Prelude_DetectTime (time);
CREATE INDEX prelude_analyzertime_index
      ON Prelude_AnalyzerTime (_parent_type,time);
CREATE INDEX prelude_node_index_location
      ON Prelude_Node (_parent_type,_parent0_index,location);
CREATE INDEX prelude_node_index_name
      ON Prelude_Node (_parent_type,_parent0_index,name);
CREATE INDEX prelude_address_index_address
      ON Prelude_Address (_parent_type,_parent0_index,_index,address);
CREATE INDEX prelude_service_index_protocol_port
      ON Prelude_Service (_parent_type,_parent0_index,protocol, port);
CREATE INDEX prelude_service_index_protocol_name
      ON Prelude_Service (_parent_type,_parent0_index,protocol, name);
[...]
```

A continuación, las tablas que genera el fichero /usr/share/libpreludedb/classic/pgsql.sql. En negrita se denotan los índices anteriormente definidos.

Tablas generadas por **`/usr/share/libpreludedb/classic/pgsql.sql`**

<table>
<tr><td>_format
(name, version)</td><td>Prelude_Alert
(_ident, messageid)</td><td>Prelude_OverflowAlert
(_message_ident,
program, size, buffer)</td><td>Prelude_ToolAlert
(_message_ident,
name, command)</td></tr>
<tr><td>Prelude_Alertident
(_message_ident,
_index, _parent_type,
alertident, anayzerid)</td><td>Prelude_CorrelationAlert
(_message_ident, name)</td><td>Prelude_Checksum
(_message_ident,
_parent0_index,
_parent1_index, _index,
algorithm, value,
checksum_key)</td><td>Prelude_Analyzer
(_message_ident,
_parent_ident, _index,
analyzerid, name,
manufacturer, model,
version, class, ostype,
osversion)</td></tr>
<tr><td rowspan="2">Prelude_File
(_message_ident,
_parent0_index, _index,
ident, path, name,
category, create_time,
create_time_gmtoff,
modify_time,
modify_time_gmtoff,
access_time,
access_time_gmtoff,
data_size, disk_size,
fstype, file_type)</td><td>Prelude_FileAccess
(_message_ident,
_parent0_index,
_parent1_index, _index)</td><td>Prelude_FileAccess_Permission
(_message_ident,
_parent0_index,
_parent1_index,
_parent2_index, _index)</td><td>Prelude_Linkage
(_message_ident,
_parent0_index,
_parent1_index, _index,
category, name, path)</td></tr>
<tr><td>Prelude_Impact
(_message_ident,
description, severity,
completion, type)</td><td>Prelude_Action
(_message_ident,
_index, description,
category)</td><td>Prelude_Node
(_message_ident,
_parent_type,
_parent0_index, ident,
category, location, name)</td></tr>
</table>

Tablas generadas por **`/usr/share/libpreludedb/classic/pgsql.sql`**

Prelude_Inode (**_message_ident, _parent0_index, _parent1_index,** change_time, change_time_gmtoff, number, major_device, minor_device, c_major_device, c_minor_device)	Prelude_Heartbeat (_ident, messageid, heartbeat_interval)	Prelude_Assessment (**_message_ident**)	Prelude_Confidence (_message_ident, confidence, rating)
	Prelude_CreateTime (**_message_ident, _parent_type**, time, usec, gmtoff)	Prelude_DetectTime (_message_ident, time, usec, gmtoff)	Prelude_AnalyzerTime (**_message_ident, _parent_type**, time, usec, gmtoff)
Prelude_AdditonalData (**_message_ident, _parent_type, _index**, type, meaning, data)	Prelude_User (**_message_ident, _parent_type, _parent0_index**, ident, category)	Prelude_UserId (**_message_ident, _parent_type, _parent0_index, _parent1_index, _parent2_index, _index**, ident, type, name, tty, number	Prelude_Address (**_message_ident, _parent_type, _parent0_index, _index**, ident, category vlan_name, vlan_num, address, netmask)
Prelude_Classification (_message_ident, ident, text)	Prelude_Reference (**_message_ident, _index**, origin, name, url, meaning)	Prelude_Source (**_message_ident, _index**, ident, spoofed, interface)	Prelude_Target (**_message_ident, _index**, ident, decoy, interface)
Prelude_Process (**_message_ident, _parent_type, _parent0_index,** ident, name, pid, path)	Prelude_ProcessArg (**_message_ident, _parent_type, _parent0_index, _index,** arg)	Prelude_ProcessEnv (**_message_ident, _parent_type, _parent0_index, _index,** env)	Prelude_Service (**_message_ident, _parent_type, _parent0_index,** ident, ip_version, name, port, iana_protocol_number, iana_protocol_name, portlist, protocol)
Prelude_WebService (**_message_ident, _parent_type, _parent0_index,** url, cgi, http_method)	Prelude_WebServiceArg (**_message_ident, _parent_type, _parent0_index, _index,** arg)	Prelude_SnmpService (**_message_ident, _parent_type, _parent0_index,** snmp_oid, community, security_name, context_name, context_engine_id, command)	

En total son 37 tablas en la que se pueden almacenar todos los atributos de todos los tipos de alertas. En Prelude_AdditionalData se pueden definir nuevos campos

4.2.4.Prelude-Manager.

Es la aplicación que recopila y normaliza eventos de seguridad o alertas. Es un servidor de gran capacidad que almacena en una base de datos SQL las alertas de sensores distribuidos. También provee de la capacidad de reenviar alertas recibidas a otros servidores Prelude-Manager. A todo lo anterior se suma la posibilidad de filtrar dichos eventos recibidos, así como definir acciones específicas para alertas concretas.

4.2.5.Prelude-LML.

Es un analizador de mensajes y colector de eventos Syslog.

Prelude-LML es un analizador de mensajes basado en firmas que monitoriza los ficheros de logs y mensajes de Syslog recibidos bajo actividad sospechosa. Maneja eventos generados por un gran conjunto de components, incluyendo, entre otros: Cisco PIX, Clamav, Dell-OM, Grsecurity, Honeyd, Ipchains, Netfilter, Ipfw, Nokia IPSO, Apache, MS-SQL, Nagios, Norton Antivirus Corporate Edition, PAM, Portsentry, Postfix, ssh, etc.

Prelude-LML se escribió con el propósito de integrar fácilmente productos que no tienen salida en IDMEF, recolectandolos y rellenando la información de los campos IDMEF-XML gracias a unas fórmulas ([16]) definidas para la sintaxis de cada sensor. Por ejemplo, el sensor de Apache transmite los mensajes cuyo formato comienza como sigue: "hostname - - [timestamp{20} ...etc". Prelude-LML, conociendo la posición en la que recibe la fecha en los mensajes de Apache, y sabiendo en qué posición del mensaje IDMEF debe insertarla, es capaz de transformar las alertas del formato inicial al estándar derivado de XML IDMEF.

4.2.6.Sensores.

Los componentes cuyos eventos puede manejar el Prelude-LML también se consideran sensores, pero en este apartado se detallan los que pueden transmitir su salida en formato IDMEF directamente, es decir, son sensores con plug-in de Prelude. Se trata de *Snort, PAM, Pflogger, Samhain, Nepenthes* y en desarrollo *Sancp* y *Libsafe*.

4.2.6.1. Snort.

Es un sistema de detección de intrusiones que se puede compilar con la opción habilitada de Prelude para que genere alertas en formato IDMEF. Es capaz de detectar y por tanto generar alertas según las reglas que tenga definidas en el `/etc/snort/rules`. A continuación, las líneas del fichero de configuración que deberán habilitarse para el correcto funcionamiento.

`/etc/snort/snort.conf` (Fragmento)

