《+》
安全加社区
公益
译文
项目
2017

入侵检测消息交换格式（IDMEF）

文档信息			
原文名称	The Intrusion Detection Message Exchange Format (IDMEF)		
原文作者	France Telecom H. Debar SecureWorks, Inc. D. Curry, Guardian, B. Feinstein	原文发布日期	2007 年 3 月
作者简介	无		
原文发布单位	互联网工程任务组（IETF）		
原文出处	http://www.ietf.org/rfc/rfc4765.txt		
译者	小蜜蜂公益翻译组	校对者	小蜜蜂公益翻译组

免责声明

• 本文原文来自于互联网的公共方式，由"安全加"社区出于学习交流的目的进行翻译，而无任何商业利益的考虑和利用，"安全加"社区已经尽可能地对作者和来源进行了通告，但不保证能够穷尽，如您主张相关权利，请及时与"安全加"社区联系。

•"安全加"社区不对翻译版本的准确性、可靠性作任何保证，也不为由翻译不准确所导致的直接或间接损失承担责任。在使用翻译版本中所包含的技术信息时，用户同意"安全加"社区对可能出现的翻译不完整、或不准确导致的全部或部分损失不承担任何责任。用户亦保证不用做商业用途，也不以任何方式修改本译文，基于上述问题产生侵权行为的，法律责任由用户自负。

本文状态

本文为互联网社区的实验协议，而非任何形式的联网标准，尚需讨论并根据建议进行改进。本文的分发不受限制。

版权声明

IESG 说明

本文内容由互联网工程任务组（IETF）议定，之后被认定为标准跟踪协议。本文并不会成为任何级别的互联网标准。对于本文用于任何目的的适当性的界定，IETF 在此声明，并不知晓 RFC 是否适用于某一特定目的。特别是，发布通知前 IETF 并未进行全面评审，评估其安全性、拥塞控制及是否与已部署协议无法交互等。IESG 选择发布该文档，是为了在工作组完成相应工作时进行书面记录，以鼓励技术开发和实验。读者应谨慎评估本文在实施和部署方面的价值。

摘要

《入侵检测消息交换格式（IDMEF）》就共享信息给入侵检测与响应系统以及可能与这些系统进行交互的管理系统规定了数据格式与交换过程。

本文描述了表示入侵检测系统输出信息的数据模型，并解释了使用此模型的基本原理。本文介绍了可扩展标记语言（XML）中的数据模型实现，进行了 XML 文档类型定义，并提供示例。

目录

1. 引言

《入侵检测消息交换格式（IDMEF）》旨在定义标准的数据格式，自动化入侵检测系统使用该格式对可疑事件发出告警。开发该标准格式可实现商业系统、开源系统和研究系统之间的互通性，允许用户根据各个系统的优缺点进行混合部署，以达到最佳实现效果。

显然，IDMEF 适用于入侵检测分析器（或称为“传感器”）和接收告警的管理器（或称为“控制台”）之间的数据信道。但是，IDMEF 在其他地方也可发挥作用，比如：

- 单一的数据库系统在应用 IDMEF 后可存储来自各种入侵检测产品的结果的数据库系统的结果，能够从全局角度而不只是片面地进行数据分析和活动报告。
- 事件关联系统在应用 IDMEF 后可接收来自各种入侵检测产品的告警，与只接收单一产品告警的系统相比，可执行更复杂的交叉关联和确认。
- 图形用户界面可以显示来自各种入侵检测产品的告警，用户可以在一个屏幕上监控所有产品，并且只需学习一个接口；以及
- 通用数据交换格式便于不同组织（用户、厂商、响应团队和执法部门）之间进行数据交换和沟通。

在选择实现方法时应考虑 IDMEF 用途的多样性。

1.1 关于 IDMEF 数据模型

IDMEF 数据模型以面向对象的形式表示入侵检测分析器发送给管理器的告警数据。

1.1.1. 数据模型解决的问题

数据模型解决了与入侵检测告警数据表示方法相关的几个问题：

- 告警信息固有的异构性。有些告警提供的信息非常少，比如事件起源、目的地址、名称和时间。有些告警的信息非常多，比如端口、服务、进程、用户信息等。表示这些信息的数据模型必须能够灵活满足不同的需求。

一般而言，面向对象的模型可通过聚合和子类化扩展。数据模型实现通过聚合或子类化进行扩展时（这样改是因为扩展的是模型，而不是分类，通过增加分类来扩展模型），即使不了解新增内容，也能够理解数据模型定义的信息子集。子类化和聚合提供了可扩展性，同时也保持了模型的一致性。

入侵检测环境的多变性。有些分析器通过分析网络流量来检测攻击，而有些则利用操作系统日志或应用审计跟踪信息。不同信息源的分析器针对同一攻击发送的告警信息也不尽相同。

数据模型定义了可适应不同分析器之间数据源差别的支持类。尤其是，通过 Node、

Process、Service 和 User 类来表示告警攻击源和攻击目标。

- 分析器的能力不尽相同。根据环境需求，用户可以安装一个提供很少告警信息的轻量级分析器，也可以安装较为复杂的分析器。虽然后者对运行系统有更大的影响，但能够提供更详细的告警信息。数据模型必须允许将数据格式转换为入侵检测分析器以外的其他工具所使用的格式，以便进一步处理告警信息。

数据模型定义了基本文档类型定义（DTD）的扩展，告警可简可繁。扩展通过子类化或新类实现。

- 操作环境不尽相同。根据所用的网络或操作系统的类型，所观察和上报的攻击具有不同特征。数据模型应适应这些差异。

Node 和 Service 支持类可提供重要的上报灵活性。如果必须上报附加信息，可以定义子类，赋予数据模型附加属性。

- 商业化厂商的目标不尽相同。出于各种原因，厂商可能希望对特定类型的攻击提供较多或较少信息。

面向对象的方法提供了这种灵活性，而子类化规则保持了模型的完整性。

1.1.2. 数据模型设计目标

设计数据模型的目标是为告警提供标准表达方式，并描述简单告警和复杂告警之间的关系。

1.1.2.1. 事件表达

当入侵检测分析器检测到某些异常事件时，会上报告警信息。数据模型的目标是提供对这些告警信息的标准化表达。分析器能力不同，生成的告警内容也不同，有的简单，有的复杂。

1.1.2.2. 内容驱动

数据模型的设计是以内容为驱动的。这意味着将引进新的对象以适应附加内容，而不是告警之间的语义差异。这是一项重要的目标，因为对计算机漏洞的分类和命名是一项既困难又主观的工作。

数据模型必须清晰明确。这意味着，我们允许分析器的准确度有差异（例如，针对同一事件，一个分析器报告的信息可能比另一个分析器报告的信息要多），但不允许它们在同一事件的两个告警中出现相互矛盾的信息描述（例如，两个分析器上报的同一信息子集必须一致，而且插入到告警数据结构中相同的占位符中）。当然，通常可以在告警中而不是告警的扩展字段插入所有“有趣的”事件信息，但这种做法降低了互通性，应尽可能避免。

1.1.2.3. 告警之间的关系

入侵检测告警可在多层传输。本文适用性广，覆盖范围从非常简单的告警（例如，这些

告警是系统中的单一行动或操作造成的，如失败登录）到非常复杂的告警（例如，数个事件组合生成的告警）。

因此，数据模型必须提供一种方法，识别复杂告警（合并了多个简单告警）内容中的简单告警。

1.2 IDMEF XML 实现

入侵检测工作组（IDWG）最早曾提出两个建议实现 IDMEF：用 SMI（管理信息结构）描述 SNMP（简单网络管理协议）MIB 和使用 DTD（文档类型定义）描述 XML 文档。

IDWG 在 1999 年 9 月和 2000 年 2 月的会议上分别对这两个建议进行了评审，并在后一次会议上达成一致，认定 XML 最符合 IDWG 的要求。

1.2.1. 可扩展标记语言

XML 作为 SGMI（标准通用标记语言）的简化版本，是 ISO 8879 标准定义的文本标记语法。作为一种表示和交换网络文档及数据的语言，XML 能够有效地解决 HTML 面临的很多问题，所以获得了业界的普遍青睐。1998 年 2 月 10 日，WWW 联盟（W3C）将 XML 作为一项建议公布于众。

XML 是一种元语言—即描述其他语言的语言，它允许应用程序定义自己的标记，还可以为不同类型的文档和应用程序定义定制标记语言。XML 不同于 HTML。HTML 有一套预设的固定标识符，必须进行修改才能满足特殊用途。XML 和 HTML 都使用元素（标签，即被尖括号“>”和“<”包起来的标识符）和属性（以“名=值”的形式出现）。在 HTML 中“<p>”的意思为“段落”，而在 XML 中可能的意思为“段落”、“人物”或“鸭嘴兽”，或者无任何意义，这取决于应用程序本身。

注意：XML 不仅提供了声明文档标记和结构的语法（如定义元素和属性，指定它们出现的顺序等），还提供了在文档中使用该标记的语法。由于标记声明和标记看似截然不同，很多人会不清楚哪个语法才是 XML。答案是，两者都是 XML，因为它们其实都是同一语言的一部分。

为清晰起见，泛指时，本文用“XML”和“XML 文档”；特指时，用“IDMEF 标记”，指元素（标记）和描述 IDMEF 消息的属性。

XML 发布后，W3C 发布了第二版，其中定义了命名空间在 XML 文档中的使用方法。XML 命名空间是一组由统一资源标识符（URI）定义的名称。使用命名空间时，每个标签都标识了命名空间来源。因此，在同一文档中可能会出现来自具有相同名称的不同命名空间的标签。例如，一个文档可同时包含“usa:football”和“europe:football”标签，而两个标签的意义各不相同。

因为预期 XML 命名空间会得到广泛应用，本文介绍了用以识别 IDMEF 命名空间的 URI。

1.2.2. XML 文档的 IDMEF 实现原理

使用或开发基于 XML 的应用程序的目的是多种多样的，包括在不同领域之间进行电子数据交换、金融数据交换、电子名片、日历和日程安排、企业软件分发、网络“推送”技术，以及用于化学、数学、音乐、分子动力学、天文学、图书与期刊出版、网络发布、气象观测、房地产交易等的标记语言。

XML 很灵活，适用于这些应用程序，同时也适用于 IDMEF 的实现。选择 XML 来实现 IDMEF 的具体原因如下：

- XML 允许开发自定义语言来描述入侵检测告警。XML 还定义了一种扩展该语言的标准方法，可用于对本文的后期修订（“标准”扩展）或满足特定厂商的使用要求（“非标准”扩展）。
- 用于处理 XML 文档的商业与开源工具得到应用。用于开发 XML 解析及/或验证工具和 API 的语言多样，包括 Java、C、C++、Tcl, Perl、Python 和 GNU Emacs Lisp。这些工具可通过多种渠道获得，产品开发者能够更容易、更快地采用 IDMEF。
- XML 符合 RFC 4766 IDMEF 要求 4.1，消息格式支持完全的国际化和本地化。XML 标准要求支持 ISO/IEC 10646（通用多八位编码字符集）的 UTF-8 和 UTF-16 编码以及 Unicode，从而使所有的 XML 应用程序（以及所有符合 IDMEF 的应用程序）与这些常用的字符编码相兼容。

XML 还支持根据元素标识每个元素内容的书写语言，使 IDMEF 很容易地适应产品的“自然语言支持”版本。

- XML 符合 RFC 4766 IDMEF 要求 4.2，消息格式支持过滤和汇总。XSL 是一种样式语言。XML 与 XSL 集成，信息可相互结合、被丢弃和被重新组织。
- 目前 W3C 和其他组织正在进行的 XML 开发项目，将提供面向对象的扩展、数据库支持和其他有用的功能。如果在 XML 中执行 IDMEF，IDMEF 可立即获得这些功能。
- XML 是免费的，没有版税和许可费用。

2. 文档约定与通知

本文档中的关键用词必须（或“‘须”）”、“不可以”、“要求”、“应”、“不得”、“应该”、“不应”、“建议”、“可以（或“可能”、“可”）”及“可选”的含义与 RFC 2119 的规定一致。

“符合 IDMEF 的应用程序”指以本文规定格式读和/或写消息的程序或程序组件，如分析器或管理器。

“IDMEF 文档”是遵守本文要求的消息，也是两个或更多 IDMEF 应用程序交换的消息。“IDMEF 消息”是“IDMEF 文档”的另一种说法。

3. 标记约定与格式问题

本文使用以下三种符号：统一建模语言用于描述数据模型，XML 用于描述 IDMEF 文档中所用的标记，以及 IDMEF 标记用于表示文档本身。

3.1 IDMEF XML 文档

本节描述 IDMEF XML 文档格式规则。这些规则中的大多数是从 XML 文档格式化规则中“继承”而来的。

3.1.1. 文档 Prolog

下面将介绍 IDMEF XML 文档 Prolog 的格式。

3.1.1.1. XML 声明

符合 IDMEF 的应用程序之间交换的 IDMEF 文档开头必须包含 XML 声明，并且必须明确所使用的 XML 版本。建议加入所使用的编码规范。

因此，IDMEF 消息应以下面的内容开头：

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"

