3.DRAFT IDMEF. NORMALIZACIÓN DE LOGS DE SEGURIDAD.

3.1.Introducción.

La primera fase es unificar en un formato común los mensajes generados por los distintos sensores. Buscando documentación relacionada con "data collection for secure monitoring", "intrusion detection exchange" y temas relacionados se puede encontrar el *draft* titulado:

The Intrusion Detection Message Exchange Format

draft-ietf-idwg-idmef-xml-14

Dicho draft, desarrollado por miembros de France Telecom, describe un modelo que engloba todos los campos de los mensajes de seguridad ([3]) de modo que se puede representar la información exportada por los sistemas de detección de intrusiones.

La implementación de dicho modelo se hace en XML (Extensible Markup Language).

3.2.Modelo IDMEF. Generalidades.

El propósito de dicho modelo es solucionar los problemas derivados de un sistema heterogéneo de sensores y datos.

3.2.1.Problemas del modelo de datos solucionados con IDMEF.

- Las información de cada dispositivo es inherentemente heterogénea. Algunas alertas son definidas con muy poca información como origen, destino, nombre y hora del evento, mientras que otras proporcionan mucha más información como puertos de servicio, procesos, usuarios etc. El modelo de datos de IDMEF es mucho más flexible para acomodarse a las diferentes necesidades.
- Un modelo orientado a objetos es extensible y escalable por medio de la agregación y la herencia de clases. Si una implementación del modelo de datos es heredada por una clase nueva, bien por agregación o por subclases, aunque esta nueva clase no comprenda determinadas extensiones o estructuras de mensajes, lo hará por herencia. Así se preserva la consistencia del modelo. IDMEF propone un modelo orientado a objetos.
- Los entornos de detección de intrusiones son diferentes. De este modo, se pueden detectar ataques analizando tráfico de red, mensajes del sistema operativo o de aplicaciones específicas. Estas alertas obviamente no contendrán la misma información. El modelo de IDMEF soporta clases que se adaptan a las diferencias de los datos originales. Origen y destino es, para las alertas, una combinación de las clases: *Node, Process, Service,* y *User*.
- Las capacidades de cada analizador son diferentes, dependiendo del tipo de entorno. El modelo de datos permite la conversión de formatos usada por herramientas de distintos IDS.
- Los entornos de trabajo son diferentes, tanto por tipo de red como de sistema operativo. Esto hace que los ataques se emitan con diferentes características. El modelo de datos se acomoda

[3] Además, este draft está sujeto a las provisiones de la sección 3 de la RFC 3667

a estas diferencias y define un campo en el que se puede almacenar el sistema operativo y la versión del núcleo en determinados casos.

- Otra variación es el vendedor o comercial, por lo que unos fabricantes pueden proveer de más información que otros.

En definitiva, el modelo de datos proveerá de una infraestructura para aplicaciones que puedan establecer relaciones entre alertas simples y complejas.

3.2.2. Razonamiento para implementar IDMEF en XML.

El objetivo del modelo de datos es proveer de un estándar de representación de la información que informa un analizador de detección de intrusiones cuando detecta un evento inusual. Estas alertas pueden ser simples o complejas dependiendo de las capacidades del analizador.

Las aplicaciones basadas en XML estás siendo usadas o desarrolladas para una amplia variedad de propuestas, incluyendo intercambio de datos electrónicos en muchos campos como el intercambio de datos financieros, planificación, distribución de software de empresas, lenguajes de marcas para química, matemáticas, música, dinámica molecular, astronomía, publicaciones web, observaciones meteorológicas etc. La flexibilidad de XML lo convierte en una buena elección para estas aplicaciones; además, existen otras razones específicas para implementar IDMEF usando XML:

- XML permite un lenguaje personalizado para ser desarrollado específicamente con el propósito de describir alertas de detección de intrusiones. También define un modo estándar de extender el lenguaje para revisiones posteriores del estándar o para una revisión específica de IDMEF de un fabricante (“non-standard” extensions”).
- Hay una amplia variedad de herramientas software para el procesamiento XML en formato comercial y software libre. Hay disponibles numerosas herramientas y APIs para hacer *parsing* ([4]) o validación XML en una gran variedad de lenguajes, incluyendo: Java, C, C++, Tcl, Perl, Python, and GNU Emacs Lisp.
- Los mensajes IDMEF requieren codificación en UTF-8 y UTF-16 de ISO/IEC 10646 y Unicode, compatibles con XML, siendo la primera mencionada la opción por defecto. Estos mensajes soportan completa internacionalización y localización.
- Otro requerimiento de los mensajes IDMEF es que debe soportar filtrado y agregación. La integración de XML con XSL, permitiendo combinar, descartar y reorganizar mensajes.
- Los proyectos para el desarrollo de la salida XML proveerán extensiones orientadas a objetos, soporte para bases de datos y otras características.

3.3. El modelo de datos IDMEF y la DTD de XML.

En este apartado se describe un resumen de los componentes individuales del modelo de datos IDMEF con diagramas UML para mostrar las relaciones entre ellos. El apartado DTD muestra como el modelo es traducido a XML.

[4] También traducido por algunos autores como parseo (de parsear), se trata de rellenar los campos de un mensaje con un formato definido, con los valores adecuados para encapsularlo y depositarlo en un flujo de salida.

3.3.1.Perspectiva general del modelo de datos.

La relación entre los principales componentes del modelo de datos se muestran en el anexo 1: diagrama UML del modelo IDMEF, donde se han omitido atributos e indicadores de ocurrencias.

La superclase de todos los mensajes IDMEF es *IDMEF-Message*. Cada tipo de mensaje es una subclase de esta superclase. Hay dos tipos de mensajes: Alertas y mensajes periódicos de información de datos o de gestión conocidos como *heartbeats* o mensajes de latencia o pulsación.

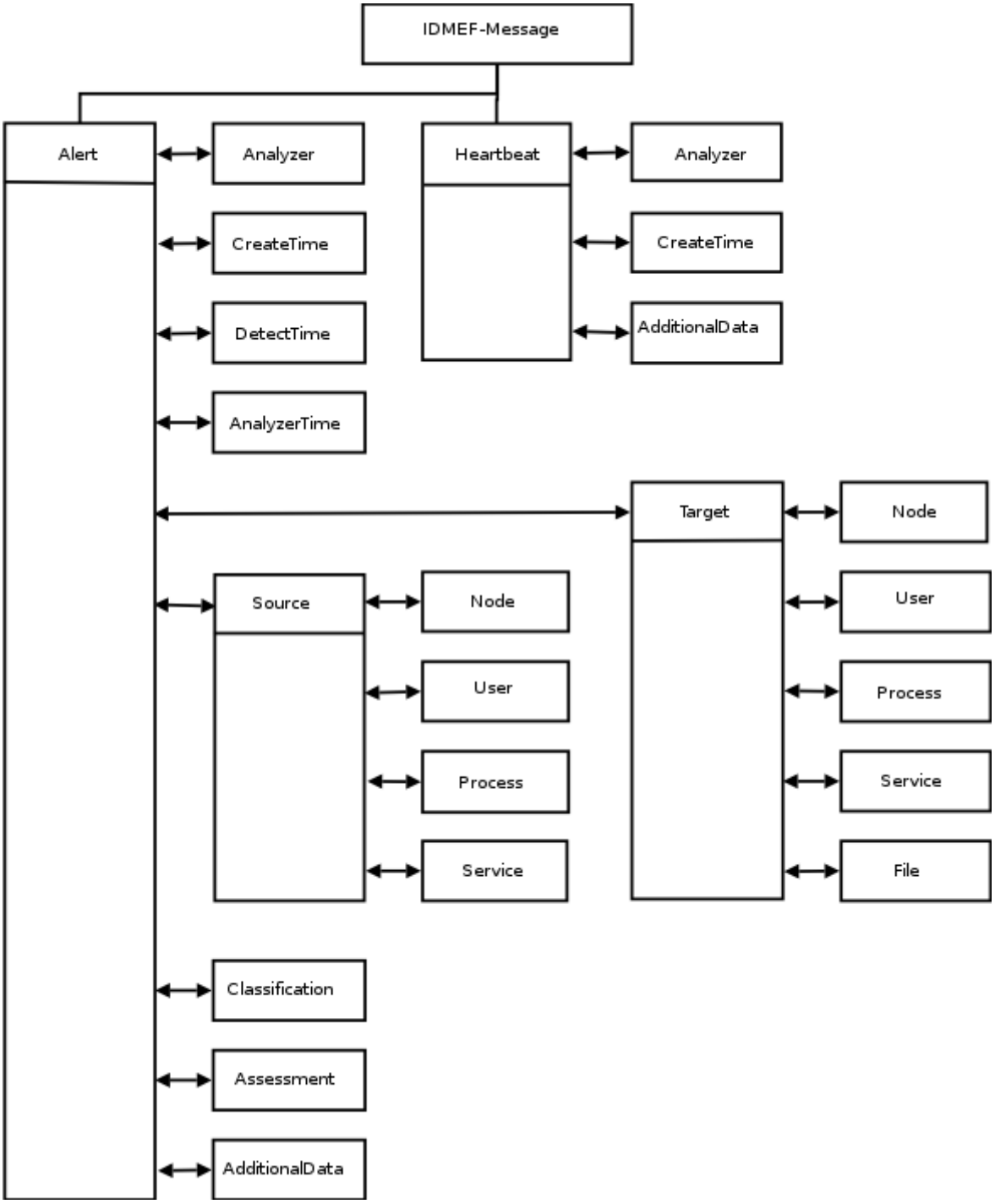

Figura 2: Modelo de datos.

El modelo de datos no especifica como debe clasificarse o identificarse una alerta. Por ejemplo, un analizador puede identificar que un escaneo de puertos es un ataque simple contra muchos puertos mientras otro analizador pudiera identificarlo como múltiples ataques desde un mismo origen. Sin embargo, una vez que un analizador ha determinado el tipo de alerta y planea enviarla, el modelo de datos dictamina cómo debería ser formateado dicho mensaje.

3.3.2. Clase *IDMEF-Message*.

Todos los mensajes IDMEF son instancias de la clase *IDMEF-Message*. Es la clase de más alto nivel de todo el modelo de datos IDMEF, así como del DTD.

Hay actualmente dos tipos de mensajes IDMEF, lo cual se corresponde con dos subclases de la superclase *IDMEF-Message*: ***Alert*** (alerta) y ***Heartbeat*** (mensaje de latencia).

Debido a que la definición del tipo de documento (DTD) no soporta la estructuración en superclase y subclases, la relación de herencia entre *IDMEF-Message* y las subclases *Alert* o *Heartbeat* mostrada en la figura 1 se reemplaza por una relación de agregación declarada en la definición del tipo de documento IDMEF como sigue:

```
<!ENTITY % attlist.idmef                "
      version            CDATA                    #FIXED    '1.0'
">
<!ELEMENT IDMEF-Message                 (
      (Alert | Heartbeat)*
)>
<!ATTLIST IDMEF-Message
      %attlist.idmef;
>
```

La clase *IDMEF-Message* tiene un atributo simple: la versión. Las aplicaciones que especifiquen un valor para este atributo en la actual versión deberán forzosamente poseer el valor “1.0”.

