Intrusion Detection Working Group G. Mansfield/D. Curry
draft-ietf-idwg-xmlsmi-01.txt Cyber Solutions/ISS
Expires: February 21, 2001 August 22, 2000

Intrusion Detection Message Exchange Format
Comparison of SMI and XML Implementations

Status of this Memo

Abstract

The purpose of the Intrusion Detection Message Exchange Format (IDMEF) is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to the management systems which may need to interact with them. The goals and requirements of the IDMEF are described in [2] and [3].

Two implementations of the IDMEF data format have been proposed: one using the Structure of Management Information (SMI) to describe a MIB, and the other using a Document Type Definition (DTD) to describe XML documents. Both representations appear to have their good and bad traits, and deciding between them is difficult.

To arrive at an informed decision, the working group tasked the authors to identify and analyze the pros and cons of both approaches, and to present the results in the form of an Internet-Draft.

The initial version of this draft was reviewed by the IDWG at the February, 2000 interim meeting where it was tentatively decided that the XML/DTD solution was best at fulfilling the IDWG requirements. This decision was finalized at the March, 2000 IETF IDWG meeting.

TABLE OF CONTENTS

1. Methods for communicating intrusion detection alert data

The requirements document, [3], having been forwarded to the IESG for consideration as an Informational RFC, defines the intrusion detection model that we are assuming: One or more sensors monitor some number of data sources for signs of intrusions, and report their observations to one or more analyzers. When an analyzer determines somehow that these observations represent a suspicious event, it sends an alert to one or more managers. A manager may respond to an alert by notifying an operator, investigating further by exchanging data with its peers, querying the source of the alert, or communicating the event to a higher level manager.

The format of the alert sent by the analyzer to the manager, and the method of communicating it, are what the IDMEF proposes to standardize [2].

In discussions within the working group, two different modes of operation have been suggested for communicating alert data between analyzers and managers. It should be noted that the IDMEF concerns itself only with the format of the alert, and not the design of the system that delivers them, or the protocols used to do so. However, it is important to have an idea of the communications mode(s) that will be used by intrusion detection systems when choosing an intrusion detection alert format, because some formats may support certain modes better than others.

1.1 Tell-only

This mode provides for a unidirectional communications flow, from an analyzer to one or more managers, as shown in Figure 1. (Managers may also pass the alert to other managers, in a hierarchical arrangement.)

```
                                       +-----------------------+
                                      /                        V
      +--------+         +----------+ /      +---------+  +---------+
      |        | report  |          |/ alert |         |  |         |
      | Sensor |-------->| Analyzer |------->| Manager |  | Manager |
      |        |         |          |        |         |  |         |
      +--------+         +----------+        +---------+  +---------+
   IDS #1                                         |
   ...............................................|..................
   IDS #2                                         | alert
                                                  V
                                             +---------+
                                             |         |
                                             | Manager |
                                             |         |
                                             +---------+
```

Figure 1. Tell-only mode.

When an analyzer detects an event (or some sequence of events) that must be communicated to a manager, it uses an alert message to do this. The analyzer places the available information (usually all of it, but sometimes not) about the event(s) into the alert, and sends it to the manager. Once the alert has been sent to the manager, the analyzer generally "forgets" about both the alert and the events that led up to it.

The principal advantage to this mode is that analyzers can be kept both simple and small, allowing them to be deployed with less impact on performance, and using smaller and less expensive hardware. The principal disadvantage is that a manager is "stuck" with whatever the analyzer sends it -- this may be not enough information, or it may be too much. The latter case can be particularly problematic; the possibility exists for a poorly configured analyzer to inundate a manager with messages the manager has no use for, but cannot ignore.

The IDMEF requirements document requires a tell-only mode for the communication of IDMEF messages (Section 3.1, paragraph 4). Most intrusion detection products on the market today implement the tell-only model.