  xmlns:idmef="http://iana.org/idmef"/>
```

在内部使用时，符合 IDMEF 的应用程序可能会省略 XML 声明内以节省空间。只有消息被发送到另一个目的地址（例如，Web 浏览器）时才会加上 XML 声明。我们不推荐这种做法，除非能够保证每条消息的版本和编码信息都不会丢失。

为确保有效性（见 6.1 节），XML 文档必须包含文档类型定义。然而，这对符合 IDMEF 的应用程序而言是一项重大开销，无论是在带宽消耗方面，还是对 XML 处理器的要求方面（不仅解析声明本身，还要解析引用的 DTD）。

因此，用户可以选择让分析器和管理器在带外就交换消息（此处定的标准消息或具有扩展名的消息）所使用的文档类型定义达成一致，然后在 IDMEF 消息中省略文档类型定义。此类协商方式不在本文讨论之列。注意：在进行任何此类协商时必须谨慎，因为管理器可能要接受来自很多不同分析器的消息，而且每条消息都会使用一个具有不同扩展集的 DTD。

3.1.2. IDMEF 中的字符数据处理

出于可移植性的原因，IDMEF 格式的应用程序和 IDMEF 消息不应使用除 UTF-8 和 UTF-6 之外的字符编码。与 XML 标准保持一致，如果 IDMEF 消息中并未指定编码机制，则默认是 UTF-8 编码。

注意：ACSII 字符集是 UTF-8 编码的子集，因此也可用于对 IDMEF 消息进行编码。

根据 XML 标准，使用 UTF-16 编码的 IDMEF 文档必须以 ISO/IEC 10646 附录 E 和 Unicode 附录 B（“零宽无间断间隔”字符，#xFEFF）中描述的字节顺序标记开头。

3.1.2.1. 字符实体应用

我们建议当数据中写到“&”、“<”、“>”和“‘”（单引号）等字符时，符合 IDMEF 的应用程序应使用这些字符的实体引用形式（见 3.2.3.1 节），以避免造成误解。

3.1.2.2. 空白处理

所有 IDMEF 元素都必须支持“xml:space”属性。

3.1.3. IDMEF 中的语言

符合 IDMEF 的应用程序必须指定其内容的编码语言；一般可以通过指定元素的“xml:lang”属性，并让其他所有元素“继承”这一定义来实现。

3.2 IDMEF 数据类型

在 XML IDMEF 消息中，所有数据都以“文本”（而不是“二进制”）来表示，因为 XML 是一种文本格式化语言。然而，我们在数据模型中提供了类属性的分类信息，使读者明了模型针对每个属性希望获得的数据类型。

在 XML IDMEF 消息中，模型中的每个数据类型都有特定的格式要求；本节将介绍这些要求。

3.2.1. 整型量

整型属性用 INTEGER 数据类型表示。整型数据必须使用十进制或十六进制编码。

十进制整型编码使用数字 0–9 和一个可选符号（“+”或“-”），例如，123 和-456。

十六进制整数使用数字 0–9 和字母 a–f（或 A–F），并且以字符“0x”开头，例如：0x1a2b。

3.2.2. 实型数

实型（浮点型）属性用 REAL 数据类型表示。实型数据必须使用十进制编码。

实型编码是 POSIX 1003.1 的“strtod”库函数的编码：首先是可选符号（“+”或“-”），然后是非空的十进制数字串（有时可以包含一个基数字符），再然后就是可选的指数部分。指数部分包括一个“e”或“E”，然后是可选符号，再然后是一个或多个十进制数字。例如，123.45e02 和-567,89e-03。

符合 IDMEF 的应用程序必须支持“.”和“,”的基数字符。

3.2.3. 字符和字符串

单个字符属性用 CHARACTER 数据类型表示。已知长度的多个字符属性用 STRING 串数据类型表示。

除了偶尔需要使用字符引用（见 3.2.3.1 和 3.2.3.2 节），字符和字符串数据没有特定的格式要求。

3.2.3.1. 字符实体引用

在 XML 文档中，某些字符在一些上下文中具有特殊的含义。如果要在这样的上下文中使用这些字符的实际含义，必须使用特殊的转义序列（即所谓的实体引用）。

有时需要转义的字符及其实体引用如下：

字符	实体引用
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

3.2.3.2. 字符代码引用

通过使用字符引用，XML 文档可包含 ISO/IEC 10646 和 Unicode 标准定义的任意字符。字符引用以“&”和“#”开始，以“.”结束。这些字符之间插入字符代码。

如果字符代码以“x”开头，表示为十六进制（Base 16）；否则，就是十进制（Base 10）。例如，与符号（&）被编码为“&#38”或“&#0026”；小于号（<）被编码为“&#60”或“&#x003C”。

ISO/IEC 10646 和 Unicode 标准中，指定的任何一个、两个或四个字节的字符均可包含在

使用这项技术的文档中。

3.2.4. 字节

二进制数据用 BYTE（和 BYTE[]）数据类型表示。二进制数据必须完全使用 base64 编码。

3.2.5. 枚举类型

枚举类型用 ENUM 数据类型表示，包括一系列有序的可接受值。

3.2.6. 日期-时间字符串

日期-时间字符串用 DATETIME 数据类型表示。每个日期-时间字符串标记的是特定的时刻，不支持一段时间。

日期-时间字符串符合 ISO 8601: 2000 子集格式，如下所示。括号中的引用部分参照 ISO 8601:2000 标准中的章节内容。

1. 日期格式必须如下：

YYYY-MM-DD，其中 YYYY 是四位数，表示年份；MM 是两位数，表示月份（01–12）；DD 是两位数，表示日期（01–31）。（5.2.1.1 节，“完整表达—扩展格式”。）

2. 时间格式必须如下：

hh:mm:ss，其中 hh 是两位数，表示小时（00–24）；mm 是两位数，表示分钟（00–59）；ss 是两位数，表示秒（00–60）。（5.3.1.1 节，“完整表达—扩展格式”。）

注意：午夜有两种表示方法：00:00:00 和 24:00:00。符合 IDMEF 的应用程序必须支持这两种表达，但尽可能使用 00:00:00。

注意：之所以使用这种格式是考虑到闰秒。正闰秒是介于 23:59:59Z 和 24:00:00Z 之间，表示为 23:59:60Z。负闰秒是跳过 23:59:59Z。符合 IDMEF 的应用程序必须支持闰秒。

3. 时间格式可包括十进制小数秒，如下所示：

hh:mm:ss.ss，或

hh:mm:ss,ss

十进制符号后的位数应满足实际需要（至少一位）。小时和分钟不支持十进制小数位（见 5.3.1.3 节，“十进制小数位的表示方法”）。

符合 IDMEF 的应用程序必须支持使用十进制符号（“.”和“,”）。

注意：小数位数并不代表精确的程度，比如 00.100000、0,1000 和 00.1 所代表的含义是一样的。

4. 对于时间的格式要求是，必须明示其是否为国际协调时间（UTC），若不是，应给出

指定时间与 UTC 之间的时间差。

* UTC 时间字符串后面必须加上字母 Z，如下所示：

hh:mm:ssZ

hh:mm:ss.ssZ

hh:mm:ss,ssZ

（见 5.3.3 节，“国际协调时间（UTC）—扩展格式”。）

* 如果早于或等于 UTC 时间，则在时间字符串后面加上“+”；如果时间晚于 UTC 时间，则在时间字符串后面加上“-”。并且，在“+”和“-”后面加上小时和分钟数，表示与 UTC 的时间差。如下所示：

hh:mm:ss+hh:mm

hh:mm:ss-hh:mm

hh:mm:ss.ss+hh:mm

hh:mm:ss.ss-hh:mm

hh:mm:ss,ss+hh:mm

hh:mm:ss,ss-hh:mm

与 UTC 之间的时间差必须用小时和分钟数表示，即使是 0 分钟。如果为+00:00，表明该时间与 UTC 时间同步。（见 5.3.4.2 节，“本地时间和与 UTC 的区别—扩展格式”。）

5. 日期-时间字符串用字母 T 将日期字符串和时间字符串连接起来。如下所示：

YYYY-MM-DDThh:mm:ssZ

YYYY-MM-DDThh:mm:ss.ssZ

YYYY-MM-DDThh:mm:ss,ssZ

YYYY-MM-DDThh:mm:ss+hh:mm

YYYY-MM-DDThh:mm:ss-hh:mm

YYYY-MM-DDThh:mm:ss.ss+hh:mm

YYYY-MM-DDThh:mm:ss.ss-hh:mm

YYYY-MM-DDThh:mm:ss,ss+hh:mm

YYYY-MM-DDThh:mm:ss,ss-hh:mm

（见 5.4.1 节，“完整表达—扩展格式”。）

总之，IDMEF 日期-时间字符串必须遵守上述九类模板中的一类。

3.2.7 NTP 时间戳

NTP 时间戳用 NTPSTAMP 数据类型表示，在 RFC 1305 和 RFC 4330 中有详细描述。NTP 时间戳是 64 位无符号定点数。前 32 位是整数部分，后 32 位是小数部分。

在 IDMEF 消息中，NTP 时间戳必须编码为两个 32 位十六进制的数值，用英文句号（.）分开。例如，0x12345678.0x87654321。

了解更多关于 NTP 时间戳的信息，请参见 6.4 节。

3.2.8. 端口列表

端口列表用 PORTLIST 数据类型表示，包括用逗号分开的数字（单个整型量）和范围（N-M 表示端口 N 到 M，包括 N 和 M）。一个列表中可以包括任何数字或范围的组合，例如，5-25,37,42,43,53,69-119,123-514。

3.2.9. 唯一标识符

本规范中使用两类唯一标识符，并且都是用 STRING 数据类型表示。

这两类标识符在相关 XML 元素中作为属性来实现，必须具有以下唯一值：

1. 如果对 Analyzer 类（见 4.2.4.1 节）的 analyzerid 属性设值，该值必须在分析器所属的入侵检测环境中具有唯一性。

analyzerid 属性不需要全局唯一，只需在分析器所在的入侵检测环境中唯一即可。在不同入侵检测环境中的两个分析器可以具有相同的 analyzerid 属性值。

缺省值 0 表示分析器不能生成唯一标识符。

2. 如果分析器支持生成消息标识符，则必须通过 analyzerid 和 messageid 来唯一识别告警和心跳信息（见 4.2.2 和 4.2.3 节）。

3. 如果 Classification、Source、Targe、Node、User、Process、Service、File、Address 和 User ID 类的 ident 属性设值，该值必须在分析器发送的所有消息中具有唯一性。

ident 属性值对标识对象的任一数据组合（而非任一对象）必须是唯一的。对象可能会有多个 ident 值。例如，用名称标识的主机、用地址标识的主机和用名称+地址标识的主机会有不同的 ident 值。此外，针对同样的信息，不同的分析器生成的值也可能不同。

ident 属性为特定分析器发送的 ident 值提供了唯一标识符。然而，当与分析器的 analyzerid 值相结合时，即可创建在入侵检测环境中的唯一值。同样，不要求 ident 值全局唯一。

缺省值 0 表示分析器不能生成唯一标识符。

关于创建这些属性中唯一值的规范，不在本文讨论之列。

4. IDEMF 数据模型与 DTD

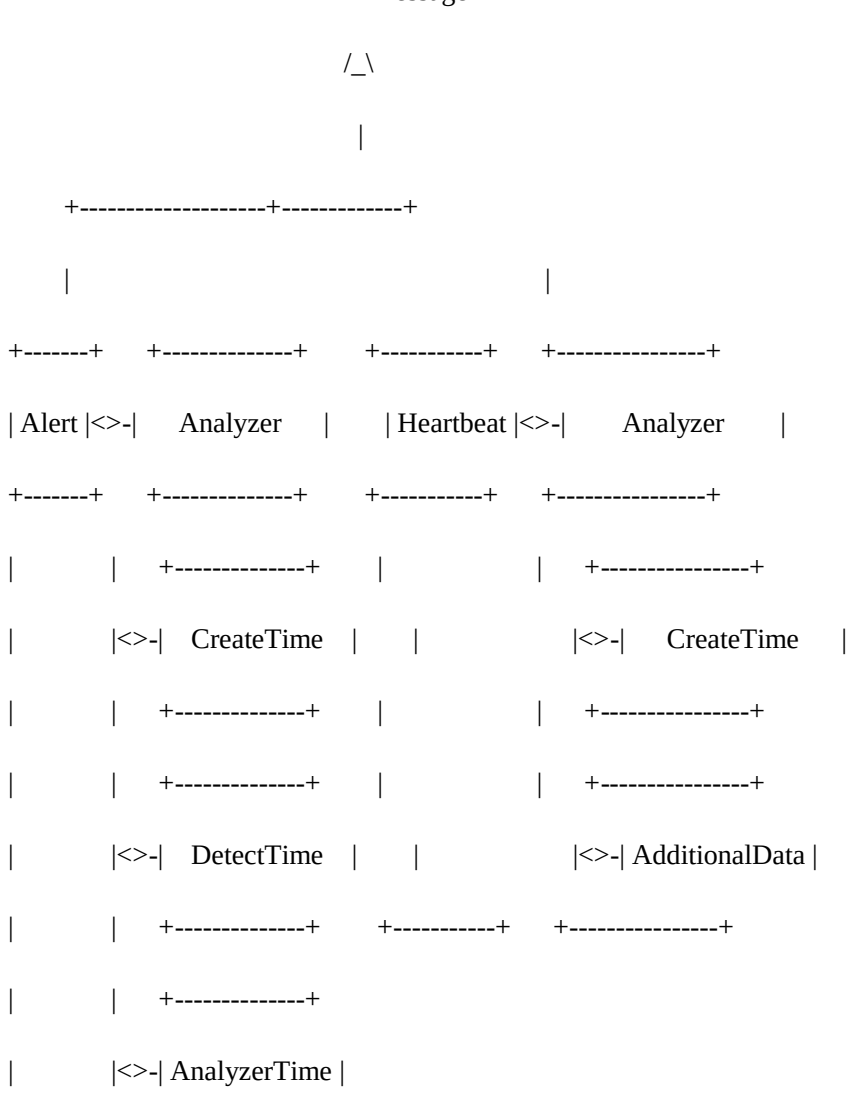
本节将详细介绍 IDMEF 数据模型的各个组件。模型的统一建模语言（UML）图展示了组件的相互关系。IDMEF DTD 相关章节介绍模型如何转化为 XML。

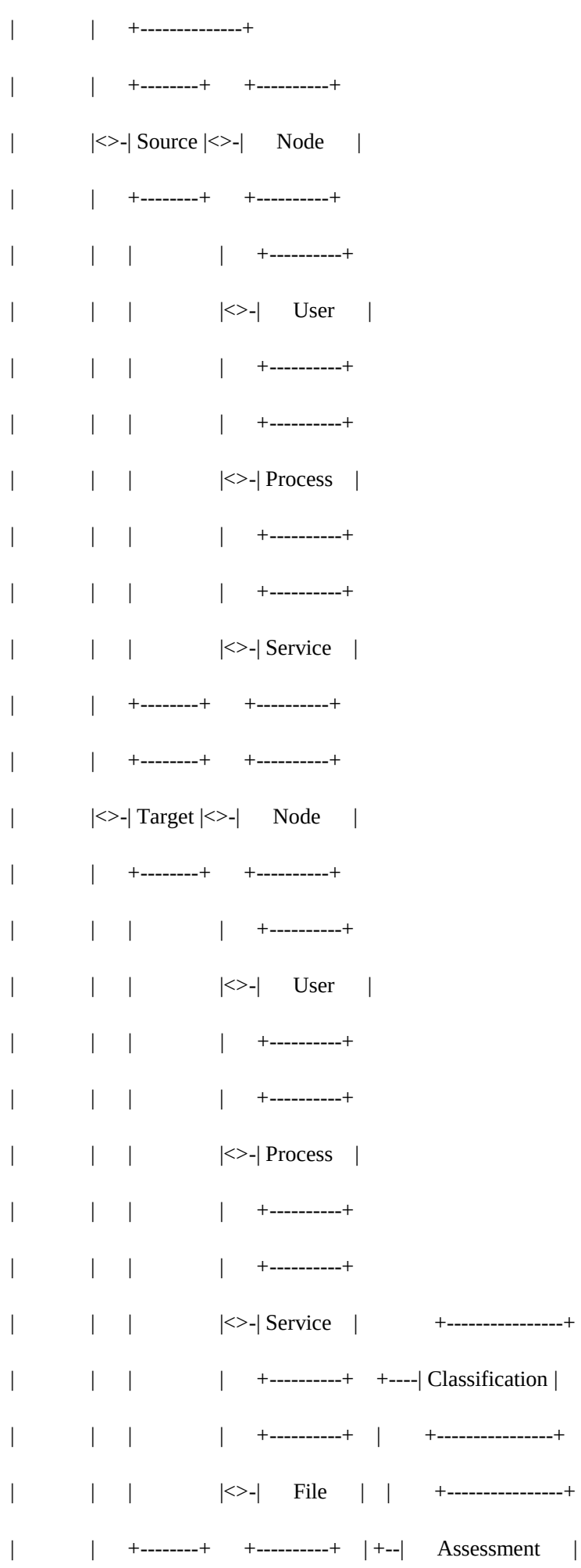
4.1 数据模型概述

IDMEF 数据模型各个主要部分之间的关系如图 1 所示（省略了发生指标和属性）。

所有 IDMEF 消息的顶层类是 IDMEF-Message，每一种类型的消息都是该类的子类。IDMEF 目前定义了两种类型的消息：Alert（告警）和 Heartbeat（心跳）。这两种消息又分别包括各自的子类，以提供更详细的消息。

需要注意的是，IDMEF 数据模型并没有对告警的分类和标识方法进行说明。例如，对一个端口的扫描，一个分析器可能将其确定为一个多目标的单一攻击，而另一个分析器可能将其确定为来自同一个源的多次攻击。只有一个分析器识别了发送的告警类型，数据模型才能规定告警格式。





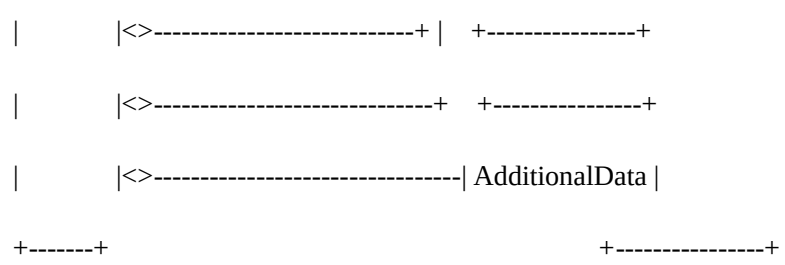


图 1 数据模型概览

4.2 消息类

以下章节将介绍各个消息类。

4.2.1. IDMEF-Message 类型

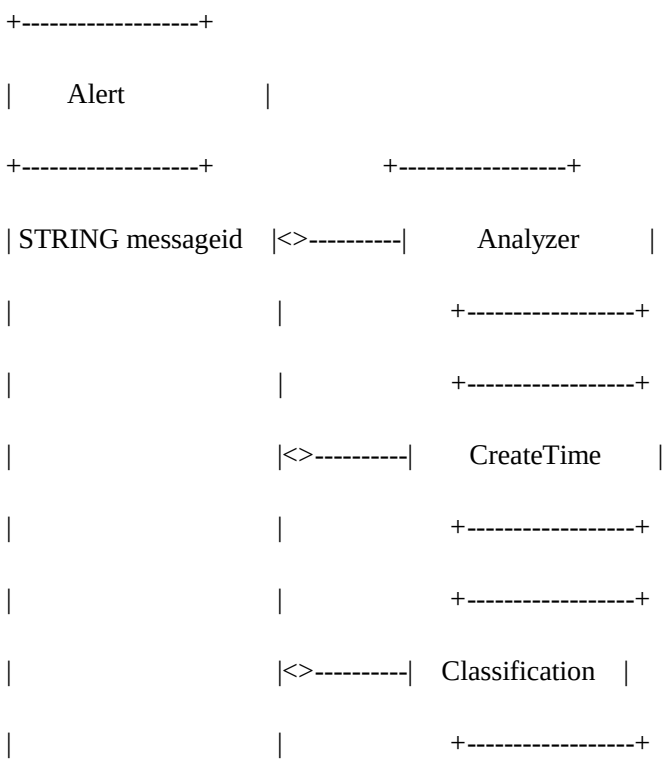
所有 IDMEF 消息都是 IDMEF 数据模型及 IDMEF DTD 的顶层类，IDMEF-Message 类的实例。目前，IDMEF-Message 有两类（子类）：Alert（告警）和 Heartbeat（心跳）。

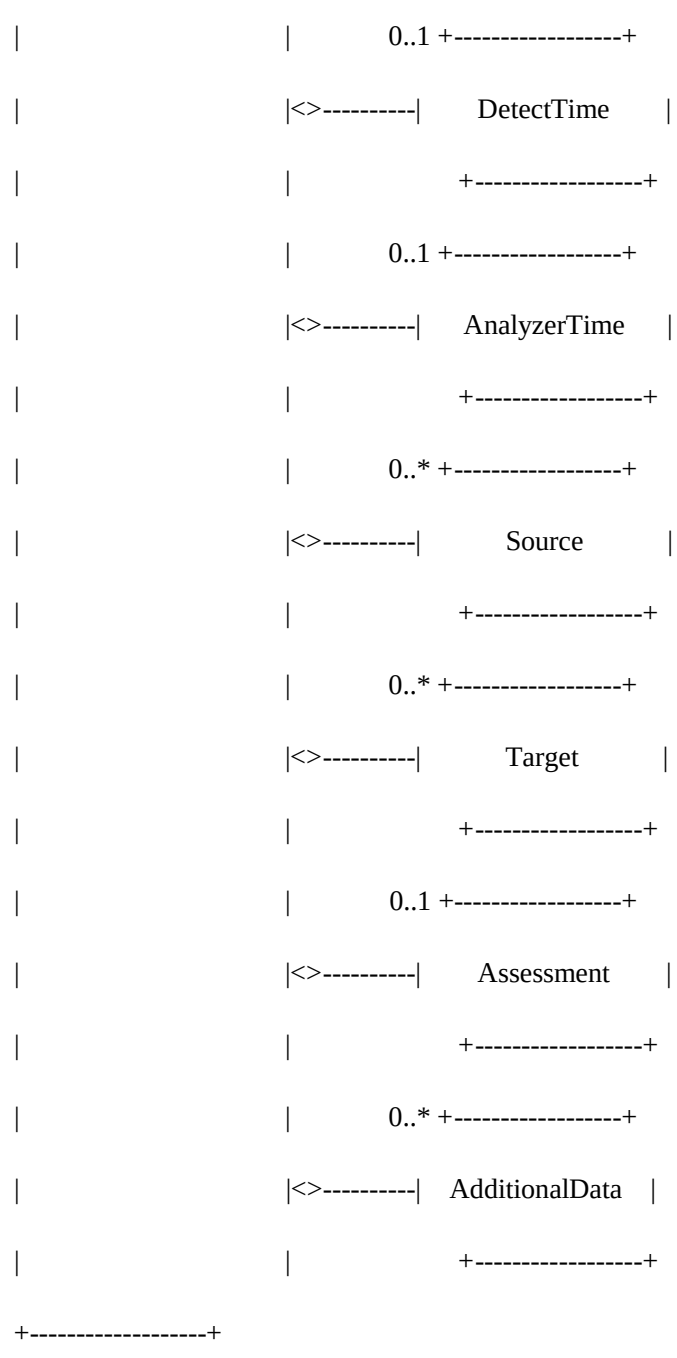
IDMEF-Message 类只有一个属性：version（版本），即消息遵循的 IDMEF-Message 规范（本文）的版本。为该属性指定值的应用程序必须设置为“1.0”。

4.2.2. Alert 类

通常，分析器检测到配置类型的攻击事件之后，就会将告警消息发送给管理器。一条告警消息可以对应一个事件，也可以对应多个事件，这是由分析器决定的。告警在外部事件发生后给出。

告警消息包含多个聚合类，见图 2。关于对聚合类的描述，见 4.2.4、4.2.5 和 4.2.6 节。





```
|    OverflowAlert    |

+------------------+
```

图 2 Alert 类

Alert 类由以下聚合类组成：

Analyzer

该类的数量为 1，表示产生告警的分析器的标识信息。

CreateTime

该类的数量为 1，表示生成告警的时间。告警中可能会包含 3 个时间，只有 CreateTime 是必选的。

Classification

该类的数量为 1，提供告警名称或管理器用于识别告警的其他信息。

DetectTime

该类的数量为 0 或 1，表示检测到告警的时间。若存在多个事件触发告警，该时间指检测到第一个事件的时间。在某些情况下，该时间与 CreateTime 值相同。

AnalyzerTime

该类的数量为 0 或 1，表示分析器的当前时间（见 6.3 节）。

Source

该类的数量为 0 或多个，表示触发告警的事件源。

Target

该类的数量为 0 或多个，表示触发告警的事件目标。

Assessment

该类的数量为 0 或 1，提供事件影响、分析器的响应行动及分析器对评估的置信度等方面的信息。

AdditionalData

该类的数量为 0 或多个，表示由分析器提供的、不适合包含在数据模型中的信息。这些信息可能只是少量的零碎信息，也可能是通过扩展 IDMEF 提供的大量信息（见第 5 节）。

Alert 类在 IDMEF DTD 中的定义如下：

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

'0' %attlist.global; >

Alert 类有一个属性：messageid。

可选。该属性为告警的唯一标识。详情请参见 3.2.9 节。

4.2.2.1. ToolAlert 类

ToolAlert 类包含与使用的攻击工具相关的附加信息或木马等恶意程序相关的信息，在分析器识别攻击工具时可能会用到这些信息。用该类可将之前发送的告警进行分组，表明“这些告警产生的原因均是因为某人使用了这个工具”。

ToolAlert 类由三个聚合类组成，如图 3 所示。

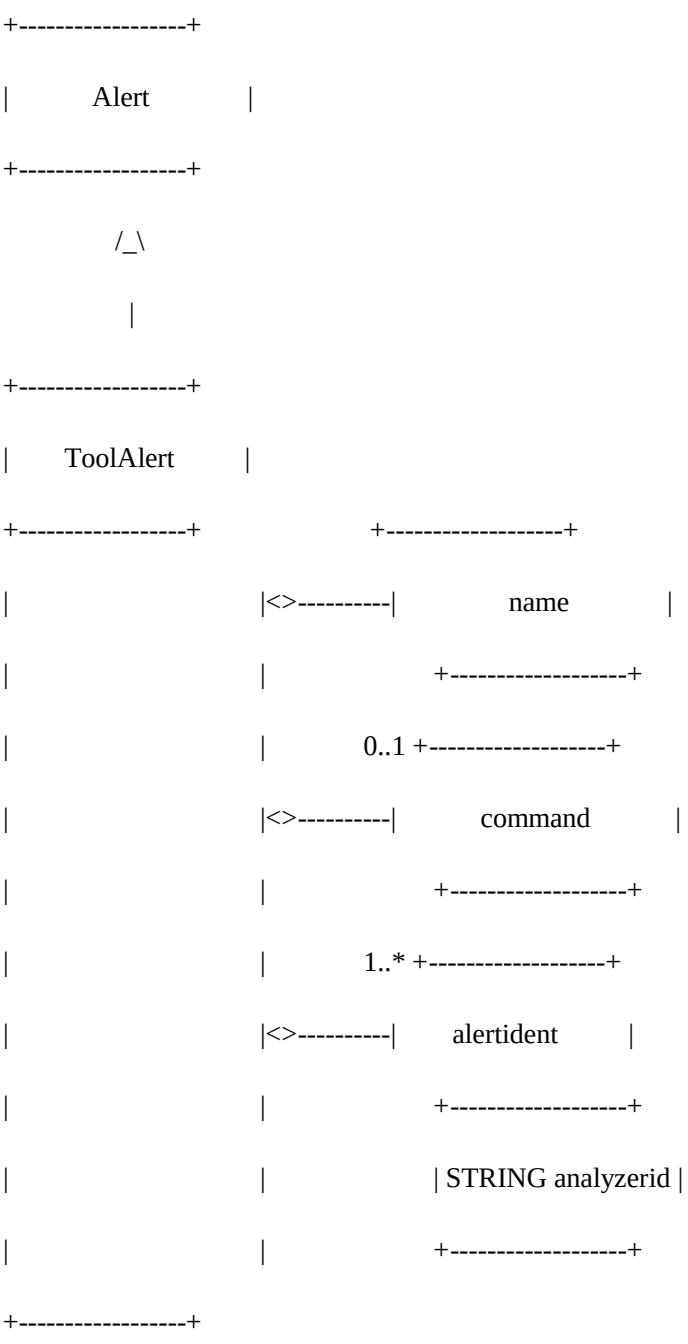


图 3 ToolAlert 类

ToolAlert 由以下聚合类组成：

name

该类的数量为 1，值为字符串型，表示将告警分组的原因，例如特定工具的名字。

command

该类的数量为 0 或 1，值为字符串型，表示执行黑客工具的命令，如 BackOrifice ping。

alertident

该类的数量为 1 或多个，值为字符串型，它是与告警相联系的告警识别符的列表。因为单个分析器发送的告警中的告警识别符是唯一的，“alertident”中的“analyzerid”属性选项被用于标识产生特定告警的分析器。如果分析器没有提供“analyzerid”属性，表明告警可能来自于与发送 ToolAlert 类相同的分析器。

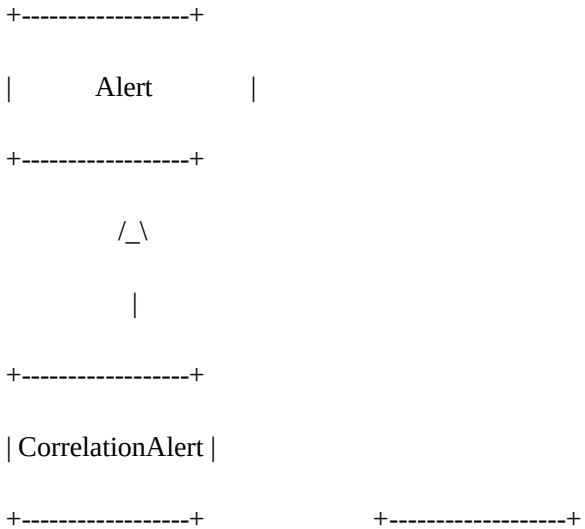
ToolAlert 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT ToolAlert                    (
    name, command?, alertident+
  )>
<!ATTLIST ToolAlert
    %attlist.global;
  >
```


4.2.2.2. CorrelationAlert 类

CorrelationAlert 类包含与报警信息相关的附加信息。用该类可将之前发送的告警进行分组，表明“这些告警是相关的”。

CorrelationAlert 类由两个聚合类组成，如图 4 所示。



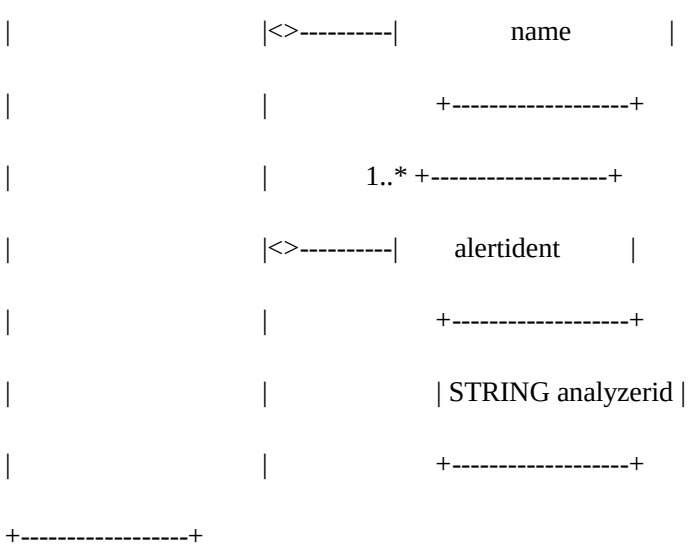


图 4 CorrelationAlert 类

CorrelationAlert 类由以下聚合类组成：

name

该类的数量为 1，字符串型，表示对告警进行分组的原因，如特定的关联方法。

alertident

该类的数量为 1 或多个，值为字符串型，它是与告警相关的告警识别符的列表。因为单个分析器发送的告警中的告警识别符是唯一的，“alertident”中的“analyzerid 属性选项被用于标识产生特定报警的分析器。” 如果分析器没有提供“analyzerid”属性，表明告警可能来自于与发送 CorrelationAlert 类相同的分析器。

CorrelationAlert 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT CorrelationAlert (
    name, alertident+
  )>
<!ATTLIST CorrelationAlert
    %attlist.global;
  >
```

4.2.2.3. OverflowAlert 类

OverflowAlert 类包含与缓冲区溢出攻击相关的信息，使用该类，分析器可提供与缓冲区溢出攻击相关的详细信息。

OverflowAlert 类由三个聚合类组成，如图 5 所示。

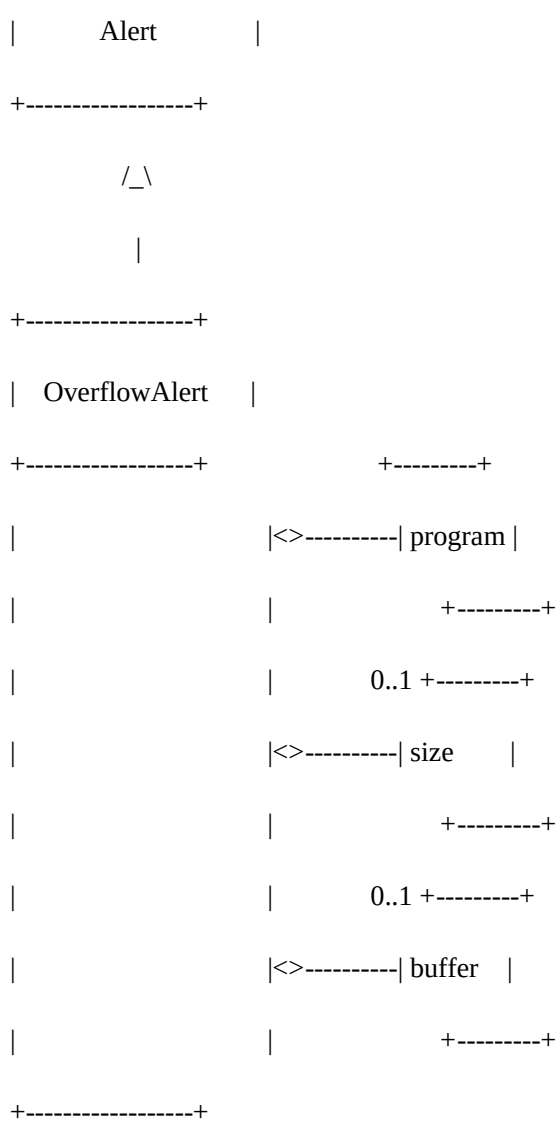


图 5 OverflowAlert 类

OverflowAlert 类由以下聚合类组成：

program

该类的数量为 1，值为字符串型，表示缓冲区溢出攻击试图运行的程序（注意：不是受到攻击的程序）。

size

该类的数量为 0 或 1，值为整型，表示缓冲区溢出的字节的大小（如攻击者发送的字节数）。

buffer

该类的数量为 0 或 1，值类型为 BYTE[]，表示部分或全部的缓冲区溢出的字节（取决于分析器可抓取的字节数量）。

OverflowAlert 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT OverflowAlert                    (
```

```
    program, size?, buffer?
  )>
<!ATTLIST OverflowAlert
    %attlist.global;
  >
```

4.2.3. Heartbeat 类

分析器使用心跳消息向管理器表明其目前状况。心跳信息周期性地发送给管理器，比如每 10 分钟或每小时。管理器能收到分析器的心跳消息，则断定分析器工作正常；若未收到心跳消息（或连续的心跳信息），则断定分析器或者网络连接失效。