3.3.2.1. Clase Alerta (*Alert*).

En general, cada vez que un analizador detecta un evento que ha sido configurado para ser visto, envía un mensaje de alerta al gestor. Dependiendo del analizador, el mensaje de alerta puede corresponder a la detección de un evento simple o múltiple, y se transmite de forma asíncrona.

Un mensaje de alerta está compuesto por varias clases agregadas, como se muestra en la Figura 2.

Las clases agregadas para la formación de esta clase son:

- *Analyzer*: Único. Almacena información de identificación para el analizador que originó la alerta.
- *CreateTime*: Único. Posee la fecha en la que la alerta fue creada. De las tres fechas que pueden proveerse con una alerta, ésta es la única que es requerida.
- *Classification*: Exactamente uno. El “nombre” de la alerta u otra información perteneciente al gestor para determinar qué es.
- *DetectTime*: Puede ser cero o uno. El el tiempo en el que la alerta fue detectada. En caso de tener más de un evento, se trata del tiempo del primer evento detectado. En determinadas circunstancias, esto puede no ser el mismo valor que *CreateTime*.
- *AnalyzerTime*: Cero o uno. El valor de tiempo actual.
- *Source*: Cero o más. Las fuentes de los dispositivos que originaron los eventos.
- *Target*: Cero o más. Las fuentes de los dispositivos destinatarios de las alertas.

- *Assessment*: Cero o uno. Información sobre el impacto de los eventos, acciones llevadas a cabo por el analizador responsable de los mismos y la confianza de esta evaluación.
- *AdditionalData*: Cero o más. Información incluída por el analizador que no se ajuste al modelo de datos. Esto podría ser una parte atómica de los datos, o un gran conjunto de datos dados por una extensión de IDMEF.

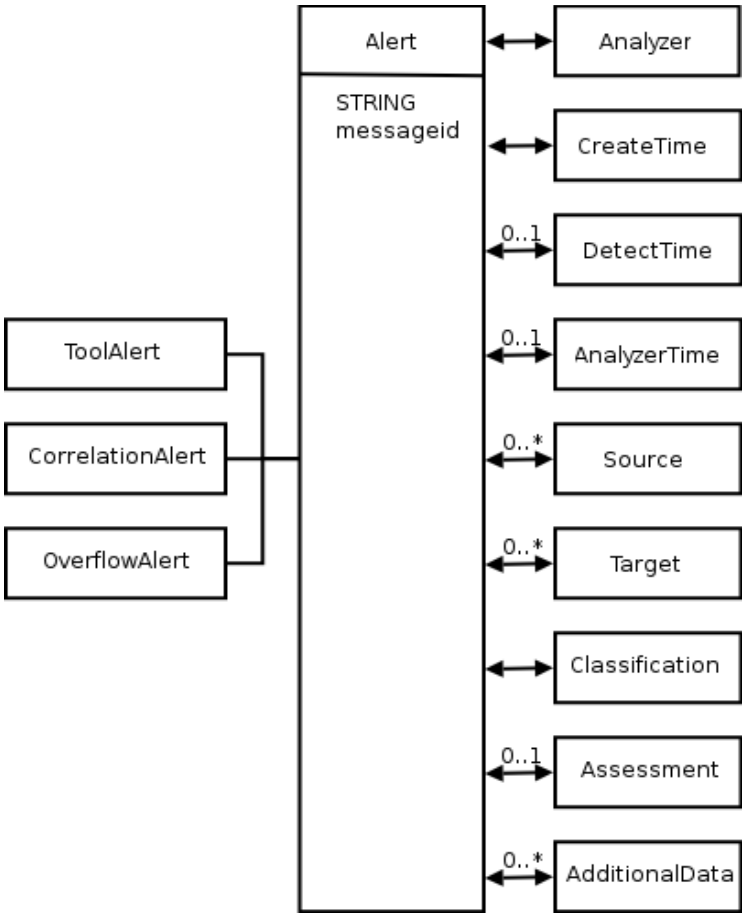

Figura 3: La clase *Alert*.

Como la definición del tipo de documento (DTD) no soporta la estructuración en clases y subclases, la relación de herencia entre *Alert* y las subclases *ToolAlert, CorrelationAlert*, y *OverflowAlert* mostradas en la figure3 ha sido reemplazada con una de agregación.

La clase *Alert* es representada en XML (en el DTD) como sigue:

```
<!ELEMENT Alert(
      Analyzer, CreateTime, DetectTime?, AnalyzerTime?,
      Source*, Target*, Classification,  Assessment?, (ToolAlert |
        OverflowAlert | CorrelationAlert)?, AdditionalData*
)>
 <!ATTLIST Alert
       messageid          CDATA                  '0'
      >
```

La clase Alerta tiene un único atributo. Se trata de *messageid*, único identificador de la alerta, el cual es opcional.

3.3.2.1.1 La clase *ToolAlert* (Alertas de herramientas).

La clase *ToolAlert* contiene información adicional relacinada con el uso de herramientas de ataque o programas malévolos como caballos de Troya, y puede ser usada por el analizador cuando es capaz de identificar dichas herramientas. Se pretende agrupar juntas una o más alertas previamente transmitidas para decir que todas ellas fueron el resultado de que alguien usó una herramienta determinada.

Est clase está compuesta por tres clases agregadas, tal y como se muestra en la figura 3:

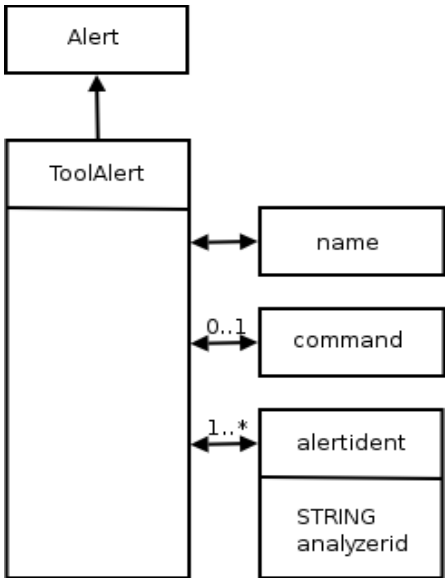

Figura 4. Clase *ToolAlert*.

Las clases agregadas que componen la clase *ToolAlert* son el nombre (name), comando (command) y el identificador de alerta (alertident). Éste último puede ser múltiple, listando en dicho caso los identificadores relacionados con la alerta. Una alerta identifica de manera única a los eventos enviados por un analizador, y además el *analyzerid* nos permite identificar los analizadores que enviaron la misma alerta. El comando es la operación pedida por la herramienta, como por ejemplo un ping de BackOrifice.

Esto se representa en la definición de tipo de documento XML de la siguiente manera:

```
<!ELEMENT ToolAlert                    (
      name, command?, alertident+
)>
<!ELEMENT alertident      (#PCDATA) >
<!ATTLIST alertident
      analyzerid          CDATA                  #IMPLIED
>
```

3.3.2.1.2 La clase *CorrelationAlert*.

La clase de correlación de alertas porta información adicional relacionada de correlación, lo cual se entiende como la agrupación de una o más alertas previamente enviadas juntas, para dejar patente que existe una relación entre todas ellas.

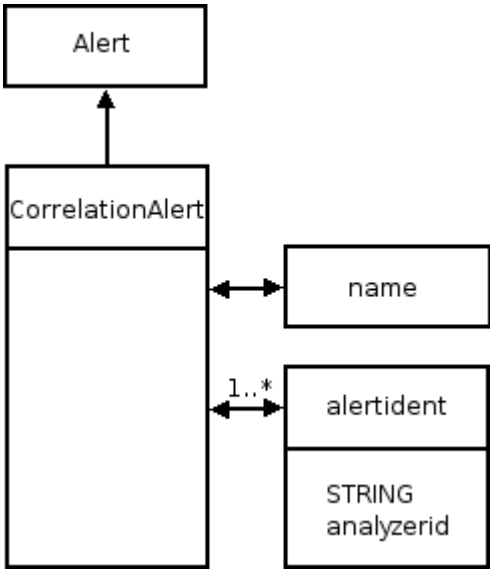

Figura 5: Clase *CorrelationAlert*.

Las clases agregadas que componen la clase de correlación de alertas son el nombre (name) y el/los identificador/es de alerta. Éste último es una lista de los identificadores de alerta relacionados con esta alerta. La cadena *analyzerid* identifica al analizador que transmitió la alerta.

Esto se define en XML como sigue:

```
<!ELEMENT CorrelationAlert              (
      name, alertident+
)>
<!ELEMENT alertident       (#PCDATA) >
<!ATTLIST alertident
      analyzerid           CDATA                    #IMPLIED
>
```

3.3.2.1.3 La clase *OverflowAlert.*

La clase de alertas de desbordamiento porta información relacionada con ataques a los buffers de desbordamiento. Se entiende como activar un analizador para proveer de los detalles del ataque de desbordamiento.

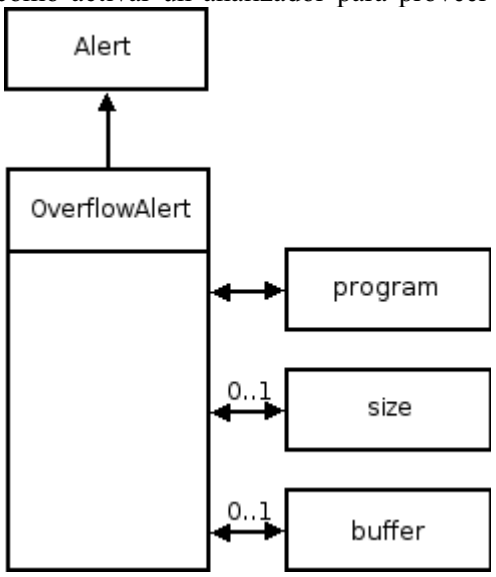

Figura 6: Clase *OverflowAlert*.

Las clases agregadas para la formación de esta clase son:

- *Program*: Única. Almacena el programa que originó el ataque por desbordamiento en forma de cadena.
- *Size*: Puede ser cero o uno. Almacena en un entero el número de bytes del desbordamiento, es decir, el número de bytes que envió el atacante.
- *Buffer*: Puede ser nulo o una cadena de bytes con todos los datos que el analizador pueda capturar que han sido desbordados.

El hecho de que el tamaño y buffer pueden ser nulos se define en el XML DTD con una interrogación:

```
<!ELEMENT OverflowAlert
(      program, size?, buffer?      )
>
```

3.3.2.2. La clase de pulsaciones (*Heartbeat*).

Los analizadores usan los mensajes de pulsaciones para indicar su estado actual a los gestores. Se entiende que los pulsos se transmiten con periodicidad, cada diez minutos o cada hora. La recepción de un pulso o "latido de corazón" de un analizador indica al gestor que se está activo. La falta o carencia de uno (o varios consecutivos) de estos pulsos indica que el analizador o su conexión de red ha caído.

Todos los gestores deben admitir la recepción de mensajes de latido de corazón. Sin embargo, el uso de estos mensajes por parte de los analizadores es opcional. Los desarrolladores de software para el gestor deberían permitir que dichos programas fueran configurados para permitir o no estos pulsos.

Un mensaje de pulso o latido de corazón está compuesto por varias clases agregadas, como se muestra en la figura 6.