1.2 Tell-and-ask

This mode provides for bidirectional communication -- from an analyzer to one or more managers and also from the managers to the analyzers, for alert data exchange, as shown in Figure 2.

```
                                      +-------------------------+
                                     /                          V
      +--------+       +----------+ /          +---------+  +---------+
      |        |report |          |/  alert    |         |  |         |
      | Sensor |------>| Analyzer |----------->| Manager |  | Manager |
      |        |       |          |            |         |  |         |
      |        |       |          |   query    |         |  |         |
      |        |       |          |<-----------|         |  |         |
      |        |       |          |  response  |         |  |         |
      |        |       |          |----------->|         |  |         |
      |        |       |          |            |         |  |         |
      +--------+       +----------+            +---------+  +---------+
   IDS #1                                           |
   .................................................|..................
   IDS #2                                           | alert
                                                    V
                                               +---------+
                                               |         |
                                               | Manager |
                                               |         |
                                               +---------+
```

Figure 2. Tell-and-ask mode.

As with the tell-only mode, when an analyzer detects an event (or some sequence of events) that must be communicated to a manager, it uses an alert message to do this. However, instead of sending all the available event information along with the alert, the analyzer may choose to send only certain data about the alert (type, time, priority, etc.). The decision about which information is sent in the initial alert, and which is not, will depend on the local circumstances and configuration.

If the manager receiving the alert is "interested" in the events that caused the alert to be generated, it can (either automatically or under the control of an operator) contact the analyzer and request additional information. The manager might ask for:

- more information on the alert, e.g., pointers provided as URLs in the alerts from the agents (using appropriate protocols such as HTTP, FTP, etc.);

- more host-related information on the circumstances under which the intrusion/attack was detected -- this may involve fetching further information from the various databases (e.g., MIBs) of the entity that originated the notification; or

- more network-related information on the circumstances under which the intrusion/attack was detected -- this may involve fetching further information from the various databases (e.g., MIBs) of the relevant network entities in the network.

The principal advantage of this mode of operation is its flexibility. Alerts can be made smaller. This approach addresses the inundation problem described above at the expense of additional network traffic and analyzer and manager complexity. Moreover, since the query facility is available the analyzers may choose to retain some or all relevant information, e.g., analysis-logs, packet-dumps, etc. for some period of time. It may also provide any information it has and in which the manager is interested. The principal disadvantage of this mode is that analyzers will be relatively more complex compared to those operating in the tell-only mode. They must store historical data for some "reasonable" period of time, either in memory or on disk, and be able to retrieve that data when asked, all the while still performing their primary function, detection of intrusions.

The IDMEF requirements document indicates that the IDMEF data format may support the tell-and-ask communications mode (Section 3.1, paragraph 4).

A Remote Monitoring (RMON) device, a fairly popular network managent agent, can be configured to function as an IDS (with limited functionality) and is an example of an IDS that operates in the tell-and-ask mode.

2. Overview of proposed implementations

Two implementations of the IDMEF data format have been proposed: one using the Structure of Management Information (SMI) to describe a MIB, and the other using a Document Type Definition (DTD) to describe XML documents. The implementations are presented briefly in this section; for more detail on either implementation consult the relevant Internet-Drafts [4, 5].

2.1 SMI

Network management involves monitoring and manipulating information about the elements to be managed such as hosts, routers, etc. These elements are monitored and controlled by accessing the related management information -- the Management Information Base (MIB), which is abstracted as a collection of Managed Objects (MO). A detailed introduction to the current SNMP Management Framework can be found in [7].

Collections of related managed objects are defined in MIB modules using the mechanisms defined in the SMI [9].

The SMI identifies the datatypes that can be used in the MIB and specifies how resources within the MIB can be represented and named. SMI uses an adapted subset of OSI's Abstract Syntax Notation One (ASN.1) [10]. The basic structure types provided allow representation of scalars and two dimensional arrays of scalars.

Every object in a MIB has an associated identifier of ASN.1 type OBJECT IDENTIFIER which serves as its name. The name space is a tree-structure rooted at the "iso" object which has the identifier 1 (it refers to the ASN.1 document itself). A row of a table is identified by the value(s) of the index(es) of the table.