所有的管理器必须支持心跳消息的接收，但是分析器是否使用这些消息则是可选的。管理器软件应可针对具体分析器配置是否使用心跳信息。

心跳消息由多个聚合类组成，见图 6。关于对聚合类的描述，详见 4.2.4 和 4.2.5 节。

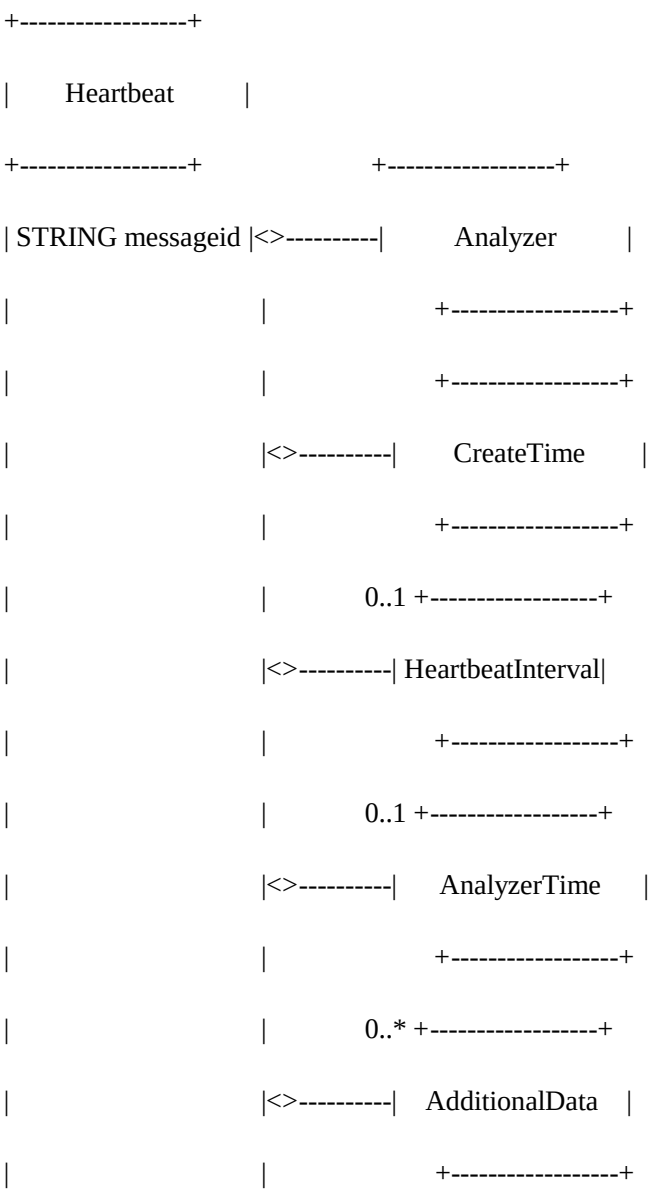


图 6 Heartbeat 类

Heartbeat 类由以下聚合类组成：

Analyzer

该类的数量为 1，表示产生心跳消息的分析器的标识信息。

CreateTime

该类的数量为 1，表示生成心跳消息的时间。

HeartbeatInterval

该类的数量为 0 或 1，表示生成心跳消息的时间间隔。

AnalyzerTime

该类的数量为 0 或 1，表示分析器的当前时间（见 6.3 节）。

AdditionalData

该类的数量为 0 或多个，表示由分析器提供的、不适合包含在数据模型中的信息。这些信息可能只是少量的零碎信息，也可能是通过扩展 IDMEF 提供的大量信息（见第 5 节）。

Heartbeat 类在 IDMEF DTD 中的定义如下：

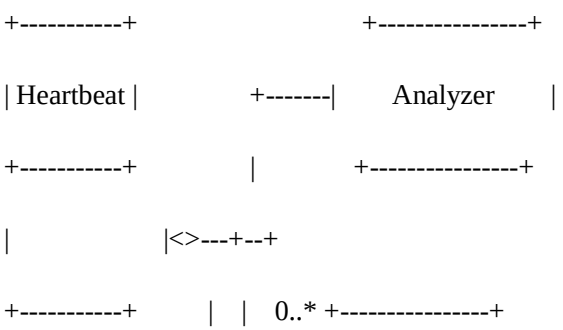
<!ELEMENT Heartbeat (Analyzer, CreateTime, HeartbeatInterval?, AnalyzerTime?, AdditionalData*)><!ATTLIST Heartbeat messageid CDATA '0' %attlist.global; >

Heartbeat 类只有一个属性：messageid。

可选。该属性为心跳信息的唯一标识。详情请参见 3.2.9 节。

4.2.4. Core 类

Core 类包括 Analyzer、Source、Target、Classification 和 AdditionalData，是告警和心跳消息的主要部分，如图 7 所示。



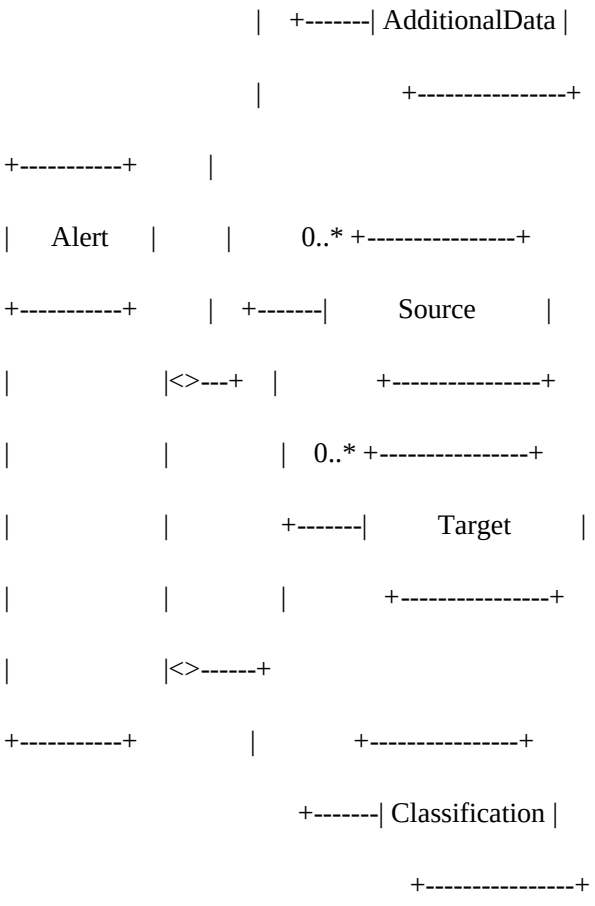
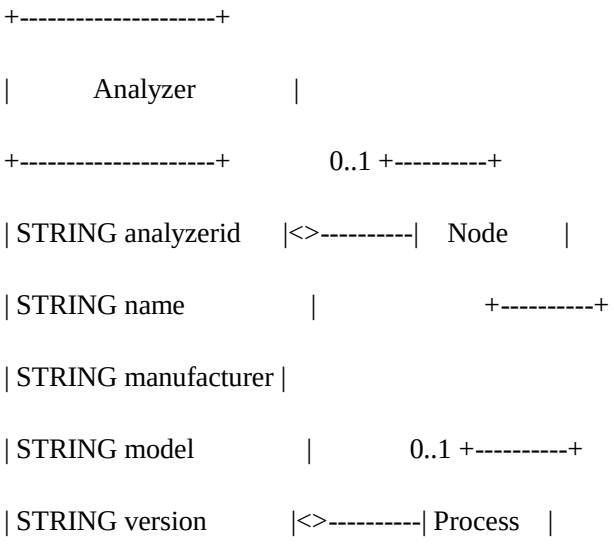


图 7 Core 类

4.2.4.1. Analyzer 类

Analyzer 类标识了生成告警或心跳消息的分析器。对于每一条告警或心跳消息，只有一个分析器被编码，并且必须是生成这条告警或心跳消息的分析器。虽然不禁止使用分层入侵检测系统（告警在树形结构中向上转发），IDMEF 数据模型不提供消息生成分析器与消息接收管理器之间的“中继”分析器识别信息。

Analyzer 类由三个聚合类组成，如图 8 所示。



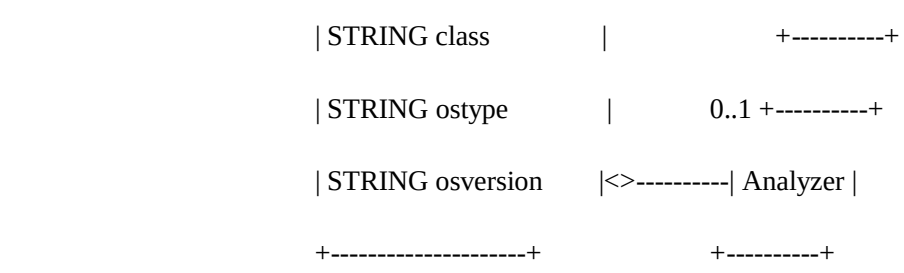


图 8 Analyzer 类

Analyzer 类由以下聚合类组成：

Node

该类的数量为 0 或 1，表示分析器所在的主机或设备的信息（如网络地址、网络名称等）。

Process

该类的数量为 0 或 1，表示分析器执行的进程的信息。

Analyzer

该类的数量为 0 或 1，表示消息可能经过的分析器的信息。该机制的原理是，当管理器接收到告警并且想要将告警转发至另一分析器时，它需要将初始分析器信息替换为自己的信息。

为了保留原始分析器的信息，可将其包含在新的分析器定义中。这样，就可以追踪分析器了。

Analyzer 类在 IDMEF DTD 中的定义如下：

```
<!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
        %attlist.global;
    >
```

Analyzer 类有 8 种属性：

analyzerid

可选（详见下文）。该属性为分析器的唯一标识。详情请参见 3.2.9 节。

该属性只是“部分”可选。如果分析器利用其它类的 ident 属性为对象提供唯一标识符，那么也必须提供有效的 analyzerid 属性。这项要求是根据 ident 属性的唯一性来规定（它们仅在特定 analyzerid 上下文中具有唯一性）的。然而，如果分析器没有 ident 属性，则可以不用 analyzerid 属性。

name

可选。表示分析器的名称，它比 analyzerid 更易理解。

manufacturer

可选。表示分析器软件和/或硬件的生产商。

model

可选。表示分析器软件和/或硬件的型号名称/数量。

version

可选。表示分析器软件和/或硬件的版本。

class

可选。表示分析器软件和/或硬件所属的类。

ostype

可选。表示操作系统名称。在 POSIX 1003.1 标准的系统中，操作系统名称指的是 uname()系统调用通过 utsname.sysname 返回的值，或 uname –s 命令的输出值。

osversion

可选。表示操作系统版本。在 POSIX 1003.1 标准系统中，操作系统名称指的是 uname()系统调用通过 utsname.release 返回的值，或 uname –r 命令的输出值。

manufacturer、model、version 和 class 属性的内容是由具体的厂商决定的，但也可以放在一起来标识分析器的不同类别（也许还能用于确定其他具体厂商的 IDMEF 消息的字段内容。）

4.2.4.2. Classification 类

Classification 类提供告警名称或管理器用于识别告警的其他信息。该名称由告警提供者确定。

Classification 类由一个聚合类组成，如图 9 所示。

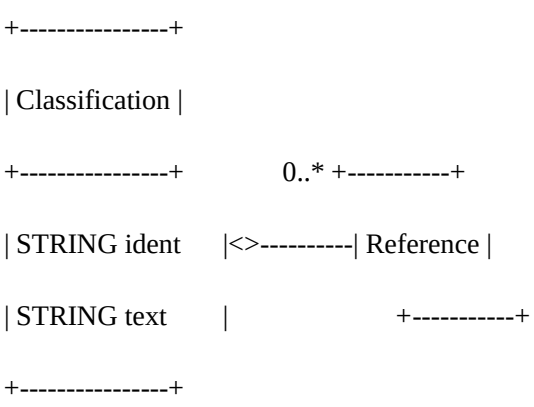


图 9 Classification 类

Classification 类由以下聚合类组成：

Reference

该类的数量为 0 或多个，表示消息信息，指向可提供告警背景信息的外部文档地址。

Classification 类在 IDMEF DTD 中的定义如下：

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

Classification 类有两种属性：

ident

可选。该属性为类别的唯一标识。详情请参见 3.2.9 节。

text

必选。厂商提供的识别告警信息的字符串。

4.2.4.3. Source 类

Source 类包括产生告警的可能的事件源。一个事件可能有多个源（如 DDoS 攻击）。

Source 类由四个聚合类组成，如图 10 所示。

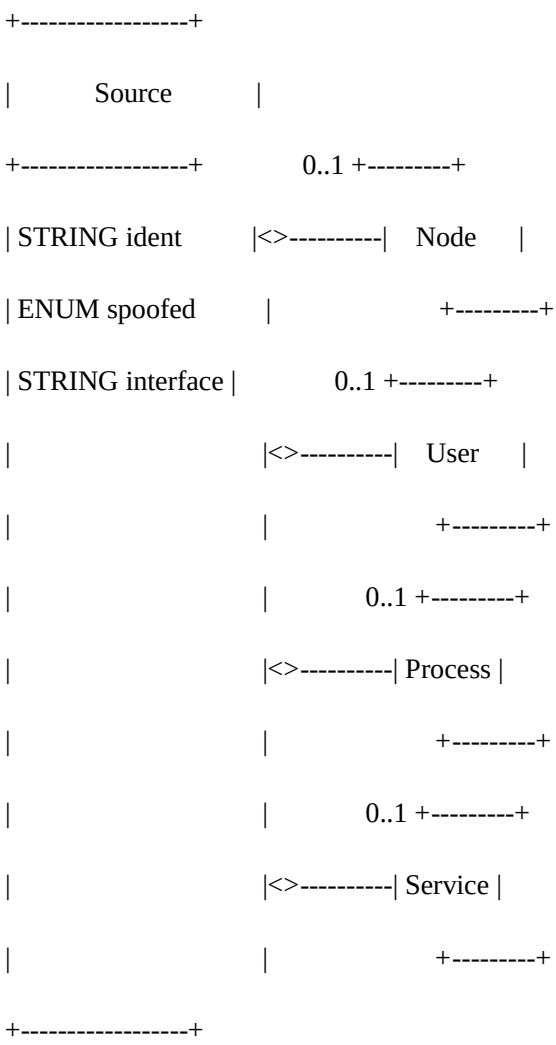


图 10 Source 类

Source 类由以下聚合类组成：

Node

该类的数量为 0 或 1，表示造成攻击事件的主机或设备信息（如网络地址、网络名称等）。

User

该类的数量为 0 或 1，表示造成攻击事件的用户信息。

Process

该类的数量为 0 或 1，表示造成攻击事件的进程的信息。

Service

该类的数量为 0 或 1，表示攻击事件中涉及的网络服务。

Source 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT Source                    (
    Node?, User?, Process?, Service?
  )>
<!ATTLIST Source
    ident               CDATA               '0'
    spoofed             %attvals.yesno;     'unknown'
    interface           CDATA               #IMPLIED
    %attlist.global;
  >
```

Source 类有三种属性：

ident

可选。该属性为源的唯一标识。详情请参见 3.2.9 节。

spoofed

可选。根据分析器的判断，表明攻击源是否是伪造地址，以隐藏真实的攻击源头。该属性的值如下所示，默认为“unknown。”（详情请参见第 10 节）。

序号	关键字	说明
0	unknown	源信息准确性未知
1	yes	源地址为诱骗地址
2	no	源地址为真实地址

interface

可选。基于网络的分析器若具有多个接口可使用该属性表明攻击源位于哪个接口。

4.2.4.4. Target 类

Target 类包括产生告警的可能的事件目标。一个事件可能有多个目标（如 port sweep 攻击）。

Target 类由四个聚合类组成，如图 11 所示。

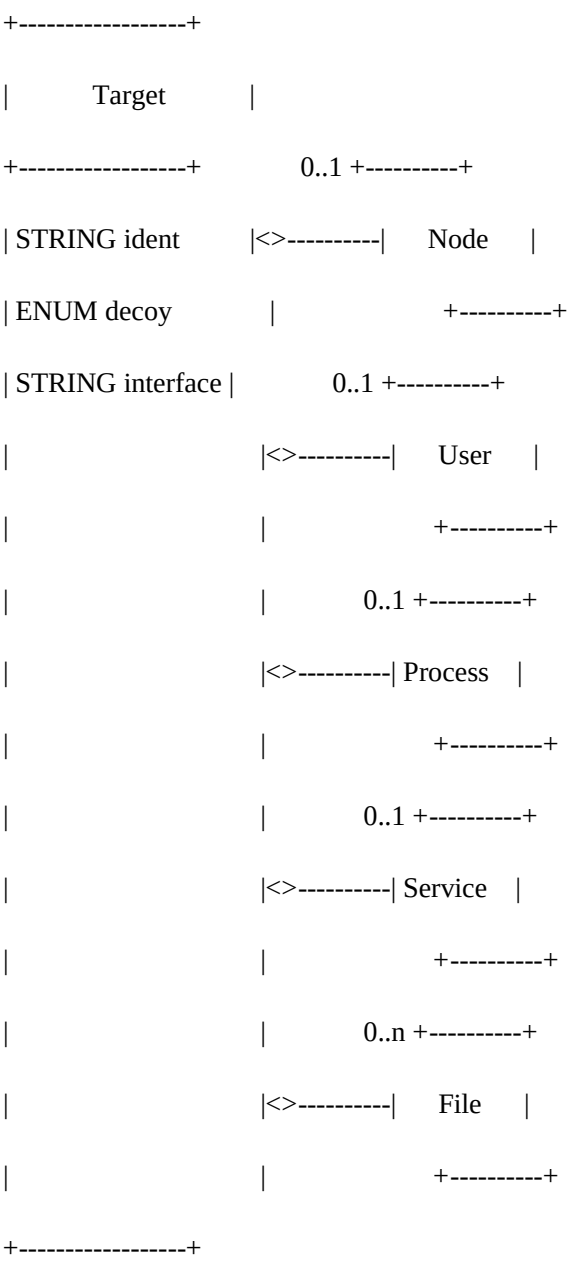


图 11 Target 类

Target 类由以下聚合类组成：

Node

该类的数量为 0 或 1，表示攻击事件针对的主机或设备信息（如网络地址、网络名称等）

User

该类的数量为 0 或 1，表示攻击事件针对的用户信息。

Process

该类的数量为 0 或 1，表示攻击事件针对的进程信息。

Service

该类的数量为 0 或 1，表示攻击事件中涉及的网络服务。

File

可选。表示攻击事件中涉及的文件信息。

Target 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT Target                    (
    Node?, User?, Process?, Service?, File*
  )>
<!ATTLIST Target
    ident               CDATA               '0'
    decoy               %attvals.yesno;     'unknown'
    interface           CDATA               #IMPLIED
    %attlist.global;
  >
```

该类有三种属性：

ident

可选。该属性为目标的唯一标识。详情请参见 3.2.9 节。

decoy

可选。根据分析器的判断，表明目标是否是诱骗地址。该属性的值如下所示，默认为“unknown。”（详情请参见第 10 节）。

+------+---------+-------------------------------------+

序号	关键字	说明
0	unknown	目标信息准确性未知
1	yes	目标地址为诱骗地址
2	no	目标地址为真实地址

interface

可选。基于网络的分析器若具有多个接口可使用该属性表明攻击目标位于哪个接口。

4.2.4.5. Assessment 类

Assessment 类用于提供分析器对攻击事件的评估，包括事件影响、针对事件所采取的行动和评估的置信度。

Assessment 类由三个聚合类组成，如图 12 所示。

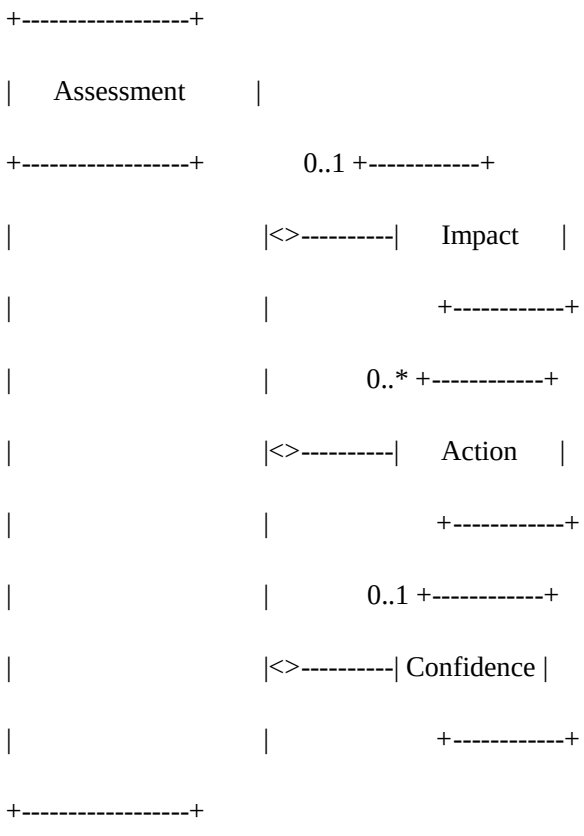


图 12 Assessment 类

Assessment 类由以下聚合类组成：

Impact

该类的数量为 0 或 1，表示分析器对攻击事件对目标的影响评估。

Action

该类的数量为 0 或多个，表示分析者针对事件所采取的行动。

Confidence

该类的数量为 0 或 1，表示分析者对事件影响评估的置信度。

Assessment 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT Assessment                    (
    Impact?, Action*, Confidence?
  )>
<!ATTLIST Assessment
    %attlist.global;
  >
```

4.2.4.6. AdditionalData 类

AdditionalData 类用于提供数据模型不能表达的信息。在只需发送少量附加信息的情况下，AdditionalData 可用于提供原子数据（整型量和字符串等）；还可用于扩展数据模型和 DTD，以支持复杂数据（如数据包头）的传输。关于扩展数据模型和 DTD，详见第 5 节。

序号	关键字	说明
0	boolean	本元素包含一个布尔值，即 true 或 false 字符串
1	byte	元素内容为一个 8 位单字节（见 3.2.4 节）
2	character	元素内容为单个字符

			（见 3.2.3 节）
	3	date-time	元素内容为 date-time 字符串
			（见 3.2.6 节）
	4	integer	元素内容为整型量（见
			（3.2.1 节）
	5	ntpstamp	元素内容为 NTP 时间戳（见
			（3.2.7 节）
	6	portlist	元素内容为端口列表（见
			（3.2.8 节）
	7	real	元素内容为实型数（见
			（3.2.2 节）
	8	string	元素内容为字符串（见
			（3.2.3 节）
	9	byte-string	元素内容为字节[]（见 3.2.4 节
	10	xmltext	元素内容为 XML 标记数据（见
			5.2 节）

AdditionalData 元素在 IDMEF DTD 中的描述为：

<!ENTITY % attvals.adtype "

```
        ( boolean | byte | character | date-time | integer | ntpstamp |
          portlist | real | string | byte-string | xmltext )
      ">

    <!ELEMENT AdditionalData        (
      (boolean | byte       | character | date-time |
       integer | ntpstamp  | portlist  | real      |
       string  | byte-string | xmltext  )
     )>

    <!ATTLIST AdditionalData
        type            %attvals.adtype;    'string'
        meaning         CDATA               #IMPLIED
        %attlist.global;
      >
```

AdditionalData 类有一个属性：

meaning

可选。描述元素内容含义的字符串。这些值取决于厂商或实现情况。确保管理器能够理解分析器所发送的字符串的含义的方法不在本文讨论之列。本文并未提供可接受含义的关键字列表，但后续版本可能提供该列表。

4.2.5. 时间类

数据模型提供三个时间类。这些类是告警类和心跳类的元素。

时间类在 IDMEF DTD 中的定义如下：

```
    <!ELEMENT ntpstamp          (#PCDATA)       >
    <!ATTLIST ntpstamp          %attlist.global;  >

    <!ELEMENT CreateTime        (#PCDATA) >
```

```
<!ATTLIST CreateTime
    ntpstamp        CDATA        #REQUIRED
    %attlist.global;
  >
<!ELEMENT DetectTime        (#PCDATA) >
<!ATTLIST DetectTime
    ntpstamp        CDATA        #REQUIRED
    %attlist.global;
  >
<!ELEMENT AnalyzerTime      (#PCDATA) >
<!ATTLIST AnalyzerTime
    ntpstamp        CDATA        #REQUIRED
    %attlist.global;
  >
```

有关<CreateTime>元素内容的日期和时间格式，请参见 3.2.6 节。

若元素内容的日期和时间与 NTP 时间戳不同（这种情况不应发生），必须以 NTP 时间戳的值为准。

4.2.5.1. CreateTime 类

CreateTime 类表示分析器创建告警或心跳的日期和时间。

4.2.5.2. DetectTime 类

DetectTime 类表示分析器检测到触发告警的事件的日期和时间。若存在多个事件触发告警，该时间指第一个事件的检测时间（该类表示的时间可能与 CreateTime 类的时间相同或不同；分析器检测到此类事件后不必立即发送告警）。

4.2.5.3. AnalyzerTime 类

AnalyzerTime 类表示分析器的当前日期和时间。该类的值应在消息传输过程中晚些时候设置，最好在传输前最后一刻填写。

有关如何使用<AnalyzerTime>在分析器和管理器之间进行基本时间同步，请参见 6.3 节。

4.2.6. 评估类

该数据模型提供分析器的三种事件评估类型。这些类是 Assessment 类的聚合类。

4.2.6.1. Impact 类

Impact 类提供分析器关于事件对攻击目标的影响的评估。该类在 IDMEF DTD 中的定义如下：