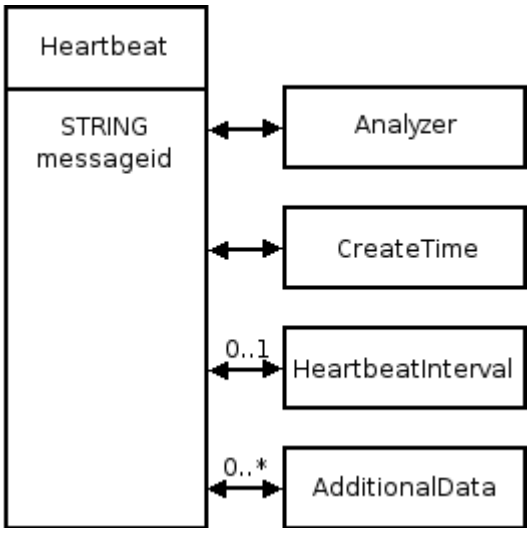

Figura 7: Clase *Heartbeat*.

Las clases agregadas que conforman la de pulsos de corazón son:

- *Analyzer*: Único. Almacena información de identificación para el analizador que originó el pulso de *heartbeat*.

- *CreateTime*: Campo con valor único correspondiente al instante cuando el mensaje de pulsación fue creado.
- *AnalyzerTime*: Puedes existir o no, almacenando la hora actual del analizador.
- *HeartbeatInterval*: También puede existir o no, almacenando en intervalo en segundos entre el cual son generados los mensajes de pulsación.
- *AdditionalData*: Puede estar vacío o ser múltiple. Contiene la información incluida por el analizador la cual no se corresponde con el modelo de datos. Esto puede ser conjunto de datos indivisible o una gran cantidad de datos dadas a través de una extensión de IDMEF.

Esta clase se representa como sigue en el XML DTD:

```
<!ELEMENT Heartbeat                    (
      Analyzer, CreateTime, HeartbeatInterval?, AnalyzerTime?,
      AdditionalData*
)>
<!ATTLIST Heartbeat
      messageid          CDATA                  '0'
>
```

La clase *Heartbeat* tiene un atributo, opcional, de nombre *messageid*, el cual corresponde a un identificador único para la pulsación.

3.3.2.3. Las clases de núcleo (Core Classes).

Las clases de núcleo son las partes principales de las clases de alerta y pulsación (*Alert* y *Heartbeat*). Dichas partes principales son analizador (*Analyzer*), fuente (*Source*), destino (*Target*), classification (*Classification*) y datos adicionales (*AdditionalData*).

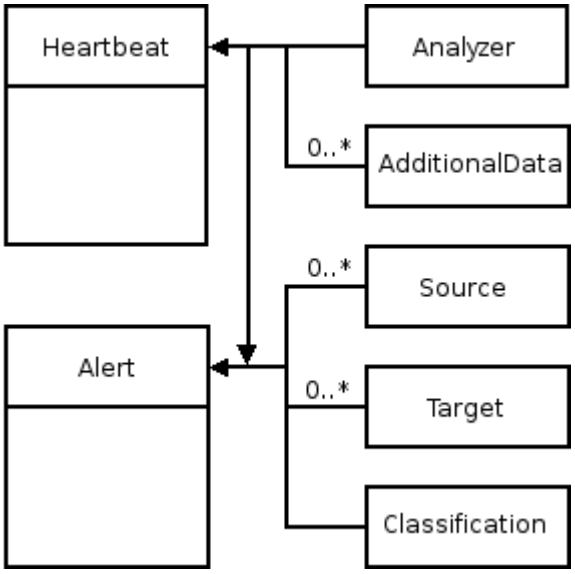

Figura 8: Las clases de núcleo.

3.3.2.3.1 La clase de analizador (*Analyzer*).

La clase *Analyzer* identifica el analizador desde el cual se origina el mensaje de alerta o pulsación. Sólo un analizador puede ser codificado por cada alerta o pulsación. Aunque el modelo de datos IDMEF no previene del uso de sistemas de detección de intrusiones jerárquicos (donde las alertas se

almacenan y relacionan en forma de árbol). No obstante, en el caso de reenvío de alertas de un analizador a otro no se dispone de ningún mecanismo para almacenar el analizador originario.

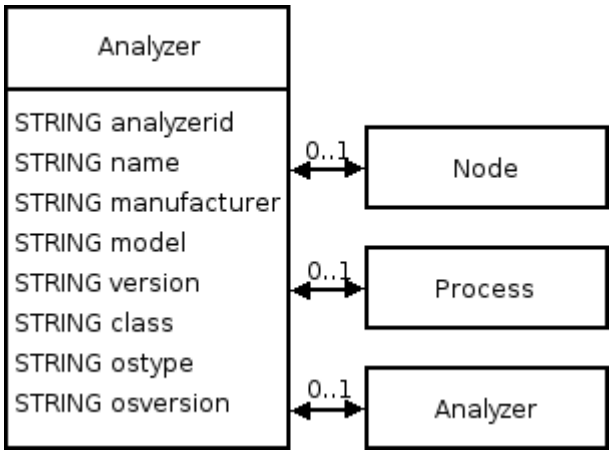

Figura 9: Clase de analizador.

Las clases agregadas que componen la de analizador son:

- *Node*: El registro en la base de datos correspondiente puede estar vacío o ser un valor no nulo y único. Contiene información sobre el dispositivo sobre el cual reside el analizador (dirección de red, nombre de red etc).
- *Process*: Cero o uno. Información sobre el proceso en el que el analizador se está ejecutando.
- *Analyzer*: Cero o uno. Información sobre el analizador desde el que se envió el mensaje. La detrás del mecanismo es que cuando un gestor recive una alerta y quiere retransmitirla a otro analizador, necesita sustituir la información del analizador original por la suya propia. Para preservar la información del analizador original, debe ser incluida en la nueva definición del analizador. Esto permitirá trazar una ruta por los que ha pasado el mensaje.

Todo esto se representa en el DTD de XML como sigue:

```
<!ELEMENT Analyzer                    (
      Node?, Process?, Analyzer?
)>
<!ATTLIST Analyzer
      analyzerid          CDATA                  '0'
      name                CDATA                  #IMPLIED
      manufacturer        CDATA                  #IMPLIED
      model               CDATA                  #IMPLIED
      version             CDATA                  #IMPLIED
      class               CDATA                  #IMPLIED
      ostype              CDATA                  #IMPLIED
      osversion           CDATA                  #IMPLIED
>
```

La clase analizador tiene ocho atributos: *analyzerid* (identificador único para el analizador). Dicho atributo es parcialmente opcional. Si el analizador hace uso del atributo "ident", el mismo que se usa en otras clases para identificar de manera única un objeto, debe proveer de un "analyzerid" válido ([5]).

El nombre (*name*) corresponde a uno explícito para el analizador. El resto de los atributos son también opcionales *manufacturer* (fabricante del analizador), *model* (nombre o número del modelo del software y/o hardware del analizador), *version* (número de versión del software y/o hardware del analizador), *class* (clase del analizador), *ostype* (nombre del sistema operativo, correspondiente al

[5] Este requerimiento es dictaminado por el requerimiento de unicidad del atributo *ident*.

valor devuelto por el comando “uname -s” o por la llamada al sistema uname()), *osversion* (versión del sistema operativo, correspondiente al valor devuelto por “uname -r”)

3.3.2.3.2 La clase de clasificación (*Classification*).

La clase de clasificación provee del “nombre” de una alerta, u otra información perteneciente al gestor para determinar qué es. Este nombre es seleccionado por el proveedor de la alerta. Las clases agregadas que la componen se muestran en la figura 9.

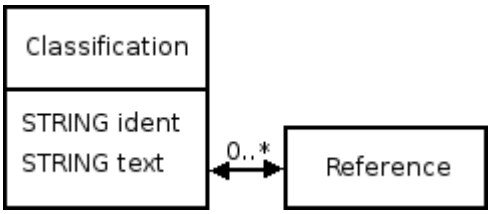

Figura 10: Clase de clasificación

La clase *Reference* es la única clase agregada que conforman la de clasificación. Puede ser cero o múltiple, y almacena la información sobre el mensaje apuntando a un lugar externo de documentación como un enlace a un sitio web, por ejemplo.

La representación en XML DTD es la siguiente:

```
<!ELEMENT Classification                (
      Reference?
)>
<!ATTLIST Classification
      ident               CDATA                   '0'
      text                CDATA                   #REQUIRED
>
```

Los atributos de la clase son *ident* y *text*. El primero de ellos es opcional pero si existe, deberá ser único. El texto del mensaje es requerido para identificar un mensaje provisto por un vendedor específico.

3.3.2.3.3 La clase de referencia (*Reference*).

La clase de referencia provee de un “nombre” de una alerta o de la información para determinar qué es. Conforma la clase de clasificación y está compuesta a su vez por dos clases agregadas:

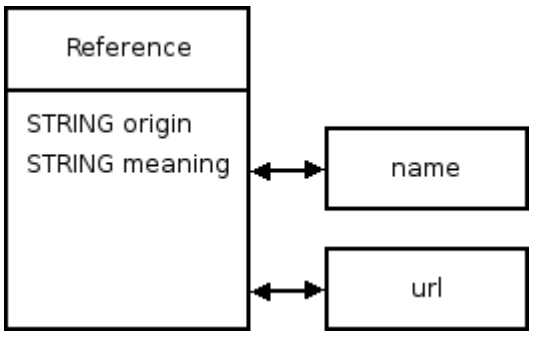

Figura 11: Clase de referencia.

- *name*: Exactamente uno, tipo cadena. Es el nombre de la alerta, de uno de los origenes listados debajo.

- *url*: Exactamente almacena una URL en la cual el gestor (o el operador humano del mismo) pueda encontrar información adicional sobre la alerta. El documento apuntado por la URL puede incluir una descripción en profundidad de el ataque, y otra información considerada relevante por el vendedor.

Esto se representa en el DTD de XML como sigue:

```
<!ENTITY % attvals.origin                "
      ( unknown | vendor-specific | user-specific | bugtraqid |
      cve | osvdb )
">
<!ELEMENT Reference                (
      name, url
)>
<!ATTLIST Reference
      origin              %attvals.origin;        'unknown'
>
```

La clase de referencia tiene dos atributos: *origin*, el cual es requerido y corresponde a la fuente de la cual se originó el nombre de la alerta, y *meaning* (significado de la referencia, comprendido por el proveedor de la alerta. Sólo es válido el valor de este campo si *origin="vendor-specific"* o *origin="user-specific"*. Los valores permitidos por este atributo *origin* se muestran en la tabla 1.

Valor	Palabra clave	Descripción
0	Unknown	El origen del nombre no es conocido
1	Vendor-specific	Un nombre de vendedor específico (y por lo tanto, una URL); esto puede ser usado para proveer la información de un producto específico.
2	User-specific	Un nombre de usuario específico (y por tanto, URL); esto puede ser usado para proveer de información para una instalación específica.
3	Bugtraqid	El identificador de base de datos de vulnerabilidad de focos de seguridad (Bugtraq) http://www.securityfocus.com/vdb
4	Cve	Nombres de vulnerabilidades y exposiciones comunes. (http://www.cve.mitre.org/)
5	Osvdb	Base de datos de vulnerabilidades de software libre. (http://osvdb.org)

Tabla 1 Posibles valores y significados para el campo ***origin***.

3.3.2.3.4 La clase de fuente (*Source*).

La clase de fuente contiene información sobre las posibles fuentes de los eventos que generaron una alerta. Un evento puede tener más de una fuente, como en ataques distribuidos. Esta clase está compuesta por las siguientes clases agregadas, tal y como puede contemplarse en la figura 11.