```
[...]
# alert_syslog: log alerts to syslog
output alert_syslog: LOG_AUTH LOG_ALERT

output alert_prelude: profile=snort sensor_name=snort

output alert_unified: filename /dev/null
output log_unified: filename /dev/null
[...]
```

4.2.6.2. Samhain.

Samhain es una plataforma en código abierto para verificación de la integridad de ficheros centralizada para máquinas basadas en detección de intrusiones sobre sistemas POSIX (Unix, Linux y Cygwin/Windows). Ha sido diseñada para monitorizar múltiples equipos con diferentes sistemas operativos desde una ubicación centralizada, aunque también puede usarse como aplicación independiente en una máquina individual.

[16] Estas fórmulas son expresiones regulares de perl *pcre* (Perl Common Regular Expressions) definidas en el fichero de configuración de Prelude-LML.

4.3.Comparativa Prelude - Foresight.

Tras un estudio teórico se decide utilizar Prelude por muchas razones, entre las cuales:

4.3.1.Salida cifrada SSL en XML.

Foresight no tiene finalizado el plugin de salida cifrada con Snort u otros sensores IDS, mientras Prelude posee la salida en formato normalizado IDMEF, que es una derivación de XML, versátil y con conexiones cifradas por SSL.

4.3.2.Base de datos escalable.

El esquema de la base de datos no es escalable para Foresight. Si se comprueba en el fichero pgsql.sql de Libpreludedb, hay muchas tablas con pocas columnas (optimiza el tiempo de acceso a consultas con varios joins o productos cartesianos) y pocas filas ya que se definen muchos campos sin la etiqueta "*not null*" en SQL, lo cual aprovecha el código de libpreludedb para no insertar las filas que tienen valores nulos. Esto permite a Prelude soportar cientos de miles de eventos.

4.3.3.Colector escalable.

Foresight propone un modelo de gestión centralizada en un servidor Prophet, colector de eventos de seguridad. Prelude, propone un colector de alertas Prelude-Manager que puede ser gestor centralizado y receptor de logs y/o sensor de otro Prelude-Manager. Es decir, Prelude proporciona la posibilidad de conectar varios Prelude-Managers en cascada siendo uno sensor del otro, pudiendo manejar cada uno una base de datos propia, todos conectados a la misma, sólo conectado el sistema colector final...etc.

4.3.4.Compatibilidad con resto de sensores.

La documentación de Foresight no especifica compatibilidad con un conjunto extenso de sensores, ni que es versátil ante la aparición de nuevos dispositivos de seguridad. En Prelude se pueden definir reglas para cualquier dispositivo que no tenga salida cifrada y normalizada, las cuales se especifican como expresiones regulares de Perl en el fichero de configuración de Prelude-LML.

A lo anterior hay que añadir que cada vez más sensores están desarrollando un plugin para Prelude con salida en IDMEF (estándar de normalización en XML y cifrado SSL), como *Snort*, *Samhain*, *Nessus*, *Nagios*, *NIDS*, *Nepenthes*, *Pflogger* y en actual desarrollo *Sancp* entre otros.

4.3.4.1. Plug-in de Snort.

Foresight no posee en la actualidad plug-in para Snort-IDS, cosa que está desarrollando en la actulidad, mientras Prelude sí lo posee. Además, en Foresight hay que hacer modificaciones a un preprocesador de p0f que se ha desarrollado para Snort.

4.3.5.Documentación.

La documentación de Foresight no es tan extensa y es más difícil de conseguir, ya que tuvo que ser pedida por email a Josh Berry, de Penson Financial Services (josh.berry@netschematics.com). Prelude, sin embargo, posee una página con manuales, foros, listas de correo resolviendo problemas y dudas de uso, desarrollo etc. Por todo ello se opta por utilizar Prelude en lugar de Foresight.