```
<!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'
    %attlist.global;
  >
```

该类有三种属性：

severity

该属性表示事件的相对风险级别评估。该属性的值如下所示，无缺省值（见第 10 节）。

序号	关键字	说明

0	info	通知有活动进行
1	low	低级
2	medium	中级
3	high	高级

completion

该属性表示分析器判断该事件是否成功。该属性的值如下所示，无缺省值（见第 10 节）。

序号	关键字	说明
0	failed	事件未成功
1	succeeded	事件成功

type

该属性表示事件类型，属于广义分类范畴。该属性的值如下所示，默认为“other”（见第 10 节）。

序号	关键字	说明

0	admin	尝试获得或
		已获得管理权限
1	dos	试图发起或已完成 DoS 攻击
2	file	试图操作文件或已完成文件操作
3	recon	试图进行 或已完成侦测
4	user	试图获取或成功获取用户权限
5	other	除以上类型之外的其他类型

这三个属性可选。该元素本身可能为空。若分析器能提供其他详情，该元素可能包括影响描述。

4.2.6.2. Action 类

Action 类描述分析器对事件采取的响应动作。该类在 IDMEF DTD 中的定义如下：

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

该类只有一个属性：

category

该属性表示动作类型，值如下所示，默认为“other”。（见第 10 节）。

序号	关键字	说明
0	block-installed	安装了某种阻断，阻止攻击到达目的地。阻断可能针对端口、地址等，或禁用用户账号。
1	notification-sent	对带外发送了某种类型的通知消息（通过传呼、e-mail 等），不包括本告警的传输。
2	taken-offline	系统、计算机或用户被迫下线，因为计算机被关闭或用户已登出。
3	other	除以上类型之外的其他类型。

该元素本身可能为空。若分析器能提供其他详情，该元素可能包含动作描述。

4.2.6.3. Confidence 类

Confidence 类表示分析器对分析有效性的最优评估。该类在 IDMEF DTD 中的定义如下：

```
<!ENTITY % attvals.rating                    "
    ( low | medium | high | numeric )
```

```
    ">

    <!ELEMENT Confidence              (#PCDATA) >
    <!ATTLIST Confidence
        rating                  %attvals.rating;          'numeric'
        %attlist.global;
      >
```

该类只有以下一个属性：

rating

该属性表示分析器对其分析有效性的评级，属性值如下所示，默认为“numeric”（见第 10 节）。

序号	关键字	说明
0	low	分析器不信任其有效性
1	medium	分析器对其有效性有一定的信心
2	high	分析器对其有效性很有信心
3	numeric	分析器提供后验概率值表示对其有效性的信心

该元素仅在分析器生成有用信息时才使用。仅输出粗糙化启发式的系统应使用“low”(低)、“medium”（中）或“high”（高）等级值。这种情况下，应省略元素内容。

生成合理概率评估的系统应使用数字等级值，并在元素内容中明确数字类型的置信度值。数值应反映出后验概率（指基于检测系统过滤的数据以及系统使用的模型得出的攻击发生概率）。数值应为范围在 0.0–1.0 之间的浮点数。数字的位数应限制为单精度浮点数表示的位数，详情请参见 3.2.2 节。

注意：不同类型的分析器可能会通过不同方法计算置信度值，并且很多情况下，不应比较不同分析器计算出的置信度值（例如，分析器基于不同方法计算或表示置信度或者分析器类型不同或配置存在差别）。当实现处理置信度值（如事件关联器）的系统时，应切记不要做系统运行环境不支持的比较或假设。

4.2.7. 支持类

支持类在核心类中占多数，在核心类之间共享。

4.2.7.1. Reference 类

Reference 类提供告警名称或其他信息供管理器进行识别。

Reference 由两个聚合类组成，如图 13 所示。

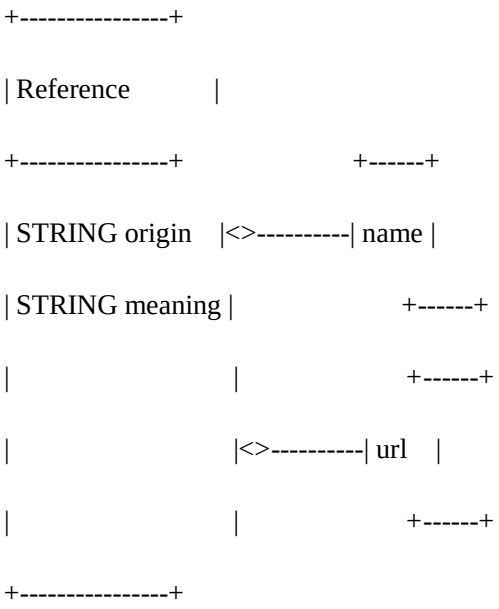


图 13 Reference 类

该类由以下聚合类组成：

name

Reference 类中包含一个 name 类。该类为字符串型，表示告警名称，隶属于以下来源类型的一种。

url

Reference 类中包含一个 url 类。该类为字符串型，表示一个 URL，管理器（或管理器的人类操作员）可从该 URL 中获得告警的其他信息。该 URL 指向的文档可能会包含攻击的详

细描述、相应的防范措施或厂商认为相关的其他信息。

该类在 IDMEF DTD 中的定义如下：

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

Reference 类有两种属性：

origin

必选。该属性表示告警名称的来源。该属性的值如下所示，缺省值为“unknown”（见第 10 节）。

序号	关键字	说明
0	unknown	名称来源未知
1	vendor-specific	与特定厂商相关的名称（URL）；用于提供与具体产品相关的信息
2	user-specific	与特定用户相关的名称（URL）；用于提供与具体安装相关的信息
3	bugtraqid	SecurityFocus （“Bugtraq”）漏洞库识别符

| | | (http://www.securityfocus.com/bid) |

| 4 | cve | 通用漏洞披露 |

| | | （CVE）名（http://cve.mitre.org/） |

| 5 | osvdb | 开源漏洞数据库 |

| | | (http://www.osvdb.org) |

+------+----------+--+

meaning

可选。该属性表示告警提供商对参考的解释。该属性仅在“来源”（origin）属性设置为 vendor-specific 或 user-specific 时有效。

4.2.7.2. Node 类

Node 类识别主机和其他网络设备（路由器、交换机等）。

Node 类由三个聚合类组成，如图 14 所示。

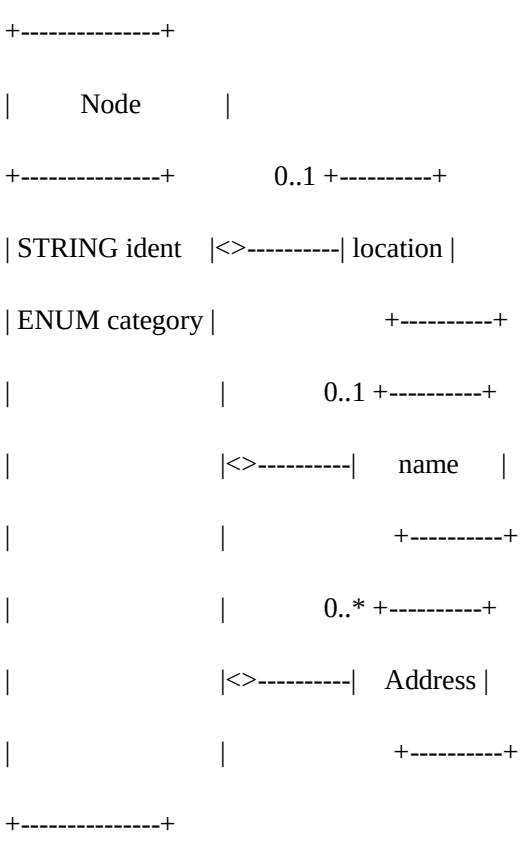


图 14 Node 类

Node 类由以下聚合类组成：

location

Node 类可包含一个 location 类，也可将其省略。该类为字符串型，表示设备位置。

name

Node 类可包含一个 name 类，也可将其省略。该类为字符串型，表示设备名称。若 Address 类未设置，必须包含该类，提供设备名称。

Address

Node 类可包含一个 Address 类，也可将其省略。该类表示设备的网络地址或硬件地址。

若不提供以上 name 类，必须指定至少一个地址。

Node 类在 IDMEF DTD 中的定义如下：

```
<!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'
    %attlist.global;
  >
```

Node 类有两种属性：

ident

可选。该属性为节点的唯一标识。详情请参见 3.2.9 节。

category

可选。该属性表示提供名称信息的域（若相关）。该属性的值如下表所示，默认为“unknown”（见第 10 节）。

```
+------+----------+--------------------------------------+
| 序号 | 关键字          | 说明                                   |
+------+----------+--------------------------------------+
```

0	unknown	域未知或不相关
1	ads	Windows 2000 高级目录服务
2	afs	Andrew 文件系统（Transarc）
3	coda	Coda 分布式文件系统
4	dfs	分布式文件系统（IBM）
5	dns	域名系统
6	hosts	本地 hosts 文件
7	kerberos	Kerberos 域
8	nds	Novell 目录服务
9	nis	网络信息服务（Sun）
10	nisplus	增强网络信息服务（Sun）
11	nt	Windows NT 域
12	wfw	Windows for Workgroups

4.2.7.2.1. Address 类

Address 类表示网络、硬件和应用的地址。

Address 类由两个聚合类组成，如图 15 所示。

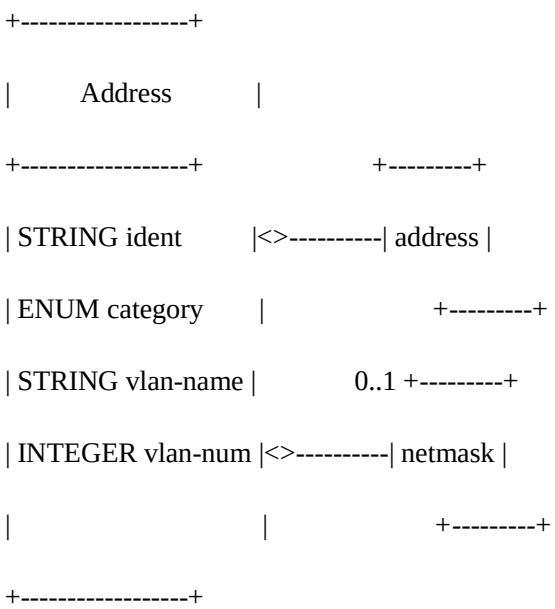


图 15 Address 类

Address 类由以下聚合类组成：

address

Address 类包含一个 address 类。该类为字符串型，表示地址信息。该类的数据格式取决于类别（category）属性。

netmask

Address 类可包含一个 netmask 类，也可将其省略。该类为字符串型，表示地址（若有）的网络掩码。

Address 类在 IDMEF DTD 中的定义如下：

```
<!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
    %attlist.global;
  >
```

Address 类有四种属性：

ident

可选。该属性为地址的唯一标识。详情请参见 3.2.9 节。

category

可选。该属性表示地址类型。该属性的值如下表所示，默认为“unknown”（见第 10 节）。

序号	关键字	说明
0	unknown	地址类型未知
1	atm	异步传输模式网络地址
2	e-mail	电子邮件地址（(RFC 822）
3	lotus-notes	Lotus Notes 电子邮件地址
4	mac	媒体访问控制（MAC）地址
5	sna	IBM 共享网络架构（SNA）地址
		address
6	vm	IBM VM ("PROFS")电子邮件地址
7	ipv4-addr	点分十进制表示的 IPv4 主机地址
		（a.b.c.d）
8	ipv4-addr-hex	十六进制表示的 IPv4 主机地址
9	ipv4-net	点分十进制表示的 IPv4 网络地址
		数字、斜线、有效位数
		（a.b.c.d/nn）
10	ipv4-net-mask	点分十进制表示的 IPv4 网络地址
		数字、斜线、点分十进制表示的
		网络掩码 （a.b.c.d/w.x.y.z）
11	ipv6-addr	IPv6 主机地址
12	ipv6-addr-hex	十六进制表示的 IPv6 主机地址
13	ipv6-net	IPv6 网络地址、斜线、有效位数
14	ipv6-net-mask	IPv6 网络地址、斜线、网络掩码

vlan-name

可选。该属性表示地址所属的虚拟局域网（VLAN）的名称。

vlan-num

可选。该属性表示地址所属的 VLAN 号。

4.2.7.3. User 类

User 类用于描述用户。该类主要用作 UserId 聚合类的“容器”类，如图 16 所示。

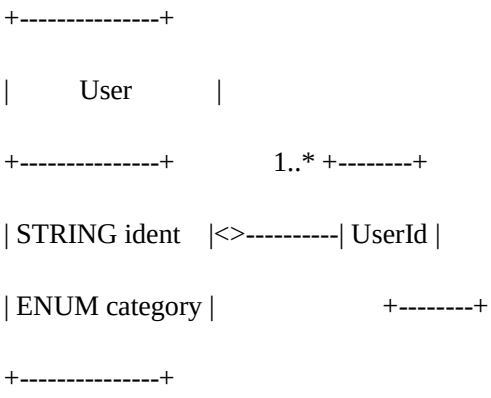


图 16 User 类

User 类包含以下聚合类：

UserId

User 类可包含一个或多个 UserId 类。该类为用户标识，如“type”属性所示（见 4.2.7.3.1 节）。

该类在 IDMEF DTD 中的定义如下：

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

%attlist.global;

>

User 类有两种属性：

ident

可选。该属性为用户的唯一标识。请参见 3.2.9 节。

category

可选。该属性表示用户类型。该属性的值如下所示，默认为“unknown”（见第 10 节）。

序号	关键字	说明
0	unknown	用户类型未知
1	application	应用用户
2	os-device	操作系统或设备用户

4.2.7.3.1. UserId 类

UserId 类提供用户的特定信息。User 类可利用多个 UserId 类表示用户之间的过渡或提供用户（或进程）权限的完整信息。

UserId 类由两个聚合类组成，如图 17 所示。

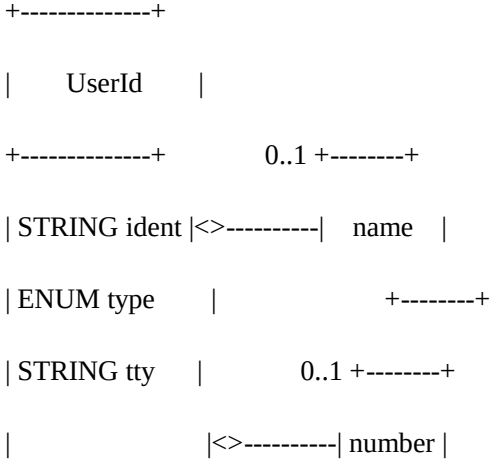


图 17 UserId 类

UserId 类由以下聚合类组成：

name

UserId 类可包含一个 name 类，也可将其省略。该类为字符串型，表示用户名或组名。

number

UserId 类可包含一个 number 类，也可将其省略。该类为整型，表示用户编号或组号。

UserId 类在 IDMEF DTD 中的定义如下：

```
<!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
      %attlist.global;
    >
```

UserId 类有三种属性：

ident

可选。该属性为用户 ID 的唯一标识，详情请参见 3.2.9 节。

type

可选。该属性表示用户类型。该属性的值如下所示，默认为“original-user”（见第 10

节）。

序号	关键字	说明
0	current-user	用户或进程当前使用的用户 ID。在 UNIX 系统中，一般为“真实的”用户 ID。
1	original-user	当前上报用户或进程的真实身份。在进行审计和支持从“audit id”令牌中提取用户 ID 的系统中，应使用该值。若系统不支持此种操作且用户已登录系统，则应使用“login id”。
2	target-user	用户或进程试图获取的用户身份。以 UNIX 系统为例，当用户在系统中试图使用 su、rlogin、telnet 等 ID 时，适用该值。
3	user-privs	用户或进程可使用的另一个用户 ID 或与文件权限关联的用户 ID。在 UNIX 系统中，对用户或进程而言，为“有效的”用户 ID；对文件而言，为所有者权限。可用多个此类 UserId 元素指定多项权限。
4	current-group	用户或进程当前使用的组 ID（若适用）。在 UNIX 系统中，一般指“真实的”组 ID。
5	group-privs	组或进程可使用的另一个组 ID 或与文件权限关联的组 ID。在 UNIX 系统中，对组或进程而言，为“有效的”组 ID；对文件而言，为组权限。在 BSD 系列的 UNIX 系统中，使用多个此类 UserId 元素覆盖“组列表”的所有组 ID。

6	other-privs	对用户、组或进程不适用，仅适用于文件。
		用户不匹配文件的用户或组权限时，
		分配给用户这种文件权限。在 UNIX 系统中，
		指“world”权限。

+------+--------------+--+

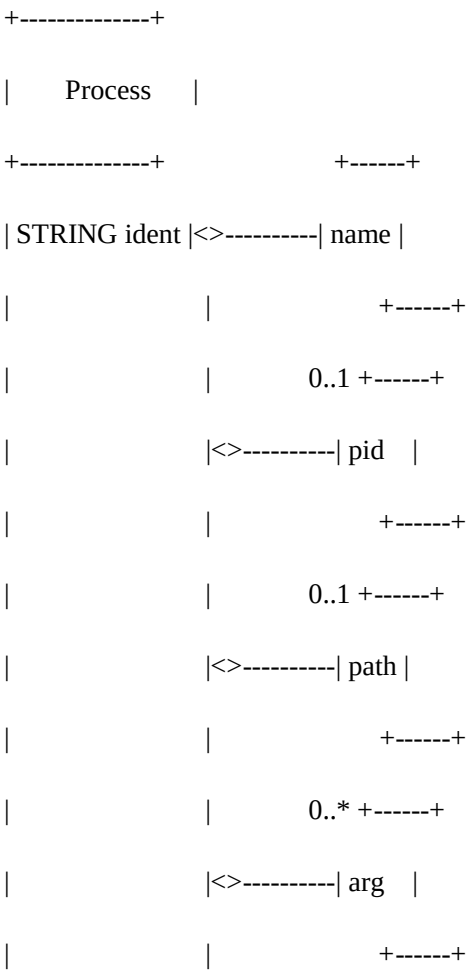
tty

可选。该属性为字符串型，表示用户正在使用的终端设备（TTY）。

4.2.7.4. Process 类

Process 类表示源、目的和分析器上正在执行的进程。

Process 类由五个聚合类组成，如图 18 所示。



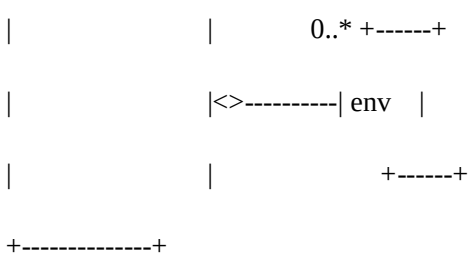


图 18 Process 类

Process 类由以下聚合类组成：

name

Process 类包含一个 name 类。该类为字符串型，表示正在执行的程序的名称。该类指定程序的短名称，程序路径和参数信息由其他类提供。

pid

Process 类可包含一个 pid 类，也可将其省略。该类为整型，表示进程 ID。

path

Process 类可包含一个 path 类，也可将其省略。该类为字符串型，表示正在执行的程序的完整路径。

arg

Process 类可包含一个 arg 类，也可将其省略。该类为字符串型，表示程序的命令行参数。可多次使用该类指定多个参数（参数的指定顺序被视为其出现的顺序）。

env

Process 类可包含一个 env 类，也可将其省略。该类为字符串型，表示进行进程相关的环境字符串，一般格式为：VARIABLE=value。可多次使用该类指定多个环境字符串。

Process 类在 IDMEF DTD 中的定义如下：

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

Process 类有一个属性：

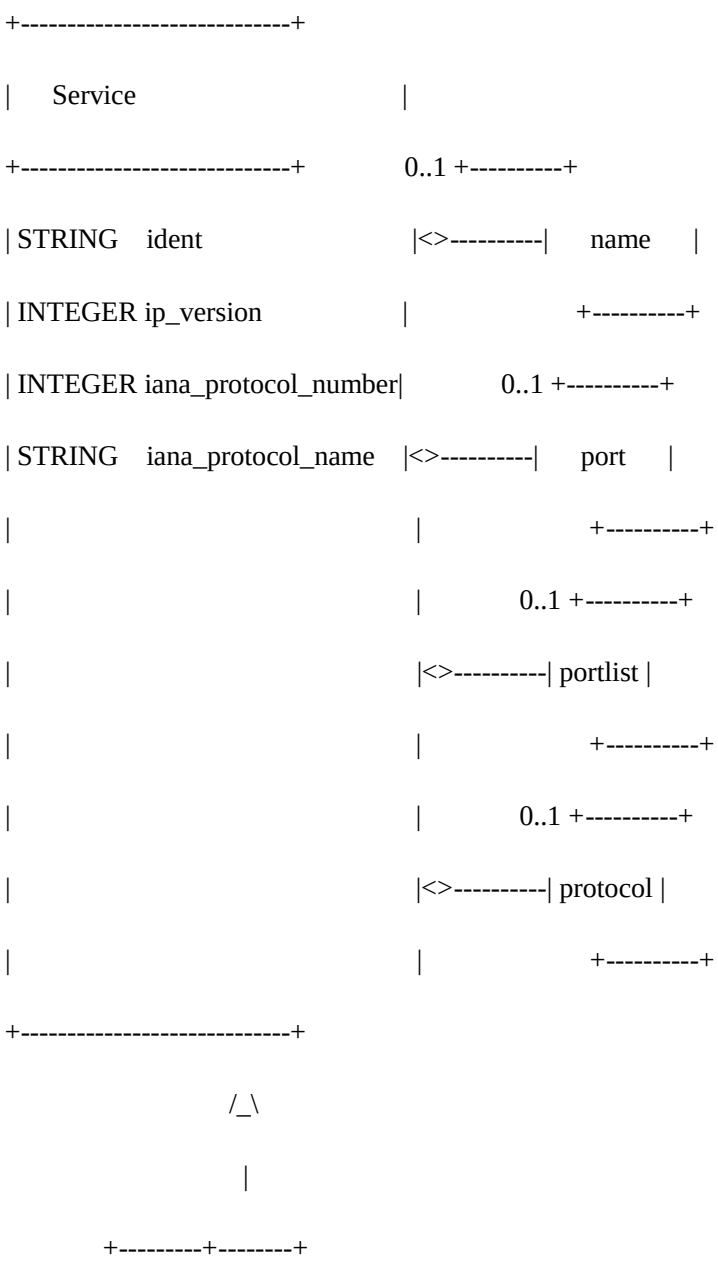
ident

可选。该属性为进程的唯一标识。详情请参见 3.2.9 节。

4.2.7.5. Service 类

Service 类表示源和目的上的网络服务。该类可通过服务名称、端口和协议表示服务。若 Service 类作为 Source（源）的聚合类出现，表示该服务为相关活动的发起方，且该服务“隶属于”Node（节点），Process（进程）和 User（用户）信息包含在 Source 中。同样，若 Service 类作为 Target（目的）的聚合类出现，表示该服务为相关活动的目的方，且该服务“隶属于”Node，Process 和 User 信息包含在 Target 中。若 Service 类在 Source 和 Target 上均出现，则源和目的上的服务信息应相同。如源和目的的服务信息相同且执行者仅希望在源或目的上执行服务，则执行者应将 Service 类指定为 Target 类的一个聚合类。

Service 类由四个聚合类组成，如图 19 所示。



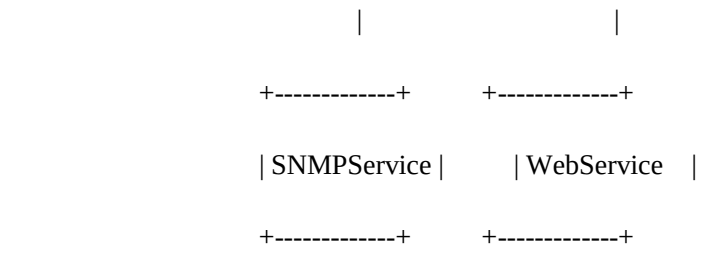


图 19 Service 类

Service 类由以下聚合类组成：

name

Service 类可包含一个 name 类，也可将其省略。该类为字符串型，表示服务名称。应尽可能使用 IANA 列表中的公认端口的名称。

port

Service 类可包含一个 port 类，也可将其省略。该类为整型，表示使用的端口号。

portlist

Service 类可包含一个 portlist 类，也可将其省略。该类为端口列表，列出所使用的端口。有关端口列表的格式规则，请参见 3.2.8 节。若指定了该类，必须对该列表的所有元素应用 iana_protocol_number 和 iana_protocol_name。

protocol

Service 类可包含一个 protocol 类，也可将其省略。该类为字符串型，表示正在使用的协议的其他信息。当指定了 Service 类的 iana_protocol_number 和/或 iana_protocol_name 属性时，该协议字段用于设置正在使用的协议的其他信息。

Service 类必须由名称（或端口）或端口列表表示。协议可选。不支持其他字段组合。

Service 类在 IDMEF DTD 中的定义如下：