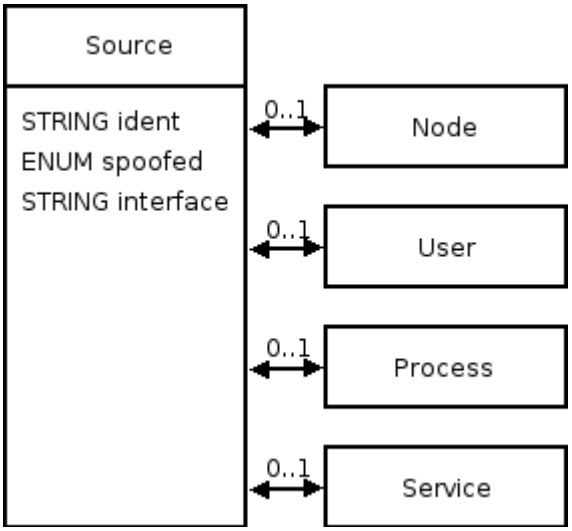

Figura 12: Clase de fuente.

- *Node*: Información sobre la máquina o dispositivo que parece ser el causante del evento (dirección de red, nombre, etc)
- *User*: Información sobre el usuario que parece ser el causante del evento/s.
- *Process*: Información sobre el proceso que parece ser el causante del evento/s.
- *Service*: Información sobre el servicio de red involucrado en el/los evento/s.

Representación en XML:

```
<!ENTITY % attvals.yesno                "
      ( unknown | yes | no )
">
<!ELEMENT Source                        (
      Node?, User?, Process?, Service?
)>
<!ATTLIST Source
      ident             CDATA                   '0'
      spoofed           %attvals.yesno;         'unknown'
      interface         CDATA                   #IMPLIED
>
```

Atributos: *ident* (identificador único para esta fuente), *spoofed* (indicada si la fuente es una trampa. Los valores posibles se detallan en la tabla 2), y *interface* (Puede ser usado por un analizador basado en red con múltiples interfaces e indicar qué interfaz fue detectada).

Valor	Palabra clave	Descripción
0	Unknown	Exactitud de la información sobre la fuente desconocida.
1	Yes	Se cree que la fuente es una trampa.
2	No	Se cree que la fuente es “auténtica”.

Tabla 2 Posibles valores y significados para el atributo ***spoofed****.*

3.3.2.3.5 La clase de objetivo (*Target*).

Contiene información sobre los posibles objetivos de los eventos que generaron una alerta. Un evento puede tener más de un destino, como el caso de un escaneo de puertos, por ejemplo.

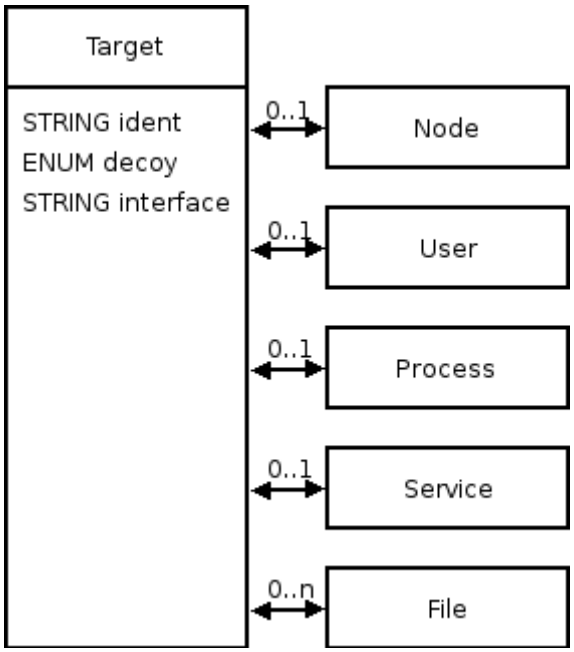

Figura 13: Clase de objetivo.

Las clases agregadas son las mismas que para la clase de fuente, a excepción de la inclusión opcional del la de fichero (*file*), en el caso de que haya uno o más ficheros involucrados en el evento.

```
<!ENTITY % attvals.yesno                "
      ( unknown | yes | no )
">
<!ELEMENT Target                        (
      Node?, User?, Process?, Service?, File*
)>
<!ATTLIST Target
      ident             CDATA                   '0'
      decoy             %attvals.yesno;         'unknown'
      interface         CDATA                   #IMPLIED
>
```

Atributos: *ident* (identificador único para el objetivo), *decoy* (indicada si el destino es una trampa. Los valores posibles se detallan en la tabla 3), y *interface* (Puede ser usado por un analizador basado en red con múltiples interfaces e indicar qué interfaz fue detectada).

Valor	Palabra clave	Descripción
0	Unknown	Exactitud de la información sobre la fuente desconocida.
1	Yes	Se cree que la fuente es una trampa.
2	No	Se cree que la fuente es “auténtica”.

Tabla 3 Posibles valores y significados para el atributo ***decoy****.*

3.3.2.3.6 La clase de Veredicto (*Assessment*).

Esta clase se usa para proveer del veredicto de un evento (su impacto, acciones tomadas en consecuencia y confidencia).

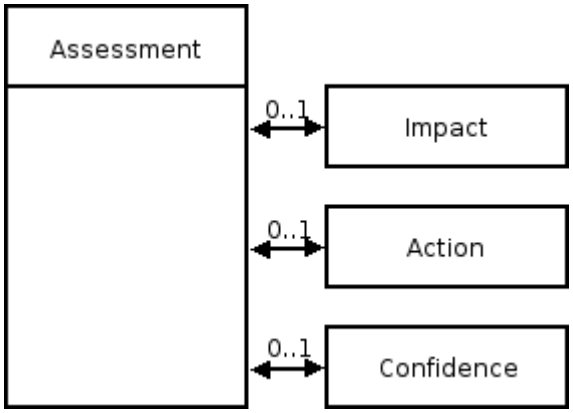

Figura 14: Clase *Assessment*.

Las clases agregadas que conforman a esta clase son, tal y como puede observarse en la figura anterior: *Impact*, *Action* y *Confidence*.

- *Impact*: El veredicto del analizador sobre el impacto del evento a un destino.
- *Action*: Acción/es tomada/s por el analizador como respuesta al evento.
- *Confidence*: Una medida de la confianza que el analizador tiene en la evaluación de un evento.

Esto se define en XML tal y como sigue:

```
<!ELEMENT Assessment                    (
      Impact?, Action*, Confidence?
)>
```

3.3.2.3.7 La clase de datos adicionales (*AdditionalData*).

La clase de datos adicionales es usada para proveer de información que no puede ser representada por el modelo de datos, la cual puede ser atómica (tipos enteros, cadenas etc), en casos que sólo necesita enviarse pequeñas cantidades de información adicional. También puede usarse para extender el modelo de datos con el fin de que el DTD soporte la transmisión de datos complejos.

Esta clase se define en XML con las siguientes líneas:

```
<!ENTITY % attvals.adtype               "
      ( boolean | byte | character | date-time | integer |
        ntpstamp | portlist | real | string | byte-string | xml )
">
<!ELEMENT AdditionalData            ANY >
<!ATTLIST AdditionalData
      type              %attvals.adtype;        'string'
      meaning           CDATA                   #IMPLIED
>
```

Atributos: son el tipo (*type*), y el significado del mismo (*meaning*). El primero es obligatorio y permite los valores que se muestran en la tabla 3, siendo el valor por defecto “string”. El segundo atributo es opcional y describe el significado del contenido de un elemento. Estos valores serán dependientes la implementación del vendedor.

Valor	Palabra clave	Descripción
0	Boolean	El elemento contiene un valor booleano: “true” o “false”.
1	Byte	El elemento contenido es una palabra simple de 8 bits.
2	Character	El elemento contenido es un carácter simple.
3	Date-time	El elemento contenido es una cadena de fecha.
4	Integer	El elemento contenido es un entero.
5	Ntpstamp	El elemento contenido es un *NTP timestamp*.
6	Portlist	El elemento contenido es una lista de puertos.
7	Real	El elemento contenido es un númeor real.
8	String	El elemento contenido es una cadena.
9	Byte-string	El elemento contenido es un vector de palabras byte[]
10	Xml	El elemento contenido es un dato XML.

*Tabla 4 Valores del campo **type** en la clase de datos adicionales.*

3.3.2.4. Las clases de tiempo.

El modelo de datos provee tres clases para representar el tiempo. Estas clases son agregadas de las clases de alerta y pulsación (*Alert* y *Heartbeat*).

3.3.2.4.1 La clase de tiempo de creación (*CreateTime*).

La clase de tiempo de creación es usada para indicar la fecha y hora de la alerta o pulsación cuando fue creada por el analizador. Se representa en XML como sigue:

```
<!ELEMENT CreateTime          (#PCDATA) >
<!ATTLIST CreateTime
      ntpstamp            CDATA                   #REQUIRED
>
```

El único atributo es requerido: *ntpstamp*. El NTP timestamp representa la misma fecha y hora que el elemento contenido.

Si la fecha y hora representada por el elemento contenido y el NTP timestamp difieren (algo que nunca debería suceder), debe usarse el valor en el *NTP timestamp*.

3.3.2.4.2 La clase de tiempo de detección (*DetectTime*).

La clase de detección de tiempo es usada para indicar la fecha y hora de los eventos que producen una alerta que fue detectada por el analizador. En caso de más de un evento, la hora almacenada es la del primer evento. Esta hora podría o no coincidir con la de *CreateTime*, ya que no se requiere que los analizadores envíen las alertas de forma inmediata tras la detección. Todo esto se representa en XML DTD como sigue:

```
<!ELEMENT DetectTime          (#PCDATA) >
<!ATTLIST DetectTime
      ntpstamp            CDATA                   #REQUIRED
>
```

Atributos: Sólo posee un atributo y requerido siempre. Se trata de *NTP timestamp* el cual representa la fecha y hora del elemento contenido. Si el dato y hora representado por el elemento contenido y el *NTP timestamp* difieren (algo que al igual que en la clase de creación de tiempos, tampoco debería suceder nunca), el valor usado debe ser el del NTP timestamp.

3.3.2.4.3 La clase de tiempo de analizador (*AnalyzerTime*).

La clase de tiempo de analizador se usa para indicar la fecha y hora actual del analizador. Su valor debería ser rellenado lo más tarde posible en el proceso de transmisión del mensaje (idealmente justo después de colocar el mensaje en el cable.

```
<!ELEMENT AnalyzerTime        (#PCDATA) >
<!ATTLIST AnalyzerTime
      ntpstamp            CDATA                    #REQUIRED
>
```

Atributos: De manera análoga a la clase de tiempo de detección y creación posee unicamente el atributo *NTP timestamp*, el cual debe ser también el valor por defecto tomado en el caso de que no coincidiesen "timestamp" con el valor representado. Puede ser usado como método rudimentario de sincronización entre analizadores y gestores.

3.3.2.5. Las clases de Veredicto (*Assessment*).

El modelo de datos provee tres tipos de "assessments" que un analizador puede hacer sobre un evento.

3.3.2.5.1 La clase de impacto (*Impact*).

Esta clase se usa para proveer del "assessment" del analizador del impacto del evento sobre el objetivo. Se representa como sigue en XML:

```
<!ENTITY % attvals.severity             "
      ( info | low | medium | high )
">
<!ENTITY % attvals.completion           "
      ( failed | succeeded )
">
<!ENTITY % attvals.impacttype           "
      ( admin | dos | file | recon | user | other )
">
<!ELEMENT Impact              (#PCDATA) >
<!ATTLIST Impact
      severity            %attvals.severity;      #IMPLIED
      completion          %attvals.completion;    #IMPLIED
      type                %attvals.impacttype;    'other'
>
```

Atributos: *severity* (estimación de la gravedad relativa del evento. Los posibles valores se detallan en la tabla 5 ([6])), *completion* (campo booleano en el que el analizador indica si el evento se completó o no) y *type* (tipo de impacto, según sea de administrador, dispositivo, fichero etc. El valor por defecto es "other", y los valores posibles se encuentran en la tabla 6).

[6] No hay valor por defecto para el campo de *severity* ni para el de *completion*..

Valor	Palabra clave	Descripción
0	Info	La alerta representa actividad informacional. Por ejemplo, el borrado de un fichero de configuración, de log o sistema.
1	Low	Gravedad baja.
2	Medium	Gravedad media.
3	High	Gravedad alta.

*Tabla 5 Posibles valores y significados para el atributo **severity**.*

Valor	Palabra clave	Descripción
0	Admin	Han sido obtenidos o pretendidos por una tentativa los privilegios de administrador
1	Dos	A denial of service was attempted or completed
2	File	Una acción de un fichero fue ATTEMPTED o completada.
3	Recon	A reconnaissance probe was attempted or completed.
4	User	Han sido obtenidos o pretendidos por una tentativa los privilegios de usuario
5	Other	Cualquier tipo no contenido en las anteriores categorías.

*Tabla 6 Posibles valores y significados para el atributo **type**.*

Todos los atributos son opcionales. Cada elemento podría estar vacío o contener una descripción textual del impacto si el analizador es capaz de proveer detalles adicionales.

3.3.2.5.2 La clase de acción (*Action*).

La clase de acción se usa para describir alguna acción tomada por el analizador en respuesta al evento. Se representa en XML DTD como sigue:

```
<!ENTITY % attvals.actioncat            "
      ( block-installed | notification-sent | taken-offline |
        other )
">
<!ELEMENT Action              (#PCDATA) >
<!ATTLIST Action
      category           %attvals.actioncat;     'other'
>
```

El único atributo de esta clase es la categoría (*category*), la cual indica el tipo de la acción llevada a cabo. Los valores permitidos se muestran abajo, siendo “other” el valor por defecto.

Valor	Palabra clave	Descripción
0	Block-installed	Un bloque de algún SORT fuen instalado para prevenir un ataque. El bloque podría ser un bloque de puertos, de direcciones, etc, o de desactivación de un usuario.
1	Notification-sent	Un mensaje de noficación sobre algo ordenado que fue enviado fuera de banda (vía paginador, e-mail, etc). No incluye la transmisión de esta alerta.
2	Take-offline	Un sistema, computadora o usuario pasó a fuera de línea, y también cuando la computadora se apagó o un usuario se salió de su sesión.
3	Other	Todo lo no comprendido en las anteriores opciones.

*Tabla 7 Posibles valores y significados para el atributo **category**.*

3.3.2.5.3 La clase de confianza (*Confidence*).

La clase de confianza se usa para representar la mejor estimación del analizador para validar su análisis. Se representa en XML como sigue, en la definición del tipo de documento.

```
<!ENTITY % attvals.rating                "
      ( low | medium | high | numeric )
">
<!ELEMENT Confidence          (#PCDATA) >
<!ATTLIST Confidence
      rating              %attvals.rating;        'numeric'
>
```

Atributo: *rating*. Es el único atributo de la clase y representa la valoración de la capacidad de un analizador para validar un análisis. Los valores permitidos se muestran en la tabla 8, siendo el valor por defecto “numeric”.

Valor	Palabra clave	Descripción
0	Low	El analizador tiene poca confianza en su capacidad de validación.
1	Medium	El analizador tiene confianza media para su capacidad de validación.
2	High	El analizador tiene gran confianza en su capacidades de validación.
3	Numeric	El analizador ha provisto de un valor posterior de probabilidad indicando su confianza en su capacidad de validación.

*Tabla 8 Posibles valores y significados para el atributo **rating**.*

Este elemento debería ser usado sólo cuando el analizador pudiera producir información significativa. Los sistemas sólo deberían usar “low”, “medium” o “high” como valor de *rating*. En este caso, el elemento contenido debería ser omitido.

Los sistemas capacitados para producir estimaciones de probabilidad razonables deberían usar el valor “numeric” en el campo *rating* e incluir un valor numérico en el valor de la confianza del

elemento contenido. Este valor numérico debería reflejar una probabilidad posterior (la probabilidad de que haya ocurrido un ataque dado por la vista de los datos por parte del sistema de detección y el modelo usado por el sistema). Es un número en coma flotante entre 0.0 y 1.0, ambos inclusive ([7]).

3.3.2.6. Las clases de soporte (*Support*).

Las clases de soporte conforman la mayor parte de las clases del núcleo, y se comparten entre ellas.

3.3.2.6.1 La clase nodo (*Node*).

La clase nodo se usa para identificar computadoras y otros dispositivos de red como encaminadores, puentes etc. Esta clase está compuesta por tres clases agregadas, mostradas en la figura siguiente:

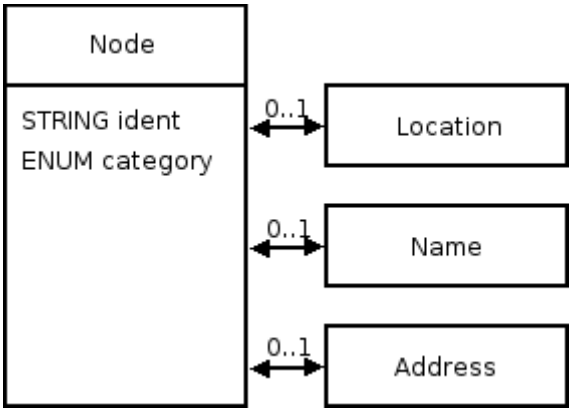

Figura 15: La clase Nodo.

A continuación se detalla una breve descripción de las clases agregadas:

- *Location*: Localización del equipo (variable tipo cadena).
- *Name*: El nombre del equipo. Esta información debe ser facilitada si no se da información sobre la dirección.
- *Address*: La dirección de red (IP) o física del equipo (MAC). Debe especificarse una de las direcciones a menos que se haya facilitado un nombre.

La representación en el DTD de XML queda de la siguiente manera:

```
<!ENTITY % attvals.nodecat                "
      ( unknown | ads | afs | coda | dfs | dns | hosts |
        kerberos | nds | nis | nisplus | nt | wfw )
">
<!ELEMENT Node                            (
      location?, (name | Address), Address*
)>
<!ATTLIST Node
      ident               CDATA                   '0'
      category            %attvals.nodecat;       'unknown'
>
```

[7] El número de dígitos debería estar limitado por el valor representable de un número en coma flotante de precisión simple.

Debería notarse que diferentes tipos de analizadores podrían computar valores de confianza en distintos caminos y que en la mayoría de casos, los valores de confianza de distintos analizadores no deberían ser comparados (por ejemplo, si el analizador usa diferentes métodos de computación o representación de la confianza, o si hay distintos tipos de configuraciones).

Atributos: La clase Nodo tiene dos atributos, los cuales son ambos opcionales: *ident* (identificador únoco para el nodo) y *category* (Dominio desde el cual se obtuvo la información sobre el nombre, en caso de que sea relevante. Los valores permitidos se muestran en la tabla bajo estas líneas y el valor por defecto es "unknown" o desconocido).

Valor	Palabra clave	Descripción
0	Unknown	Dominio desconocido o irrelevante.
1	Ads	Windows 2000 Advanced Directory Services
2	Afs	Andrew File System (Transarc)
3	Coda	Coda Distributed File System
4	Dfs	Distributed File System (IBM)
5	Dns	Domain Name System
6	Hosts	Local hosts file
7	Kerberos	Kerberos realm
8	Nds	Novell Directory Services
9	Nis	Network Information Services (Sun)
10	Nisplus	Network Information Services Plus (Sun)
11	Nt	Windows NT domain
12	Wfw	Windows for Workgroups

Tabla 9 Posibles valores y significados para el atributo ***category***.

3.3.2.6.2 La clase de direcciones (*Address*)

La clase Dirección se usa para representar las direcciones de red, física y de aplicación. Está compuesta por dos clases agregadas, tal como se muestra en la figura siguiente:

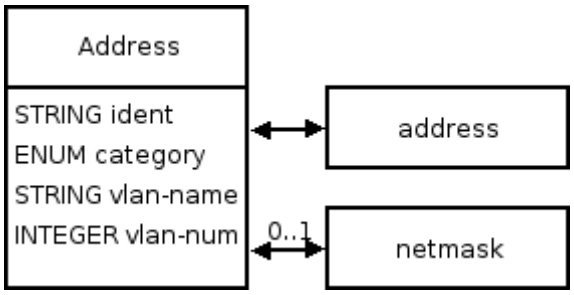

Figura 16: La clase de direcciones.

Las clases agregadas que se ven a la derecha de la figura se detallan a continuación.

- *Address*: Contiene información de la dirección. El formato viene condicionado por el atributo *category*.

- *Netmask*: Máscara de red para la dirección, si es apropiada.

Representación en XML.

```
<!ENTITY % attvals.addrcat              "
      ( unknown | atm | e-mail | lotus-notes | mac | sna | vm |
        ipv4-addr | ipv4-addr-hex | ipv4-net | ipv4-net-mask |
        ipv6-addr | ipv6-addr-hex | ipv6-net | ipv6-net-mask )
">
<!ELEMENT Address                       (
      address, netmask?
)>
<!ATTLIST Address
      ident               CDATA                   '0'
      category            %attvals.addrcat;       'unknown'
      vlan-name           CDATA                   #IMPLIED
      vlan-num            CDATA                   #IMPLIED
>
```

Atributos. La clase de direcciones contiene cuatro atributos:

ident. Un identificador único para la dirección, aunque puede atribuirse de forma opcional.

category. Corresponde al tipo de dirección representado. Los valores permitidos se muestran en la tabla 10, siendo el valor por defecto la cadena “unknown”.

vlan-name. Nombre de la red de área local virtual a la que pertenece la dirección. Opcional.

vlan-num. Número de la LAN virtual a la que pertenece la dirección. Opcional.

Valor	Palabra clave	Descripción
0	Unknown	Tipo de dirección desconocido.
1	Atm	Dirección de red de modo de transferencia asíncrono (ATM)
2	E-mail	Dirección de correo electrónico (RFC 822)
3	Lotus-notes	Dirección de correo de notas de Lotus.
4	Mac	Dirección de capa de control de acceso al medio (MAC)
5	Sna	Dirección de arquitectura de red compartida de IBM (SNA)
6	Vm	Dirección de correo IBM VM (“PROFS”)
7	Ipv4-addr	Dirección de red Ipv4 en notación decimal (a.b.c.d).
8	Ipv4-addr-hex	Dirección de red Ipv4 en notación hexadecimal.
9	Ipv4-net	Dirección de red Ipv4 en notación decimal/bits significativos (a.b.c.d/nn).
10	Ipv4-net-mask	Dirección de red/Máscara de red en notación decimal (a.b.c.d/w.x.y.z)
11	Ipv6-addr	Dirección de red Ipv6.
12	Ipv6-addr-hex	Dirección de red Ipv6 en notación hexadecimal.
13	Ipv6-net	Dirección de red Ipv6 en notación decimal/bits significativos
14	Ipv6-net-mask	Dirección de red/Máscara de red en notación decimal.