```
<!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
        %attlist.global;
    >
```

Service 类有四种属性：

ident

可选。该属性为服务的唯一标识。详情请参见 3.2.9 节。

ip_version

可选。该属性为整型，表示 IP 版本号。

iana_protocol_number

可选。该属性为整型，表示 IANA 协议号。

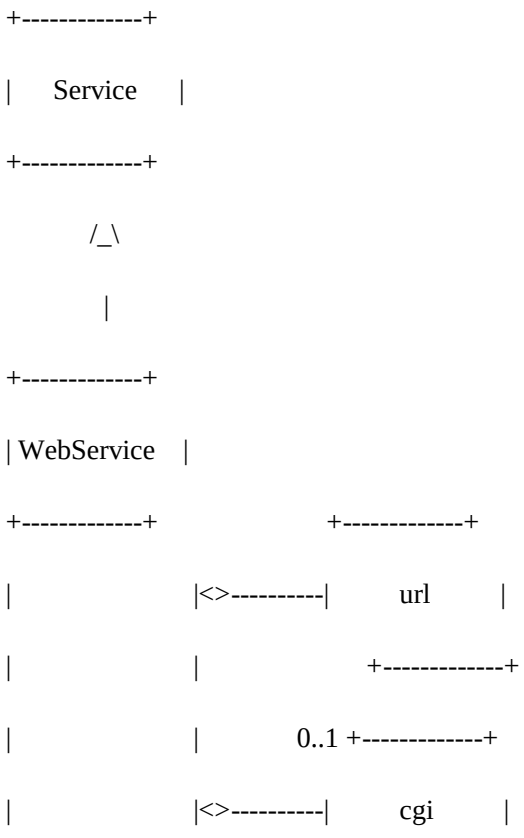
iana_protocol_name

可选。该属性为字符串型，表示 IANA 协议名称。

4.2.7.5.1. WebService 类

WebService 类描述 web 流量的其他相关信息。

WebService 类由四个聚合类组成，如图 20 所示。



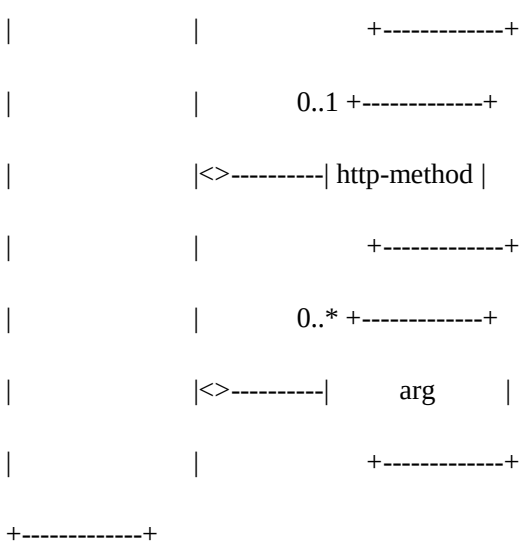


图 20 WebService 类

WebService 类由以下聚合类组成：

url

WebService 类包含一个 url 类。该类为字符串型，表示请求中的 URL。

cgi

WebService 类可包含一个 cgi 类，也可将其省略。该类为字符串型，表示不带参数的请求中的 CGI 脚本。

http-method

WebService 类可包含一个 http-method 类，也可将其省略。该类为字符串型，表示请求中采用的 HTTP 方法（PUT 或 GET）。

arg

WebService 类可包含一个 arg 类，也可将其省略。该类为字符串型，表示传递给 CGI 脚本的参数。

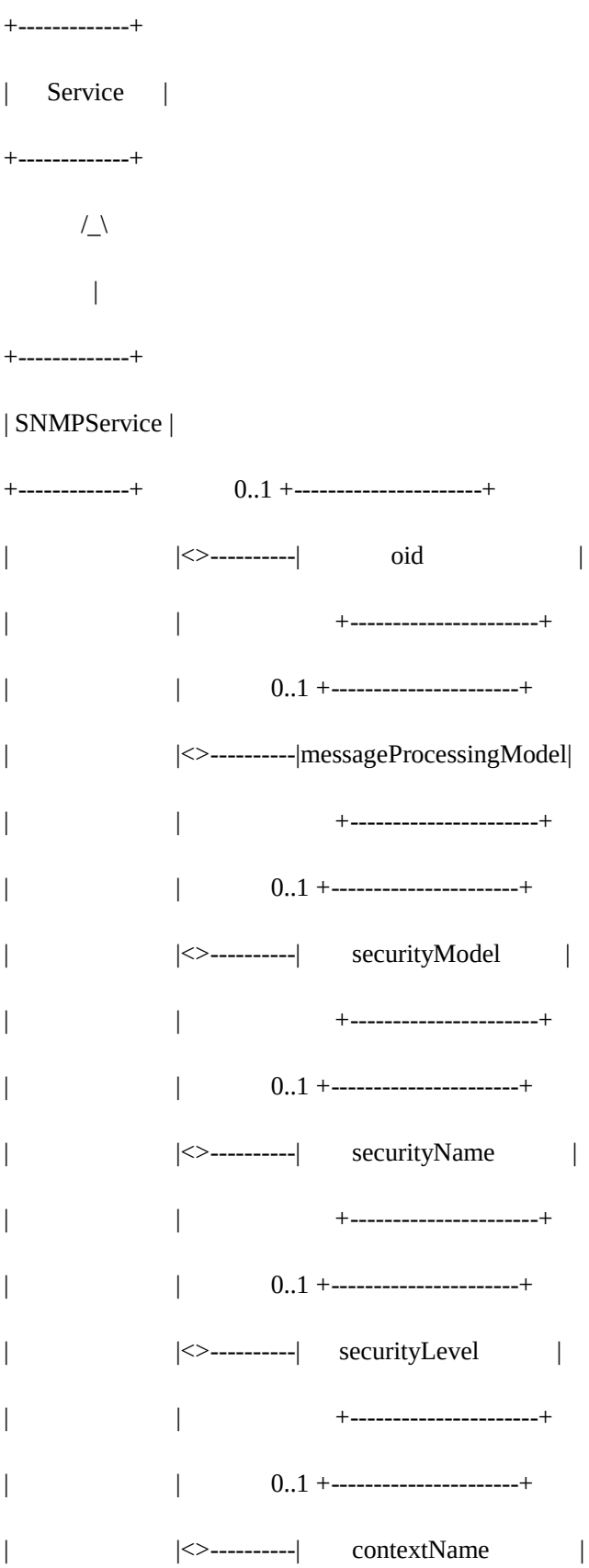
WebService 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT WebService                    (
    url，cgi?，http-method?，arg*
  )>
<!ATTLIST WebService
    %attlist.global;
  >
```

4.2.7.5.2. SNMPService 类

SNMPService 类描述 SNMP 流量的其他相关信息。有关 SNMPService 类的聚合类的描述应与 RFC 3411 和 RFC 3584 中的描述一致。

SNMPService 类由八个聚合类组成，如图 21 所示。



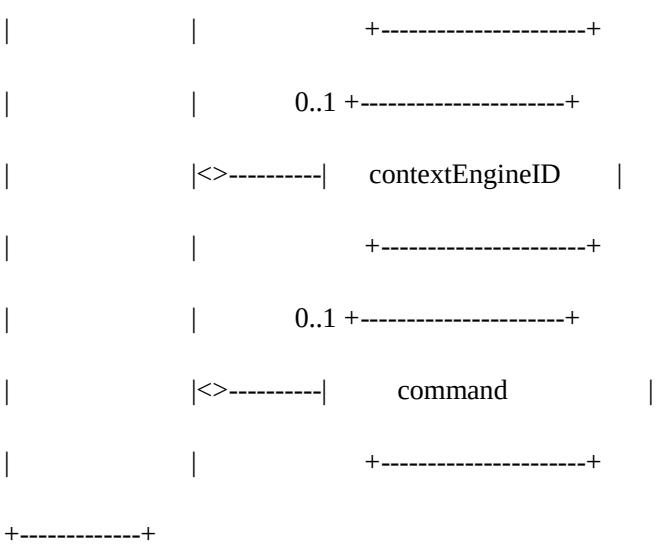


图 21SNMPService 类

SNMPService 类由以下聚合类组成：

oid

SNMPService 类可包含一个 oid 类，也可将其省略。该类为字符串型，表示请求中的对象标识。

messageProcessingModel

SNMPService 类可包含一个 messageProcessingModel 类，也可将其省略。该类为整型，表示 SNMP 版本；值 0 表示 SNMPv1；值 1 表示 SNMPv2c；值 2 表示 SNMPv2u 和 SNMPv2*；值 3 表示 SNMPv3。有关该类的值的详细描述，请参见 RFC 3411 的第 5 节。

securityModel

SNMPService 类可包含一个 securityModel 类，也可将其省略。该类为整型，表示所用的安全模型；值 0 表示任意安全模型；值 1 表示 SNMPv1；值 2 表示 SNMPv2c；值 3 表示 USM。有关该类的值的详细描述，请参见 RFC 3411 的第 5 节。

securityName

NMPService 类可包含一个 securityName 类，也可将其省略。该类为字符串型，表示对象的安全名称。详情请参见 RFC 3411 中的 3.2.2 节。

securityLevel

SNMPService 类可包含一个 securityLevel 类，也可将其省略。该类为整型，表示 SNMP 请求的安全级别。详情请参见 RFC 3411 的 3.4.3 节。

contextName

SNMPService 类可包含一个 contextName 类，也可将其省略。该类为字符串型，表示对象的上下文名称。详情请参见 RFC 3411 的 3.3.3 节。

contextEngineID

SNMPService 类可包含一个 contextEngineID 类，也可将其省略。该类为字符串型，表示对象的上下文引擎标识。详情请参见 RFC 3411 的 3.3.2 节。

command

SNMPService 类可包含一个 command 类，也可将其省略。该类为字符串型，表示发送给 SNMP 服务器的命令（如 GET 和 SET 等）。

若 SNMP 消息存在其他字段且这些字段应包含在 IDMEF 告警中，则这些字段必须位于其他数据结构中，其含义为 RFC 3411 的第 5 节中的对象定义，值位于其他数据负载中。

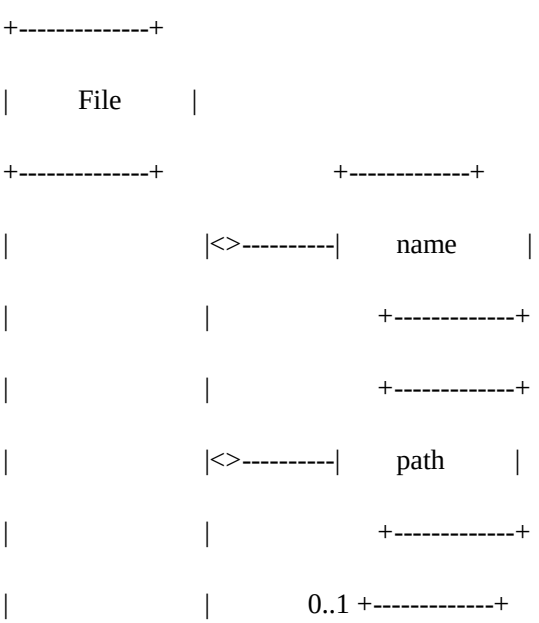
SNMPService 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT SNMPService                    (
    oid?，messageProcessingModel?，securityModel?，securityName?,
    securityLevel?，contextName?，contextEngineID?，command?
  )>
<!ATTLIST SNMPService
    %attlist.global;
  >
```

4.2.7.6. File 类

File 类提供目的上所创建、删除或修改的文件或类文件对象的特定信息。这些信息提供事件发生前或发生时的文件设置，这取决于该类的“category”属性的设置。

File 类由 11 个聚合类组成，如图 22 所示。



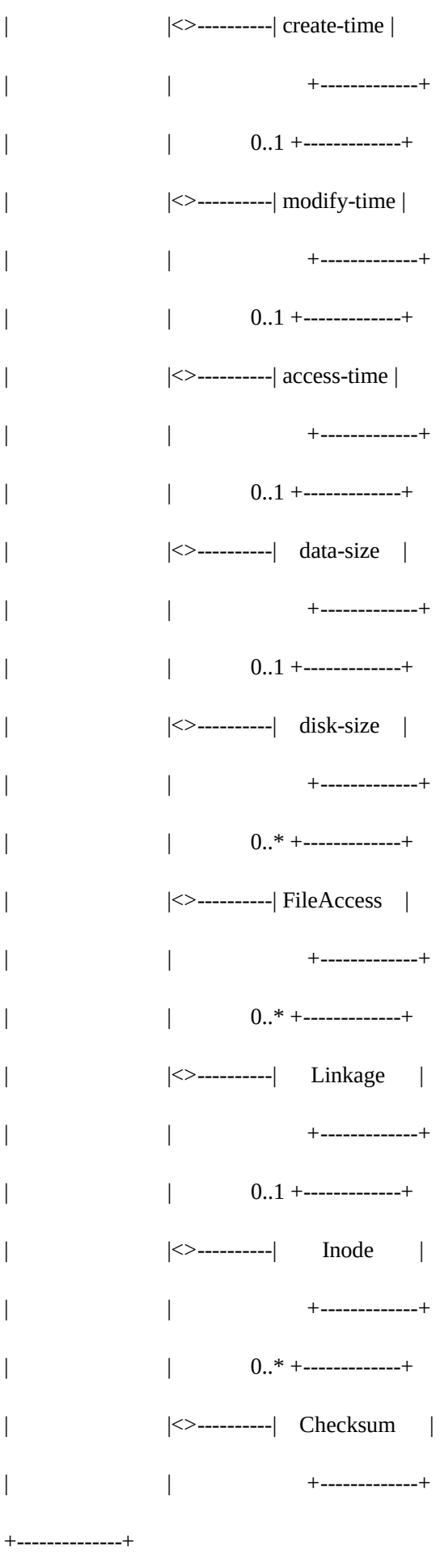


图 22 File 类

File 类由以下聚合类组成：

name

File 类包含一个 name 类。该类为字符串型，表示告警对应的文件名，不包含文件路径。

path

File 类包含一个 path 类。该类为字符串型，表示文件的完整路径，包括文件名。为方便告警处理，文件路径名应尽量基于通用原则表示。

对于 Windows 系统，远程文件的路径应基于通用命名规则（UNC）指定，而本地文件的路径应通过盘符指定如 C:\boot.ini。对于 UNIX 系统，网络文件系统上的文件路径应利用所挂载资源的名称，而非本地挂载点如 fileserver:/usr/local/bin/foo。挂载点可通过 Linkage 元素提供。

create-time

File 类可包含一个 create-time 类，也可将其省略。该类为日期时间类型，表示文件创建时间。请注意该类不是 UNIX 系统中的 st_ctime 文件属性（该属性不是文件创建时间）。UNIX 系统中的 st_ctime 文件属性位于 Inode 类中。

modify-time

File 类可包含一个 modify-time 类，也可将其省略。该类为日期时间类型，表示文件的最新修改时间。

access-time

File 类可包含一个 access-time 类，也可将其省略。该类为日期时间类型，表示文件的最新访问时间。

data-size

File 类可包含一个 data-size 类，也可将其省略。该类为整型，表示数据大小，单位为字节。一般，该类指文件大小。在 UNIX 文件系统中，该类的值等同于 stat.st_size；在 Windows 的 NTFS 文件系统中，该类的值的等同于有效数据长度（VDL）。

disk-size

File 类可包含一个 disk-size 类，也可将其省略。该类为整型，表示存储文件的磁盘的物理空间大小，单位为字节。在 UNIX 文件系统中，该类的值等于 512 与 stat.st_blocks 的乘积；在 Windows 的 NTFS 文件系统中，该类的值相当于文件结束符（EOF）。

FileAccess

File 类可包含多个 FileAccess 类，也可将其省略。该类表示文件的访问权限。

Linkage

File 类可包含多个 Linkage 类，也可将其省略。该类表示文件所链接的文件系统对象

（对文件的其他引用）。

Inode

File 类可包含多个 Inode 类，也可将其省略。该类表示文件的索引节点信息（UNIX 相关）。

Checksum

File 类可包含多个 Checksum 类，也可将其省略。该类表示文件的校验和信息。

File 类在 IDMEF DTD 中的定义如下：

```
<!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
    %attlist.global;
  >
```

File 类有四个属性（一个必选，三个可选）。

ident

可选。该属性为文件的唯一标识。详情请参见 3.2.9 节。

category

必选。该属性表示所提供的信息的情境。该属性的值如下所示，无缺省值（详情请参见第 10 节）。

序号	关键字	说明
0	current	文件信息为上报的变化发生后的信息。
1	original	文件信息为上报的变化发生前的信息。

fstype

可选。该属性表示文件所在的文件系统的类型。这一属性决定了路径名称和其他属性的含义。

序号	关键字	说明
0	ufs	伯克利 UNIX 快速文件系统
1	efs	Linux“efs”文件系统
2	nfs	网络文件系统
3	afs	Andrew 文件系统
4	ntfs	Windows NT 文件系统
5	fat16	16 位 Windows FAT 文件系统
6	fat32	32 位 Windows FAT 文件系统
7	pcfs	CD-ROM 上使用的 PC（MS-DOS）文件系统
8	joliet	Joliet CD-ROM 文件系统
9	iso9660	ISO 9660 CD-ROM 文件系统

file-type

可选。该属性表示文件类型，为 MIME 类型中的一种。

4.2.7.6.1. FileAccess 类

FileAccess 类表示文件的访问权限。该类所表示的文件权限在整个操作系统均有效。

FileAccess 类由两个聚合类组成，如图 23 所示。

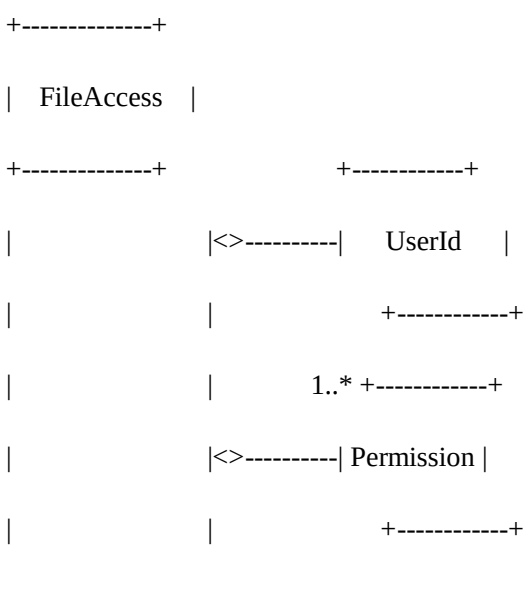


图 23 FileAccess 类

FileAccess 类由以下聚合类组成：

UserId

FileAccess 类包含一个 UserId 类。该类表示权限所适用的用户（或组）。在这种情况下，“type”属性的值必须为“user-privs”、“group-privs”或“other-privs”，不得设置为其他值。

Permission

FileAccess 类包含一个或多个 Permission 类。该类为枚举型，表示所允许的访问权限级别。该类的值如下所示，无缺省值（见第 10 节）。

序号	关键字	说明
0	noAccess	该用户不具有任何访问权限
1	read	该用户具有文件读权限
2	write	该用户具有文件写权限
3	execute	该用户有文件执行权限
4	search	该用户可搜索本文件

| | | （适用于 UNIX 目录的“执行”权限） |

| 5 | delete | 该用户可删除本文件 |

| 6 | executeAs | 该用户可以其他用户身份执行本文件 |

| 7 | changePermissions | 该用户可修改对本文件的访问权限 |

| 8 | takeOwnership | 该用户可获取本文件的所有权 |

+------+------------------+-------------------------------------+

changePermissions 和 takeOwnership 两个字符串适用于 Windows 系统。UNIX 系统中，文件所有人均具备“changePermissions”权限，即使该用户没有其他权限。Windows 系统中的“Full Control”表示列举文件的所有权限。“executeAs”字符串相当于 UNIX 系统中的 set-user-id 和 set-group-id 特性。

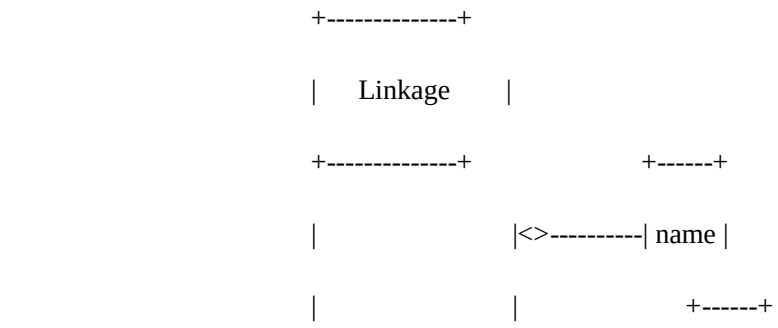
FileAccess 类在 IDMEF DTD 中的定义如下：

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


4.2.7.6.2. Linkage 类

Linkage 类表示 File 元素指定的文件与文件系统中的其他对象之间的文件系统连接。例如，若 File 元素为符号链接或快捷方式，Linkage 元素应包含该链接所指向的对象的名称。必要时，可提供关于 Linkage 元素指定的对象与另外的 File 元素的更多信息。

Linkage 类由三个聚合类组成，如图 24 所示。



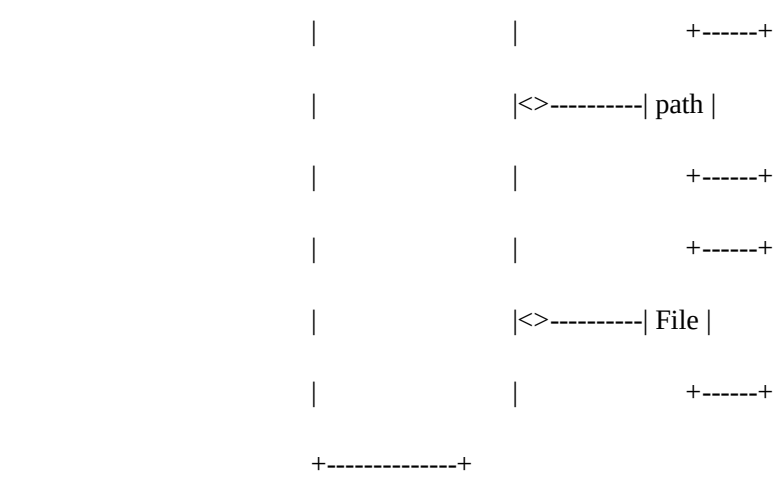


图 24 Linkage 类

Linkage 类由以下聚合类组成：

name

Linkage 类包含一个 name 类。该类为字符串型，表示文件系统对象，不包含路径。

path

Linkage 类包含一个 path 类。该类为字符串型，表示文件系统对象的完整路径，包括对象名称。为方便告警处理，路径名应尽量基于通用原则表示。

File

Linkage 类包含一个 file 类。若提供文件的更多信息，可利用<File>元素代替<name>和<path>元素。

Linkage 类在 IDMEF DTD 中的定义如下：

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

>

Linkage 类有一个属性：

category

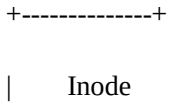
该属性表示链接所指向的对象的类型。该属性的值如下所示，无缺省值 （见第 10 节）。

序号	关键字	说明
0	hard-link	<name>元素表示本文件的另一个名称。与其他系统相比，该信息在 NTFS 系统中更易获取。
1	mount-point	父目录的<name>和<path>元素所指定的目录的别名。
2	reparse-point	仅适用于 Windows 系统；不包括符号链接与挂载点，后者是特定类型的重解析点。
3	shortcut	Windows 的“快捷方式（Shortcut）”指示的文件。快捷方式与符号链接不同，快捷方式的内容对管理器可能会很重要。
4	stream	Windows 中的备用数据流（ADS）；MacOS 中的 Fork。它作为单独的文件系统实体，被视为主<File>的扩展。
5	symbolic-link	<name>元素表示链接指向的文件。

4.2.7.6.3. Inode 类

Inode 类表示 UNIX 文件系统的索引节点包含的其他信息。

Inode 类由六个聚合类组成，如图 25 所示。



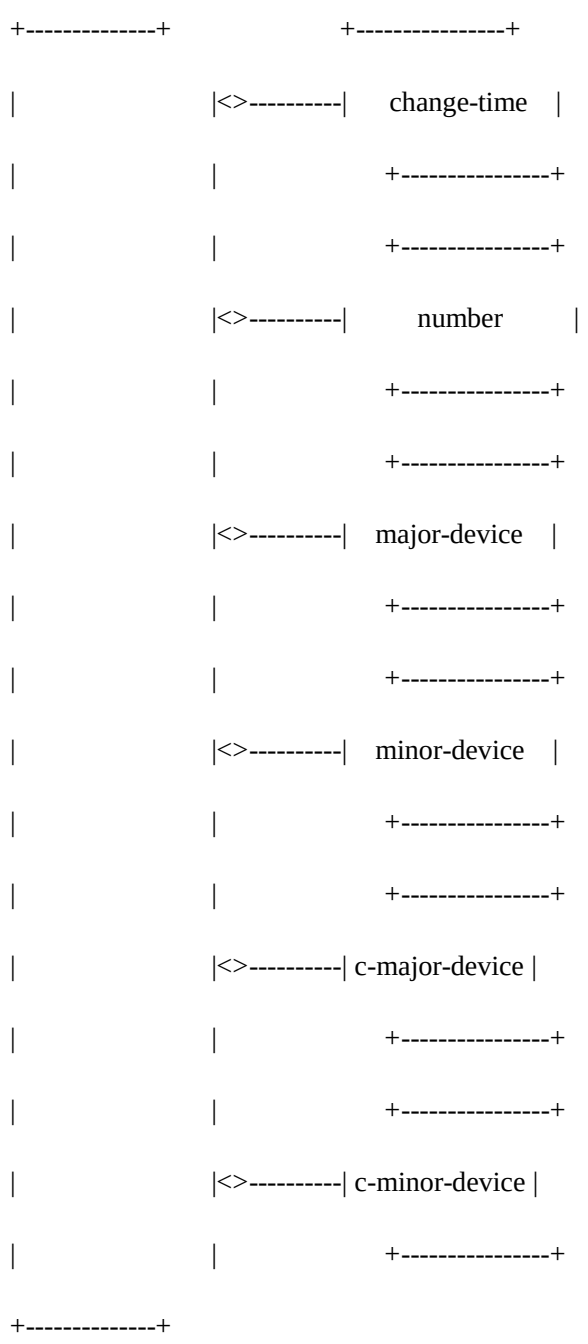


图 25 Inode 类

Inode 类由以下聚合类组成：

change-time

Inode 类可包含一个 change-time 类，也可将其省略。该类为日期时间型，表示索引节点的最新变化时间，由“struct stat”中的 st_ctime 元素指定。

number

Inode 类可包含一个 number 类，也可将其省略。该类为整型，表示索引节点编号。

major-device

Inode 类可包含一个 major-device 类，也可将其省略。该类为整型，表示文件所在设备的

主设备号。

minor-device

Inode 类可包含一个 minor-device 类，也可将其省略。该类为整型，表示文件所在设备的次设备号。

c-major-device

Inode 类可包含一个 c-major-device 类，也可将其省略。该类为整型，表示文件的主设备，前提是文件为字符设备文件。

c-minor-device

Inode 类可包含一个 c-minor-device 类，也可将其省略。该类为整型，表示文件的次设备，前提是文件为字符设备文件。

注意，<number>、<major-device>和<minor-device>必须同时指定，且<c-major-device>和<c-minor-device>必须同时指定。

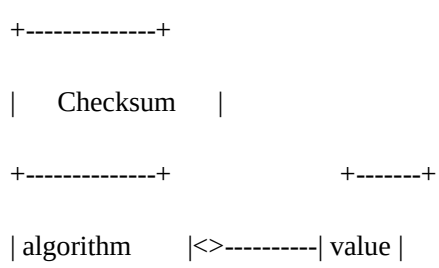
Inode 类在 IDMEF DTD 中的定义如下：

```
<!ELEMENT Inode                                    (
    change-time?，(number，major-device，minor-device)?,
    (c-major-device，c-minor-device)?
  )>
<!ATTLIST Inode
    %attlist.global;
  >
```

4.2.7.6.4. Checksum 类

Checksum 类表示文件相关的校验和信息。校验和信息可由文件完整性检查工具及其他工具提供。

Checksum 类由两个聚合类组成，如图 26 所示。



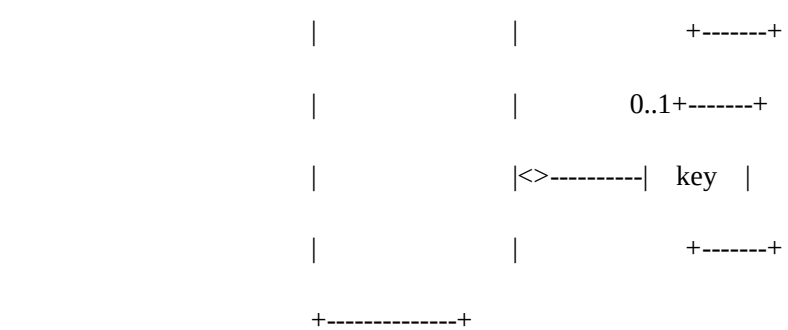


图 26 Checksum 类

Checksum 类由以下聚合类组成：

value

Checksum 类包含一个 value 类。该类为字符串型，表示校验和值。

key

Checksum 类包含一个 key 类。该类为字符串型，表示校验和密钥（若有）。

Checksum 类在 IDMEF DTD 中的定义如下：

```
<!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
      %attlist.global;
    >
```

Checksum 类有一个属性：

algorithm

该属性表示计算校验和所用的加密算法。该属性的值如下所示，无缺省值 （见第 10 节）。

序号	关键字	说明
0	MD4	MD4 算法
1	MD5	MD5 算法
2	SHA1	SHA1 算法
3	SHA2-256	256 位摘要长度的 SHA2 算法
4	SHA2-384	384 位摘要长度的 SHA2 算法
5	SHA2-512	512 位摘要长度的 SHA2 算法
6	CRC-32	32 位校验长度的 CRC 算法
7	Haval	Haval 算法
8	Tiger	Tiger 算法
9	Gost	Gost 算法

5. IDMEF 扩展

随着入侵检测系统的发展，IDMEF 数据模型和 DTD 也将随之发生变化。在演进过程中，为支持新特性，数据模型和 DTD 将进行扩展。本章将介绍这些扩展。随着这些扩展逐步成熟，数据模型和 DTD 可并入该规范的未来版本。

5.1 数据模型扩展

可通过继承和聚合两种机制对 IDMEF 数据模型进行扩展：

- 继承表示超类和子类之间的一种关系，具体指子类继承超类的所有属性、操作和关系。
- 此类关系也称为属于（is-a）或类属（kind-of）关系，子类可能具备仅适用于该子类但不适用于超类的其他属性或操作。
- 聚合是一种关联关系形式，指整体与部分之间的关系。此类关系也称为包含（part-of）关系。在这种情况下，聚合类包含其所有属性及其部分相关的很多属性（由发生指标指定）。
- 在这两种机制中，首选继承关系，因为该类关系保留了现有数据模型结构及在该结构的类中执行的操作（方法）。

请注意，IDMEF DTD 扩展规则（见 5.2 节）对数据模型进行扩展的地方进行了限制。

5.2 IDMEF DTD 扩展

IDMEF DTD 可通过以下两种方法扩展：

1. AdditionalData 类（见 4.2.4.6 节）允许实施者在告警或心跳消息中包含任意“原子”数据项（整型或字符串型）。应尽可能采用这种方法。详情请参见 7.4 和 7.5 节。

2. AdditionalData 类允许实施者利用描述任意复杂数据类型和关系的其他 DTD“模块”扩展 IDMEF DTD。扩展方法如下所示。

要利用新 DTD“模块”扩展 IDMEF DTD，必须执行以下步骤：

1. 文档声明中必须指定 DTD 位置，明确命名空间且包含扩展 DTD 的位置，然后引用命名空间。

2. 可通过定义多个命名空间和 DTD 位置，然后对其引用，进行多项扩展。

3. 扩展 DTD 必须在单独的 XML 命名空间中声明其所有元素和属性。扩展 DTD 不得在“idmef”或默认命名空间中声明任何元素或属性。

4. 扩展必须仅包含在 IDMEF 告警和心跳消息中，通过其“type”属性包含“xml”值的<AdditionalData>元素指定。例如：

下例定义并引用了“vendorco”命名空间，这样 XML 解析器可读取该扩展的 DTD。

```
<idmef:IDMEF-Message version="1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:idmef="http://iana.org/idmef"
  xmlns:vendorco="http://vendor.com/idmef"
xsi:schemaLocation="http://vendor.com/idmef http://v.com/vidmef.xsd">
 <idmef:Alert messageid="...">
      ...
 <idmef:AdditionalData type="xml" meaning="VendorExtension">
  <idmef:xml>
   <vendorco:TestVendor a="attribute of example"
    xmlns:vendorco="http://vendor.com/idmef"
xsi:schemaLocation="http://vendor.com/idmef http://v.com/vidmef.xsd">
     <vendorco:content>content element of example</vendorco:content>
   </vendorco:TestVendor>
  </idmef:xml>
 </idmef:AdditionalData>
 </idmef:Alert>
</idmef:IDMEF-Message>
```

有关 IDMEF DTD 扩展的另一实例，请参见 7.8 节。

6. 特别注意事项

本章介绍了 IDMEF 实施者必须特别注意的一些事项。

6.1 XML 的有效性及正确格式

据预计，符合 IDMEF 的应用程序在其通信中一般不涉及 IDMEF DTD。然而事实并非如此，DTD 会在 IDMEF 消息中的文档类型定义中引用。这些 IDMEF 文档格式正确且有效，如 http://www.w3.org/TR/2000/REC-xml-20001006 中的定义。

规定其他 IDMEF 文档不应包含文档序言（如 IDMEF 格式的数据库中的记录）。这些文档虽格式正确，但无效。

一般，格式正确指文档中存在一个元素，包含所有内容（如<Book>），且所有其他元素正确嵌套，不能互相交叉（如不应在一个“chapter”中嵌入另一个“chapter”）。

有效性指文档格式正确且符合特定规则（在文档类型定义中明确），例如文档中哪些元素合法以及元素之间如何嵌套等（如不应在一个“title”中嵌入另一个“chapter”）。文档只有在引用 DTD 后才有效。

XML 必须能够解析任何格式正确的文档，无论文档是否有效。验证的目的是使文档易于处理（数据解析后进行处理）。若不进行验证，文档中的元素可能会杂乱无序或处理应用无法解析作者“创建”的元素等等。

IDMEF 文档必须采用正确的格式。IDMEF 文档应在切实可行的情况下有效。

6.2 无法识别的 XML 标记

符合 IDMEF 的应用程序有时收到的 IDMEF 消息格式正确，甚至格式正确且有效，但却无法识别消息中包含的标记。存在以下两种情况：

- 虽然这些标记被识别为“合法”（有效文档），但应用程序不知晓该元素内容的语义。
- 标记根本无法识别。

若 IDMEF 消息满足格式要求（见 6.1 节），符合 IDMEF 的应用程序必须继续处理这些包含未知标记的消息。至于如何处理（或忽略）未知元素取决于具体的应用程序。

6.3 分析器与管理器之间的时间同步

本文档不关注分析器与管理器之间进行的日时钟同步，但提供以下建议和说明：

1. 所有分析器和管理器应尽量将其日时钟同步至外部源，例如 NTP（RFC 1305）或 SNTP（RFC 4330）全球定位系统（GPS）、地球同步环境卫星（GOES）、NIST 无线电台 WWV 时钟或其他一些可靠的时间标准。

2. 若外部时间同步无法实现，IDMEF 提供<AnalyzerTime>元素，用于进行基本时间同步（见下文）。

3. 符合 IDMEF 的应用程序应允许用户启用/禁用<AnalyzerTime>作为时间同步配置选项。

很多警告与通过<AnalyzerTime>进行时间同步有关：

1. <AnalyzerTime>最适用于“扁平化”环境。在这类环境中，分析器向某一级别的管理器上报。对于高级管理器、中间中继器和分析器构成的树形拓扑，问题就复杂多了。

2. 若涉及中间消息转接（管理器或其他），有以下两种场景：

- 中间设备可转发完整的 IDMEF 消息或进行聚合或关联，但不得引入延时。在这种情况下，分析器和高级管理器之间进行端到端的时间同步。
- 中间设备需进行存储或其他处理，因此可能会引入延时。在这种情况下，每一跳必须进行时间同步。这就意味着，每个中间设备必须解析 IDMEF 消息，调整所有时间值，重新组织消息，然后发送消息。

3. 在混合模式环境中，某些分析器和管理器使用外部时间同步而有些则不然，此时所有管理器和中间设备必须执行<AnalyzerTime>同步。

这是因为确定双方之间是否真正需要补偿变得急剧复杂，需了解拓扑中的其他部分。

4. 若告警存在备用路径或存储在多个位置，所记录的告警次数可能会因路径的不同而存在差别。

综上所述，<AnalyzerTime>同步在很多环境中或许仅是聊胜于无。若进行此类同步，建议采取以下步骤：

1. 当分析器或管理器发送 IDMEF 消息时，应将<AnalyzerTime>元素设置为其日时钟的当前值。此步骤在消息传输过程中应尽可能晚些时候进行，最好在传输前的最后一刻进行。

2. 管理器收到 IDMEF 消息后，应计算其日时钟与消息中包含的<AnalyzerTime>元素的时间差。应基于该时间差调整<CreateTime>和<DetectTime>元素的时间（也应调整 NTP 时间戳）。

3. 若管理器作为中间设备将 IDMEF 消息发送给更高级别的管理器，且逐跳的同步也生效，管理器应重新生成<AnalyzerTime>的值，即将该元素设置为自己的日时钟时间值。

6.4 NTP 时间戳绕回

以下摘自 RFC 4330：

注意，自 1968 年（2,147,483,648 秒）的某时间以来，设置了最高有效位（整数部分 0 位），且 64 位的字段将在 2036 年的某时间（4,294,967,296 秒）溢出。若 NTP 或 SNTP 在 2036 年仍使用，则有必要采用外部手段表示相对于 1900 年和 2036 年（及为 136 的倍数的其他年份）的时间。这里存在 200 皮秒间隔，该间隔每 136 年当 64 位字段为 0 时会被忽略，习惯上称之为无效时间戳或不可用时间戳。

符合 IDMEF 的应用程序不得发送值为 0 的 NTP 时间戳，除非为了表明该时间戳无效或不可用。若符合 IDMEF 的应用程序必须在轮转时间发送 IDMEF 消息，应用程序应等待 200 皮秒，直到时间戳存在非零值。

以下也摘自 RFC 4330：

NTP 时间戳格式已使用了 17 年，可能在今后的 40 年（秒字段会溢出）仍发挥作用。

鉴于 0 位在 1968 年设置前或许不宜对 NTP 时间戳进行归档，以下规定可延长 NTP 时间戳的使用期限：

若设置了 0 位，如果 UTC 时间在 1968–2036 年之间，UTC 时间为 1900 年 1 月 1 日 0 时 0 分 0 秒。

若未设置 0 位，如果 UTC 时间在 2036-2104 年之间，UTC 时间为 2036 年 2 月 7 日 6 时 28 分 16 秒。

注意，在计算以上对应关系时，2000 年不能视为闰年，而且也要忽略闰秒。

2036 年 2 月 7 日 6 时 28 分 16 秒后使用的符合 IDMEF 的应用程序必须遵循以上规定。

6.5 数字签名

RFC 3275 中规定了标准的 XML 数字签名处理规则和语法。XML 签名对所有类型的数据提供完整性、消息认证和/或签名者认证服务，不论是带有签名的 XML 中的数据还是其他地方的数据。

IDMEF 需求文档（RFC 4766）规定通信协议负责消息完整性和认证，不涉及消息格式。但在以下两个场景中，IDMEF 消息可能需包含数字签名：通过其他不安全协议进行 IDMEF 消息交换或必须对数字签名进行归档供以后使用。

本文档不关注 IDMEF 消息的数字签名使用规范。若需此规范，建议使用 XML 签名标准。

7. 示例

本节示例用以演示如何使用 IDMEF 编码告警数据，仅为说明之用，并不一定是编码这些特定告警的唯一（甚或“最佳”）方法，因此，不应依据这些示例进行实际的告警分类。

7.1 拒绝服务攻击

下述各例说明了如何在 IDMEF 中描述常见的拒绝服务（DoS）攻击。

7.1.1. 泪滴攻击（“Teardrop”Attack）

基于网络的泪滴攻击检测。告警的基本格式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef"
                     version="1.0">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
      <idmef:Node category="dns">
        <idmef:location>Headquarters DMZ Network</idmef:location>
        <idmef:name>analyzer01.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc723b45.0xef449129">
      2000-03-09T10:01:25.93464-05:00
    </idmef:CreateTime>
    <idmef:Source ident="a1b2c3d4">
      <idmef:Node ident="a1b2c3d4-001" category="dns">
        <idmef:name>badguy.example.net</idmef:name>
        <idmef:Address ident="a1b2c3d4-002"
                       category="ipv4-net-mask">
          <idmef:address>192.0.2.50</idmef:address>
```

```
                <idmef:netmask>255.255.255.255</idmef:netmask>
              </idmef:Address>
            </idmef:Node>
          </idmef:Source>
          <idmef:Target ident="d1c2b3a4">
            <idmef:Node ident="d1c2b3a4-001" category="dns">
              <idmef:Address category="ipv4-addr-hex">
                <idmef:address>0xde796f70</idmef:address>
              </idmef:Address>
            </idmef:Node>
          </idmef:Target>
          <idmef:Classification text="Teardrop detected">
            <idmef:Reference origin="bugtraqid">
              <idmef:name>124</idmef:name>
              <idmef:url>http://www.securityfocus.com/bid/124</idmef:url>
            </idmef:Reference>
          </idmef:Classification>
        </idmef:Alert>
      </idmef:IDMEF-Message>
```

7.1.2. “死亡之 Ping”攻击（“Ping of Death”Attack）

基于网络的死亡之 Ping 攻击检测。注意多个目标以及伪造源地址的识别。

注意：为符合 IETF 格式要求，URL 已经过裁剪。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
```

```
<idmef:Analyzer analyzerid="bc-sensor01">
   <idmef:Node category="dns">
      <idmef:name>sensor.example.com</idmef:name>
   </idmef:Node>
</idmef:Analyzer>
<idmef:CreateTime ntpstamp="0xbc71f4f5.0xef449129">
   2000-03-09T10:01:25.93464Z
</idmef:CreateTime>
<idmef:Source ident="a1a2" spoofed="yes">
   <idmef:Node ident="a1a2-1">
      <idmef:Address ident="a1a2-2" category="ipv4-addr">
         <idmef:address>192.0.2.200</idmef:address>
      </idmef:Address>
   </idmef:Node>
</idmef:Source>
<idmef:Target ident="b3b4">
   <idmef:Node>
      <idmef:Address ident="b3b4-1" category="ipv4-addr">
         <idmef:address>192.0.2.50</idmef:address>
      </idmef:Address>
   </idmef:Node>
</idmef:Target>
<idmef:Target ident="c5c6">
   <idmef:Node ident="c5c6-1" category="nisplus">
      <idmef:name>lollipop</idmef:name>
   </idmef:Node>
</idmef:Target>
<idmef:Target ident="d7d8">
```

```
        <idmef:Node ident="d7d8-1">
          <idmef:location>Cabinet B10</idmef:location>
          <idmef:name>Cisco.router.b10</idmef:name>
        </idmef:Node>
      </idmef:Target>
      <idmef:Classification text="Ping-of-death detected">
        <idmef:Reference origin="cve">
          <idmef:name>CVE-1999-128</idmef:name>
          <idmef:url>http://www.cve.mitre.org/cgi-bin/
          cvename.cgi?name=CVE-1999-128</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.2 端口扫描攻击

下述各例说明了如何在 IDMEF 中描述常见的端口扫描攻击。

7.2.1. 非法连接服务

基于主机的策略违背（试图通过“finger”获取信息）检测。注意目标服务的识别以及发起用户（如通过 RFC 1413 获取）。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-sensor01">
      <idmef:Node category="dns">
        <idmef:name>sensor.example.com</idmef:name>
```

```
    </idmef:Node>
</idmef:Analyzer>
<idmef:CreateTime ntpstamp="0xbc72541d.0x00000000">
    2000-03-09T18:47:25+02:00
</idmef:CreateTime>
<idmef:Source ident="a123">
    <idmef:Node ident="a123-01">
        <idmef:Address ident="a123-02" category="ipv4-addr">
            <idmef:address>192.0.2.200</idmef:address>
        </idmef:Address>
    </idmef:Node>
    <idmef:User ident="q987-03" category="os-device">
        <idmef:UserId ident="q987-04" type="target-user">
            <idmef:name>badguy</idmef:name>
        </idmef:UserId>
    </idmef:User>
    <idmef:Service ident="a123-03">
        <idmef:port>31532</idmef:port>
    </idmef:Service>
</idmef:Source>
<idmef:Target ident="z456">
    <idmef:Node ident="z456-01" category="nis">
        <idmef:name>myhost</idmef:name>
        <idmef:Address ident="z456-02" category="ipv4-addr">
            <idmef:address>192.0.2.50</idmef:address>
        </idmef:Address>
    </idmef:Node>
    <idmef:Service ident="z456-03">
```

```
            <idmef:name>finger</idmef:name>
            <idmef:port>79</idmef:port>
          </idmef:Service>
        </idmef:Target>
        <idmef:Classification text="Portscan">
          <idmef:Reference origin="vendor-specific">
            <idmef:name>finger</idmef:name>
            <idmef:url>http://www.vendor.com/finger</idmef:url>
          </idmef:Reference>
          <idmef:Reference origin="vendor-specific"
                          meaning="general documentation">
            <idmef:name>Distributed attack</idmef:name>
            <idmef:url>http://www.vendor.com/distributed</idmef:url>
          </idmef:Reference>
        </idmef:Classification>
      </idmef:Alert>
    </idmef:IDMEF-Message>
```


7.2.2. 简单端口扫描

基于网络的端口扫描检测。这种检测由单一分析器完成。关联引擎检测同类攻击的相关信息，详见第 7.5 节。注意，列举被扫描的端口用的是<portlist>。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer62">
      <idmef:Node category="dns">
        <idmef:location>Headquarters Web Server</idmef:location>
```

```
      <idmef:name>analyzer62.example.com</idmef:name>
    </idmef:Node>
</idmef:Analyzer>
<idmef:CreateTime ntpstamp="0xbc72b2b4.0x00000000">
    2000-03-09T15:31:00-08:00
</idmef:CreateTime>
<idmef:Source ident="abc01">
    <idmef:Node ident="abc01-01">
      <idmef:Address ident="abc01-02" category="ipv4-addr">
        <idmef:address>192.0.2.200</idmef:address>
      </idmef:Address>
    </idmef:Node>
</idmef:Source>
<idmef:Target ident="def01">
    <idmef:Node ident="def01-01" category="dns">
      <idmef:name>www.example.com</idmef:name>
      <idmef:Address ident="def01-02" category="ipv4-addr">
        <idmef:address>192.0.2.50</idmef:address>
      </idmef:Address>
    </idmef:Node>
    <idmef:Service ident="def01-03">
      <idmef:portlist>5-25,37,42,43,53,69-119,123-514
      </idmef:portlist>
    </idmef:Service>
</idmef:Target>
<idmef:Classification text="simple portscan">
    <idmef:Reference origin="vendor-specific">
      <idmef:name>portscan</idmef:name>
```

```
        <idmef:url>http://www.vendor.com/portscan</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
  </idmef:Alert>
</idmef:IDMEF-Message>
```

7.3 本地攻击

下述各例说明了如何在 IDMEF 中描述常见的本地主机攻击。

7.3.1. loadmodule 攻击

基于主机的 loadmodule 利用检测。这种攻击的原理如下：诱使 loadmodule 程序运行其他程序，因为 loadmodule 的 set-user-id 位为 root，被执行程序就会以超级用户权限运行。注意，用了<User>和<Process>来识别试图发动利用攻击的用户和该用户所使用的攻击方法。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-fs-sensor13">
      <idmef:Node category="dns">
        <idmef:name>fileserver.example.com</idmef:name>
      </idmef:Node>
      <idmef:Process>
        <idmef:name>monitor</idmef:name>
        <idmef:pid>8956</idmef:pid>
        <idmef:arg>monitor</idmef:arg>
        <idmef:arg>-d</idmef:arg>
        <idmef:arg>-m</idmef:arg>
        <idmef:arg>idmanager.example.com</idmef:arg>
        <idmef:arg>-l</idmef:arg>
```

```
        <idmef:arg>/var/logs/idlog</idmef:arg>
    </idmef:Process>
</idmef:Analyzer>
<idmef:CreateTime ntpstamp="0xbc7221c0.0x4ccccccc">
    2000-03-09T08:12:32.3-05:00
</idmef:CreateTime>
<idmef:Source ident="a1a2">
    <idmef:User ident="a1a2-01" category="os-device">
        <idmef:UserId ident="a1a2-02"
                              type="original-user">
            <idmef:name>joe</idmef:name>
            <idmef:number>13243</idmef:number>
        </idmef:UserId>
    </idmef:User>
    <idmef:Process ident="a1a2-03">
        <idmef:name>loadmodule</idmef:name>
        <idmef:path>/usr/openwin/bin</idmef:path>
    </idmef:Process>
</idmef:Source>
<idmef:Target ident="z3z4">
    <idmef:Node ident="z3z4-01" category="dns">
        <idmef:name>fileserver.example.com</idmef:name>
    </idmef:Node>
</idmef:Target>
<idmef:Classification text="Loadmodule attack"
                                  ident="loadmodule">
    <idmef:Reference origin="bugtraqid">
        <idmef:name>33</idmef:name>
```

```
            <idmef:url>http://www.securityfocus.com</idmef:url>
          </idmef:Reference>
        </idmef:Classification>
      </idmef:Alert>
    </idmef:IDMEF-Message>
```

入侵检测系统（IDS）还能识别目标用户为 root 用户，显示试图使用的命令。这种情况下，告警如下所示：