Tabla 10 Posibles valores y significados para el atributo ***category***.

3.3.2.6.3 La clase Usuario (*User*).

La clase de usuarios se usa plara describir a los usuarios con su identificador, por ejemplo. Se usa principalmente como contenedora de las clases agregadas de *UserId* como se muestra en la figura bajo estas líneas.

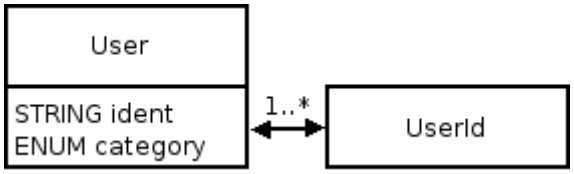

Figura 17: La clase de usuarios.

Las clases agregadas contenidas en la de usuario son:

- *UserId*: Identificación de un usuario.

Código XML para representación en el DTD:

```
<!ENTITY % attvals.usercat              "
      ( unknown | application | os-device )
">
<!ELEMENT User                          (
      UserId+
)>
<!ATTLIST User
      ident             CDATA                   '0'
      category          %attvals.usercat;       'unknown'
>
```

Atributos: la clase de usuario tiene dos atributos opcionales: *ident* (identificador único de usuario) y *category* (tipo de usuario representado).

Valor	Palabra clave	Descripción
0	Unknown	Tipo de usuario desconocido. Es el valor por defecto.
1	Application	Usuario de aplicación.
2	Os-device	Un usuario de dispositivo o sistema operativo

Tabla 11 Posibles valores y significados para el atributo ***category***.

3.3.2.6.4 La clase identificador de usuario (*UserId*).

La clase de identificador de usuario provee de información específica sobre un usuario. Más que un *UserId* puede ser usado dentro de la clase Usuario para indicar intentos de transición de un usuario a otro, o proveer de información completa sobre los privilegios de usuarios o procesos.

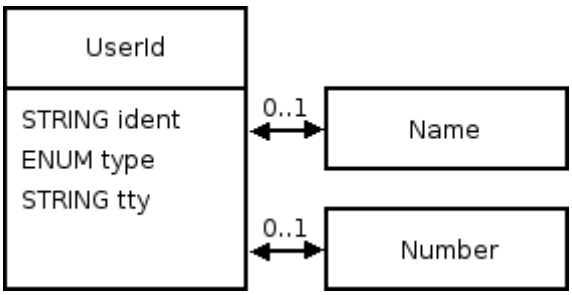

Figura 18: La clase Identificador de Usuario.

Las clases agregadas que conforman *UserId* son:

- *Name*: Nombre de usuario o grupo.
- *Number:* Número de usuario o grupo.

En la definición del tipo de documento XML queda como sigue:

```
<!ENTITY % attvals.idtype               "
      ( current-user | original-user | target-user | user-privs |
        current-group | group-privs | other-privs )
">
<!ELEMENT UserId                        (
      (name, number?) | (number, name?)
)>
<!ATTLIST UserId
      ident             CDATA                   '0'
      type              %attvals.idtype;        'original-user'
      tty               CDATA                   #IMPLIED
>
```

Atributos:

ident. Identificador único para el usuario. Puede existir de forma opcional.

type. El tipo de información de usuario representada. Los valores permitidos se muestran en la tabla 12, siendo el valor por defecto "original-user".

tty. Identificador único opcional de tipo cadena para el terminal del usuario.

Valor	Palabra clave	Descripción
0	Current-user	El identificador del usuario actual que está siendo utilizado por el usuario o proceso. En sistemas Unix, debería ser en general el identificador "auténtico".
1	Original-user	La identidad actual del usuario o proceso siendo informado. En aquellos sistemas que (a) haga algún tipo de auditoría y (b) soporte la extracción del identificador de usuario del testigo "audit id", debería usarse dicho valor "audit id". En los sistemas que no lo soporten, debería usarse el "login id".
2	Target-user	El identificador de usuario o proceso que está intentando usar, por ejemplo en el caso de sistemas Unix, "su", "rlogin", "telnet" etc.
3	User-privs	El usuario o proceso tiene la capacidad de usar otro identificador de usuario. (En sistemas Unix sería el usuario efectivo, o los permisos de propietario en un contexto de fichero).
4	Current-group	Identificador del grupo actual (si es aplicable) que está siendo usado por el usuario o proceso. (En sistemas Unix, es el id. grupo real)
5	Group-privs	Sería el equivalente al identificador de grupo efectivo en sistemas Unix. En Unix derivados de BSD, múltiples elementos de este tipo podrían ser usados para incluir los identificadores de grupo en una "lista de grupos".
6	Other-privs	No usado en contextos de usuario, grupo o proceso. Sólo usado en el contexto de ficheros.

*Tabla 12 Posibles valores y significados para el atributo **type**.*

3.3.2.7. La clase Proceso (*Process*).

La clase Proceso se usa para describir los procesos que están siendo ejecutados en los orígenes, destinos y analizadores.

Las cinco clases agregadas que lo componen se muestran en la figura 18:

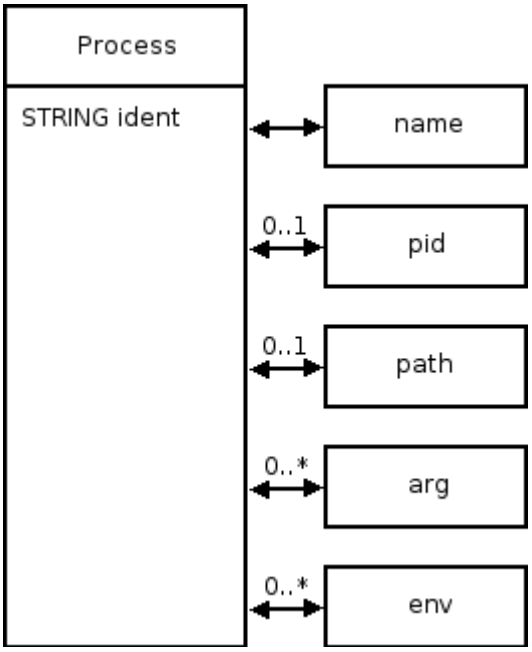

Figura 19: La clase de procesos.

Las clases agregadas que conforman la de Proceso son:

- *Name*: Siempre existente, de tipo cadena. Es el nombre del programa que está siendo ejecutado, pero no la ruta y argumento completo.
- *Pid:* Identificador del proceso.
- *Path:* Ruta completa del programa que está siendo ejecutado.
- *Arg:* Un argumento en línea de comando para el programa. Puede especificarse cero o múltiples argumentos, los cuales deben haberse producido en el mismo orden en el que son especificados.
- *Env*: Cadena de entorno asociada con el proceso; generalmente del formato "VARIABLE=valor". Se pueden especificar múltiples cadenas con distintos usos de env.

En la definición del tipo de documento XML queda como sigue:

```
<!ELEMENT Process                       (
      name, pid?, path?, arg*, env*
)>
<!ATTLIST Process
      ident             CDATA                   '0'
>
```

El único atributo es opcional y único. Se trata de un identificador para el proceso.

3.3.2.8. La clase de servicios (*Service*).

La clase Servicio describe servicios de red en orígenes y destinos. Puede identificarlos por el nombre, puerto y protocolo. Cuando ocurre un servicio como una clase agregada de la *Source*, se entiende que el servicio es una de aquellas actividades de interés que se están originando y se adjunta a la la información de nodo, proceso y usuario contenidas en la clase de fuentes. Del mismo modo, cuando ocurren un servicio como una clase agregada de la de objetivos (*Target*), se entiende que el servicio es receptor de una actividad de interés. Si el servicio sucede en ambos, en origen y destino, entonces la información de ambos lados debería ser la misma. Si la información es idéntica e implementa y se desea tener la información sólo en un sentido, debería especificarse en la clase *Target*.

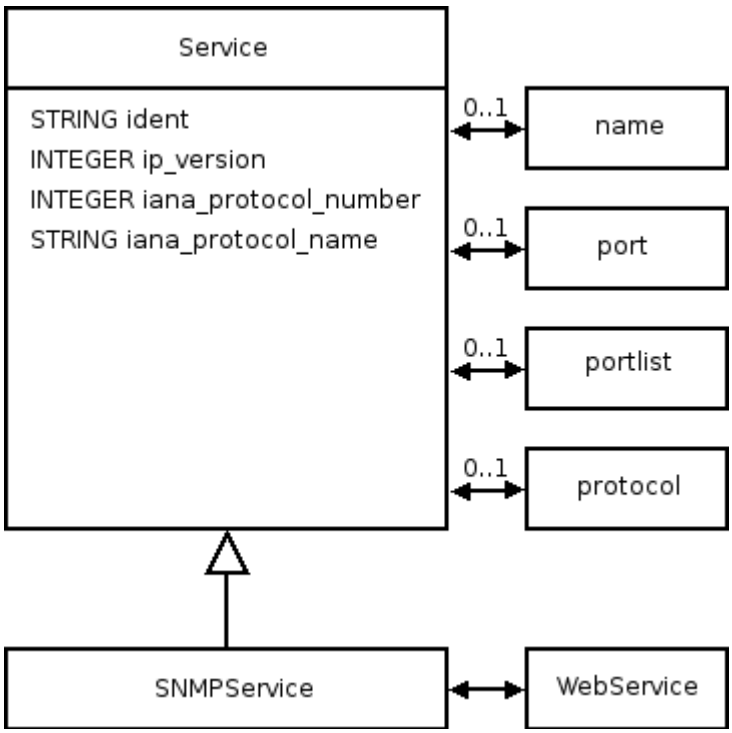

Figura 20: La clase Servicio.

Las clases agregadas que conforman la de Servicio son:

- *Name*: Nombre del servicio. Siempre que sea posible, debería usarse el nombre de la lista IANA de puertos conocidos.
- *Port*: El número del puerto que está siendo usado.
- *Portlist:* Lista de los números de puertos que están siendo usados. Las reglas de formateado. Para una lista de puertos dada, el *iana_protocol_number* e *iana_protocol_name* deben aplicarse a todos los elementos de la lista.
- *Protocol*: Información adicional sobre el protocolo que está siendo usado. El campo protocol porta información adicional relacionada con el protocolo usado cuando los atributos de servicio *iana_protocol_number* e *iana_protocol_name* son rellenados.

Un servicio debe ser especificado como, bien un nombre o puerto, o bien una lista de puertos. El protocolo es opcional en todos los casos.