```
    <?xml version="1.0" encoding="UTF-8"?>

    <idmef:IDMEF-Message version="1.0"
                         xmlns:idmef="http://iana.org/idmef">
      <idmef:Alert messageid="abc123456789">
        <idmef:Analyzer analyzerid="bc-fs-sensor13">
          <idmef:Node category="dns">
            <idmef:name>fileserver.example.com</idmef:name>
          </idmef:Node>
          <idmef:Process>
            <idmef:name>monitor</idmef:name>
            <idmef:pid>8956</idmef:pid>
            <idmef:arg>monitor</idmef:arg>
            <idmef:arg>-d</idmef:arg>
            <idmef:arg>-m</idmef:arg>
            <idmef:arg>idmanager.example.com</idmef:arg>
            <idmef:arg>-l</idmef:arg>
            <idmef:arg>/var/logs/idlog</idmef:arg>
          </idmef:Process>
        </idmef:Analyzer>
        <idmef:CreateTime ntpstamp="0xbc7221c0.0x4ccccccc">
```

```
    2000-03-09T08:12:32.3-05:00
</idmef:CreateTime>
<idmef:Source ident="a1a2">
    <idmef:User ident="a1a2-01" category="os-device">
        <idmef:UserId ident="a1a2-02" type="original-user">
            <idmef:name>joe</idmef:name>
            <idmef:number>13243</idmef:number>
        </idmef:UserId>
    </idmef:User>
    <idmef:Process ident="a1a2-03">
        <idmef:name>loadmodule</idmef:name>
        <idmef:path>/usr/openwin/bin</idmef:path>
    </idmef:Process>
</idmef:Source>
<idmef:Target ident="z3z4">
    <idmef:Node ident="z3z4-01" category="dns">
        <idmef:name>fileserver.example.com</idmef:name>
    </idmef:Node>
    <idmef:User ident="z3z4-02" category="os-device">
        <idmef:UserId ident="z3z4-03" type="target-user">
            <idmef:name>root</idmef:name>
            <idmef:number>0</idmef:number>
        </idmef:UserId>              </idmef:User>
    <idmef:Process ident="z3z4-04">
        <idmef:name>sh</idmef:name>
        <idmef:pid>25134</idmef:pid>
        <idmef:path>/bin/sh</idmef:path>
    </idmef:Process>
```

```
        </idmef:Target>
        <idmef:Classification text="Loadmodule attack"
                              ident="loadmodule">
        </idmef:Classification>
      </idmef:Alert>
    </idmef:IDMEF-Message>
```

注意，使用了类别标识。

7.3.2. phf 攻击

基于网络的 phf 攻击检测。注意，使用了<WebService>元素提供更多的攻击信息。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-sensor01">
      <idmef:Node category="dns">
        <idmef:name>sensor.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc71e980.0x00000000">
      2000-03-09T08:12:32-01:00
    </idmef:CreateTime>
    <idmef:Source ident="abc123">
      <idmef:Node ident="abc123-001">
        <idmef:Address ident="abc123-002"
                       category="ipv4-addr">
          <idmef:address>192.0.2.200</idmef:address>
```

```
      </idmef:Address>
    </idmef:Node>
    <idmef:Service ident="abc123-003">
      <idmef:port>21534</idmef:port>
    </idmef:Service>
  </idmef:Source>
  <idmef:Target ident="xyz789">
    <idmef:Node ident="xyz789-001" category="dns">
      <idmef:name>www.example.com</idmef:name>
      <idmef:Address ident="xyz789-002"
                     category="ipv4-addr">
        <idmef:address>192.0.2.100</idmef:address>
      </idmef:Address>
    </idmef:Node>
    <idmef:Service>
      <idmef:port>8080</idmef:port>
      <idmef:WebService>
        <idmef:url>
        http://www.example.com/cgi-bin/phf?/etc/group
        </idmef:url>
        <idmef:cgi>/cgi-bin/phf</idmef:cgi>
        <idmef:http-method>GET</idmef:http-method>
      </idmef:WebService>
    </idmef:Service>
  </idmef:Target>
  <idmef:Classification text="phf attack">
    <idmef:Reference origin="bugtraqid">
      <idmef:name>629</idmef:name>
```

```
            <idmef:url>
            http://www.securityfocus.com/bid/629
            </idmef:url>
          </idmef:Reference>
        </idmef:Classification>
      </idmef:Alert>
    </idmef:IDMEF-Message>
```

7.3.3. 文件修改

基于主机的竞争条件攻击检测。注意，使用了<File>元素提供攻击中用到的文件信息。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert>
    <idmef:Analyzer analyzerid="bids-192.0.2.1"
                    ostype="Linux"
                    osversion="2.2.16-3">
      <idmef:Node category="hosts">
        <idmef:name>etude</idmef:name>
        <idmef:Address category="ipv4-addr">
          <idmef:address>192.0.2.1</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc71e980.0x00000000">
      2000-03-09T08:12:32-01:00
    </idmef:CreateTime>
    <idmef:Source spoofed="no">
```

```
    <idmef:Node>
      <idmef:location>console</idmef:location>
      <idmef:Address category="ipv4-addr">
        <idmef:address>192.0.2.1</idmef:address>
      </idmef:Address>
      </idmef:Node>
  </idmef:Source>
  <idmef:Target decoy="no">
    <idmef:Node>
      <idmef:location>local</idmef:location>
      <idmef:Address category="ipv4-addr">
        <idmef:address>192.0.2.1</idmef:address>
      </idmef:Address>
    </idmef:Node>
    <idmef:User category="os-device">
      <idmef:UserId type="original-user">
        <idmef:number>456</idmef:number>
      </idmef:UserId>
      <idmef:UserId type="current-user">
        <idmef:name>fred</idmef:name>
        <idmef:number>456</idmef:number>
      </idmef:UserId>
      <idmef:UserId type="user-privs">
        <idmef:number>456</idmef:number>
      </idmef:UserId>
    </idmef:User>
    <idmef:File category="current" fstype="tmpfs">
      <idmef:name>xxx000238483</idmef:name>
```

```
<idmef:path>/tmp/xxx000238483</idmef:path>
<idmef:FileAccess>
  <idmef:UserId type="user-privs">
    <idmef:name>alice</idmef:name>
    <idmef:number>777</idmef:number>
  </idmef:UserId>
  <idmef:permission perms="read" />
  <idmef:permission perms="write" />
  <idmef:permission perms="delete" />
  <idmef:permission perms="changePermissions" />
</idmef:FileAccess>
<idmef:FileAccess>
  <idmef:UserId type="group-privs">
    <idmef:name>user</idmef:name>
    <idmef:number>42</idmef:number>
  </idmef:UserId>
  <idmef:permission perms="read" />
  <idmef:permission perms="write" />
  <idmef:permission perms="delete" />
</idmef:FileAccess>
<idmef:FileAccess>
  <idmef:UserId type="other-privs">
    <idmef:name>world</idmef:name>
  </idmef:UserId>
  <idmef:permission perms="noAccess" />
</idmef:FileAccess>
<idmef:Linkage category="symbolic-link">
  <idmef:name>passwd</idmef:name>
```

```
            <idmef:path>/etc/passwd</idmef:path>
          </idmef:Linkage>
        </idmef:File>
      </idmef:Target>
      <idmef:Classification text="DOM race condition">
        <idmef:Reference origin="vendor-specific">
          <idmef:name>DOM race condition</idmef:name>
          <idmef:url>file://attack-info/race.html
          </idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.4 系统策略违背

本例中，登录仅限于白天。若用户在晚 10 点后进行登录，会上报策略违背告警。注意，使用了<AdditionalData>元素提供被违背策略的信息。

```
  <?xml version="1.0" encoding="UTF-8"?>

  <idmef:IDMEF-Message version="1.0"
                       xmlns:idmef="http://iana.org/idmef">
    <idmef:Alert messageid="abc123456789">
      <idmef:Analyzer analyzerid="bc-ds-01">
        <idmef:Node category="dns">
          <idmef:name>dialserver.example.com</idmef:name>
        </idmef:Node>
      </idmef:Analyzer>
      <idmef:CreateTime ntpstamp="0xbc72e7ef.0x00000000">
```

```
    2000-03-09T22:18:07-05:00
</idmef:CreateTime>
<idmef:Source ident="s01">
    <idmef:Node ident="s01-1">
        <idmef:Address category="ipv4-addr">
            <idmef:address>127.0.0.1</idmef:address>
        </idmef:Address>
    </idmef:Node>
    <idmef:Service ident="s01-2">
        <idmef:port>4325</idmef:port>
    </idmef:Service>
</idmef:Source>
<idmef:Target ident="t01">
    <idmef:Node ident="t01-1" category="dns">
        <idmef:name>mainframe.example.com</idmef:name>
    </idmef:Node>
    <idmef:User ident="t01-2" category="os-device">
        <idmef:UserId ident="t01-3" type="current-user">
            <idmef:name>louis</idmef:name>
            <idmef:number>501</idmef:number>
        </idmef:UserId>
    </idmef:User>
    <idmef:Service ident="t01-4">
        <idmef:name>login</idmef:name>
        <idmef:port>23</idmef:port>
    </idmef:Service>
</idmef:Target>
<idmef:Classification text="Login policy violation">
```

```
        <idmef:Reference origin="user-specific">
          <idmef:name>out-of-hours activity</idmef:name>
          <idmef:url>http://my.company.com/policies
          </idmef:url>
        </idmef:Reference>
      </idmef:Classification>
      <idmef:AdditionalData type="date-time"
                            meaning="start-time">
        <idmef:date-time>2000-03-09T07:00:00-05:00</idmef:date-time>
      </idmef:AdditionalData>
      <idmef:AdditionalData type="date-time"
                            meaning="stop-time">
        <idmef:date-time>2000-03-09T19:30:00-05:00</idmef:date-time>
      </idmef:AdditionalData>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```


7.5 关联告警

关联引擎而非单个分析器在检测到 7.2.2 节中所述端口扫描后，发送相关告警。下例说明了如何在 IDMEF 中描述这种告警。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-corr-01">
      <idmef:Node category="dns">
```

```
        <idmef:name>correlator01.example.com</idmef:name>
    </idmef:Node>
</idmef:Analyzer>
<idmef:CreateTime ntpstamp="0xbc72423b.0x00000000">
    2000-03-09T15:31:07Z
</idmef:CreateTime>
<idmef:Source ident="a1">
    <idmef:Node ident="a1-1">
        <idmef:Address ident="a1-2" category="ipv4-addr">
            <idmef:address>192.0.2.200</idmef:address>
        </idmef:Address>
    </idmef:Node>
</idmef:Source>
<idmef:Target ident="a2">
    <idmef:Node ident="a2-1" category="dns">
        <idmef:name>www.example.com</idmef:name>
        <idmef:Address ident="a2-2" category="ipv4-addr">
            <idmef:address>192.0.2.50</idmef:address>
        </idmef:Address>
    </idmef:Node>
    <idmef:Service ident="a2-3">
        <idmef:portlist>5-25,37,42,43,53,69-119,123-514
        </idmef:portlist>
    </idmef:Service>
</idmef:Target>
<idmef:Classification text="Portscan">
    <idmef:Reference origin="vendor-specific">
        <idmef:name>portscan</idmef:name>
```

```
            <idmef:url>http://www.vendor.com/portscan</idmef:url>
          </idmef:Reference>
        </idmef:Classification>
        <idmef:CorrelationAlert>
          <idmef:name>multiple ports in short time</idmef:name>
          <idmef:alertident>123456781</idmef:alertident>
          <idmef:alertident>123456782</idmef:alertident>
          <idmef:alertident>123456783</idmef:alertident>
          <idmef:alertident>123456784</idmef:alertident>
          <idmef:alertident>123456785</idmef:alertident>
          <idmef:alertident>123456786</idmef:alertident>
          <idmef:alertident analyzerid="a1b2c3d4">987654321
          </idmef:alertident>
          <idmef:alertident analyzerid="a1b2c3d4">987654322
          </idmef:alertident>
        </idmef:CorrelationAlert>
      </idmef:Alert>
    </idmef:IDMEF-Message>
```


7.6 分析器评估

基于主机检测通过 eject 缓冲区溢出成功获取未授权 root 访问权限的情况。注意，使用了<Assessment>元素提供分析器对攻击的评估与响应信息。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert>
    <idmef:Analyzer analyzerid="bids-192.0.2.1">
```

```
</idmef:Analyzer>
<idmef:CreateTime ntpstamp="0xbc71e980.0x00000000">
   2000-03-09T08:12:32-01:00
</idmef:CreateTime>
<idmef:Source spoofed="no">
   <idmef:Node>
      <idmef:location>console</idmef:location>
      <idmef:Address category="ipv4-addr">
         <idmef:address>192.0.2.1</idmef:address>
      </idmef:Address>
   </idmef:Node>
</idmef:Source>
<idmef:Target decoy="no">
   <idmef:Node>
      <idmef:location>local</idmef:location>
      <idmef:Address category="ipv4-addr">
         <idmef:address>192.0.2.1</idmef:address>
      </idmef:Address>
   </idmef:Node>
   <idmef:User category="os-device">
      <idmef:UserId type="original-user">
         <idmef:number>456</idmef:number>
      </idmef:UserId>
      <idmef:UserId type="current-user">
         <idmef:name>root</idmef:name>
         <idmef:number>0</idmef:number>
      </idmef:UserId>
      <idmef:UserId type="user-privs">
```

```
            <idmef:number>0</idmef:number>
        </idmef:UserId>
    </idmef:User>
    <idmef:Process>
        <idmef:name>eject</idmef:name>
        <idmef:pid>32451</idmef:pid>
        <idmef:path>/usr/bin/eject</idmef:path>
        <idmef:arg>\x90\x80\x3f\xff...\x08/bin/sh</idmef:arg>
    </idmef:Process>
</idmef:Target>
<idmef:Classification
        text="Unauthorized administrative access">
    <idmef:Reference origin="vendor-specific">
        <idmef:name>Unauthorized user to superuser</idmef:name>
        <idmef:url>file://attack-info/u2s.html</idmef:url>
    </idmef:Reference>
</idmef:Classification>
<idmef:Assessment>
    <idmef:Impact severity="high" completion="succeeded"
                type="admin"/>
    <idmef:Action category="notification-sent">
        page
        </idmef:Action>
    <idmef:Action category="block-installed">
        disabled user (fred)
    </idmef:Action>
    <idmef:Action category="taken-offline">
        logout user (fred)
```

```
        </idmef:Action>
        <idmef:Confidence rating="high"/>
      </idmef:Assessment>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.7 心跳

本例为心跳消息，通知管理器自己还“活着并且处于工作状态”。注意，使用了带有“meaning”属性的<AdditionalData>元素提供补充信息。

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Heartbeat messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
      <idmef:Node category="dns">
        <idmef:location>Headquarters DMZ Network</idmef:location>
        <idmef:name>analyzer01.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc722ebe.0x00000000">
      2000-03-09T14:07:58Z
    </idmef:CreateTime>
    <idmef:AdditionalData type="real" meaning="%memused">
      <idmef:real>62.5</idmef:real>
    </idmef:AdditionalData>
    <idmef:AdditionalData type="real" meaning="%diskused">
      <idmef:real>87.1</idmef:real>
```

```
    </idmef:AdditionalData>
  </idmef:Heartbeat>
</idmef:IDMEF-Message>
```

7.8 XML 扩展

扩展 IDMEF DTD 的方法如下例所示。本例中，VendorCo 公司决定在 Node 类中增加地理信息。为此，VendorCo 创建了文档类型定义（DTD），为类定义自己的格式。

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:vendorco="http://vendor.com/idmef"
            targetNamespace="http://vendor.com/idmef"
            elementFormDefault="qualified" >

  <xsd:annotation>
    <xsd:documentation>
      Intrusion Detection Message Exchange Format (IDMEF) Extension
      for geographic information
    </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType name="NodeGeoType">
    <xsd:sequence>
    <xsd:element name="latitude"
                 type="xsd:string" />
    <xsd:element name="longitude"
                 type="xsd:string" />

    <xsd:element name="elevation"
```

```
                type="xsd:string"
                minOccurs="0"
                maxOccurs="1" />
            </xsd:sequence>
        <xsd:attribute name="node-ident"
                type="xsd:integer"
                use="required"/>
    </xsd:complexType>

    <xsd:element name="NodeGeography" type="vendorco:NodeGeoType" />

</xsd:schema>
```

VendorCo:NodeGeography 类包含三大类的地理数据：VendorCo:latitude、VendorCo:longitude 和 VendorCo:elevation。为了将该类中的信息与特定节点关联，提供了 VendorCo:node-ident 属性，这个属性的值与相关 Node 元素的 ident 属性的值相同。

为了利用这个 DTD，VendorCo 根据 5.2 节所述规则在 DTD 中定义了一个参数实体 x-vendorco，然后再引用这个实体。告警中，AdditionalData 元素含有“type”属性（值为“xml”），还包含 VendorCo 元素，如下例所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message version="1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:idmef="http://iana.org/idmef"
    xmlns:vendorco="http://v.com/idmef"
    xsi:schemaLocation="http://v.com/idmef http://v.com/geo.xsd">

  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
      <idmef:Node category="dns">
        <idmef:location>Headquarters DMZ Network</idmef:location>
```

```
      <idmef:name>analyzer01.example.com</idmef:name>
    </idmef:Node>
</idmef:Analyzer>
<idmef:CreateTime ntpstamp="0xbc723b45.0xef449129">
    2000-03-09T10:01:25.93464-05:00
</idmef:CreateTime>
<idmef:Source ident="a1b2c3d4">
    <idmef:Node ident="a1b2c3d4-001" category="dns">
      <idmef:name>badguy.example.net</idmef:name>
      <idmef:Address ident="a1b2c3d4-002" category="ipv4-net-mask">
        <idmef:address>192.0.2.50</idmef:address>
        <idmef:netmask>255.255.255.255</idmef:netmask>
      </idmef:Address>
    </idmef:Node>
</idmef:Source>
<idmef:Target ident="d1c2b3a4">
    <idmef:Node ident="d1c2b3a4-001" category="dns">
      <idmef:Address category="ipv4-addr-hex">
        <idmef:address>0xde796f70</idmef:address>
      </idmef:Address>
    </idmef:Node>
</idmef:Target>
<idmef:Classification text="Teardrop">
    <idmef:Reference origin="bugtraqid">
      <idmef:name>124</idmef:name>
      <idmef:url>http://www.securityfocus.com/bid/124</idmef:url>
    </idmef:Reference>
</idmef:Classification>
```

```
        <idmef:AdditionalData type="xml" meaning="node geo info">
          <idmef:xml>
            <vendorco:NodeGeography
              xmlns:vendorco="http://vendor.com/idmef"
        xsi:schemaLocation="http://v.com/idmef http://v.com/geo.xsd"
              vendorco:node-ident="a1b2c3d4-001">
              <vendorco:latitude>38.89</vendorco:latitude>
              <vendorco:longitude>-77.02</vendorco:longitude>
            </vendorco:NodeGeography>
          </idmef:xml>
        </idmef:AdditionalData>
      </idmef:Alert>
    </idmef:IDMEF-Message>
```

8. IDMEF 文档类型定义（规范）

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ********************************************************************
 ********************************************************************
*** 入侵检测消息交换格式（IDMEF）XML DTD1.0 版，***
***                    2006 年 3 月 7 日                    ***
***                                                         ***
*** IDMEF XML DTD 的用法与扩展在***
*** RFC 4765《入侵检测消息交换格式》中进行了描述，***
***该文作者为 H. Debar、D. Curry 和 B. Feinstein.        ***
 ********************************************************************
 ******************************************************************* -->

<!--
================================================================

================================================================ 第 1
节. 属性列表声明。
    ================================================================
    ================================================================ -
->

<!--
| IDMEF 元素属性。一般情况下，这些属性的
| 固定值会随着 DTD 新版本的发布而
| 变化。
 -->

<!ENTITY % attlist.idmef                    "
    version                 CDATA                       #FIXED      '1.0'
  ">
```

```
        <!--
        | 所有元素的属性。这些是每个元素都应有的
        | XML 属性。空格处理、语言与命名
        | 空间。
        -->
        <!ENTITY % attlist.global                    "
            xmlns:idmef          CDATA                    #FIXED
                'http://iana.org/idmef'
            xmlns                CDATA                    #FIXED
                'http://iana.org/idmef'
            xml:space            (default | preserve)     'default'
            xml:lang             NMTOKEN                  #IMPLIED
          ">    <!--
============================================================


============================================================================ 第 2
节. 属性值声明。针对特定元素的多个
    ===          属性列表的枚举值。
        ============================================================
        ============================================================ -->


        <!--
    | Action.category 属性值。
        -->
        <!ENTITY % attvals.actioncat                 "
            ( block-installed | notification-sent | taken-offline | other )
          ">


        <!--
    | Address.category 属性值。
        -->
        <!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 )
      ">

    <!--
| AdditionalData.type 属性值。
     -->
    <!ENTITY % attvals.adtype                    "
        ( boolean | byte | character | date-time | integer | ntpstamp |
          portlist | real | string | byte-string | xmltext )
      ">

    <!--
| Impact.completion 属性值。
     -->
    <!ENTITY % attvals.completion                "
        ( failed | succeeded )
      ">
    <!--
     | Values for the File.category attribute.
     -->
    <!ENTITY % attvals.filecat                   "
        ( current | original )
      ">

    <!ENTITY % attvals.fileperm "( noAccess | read | write | execute |
      search | delete | executeAs | changePermissions |
      takeOwnership)" >
    <!--
| UserId.type 属性值。
     -->
```

```
    <!ENTITY % attvals.idtype                    "
        ( current-user | original-user | target-user | user-privs |
          current-group | group-privs | other-privs )
      ">

    <!--
| Impact.type 属性值。
     -->
    <!ENTITY % attvals.impacttype                "
        ( admin | dos | file | recon | user | other )
      ">

    <!--
| Linkage.category 属性值。
     -->
    <!ENTITY % attvals.linkcat                   "
        ( hard-link | mount-point | reparse-point | shortcut | stream |
          symbolic-link )
      ">

    <!--
| Checksum.algorithm 属性值
     -->
    <!ENTITY % attvals.checksumalgos             "
          ( MD4 | MD5 | SHA1 | SHA2-256 | SHA2-384 | SHA2-512 | CRC-32 |
            Haval | Tiger | Gost )
      ">

    <!--
| Node.category 属性值。
     -->
    <!ENTITY % attvals.nodecat                   "
```

```
            ( unknown | ads | afs | coda | dfs | dns | hosts | kerberos |
              nds | nis | nisplus | nt | wfw )
          ">

    <!--
| Reference.origin 属性值
     -->
    <!ENTITY % attvals.origin                    "
            ( unknown | vendor-specific | user-specific | bugtraqid | cve |
              osvdb )
          ">

    <!--
| Confidence.rating 属性值。
     -->
    <!ENTITY % attvals.rating                    "
            ( low | medium | high | numeric )
          ">

    <!--
| Impact.severity 属性值。
     -->
    <!ENTITY % attvals.severity                  "
            ( info | low | medium | high )
          ">

    <!--
| User.category 属性值。
     -->
    <!ENTITY % attvals.usercat                   "
            ( unknown | application | os-device )
          ">
```

```
        <!--
    | yes/no 属性的值，如 Source.spoofed 和
    | Target.decoy 属性。
         -->
        <!ENTITY % attvals.yesno                    "
            ( unknown | yes | no )
          ">

        <!--
==================================================================

==================================================================== 第 3
节. 顶级元素声明。IDMEF-Message 元素
    ===              与它可包含的消息类型。
        ================================================================
        ================================================================ -
->

        <!ELEMENT IDMEF-Message                  (
            (Alert | Heartbeat)*
          )>
        <!ATTLIST IDMEF-Message
            %attlist.global;
            %attlist.idmef;
          >

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

```
            messageid           CDATA                   '0'
            %attlist.global;
          >

        <!ELEMENT Heartbeat                 (
            Analyzer, CreateTime, HeartbeatInterval?, AnalyzerTime?,
            AdditionalData*
          )>
        <!ATTLIST Heartbeat
            messageid           CDATA                   '0'
            %attlist.global;
          >

        <!--
================================================================

================================================================ 第 4
节. Alert 元素子类，为特定告警类型
    ===             提供更多的数据。
        ================================================================
        ================================================================ -
->

        <!ELEMENT CorrelationAlert              (
            name, alertident+
          )>
        <!ATTLIST CorrelationAlert
            %attlist.global;
          >