Como el DTD no soporta ha herencia en subclases, la relación de herencia entre las clases *Service, SNMPService* y *WebService* mostradas en la figura 19 se sustituyen por una relación de agregación. Esta relación se define en XML como sigue:

```
<!ELEMENT Service                        (
      (((name, port?) | (port, name?)) | portlist?, protocol?,
      SNMPService?, WebService?
)>
<!ATTLIST Service
      ident                CDATA                  '0'
      ip_version           CDATA                  #IMPLIED
      iana_protocol_number CDATA                  #IMPLIED
      iana_protocol_name   CDATA                 #IMPLIED
>
```

Atributos.

ident. Identificador único y opcional para el servicio.

ip_version. Número de versión del protocolo IP.

iana_protocol_number. Número de protocolo de IANA.

iana_protocol_name. Nombre del protocolo de IANA.

3.3.2.8.1 La clase Servicio Web (*WebService*).

La clase de servicio web porta información adicional relacionada con el tráfico web. Está compuesta por cuatro clases agregadas, como se muestra en la figura 20.

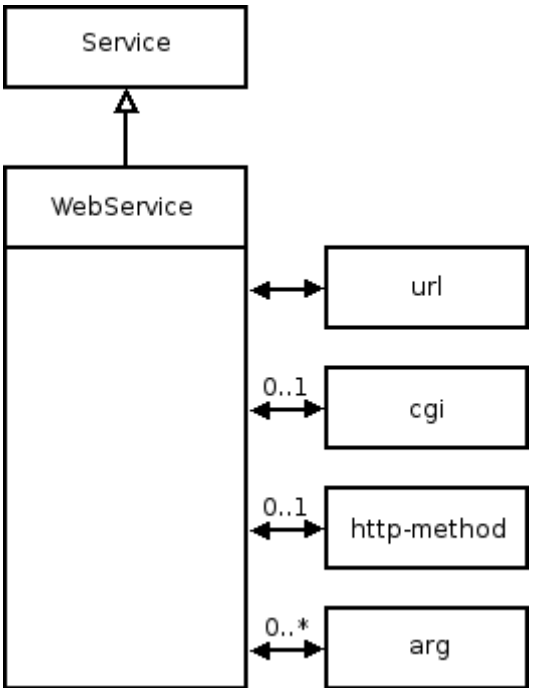

Figure 21: La clase Servicio Web

Las clases agregadas que conforman la de servicio web son:

- *Url*: Nombre del servicio. Siempre que sea posible, debería usarse el nombre de la lista IANA de puertos conocidos.
- *Cgi:* Guión de la interfaz de pasarela común (CGI) en la petición sin argumentos.

- *Http-method*: Método HTTP usado en la petición: *put*, *get*.
- *Arg*: Argumentos del guión del CGI.

Representación en XML:

```
<!ELEMENT WebService                    (
      url, cgi?, http-method?, arg*
)>
```

3.3.2.8.2 La clase Servicio SNMP.

La clase Servicio SNMP porta información adicional relacionada con el tráfico del protocolo de gestión SNMP.

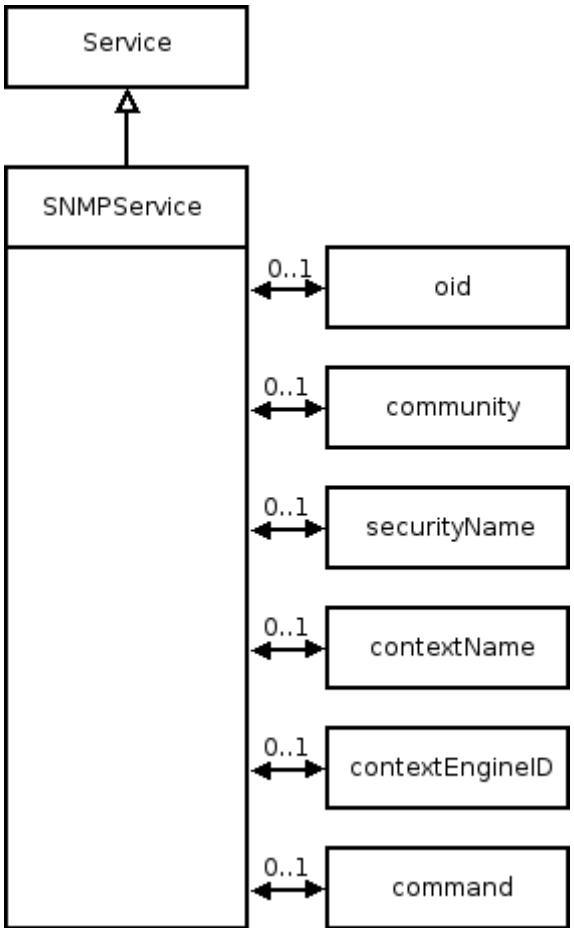

Figure 22: La clase de servicio SNMP.

Clases agregadas que conforman a la de Servicio SNMP.

- *Oid*: Identificador de objeto en la petición.
- *Community*: Cadena de la comunidad del objeto, si el tráfico usa la versión 1 o 2 de SNMP.
- *SecurityName*: Nombre de seguridad del objeto, si el tráfico usa SNMPv3.
- *ContextName*: Nombre del contexto del objeto, si el tráfico usa SNMPv3.
- *ContextEngineID*: Identificador del motor del contexto del objeto, si el tráfico usa SNMPv3.
- *Command*: Comando enviado al servidor SNMP (“get”, “set”, etc)

Código XML:

```
<!ELEMENT SNMPService                    (
      oid?, community?, command?
)>
```

3.3.2.9. La clase Fichero.

La clase de fichero provee de información específica sobre un fichero u otro objeto como ficheros pipe que haya sido creado, borrado o modificado sobre el objetivo. La descripción puede proveer cualquier configuración del fichero antes o al mismo tiempo que el evento, tal como se puede especificar en el atributo *category*.

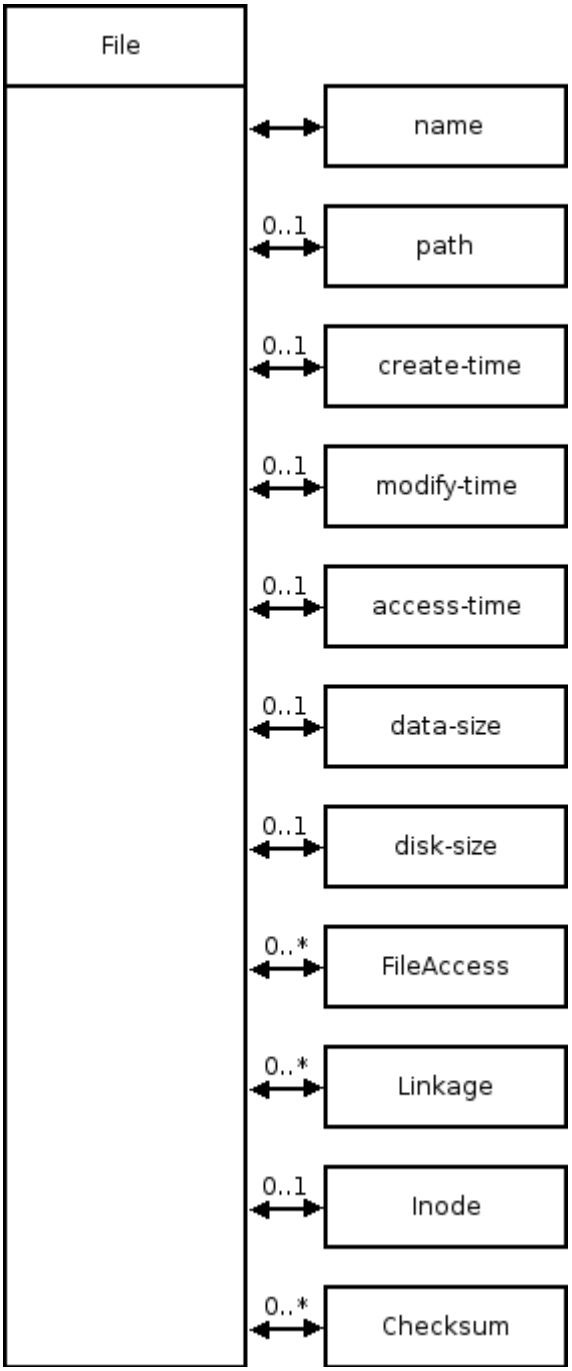

Figura 23: La clase Fichero.

Clases que conforman la de Fichero.

- *Name*: El nombre del fichero sobre el que se aplica la alerta, sin incluir la ruta al mismo.
- *Path*: Ruta completa al fichero, incluyendo el nombre. El nombre de ruta debería ser representado como "universal" en la medida de lo posible para facilitar el procesamiento de la alerta. En sistemas Windows, la ruta debería ser especificada usando la convención universal de nombrado (UNC) para ficheros remotos además de la letra de dispositivo para ficheros locales ("c:\boot.ini"). Para sistemas Unix, rutas sobre sistemas de ficheros de red deberían usar el nombre de los recursos montados en lugar del punto de montaje local ("fileserver:/usr/local/bin/foo"). El punto de montaje puede ser provisto por el elemento <Linkage>.
- *Create-time*: De tipo DATETIME, almacena la fecha y hora en la que el fichero fue creado. Nótese que es diferente del atributo "st_ctime" en sistemas Unix. Dicho atributo es contenido en la clase *Inode*.
- *Modify-time*: Fecha en la que el fichero sufrió la última modificación.
- *Access-time*: Fecha del último acceso al fichero.
- *Data-size*: El tamaño de los datos, en bytes. Para sistemas Windows, este valor corresponde al VDL, y para sistemas Unix, stat.st_size.
- *Disk-size*: Espacio físico en disco consumido por el fichero, en bytes. En sistemas de ficheros UFS (Unix file systems), este valor corresponde con 512 * stat.st_blocks. Sobre sistemas Windows NTFS, este valor se corresponde con EOF.
- *FileAccess*: Permisos de acceso al fichero.
- *Linkage*: Objetos del sistema de fichero a los que el fichero hace referencia o apunta.
- *Inode*: Información del nodo-i para ese fichero. (Sólo relevante en ficheros con estructura en nodos-i como Unix, no para particionado en FAT o NTFS.)
- *Checksum*: Información de chequeo del fichero.

Representación en el DTD de XML:

```
<!ENTITY % attvals.filecat              "
      ( current | original )
">
<!ELEMENT File                          (
      name, path, create-time?, modify-time?, access-time?,
      data-size?, disk-size?, FileAccess*, Linkage*, Inode?, Checksum*
)>
<!ATTLIST File
      ident             CDATA                   '0'
      category          %attvals.filecat;       #REQUIRED
      fstype            CDATA                   #IMPLIED
      file-type         CDATA                   #IMPLIED
>
```

Atributos.

ident: identificador de este fichero, opcional, que en el caso de existir es único.

category: El contexto para la información que está siendo provista. Los valores permitidos son mostrados en la tabla 13. No hay valor por defecto.

Valor	Palabra clave	Descripción
0	Current	La información del fichero proviene de antes del cambio informado.
1	Original	La información del fichero proviene de antes del cambio informado.

*Tabla 13 Posibles valores y significados para el atributo **category**.*

fstype: El tipo de sistema de fichero del fichero en el que reside. Este atributo gobierna cómo los nombres de ruta y otros atributos son interpretados. Los valores permitidos se muestran en la tabla 14 y tampoco hay valor por defecto.

file-type: Tipo de fichero como tipo MIME. Es de carácter opcional.