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

```
<!ATTLIST OverflowAlert
    %attlist.global;
  >

<!ELEMENT ToolAlert                    (
    name, command?, alertident+
  )>
<!ATTLIST ToolAlert
    %attlist.global;
  >

<!--
=============================================================

============================================================== 第 5 节. AdditionalData 元素。该元素允许
  ===                  告警包含额外信息，这些信息无法
  ===                  在数据模型的其他地方编码。
      ============================================================
      ============================================================ -->

<!ELEMENT AdditionalData              (
  (boolean | byte          | character | date-time |
   integer | ntpstamp      | portlist  | real      |
   string  | byte-string | xmltext    )
 )>

<!ATTLIST AdditionalData
    type                %attvals.adtype;        'string'
    meaning             CDATA                   #IMPLIED
    %attlist.global;
```

```
            >


        <!--
================================================================


================================================================ 第 6
节. 与识别实体相关的元素–分析器
    ===          （这些消息的发送者）、（攻击）
    ===          源与（攻击）目标
        ================================================================
        ================================================================ -
->


        <!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
            %attlist.global;
          >


        <!ELEMENT Classification            (
            Reference*
          )>
        <!ATTLIST Classification
            ident               CDATA               '0'
```

```
        text                CDATA                #REQUIRED
    >

<!ELEMENT Source                (
        Node?, User?, Process?, Service?
    )>
<!ATTLIST Source
        ident               CDATA                '0'
        spoofed             %attvals.yesno;      'unknown'
        interface           CDATA                #IMPLIED
        %attlist.global;
    >

<!ELEMENT Target                (
        Node?, User?, Process?, Service?, File*
    )>
<!ATTLIST Target
        ident               CDATA                '0'
        decoy               %attvals.yesno;      'unknown'
        interface           CDATA                #IMPLIED
        %attlist.global;
    >

<!ELEMENT Assessment            (
        Impact?, Action*, Confidence?
    )>
<!ATTLIST Assessment
        %attlist.global;
    >

<!--
================================================================
```

```
============================================================== 第 7 节. 用于提供实体详细信息的
    ===             辅助元素–地址、名称等
        ============================================================
        ============================================================ -->

        <!ELEMENT Reference                 (
            name, url
          )>
        <!ATTLIST Reference
            origin              %attvals.origin;        'unknown'
            meaning             CDATA                   #IMPLIED
          >

        <!ELEMENT Node                      (
            location?, (name | Address), Address*
          )>
        <!ATTLIST Node
            ident               CDATA                   '0'
            category            %attvals.nodecat;       'unknown'
            %attlist.global;
          >
        <!ELEMENT Address                   (
            address, netmask?
          )>
        <!ATTLIST Address
            ident               CDATA                   '0'
            category            %attvals.addrcat;       'unknown'
            vlan-name           CDATA                   #IMPLIED
            vlan-num            CDATA                   #IMPLIED
            %attlist.global;
```

```
    >

<!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
        %attlist.global;
    >

<!ELEMENT Permission EMPTY >
<!ATTLIST Permission
        perms                   %attvals.fileperm;      #REQUIRED
        %attlist.global;
>

<!ELEMENT FileAccess                        (
        UserId, Permission+
    )>
<!ATTLIST FileAccess
        %attlist.global;
    >

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

```
<!ATTLIST Inode
    %attlist.global;
  >

<!ELEMENT Linkage                    (
    (name, path) | File
  )>
<!ATTLIST Linkage
    category          %attvals.linkcat;          #REQUIRED
    %attlist.global;
  >

<!ELEMENT Checksum                    (
    value, key?
  )>
<!ATTLIST Checksum
    algorithm         %attvals.checksumalgos; #REQUIRED
    %attlist.global;
  >

<!ELEMENT Process                    (
    name, pid?, path?, arg*, env*
  )>
<!ATTLIST Process
    ident             CDATA                  '0'
    %attlist.global;
  >

<!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
      %attlist.global;
   >

<!ELEMENT SNMPService                              (
      oid?, messageProcessingModel?, securityModel?, securityName?,
      securityLevel?, contextName?, contextEngineID?, command?
   )>
<!ATTLIST SNMPService
      %attlist.global;
   >

<!ELEMENT User                                     (
      UserId+
   )>
<!ATTLIST User
      ident                  CDATA                  '0'
      category               %attvals.usercat;      'unknown'
      %attlist.global;
   >

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

```
        %attlist.global;
      >

    <!ELEMENT WebService                    (
        url, cgi?, http-method?, arg*
      )>
    <!ATTLIST WebService
        %attlist.global;
      >

    <!--
============================================================

============================================================ 第 8
节. 具有子元素或特殊属性的
    ===          简单元素。

============================================================
      ============================================================ -
->

    <!ELEMENT Action            (#PCDATA) >
    <!ATTLIST Action
        category          %attvals.actioncat;    'other'
        %attlist.global;
      >

    <!ELEMENT CreateTime        (#PCDATA) >
    <!ATTLIST CreateTime
        ntpstamp          CDATA                  #REQUIRED
        %attlist.global;
      >

    <!ELEMENT DetectTime        (#PCDATA) >
```

```
<!ATTLIST DetectTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;

  >

<!ELEMENT AnalyzerTime          (#PCDATA) >
<!ATTLIST AnalyzerTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;
  >

<!ELEMENT Confidence            (#PCDATA) >
<!ATTLIST Confidence
    rating              %attvals.rating;        'numeric'
    %attlist.global;
  >

<!ELEMENT Impact                (#PCDATA) >
<!ATTLIST Impact
    severity            %attvals.severity;      #IMPLIED
    completion          %attvals.completion;    #IMPLIED
    type                %attvals.impacttype;    'other'
    %attlist.global;
  >

<!ELEMENT alertident            (#PCDATA) >
<!ATTLIST alertident
    analyzerid          CDATA                   #IMPLIED
    %attlist.global;
  >
```

```
        <!--
================================================================

================================================================ 第 9
节. 不具有子元素及特殊属性的
    === 简单元素。
        ================================================================
        ============================================================ -->

        <!ELEMENT boolean          (#PCDATA)          >
        <!ATTLIST boolean          %attlist.global;   >

        <!ELEMENT byte             (#PCDATA)          >
        <!ATTLIST byte             %attlist.global;   >

        <!ELEMENT character        (#PCDATA)          >
        <!ATTLIST character        %attlist.global;   >

        <!ELEMENT date-time        (#PCDATA)          >
        <!ATTLIST date-time        %attlist.global;   >

        <!ELEMENT integer          (#PCDATA)          >
        <!ATTLIST integer          %attlist.global;   >

        <!ELEMENT ntpstamp         (#PCDATA)          >
        <!ATTLIST ntpstamp         %attlist.global;   >
        <!ELEMENT real             (#PCDATA)          >
        <!ATTLIST real             %attlist.global;   >

        <!ELEMENT string           (#PCDATA)          >
        <!ATTLIST string           %attlist.global;   >

        <!ELEMENT byte-string      (#PCDATA)          >
```

```
<!ATTLIST byte-string          %attlist.global;   >

<!ELEMENT xmltext              ANY                >
<!ATTLIST xmltext              %attlist.global;   >

<!ELEMENT access-time          (#PCDATA)          >
<!ATTLIST access-time          %attlist.global;   >

<!ELEMENT address              (#PCDATA)          >
<!ATTLIST address              %attlist.global;   >

<!ELEMENT arg                  (#PCDATA)          >
<!ATTLIST arg                  %attlist.global;   >

<!ELEMENT buffer               (#PCDATA)          >
<!ATTLIST buffer               %attlist.global;   >

<!ELEMENT c-major-device       (#PCDATA)          >
<!ATTLIST c-major-device       %attlist.global;   >

<!ELEMENT c-minor-device       (#PCDATA)          >
<!ATTLIST c-minor-device       %attlist.global;   >

<!ELEMENT cgi                  (#PCDATA)          >
<!ATTLIST cgi                  %attlist.global;   >

<!ELEMENT change-time          (#PCDATA)          >
<!ATTLIST change-time          %attlist.global;   >

<!ELEMENT command              (#PCDATA)          >
<!ATTLIST command              %attlist.global;   >
```

```
<!ELEMENT create-time (#PCDATA) >
<!ATTLIST create-time %attlist.global; >

<!ELEMENT data-size (#PCDATA) >
<!ATTLIST data-size %attlist.global; >

<!ELEMENT disk-size (#PCDATA) >
<!ATTLIST disk-size %attlist.global; >
<!ELEMENT env (#PCDATA) >
<!ATTLIST env %attlist.global; >

<!ELEMENT http-method (#PCDATA) >
<!ATTLIST http-method %attlist.global; >

<!ELEMENT location (#PCDATA) >
<!ATTLIST location %attlist.global; >

<!ELEMENT major-device (#PCDATA) >
<!ATTLIST major-device %attlist.global; >

<!ELEMENT minor-device (#PCDATA) >
<!ATTLIST minor-device %attlist.global; >

<!ELEMENT modify-time (#PCDATA) >
<!ATTLIST modify-time %attlist.global; >

<!ELEMENT name (#PCDATA) >
<!ATTLIST name %attlist.global; >

<!ELEMENT netmask (#PCDATA) >
<!ATTLIST netmask %attlist.global; >
```

```
<!ELEMENT number (#PCDATA) >
<!ATTLIST number %attlist.global; >

<!ELEMENT oid (#PCDATA) >
<!ATTLIST oid %attlist.global; >

<!ELEMENT path (#PCDATA) >
<!ATTLIST path %attlist.global; >

<!ELEMENT permission (#PCDATA) >
<!ATTLIST permission %attlist.global; >

<!ELEMENT pid (#PCDATA) >
<!ATTLIST pid %attlist.global; >

<!ELEMENT port (#PCDATA) >
<!ATTLIST port %attlist.global; >

<!ELEMENT portlist (#PCDATA) >
<!ATTLIST portlist %attlist.global; >

<!ELEMENT program (#PCDATA) >
<!ATTLIST program %attlist.global; >
<!ELEMENT protocol (#PCDATA) >
<!ATTLIST protocol %attlist.global; >

<!ELEMENT size (#PCDATA) >
<!ATTLIST size %attlist.global; >

<!ELEMENT url (#PCDATA) >
<!ATTLIST url %attlist.global; >
```

```
<!ELEMENT HeartbeatInterval (#PCDATA)          >
<!ATTLIST HeartbeatInterval %attlist.global;   >

<!ELEMENT messageProcessingModel (#PCDATA)     >
<!ATTLIST messageProcessingModel %attlist.global;>

<!ELEMENT securityModel       (#PCDATA)        >
<!ATTLIST securityModel       %attlist.global; >

<!ELEMENT securityName        (#PCDATA)        >
<!ATTLIST securityName        %attlist.global; >

<!ELEMENT securityLevel       (#PCDATA)        >
<!ATTLIST securityLevel       %attlist.global; >

<!ELEMENT contextName         (#PCDATA)        >
<!ATTLIST contextName         %attlist.global; >

<!ELEMENT contextEngineID     (#PCDATA)        >
<!ATTLIST contextEngineID     %attlist.global; >

<!ELEMENT value               (#PCDATA)        >
<!ATTLIST value               %attlist.global; >

<!ELEMENT key                 (#PCDATA)        >
<!ATTLIST key                 %attlist.global; >

<!-- End of IDMEF DTD -->
```

9. 安全考虑

本文对入侵检测系统实现之间传输安全相关信息规定了数据表达方法，虽然没有与数据格式直接相关的安全问题，数据本身可能包含安全敏感信息，这些信息的保密性、完整性与/或可用性可能需要保护。

这意味着应保护用于收集、传输、处理与存储此类数据的系统，防止其被非法使用，还应防止非法访问这些数据。此类防护不在本文讨论之列。

分析器传输 IDMEF 数据给管理器需要使用传输协议，RFC 4766 第 5 节介绍了传输协议的必要与建议安全特征。这些要求包括消息保密性、消息完整性、抗抵赖性及避免复制消息。当前既有标准也有建议协议提供这些特性。

若协议不满足 RFC 4766 第 5 节要求，却用于交换 IDMEF 消息，最好使用数字签名保证这些消息的完整性。详细信息见本文档 6.5 节。

10. IANA 考虑

第 5 节阐述了如何在 IDMEF 消息中使用 AdditionalData 类提供任意“原子”数据项以及如何通过使用 AdditionalData 新增类与属性来扩展 DTD。

有时，可能希望将某一扩展的私有或本地使用状态（通过上述机制新建的扩展）改为“标准”状态，以获得所有实现的支持。

实现方式如本节所述。

10.1 为现有属性新增值

本文规定的属性中有一些含有可能的允许值列表。为支持在此类列表中添加新值，IANA 创建了属性值库，称为“入侵检测消息交换格式（IDMEF）属性值”。

根据 RFC 2434 中的指导性要求，该库为 RFC“要求的规范”。10.1.1 节中列举了属性值库的初始值。

新建属性时，必须发布 RFC 记录类型。在 RFC 中，加入本文 10.1.2 节中的注册模板，放到“IANA 考虑”一节，填写相应字段。必须在文档中描述互通与安全问题。

在库中新增属性值时，IANA 须为值按数字顺序依次分配序号。

10.1.1. 属性注册

IDMEF 类名：引用（Reference）

IDMEF 属性名：来源（origin）

注册值：

序号	关键字	说明
0	unknown	名称来源未知
1	vendor-specific	与特定厂商相关的名称（URL）；用于提供与具体产品相关的信息
2	user-specific	与特定用户相关的名称（URL）；用于提供与具体安装相关的信息
3	bugtraqid	SecurityFocus （“Bugtraq”）漏洞库识别符 (http://www.securityfocus.com/bid)
4	cve	通用漏洞披露

		（CVE）名（http://cve.mitre.org/）
5	osvdb	开源漏洞数据库 (http://www.osvdb.org)

IDMEF 类名：源（Source）

IDMEF 属性名：伪造（spoofed）

注册值：

序号	关键字	说明
0	unknown	源信息准确性未知
1	yes	源地址为诱骗地址
2	no	源地址为真实地址

IDMEF 类名：目标（Target）

IDMEF 属性名：诱骗（decoy）

注册值：

序号	关键字	说明
0	unknown	目标信息准确性未知
1	yes	目标地址为诱骗地址
2	no	目标地址为真实地址

IDMEF 类名： AdditionalData

IDMEF 属性名：类型（type）

注册值：

序号	关键字	说明

0	boolean	本元素包含一个布尔值，即 true 或 false 字符串
1	byte	元素内容为一个 8 位单字节（见 3.2.4 节）
2	character	元素内容为单个字符（见 3.2.3 节）
3	date-time	元素内容为 date-time 字符串（见 3.2.6 节）
4	integer	元素内容为整型量（见 3.2.1 节）
5	ntpstamp	元素内容为 NTP 时间戳（见 3.2.7 节）
6	portlist	元素内容为端口列表（见 3.2.8 节）
7	real	元素内容为实数（见 3.2.2 节）
8	string	元素内容为字符串（见 3.2.3 节）
9	byte-string	元素内容为字节[]（见 3.2.4 节）
10	xmltext	元素内容为 XML 标记数据（见 5.2 节）

IDMEF 类名：影响（Impact）

IDMEF 属性名：告警级别（severity）

注册值：

序号	关键字	说明
0	info	通知有活动进行

1	low	低级
2	medium	中级
3	high	高级

IDMEF 类名：影响（Impact）

IDMEF 属性名：完成（completion）

注册值：

序号	关键字	说明
0	failed	攻击未成功
1	succeeded	攻击成功

IDMEF 类名：影响（Impact）

IDMEF 属性名：类型（type）

注册值：

序号	关键字	说明
0	admin	试图或成功获取管理权限
1	dos	试图或成功完成 DDoS 攻击
2	file	试图或成功进行文件操作
3	recon	试图或成功完成侦测
4	user	试图或成功获取用户权限
5	other	除以上类型之外的其他类型

IDMEF 类名：动作（Action）

IDMEF 属性名：类别（category）

注册值：

+------+------------------+-------------------------------------+

| 序号 | 关键字 | 说明 |

+------+------------------+-------------------------------------+

| 0 | block-installed | 安装了某种阻断，|

| | | 阻止攻击到达 |

| | | 目的地。阻断可能针对

| | | 端口、地址等，或 |

| | | 禁用用户账号。 |

| 1 | notification-sent | 对带外发送了某种类型的 |

| | | 通知消息（通过传呼、 |

| | | email 等），不包括 |

| | | 本告警的传输。 |

| 2 | taken-offline | 系统、计算机或用户 |

| | | 被迫下线，因为计算机被 |

| | | 关闭或用户已登出。 |

| 3 | other | 除以上类型之外的其他类型 |

+------+------------------+-------------------------------------+

IDMEF 类名：置信度（Confidence）

IDMEF 属性名：等级（rating）

注册值：

+------+---------+---+

| 序号 | 关键字 | 说明 |

+------+---------+---+

| 0 | low | 分析器对其有效性几乎没有信心 |

| | | validity |

| 1 | medium | 分析器对其有效性有一定的信心 |

| | | validity |

| 2 | high | 分析器对其有效性很有信心

3	numeric	分析器提供后验
		概率值，表示对其有效性的信心。

IDMEF 类名：节点（Node）

IDMEF 属性名：类别（category）

注册值：

序号	关键字	说明
0	unknown	域未知或不相关
1	ads	Windows 2000 高级目录服务
2	afs	Andrew 文件系统（Transarc）
3	coda	Coda 分布式文件系统
4	dfs	分布式文件系统（IBM）
5	dns	域名系统
6	hosts	本地 hosts 文件
7	kerberos	Kerberos 域
8	nds	Novell 目录服务
9	nis	网络信息服务（Sun）
10	nisplus	增强网络信息服务（Sun）
11	nt	Windows NT 域
12	wfw	Windows for Workgroups

IDMEF 类名：地址（Address）

IDMEF 属性名：类别（category）

注册值：

序号	关键字	说明
0	unknown	地址类型未知

1	atm	异步传输模式网络地址
2	e-mail	电子邮件地址（(RFC 822）
3	lotus-notes	Lotus Notes 电子邮件地址
4	mac	媒体访问控制（MAC）地址
5	sna	IBM 共享网络架构（SNA）地址
		address
6	vm	IBM VM ("PROFS")电子邮件地址
7	ipv4-addr	点分十进制表示的 IPv4 主机地址
		（a.b.c.d）
8	ipv4-addr-hex	十六进制表示的 IPv4 主机地址
9	ipv4-net	点分十进制表示的 IPv4 网络地址
		数字、斜线、有效位数
		（a.b.c.d/nn）
10	ipv4-net-mask	点分十进制表示的 IPv4 网络地址
		数字、斜线、点分十进制表示的
		网络掩码 （a.b.c.d/w.x.y.z）
11	ipv6-addr	IPv6 主机地址
12	ipv6-addr-hex	十六进制表示的 IPv6 主机地址
13	ipv6-net	IPv6 网络地址、斜线、有效位数
14	ipv6-net-mask	IPv6 网络地址、斜线、网络掩码

IDMEF 类名：用户（User）

IDMEF 属性名：类别（category）

注册值：

序号	关键字	说明
0	unknown	用户类型未知
1	application	应用用户
2	os-device	操作系统或设备用户

IDMEF 类名：用户 ID（UserId）

IDMEF 属性名：类别（category）

注册值：

序号	关键字	说明
0	current-user	用户或进程当前使用的用户 ID。在 UNIX 系统中，一般为“真实的”用户 ID。
1	original-user	当前上报用户或进程的真实身份。在进行审计和支持从“audit id”令牌中提取用户 ID 的系统中，应使用该值。若系统不支持此种操作且用户已登录系统，则应使用“login id”。
2	target-user	用户或进程试图获取的用户身份。以 UNIX 系统为例，当用户在系统中试图使用 su、rlogin、telnet 等 ID 时，适用该值。
3	user-privs	用户或进程可使用的另一个用户 ID 或与文件权限关联的用户 ID。在 UNIX 系统中，对用户或进程而言，为“有效的”用户 ID；对文件而言，为所有者权限。可用多个此类 UserId 元素指定多项权限。
4	current-group	用户或进程当前使用的组 ID（若适用）。在 UNIX 系统中，一般指“真实的”组 ID。
5	group-privs	组或进程可使用的另一个组 ID 或与文件权限关联的组 ID。在 UNIX 系统中，对组或进程而言，为“有效的”组 ID；对文件而言，为组权限。在 BSD 系列的 UNIX 系统中，使用多个此类 UserId 元素覆盖“组列表”的所有组 ID。
6	other-privs	对用户、组或进程不适用，仅适用于文件。

			用户不匹配文件的用户或组权限时，
			分配给用户这种文件权限。在 UNIX 系统中，
			指“world”权限。

+------+--------------+---+

IDMEF 类名：文件（File）

IDMEF 属性名：类别（category）

注册值：

+------+----------+---+

| 序号 | 关键字 | 说明 |

+------+----------+---+

| 0 | current | 上报变化后获取的文件信息 |

| 1 | original | 上报变化前获取的文件信息 |

IDMEF 类名：文件（File）

IDMEF 属性名：fstype

注册值：

+------+---------+-----------------------------------+

| 序号 | 关键字 | 说明 |

+------+---------+-----------------------------------+

| 0 | ufs | 伯克利 UNIX 快速文件系统 |

| 1 | efs | Linux“efs”文件系统 |

| 2 | nfs | 网络文件系统 |

| 3 | afs | Andrew 文件系统 |

| 4 | ntfs | Windows NT 文件系统 |

| 5 | fat16 | 16 位 Windows FAT 文件系统 |

| 6 | fat32 | 32 位 Windows FAT 文件系统 |

| 7 | pcfs | CD-ROM 上使用的 PC（MS-DOS）文件系统 |

| 8 | joliet | Joliet CD-ROM 文件系统 |

| 9 | iso9660 | ISO 9660 CD-ROM 文件系统 |

+------+---------+-----------------------------------+

IDMEF 类名： FileAccess

IDMEF 属性名：权限（permission）

注册值：

序号	关键字	说明
0	noAccess	该用户不具有任何访问权限
1	read	该用户具有文件读权限
2	write	该用户具有文件写权限
3	execute	该用户有文件执行权限
4	search	该用户可搜索本文件（适用于 UNIX 目录的“执行”权限）
5	delete	该用户可删除本文件
6	executeAs	该用户可以其他用户身份执行本文件
7	changePermissions	该用户可修改对本文件的访问权限
8	takeOwnership	该用户可获取本文件的所有权

IDMEF 类名：关联（Linkage）

IDMEF 属性名：类别（category）

注册值：

序号	关键字	说明
0	hard-link	<name>元素表示本文件的另一个名称。与其他系统相比，该信息在 NTFS 系统中更易获取。
1	mount-point	父目录的<name>和<path>元素所指定的目录的别名。
2	reparse-point	仅适用于 Windows 系统：不包括符号链接与挂载点，后者是特定类型的重解析点。
3	shortcut	Windows 的“快捷方式（Shortcut）”指示的文件。快捷方式与符号链接不同，

		快捷方式的内容对管理器可能会很重要。
4	stream	Windows 中的备用数据流（ADS）；
		MacOS 中的 Fork。它作为单独的文件
		系统实体，被视为主<File>的扩展。
5	symbolic-link	<name>元素表示链接指向的文件。

IDMEF 类名：校验和（Checksum）

IDMEF 属性名：算法（algorithm）

注册值：

序号	关键字	说明
0	MD4	MD4 算法
1	MD5	MD5 算法
2	SHA1	SHA1 算法
3	SHA2-256	256 位摘要长度的 SHA2 算法
4	SHA2-384	384 位摘要长度的 SHA2 算法
5	SHA2-512	512 位摘要长度的 SHA2 算法
6	CRC-32	32 位校验长度的 CRC 算法
7	Haval	Haval 算法
8	Tiger	Tiger 算法
9	Gost	Gost 算法

10.1.2. 注册模板

IDMEF 类名：

<提供类名，所含属性欲新增值的话，还应包含该属性，如“Address”>

IDMEF 属性名：

<欲新增值的属性的名称，如“category”>

欲定义的新属性值：

<提供欲新增的属性值名称，如“sneaker-net”>

新属性值含义：

<详细解释属性值含义，也就是说，若分析器发送了该值，它想传达给接收者什么信息>

联系人与电子邮件地址：

<申请人姓名与电子邮件地址>

10.2 新增属性与类

本文所定义的 IDMEF 类与属性在设计上尽可能地考虑到当前与近期需求。未来需要新增类或为现有类新增属性，但此类操作应谨慎。

只有当现有类与属性无法准确表达信息时，才可以新增属性或类。另外，这种新增的属性或类应仅适用于通用数据类型。对于与特定厂商相关的信息、特定类型的分析器所提供或特定类型的管理器所使用的模糊信息、“pet”属性等，不宜新增类和属性。

撰写本 RFC 时，预期需要新增类和属性的第一种情况是处理基于主机的入侵检测系统。不过，在做此类新增前，须对该类系统将要提供的数据集达成一致。

根据 RFC 2434 所提出的要求，在 IDMEF 中新增类与属性的前提是达成“IETF 共识”。

新增属性或类时，必须发布相应 RFC，获得 IESG 批准。一般情况下，IESG 会从有关人士（如相关工作组（若有））处获取欲新增类或属性的信息。申请者必须在文档中描述互通与安全问题。

入侵检测消息交换格式（IDMEF）

安全加社区

公益译文项目

2017

网络安全公益译文项目旨在分享国外先进网络安全理念，将网络安全战略性文档翻译为中文，促进国内安全组织在相关方面的思考和交流。该项目由安全加社区发起，安全加社区是国内的网络安全社区，社区欢迎网络安全人士的加入，并致力于交付网络安全问题的解决能力。