Valor	Palabra clave	Descripción
0	Ufs	Sistema de ficheros rápido de Berkeley UNIX.
1	Efs	Sistema de ficheros Linux “EFS”.
2	Nfs	Sistema de ficheros de red.
3	Afs	Sistema de ficheros de Andrew.
4	Ntfs	Sistema de ficheros de Windows NT.
5	Fat16	Sistema de ficheros 16-bit Windows FAT.
6	Fat32	Sistema de ficheros 32-bit Windows FAT.
7	Pcfs	Sistema de ficheros “PC” (MS-DOS) sobre CD-ROM.
8	Joliet	Sistema de ficheros Joliet CD-ROM.
9	Iso9660	Sistema de ficheros ISO 9660 CD-ROM.

*Tabla 14 Posibles valores y significados para el atributo **fstype**.*

3.3.2.9.1 La clase de acceso a ficheros (*FileAccess*).

La clase de acceso a ficheros representa los permisos de acceso a un fichero. Se entiende como una operación util para cruzar información contenida en dichos ficheros entre sistemas operativos.

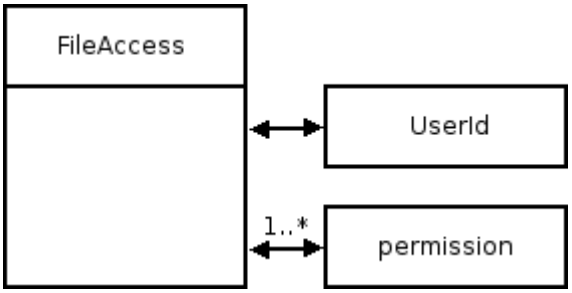

Figura 24: La clase de acceso a ficheros.

Clases agregadas que conforman la clase de acceso a ficheros.

- *UserId*: El usario (o grupo) sobre el que se aplican estos permisos. El valor de el atributo *type* debe ser “user-privs”, “group-privs” u “other-privs” según sea apropiado. Otros valores para el tipo no pueden ser usados en este contexto.
- *Permission*: Uno o más. Nivel de accesibilidad permitido. Los valores permitidos se muestran en la tabla 15. No hay valor por defecto.

Valor	Palabra clave	Descripción
0	NoAccess	No se le permite el acceso a este usuario.
1	Read	El usuario sólo tiene permiso de lectura del fichero.
2	Write	El usuario tiene acceso en escritura al fichero.
3	Execute	El usuario está habilitado para ejecutar el fichero.
4	Search	El usuario tiene la posibilidad de buscar el fichero (Se aplica a permiso “execute” en directorios en UFS).
5	Delete	El usuario tiene la capacidad de borrar el fichero.
6	ExecuteAs	El usuario tiene la capacidad de ejecutar el fichero como otro usuario. Equivale a *set-user-id* y *set-group-id* en Unix.
7	changePermissions	El usuario tiene la capacidad de cambiar los permisos de acceso a este fichero.
8	TakeOwnership	El usuario tiene la capacidad de tomar posesión de este fichero.

*Tabla 15 Posibles valores y significados para el atributo **permission**.*

Los valores “changePermissions” y “takeOwnership” representan esos conceptos en Windows. En Unix, el propietario del fichero siempre tiene permiso “changePermissions” incluso si no hay otro acceso permitido para ese usuario. “Full Control” se representa en Windows enumerando los permisos que contiene.

Representación en XML:

```
<!ENTITY % attvals.fileperm "( noAccess | read | write |
      execute | search | delete | executeAs |
      changePermissions | takeOwnership)" >
<!ELEMENT Permission EMPTY >
<!ATTLIST Permission
      perms                %attvals.fileperm;        #REQUIRED
      %attlist.global;
>
<!ELEMENT FileAccess                    (
      UserId, permission+
)>
```

3.3.2.9.2 La clase de enlazado (*Linkage*).

La clase de enlazado representa las conexiones del sistema de ficheros entre el fichero y un enlace simbólico en ese mismo sistema de ficheros. Por ejemplo, si el elemento <File> es un enlace simbólico o un acceso directo, <Linkage> debería contener el nombre del objeto al que apunta.

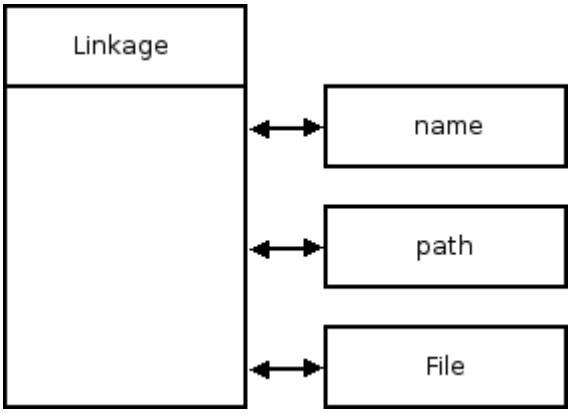

Figura 25: La clase de enlazado.

Las clases agregadas que conforman la de *Linkage* son:

- *Name*: El nombre del objeto del sistema de ficheros, sin incluir la ruta.
- *Path*: Ruta completa al objeto del sistema de ficheros, incluyendo el nombre.
- *File*: Se usa en lugar del nombre y ruta si va a incluirse información adicional.

Representación en XML:

```
<!ENTITY % attvals.linkcat              "
      ( hard-link | mount-point | reparse-point | shortcut |
        stream | symbolic-link )
">
<!ELEMENT Linkage                       (
      (name, path) | File
)>
<!ATTLIST Linkage
      category           %attvals.linkcat;       #REQUIRED
>
```

El único atributo de la clase de enlazado es la categoría (*categoría*), cuyos valores permitidos se detallan bajo estas líneas. No hay valor por defecto.

Valor	Palabra clave	Descripción
0	Hard-link	El elemento <nombre> representa otro nombre para este fichero. Esta información se puede obtener más fácilmente en sistemas NTFS que en otros.
1	Mount-point	Es un alias para el directorio especificado por los <nombre> y <ruta> del padre.
2	Reparse-point	Se aplica sólo a Windows; excluye enlaces simbólicos y puntos de montaje los cuales son tipos específicos o puntos "reparse".
3	Shortcut	El fichero es representado por un "acceso directo" de Windows. Un acceso directo se diferencia de un enlace simbólico en su contenido, que puede ser de importancia para el gestor
4	Stream	Flujo de datos suplente (ADS) en Windows o "fork" en MacOS.
5	Symbolic-link	El elemento <nombre> representa el fichero al cual el enlace apunta.

Tabla 16 Posibles valores y significados para el atributo ***category****.*

3.3.2.9.3 La clase de Nodo–I (*Inode*).

La clase de nodo-i se usa para representar información adicional contenida en un sistema de ficheros Unix de nodos-i. Está compuesta por seis clases agregadas, como se muestra en la figura 25.

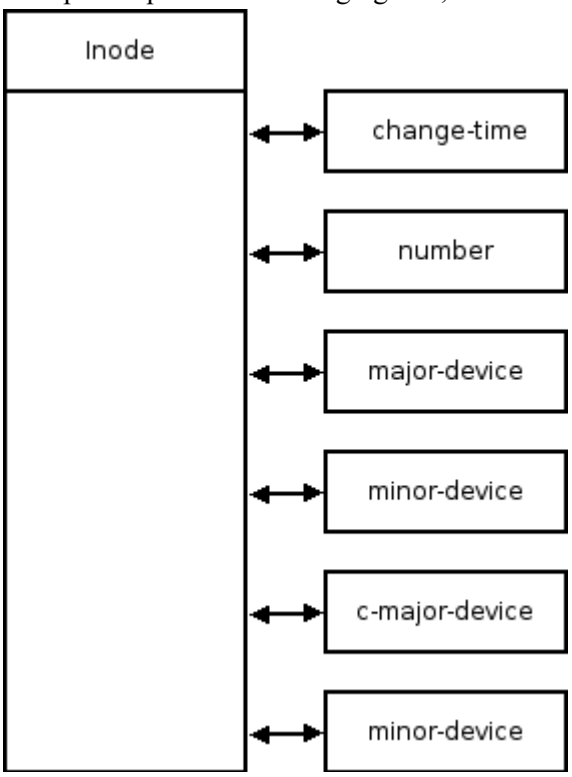

Figura 26: Clase de nodo-i.

- *Change-time*: Fecha del último cambio en el nodo-i, dado por el elemento *st_ctime* de "struct stat".
- *Number*: Número de nodo-i.
- *Major-device*: Número de dispositivo mayor del dispositivo en el que reside el fichero.
- *Minor-device*: Número de dispositivo menor del dispositivo en el que reside el fichero.
- *C-major-device:* Dispositivo mayor del fichero si es un dispositivo de carácter especial.
- *C-menor-device:* Dispositivo menor del fichero si es un dispositivo de carácter especial.

Siempre que exista uno de los dos últimos debe existir el otro, y siempre que exista el número, dispositivo mayor o dispositivo menor, deben existir los otros. Esto se representa en XML imponiendo existencia conjunta entre paréntesis:

```
<!ELEMENT Inode                         (
      change-time?, (number, major-device, minor-device)?,
      (c-major-device, c-minor-device)?
)>
```

3.3.2.9.4 Clase de suma de verificación (*Checksum*).

La clase de suma de verificación representa la información asociada con el control de redundancia del fichero. Esta información puede ser provista por un chequeo de la integridad de los ficheros. Las dos clases agregadas son las correspondientes al valor y la clave. El software *Samhain* © posee salida configurable para prelude y comprueba la consistencia del sistema de ficheros.

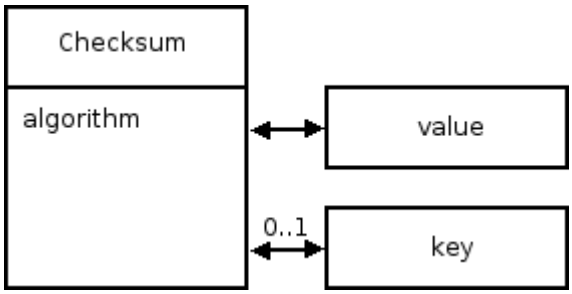

Figura 27: Clase *Checksum*.

La clase *Checksum* está formada por las siguientes clases agregadas.

- *Value:* Valor de la suma de verificación.
- *Key*: Clave de la suma de verificación, si procede.

```
<!ENTITY % attvals.checksumalgos        "
      ( MD4 | MD5 | SHA1 | SHA2-256 | SHA2-384 | SHA2-512 |
        CRC-32 | Haval | Tiger | Gost )
">
<!ELEMENT Checksum                      (
      value, key?
)>
<!ATTLIST Checksum
      algorithm          %attvals.checksumalgos; #REQUIRED
>
```

algorithm es el algoritmo criptográfico de esta clase usado para la computación de la verificación. Los valores permitidos se muestran en la tabla 17.

Valor	Palabra clave	Descripción
0	MD4	Algoritmo MD4.
1	MD5	Algoritmo MD5.
2	SHA1	Algoritmo SHA1.
3	SHA2-256	Algoritmo SHA2 con 256 bits de longitud.
4	SHA2-384	Algoritmo SHA2 con 384 bits de longitud.
5	SHA2-512	Algoritmo SHA2 con 512 bits de longitud.
6	CRC-32	Algoritmo de código de redundancia cíclica de 32 bits de longitud.
7	Haval	Algoritmo Haval.
8	Tiger	Algoritmo Tiger.
9	Gost	Algoritmo Gost.

Tabla 17 Posibles valores y significados para el atributo ***algorithm***.