Thus, the ifOperStatus object of type "operational status of an interface," in the interfaces table will have an object identifier that will look like

    1.3.6.1.2.2.1.8

    (iso.org.dod.internet.mgmt.interfaces.ifTable.ifEntry.ifOperStatus)

If the rows in the interface table are indexed by, say, the "ifIndex" object, and we are interested in the operational status of the interface which has an ifIndex = 2, then we would look for the value of the object named

    1.3.6.1.2.2.1.8.2

The definition of the ifOperStatus object in the MIB will also tell us, among other things, that the syntax object is INTEGER.

The SMI representation mechanism does not constrain applications to function in the tell-only mode or the tell-and-ask mode. The application may chose to use either or both modes. This is possible because of the granularity of the naming mechanism. In the tell-and-ask mode the analyzer may send small alert messages containing essential information to the manager. Subsequently, the analyzer will respond to queries from the manager for additional details on the alert.

The SMI-MIB implementation can also work in the tell-only mode, by sending larger messages (containing all relevant information) from the analyzer to the manager via alert messages.

2.2 XML

The Extensible Markup Language (XML) is a text markup syntax defined by the World Wide Web Consortium (W3C). It is gaining widespread attention as a language for representing and exchanging documents and data on the Internet. XML is currently being used in a variety of projects, including the Text Encoding Initiative, Microsoft Channel Definition Format, Wireless Application Protocol (WAP) Wireless Markup Language, Chemical Markup Language, Weather Observation Markup Format, Open Financial Exchange, OpenMLS (real estate), Mathematical Markup Language, Electronic Data Interchange (several projects), News Industry Text Format, and a variety of others. For a comprehensive list, of XML applications, see

http://www.oasis-open.org/cover/xml.html#xml-osd

XML is a metalanguage -- a language for describing other languages -- that enables an application (such as IDMEF) to define its own markup. XML allows the definition of a customized markup language specific to an application. This differs from HTML, for example, in which a fixed set of markup tags with preset meanings must be "adapted" to uses for which they were not intended. Both XML and HTML use tags (identifiers delimited by '<' and '>') and attributes (of the form "name='value'"). But where "<p>" always means "paragraph" in HTML, it may mean "paragraph," "person," "price," or have no meaning at all in an XML application.

Each XML application defines the tags and attributes it needs in an XML Document Type Definition (DTD). Tags are defined to identify the semantic elements of the marked-up data (e.g., paragraphs, tables, figures, section headings, footnotes, chapters, titles, etc.) The DTD also specifies how the semantic elements of the data relate to each other (e.g., a chapter may only have one title, sections may only occur inside chapters, second-level headings must follow a first-level heading, and so on).

An XML IDMEF DTD defines the tags and attributes needed to identify

the various elements of an intrusion detection alert, as set forth in the data model defined by Debar, Huang, and Donahoo [6]. A complete "document" (alert) might look like:

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

<!DOCTYPE IDMEF-Message PUBLIC "-//IETF//DTD RFCxxxx IDMEF v1.1//EN"
  "idmef-message.dtd">

<IDMEF-Message version="1.1">
  <Alert id="345097" confidence="100" impact="attempted-admin"
    method="knowledge">
    <Time offset="-5">
      <date>2000/03/09</date>
      <time>08:12:32.3</time>
    </Time>
    <Analyzer id="372">
      <Node id="987" category="dns">
        <name>fileserver.bigcompany.com</name>
      </Node>
      <Process id="956">
        <pid>8956</pid>
        <Arguments>
          <arg>monitor</arg><arg>-d</arg>
          <arg>-m</arg><arg>idmanager.bigcompany.com</arg>
          <arg>-l</arg><arg>/var/logs/idlog</arg>
        </Arguments>
      </Process>
    </Analyzer>
    <Name origin="bugtraqid">33</Name>
    <signature>loadmodule forking shell</signature>
    <Source id="5678901234">
      <User id="7890123456" category="os-device">
        <name>joe</name>
        <uid>13243</uid>
      </User>
      <Process id="6789012345">
        <name>loadmodule</name>
        <path>/usr/openwin/bin</path>
      </Process>
    </Source>
    <Target id="3456789012">
      <Node id="4567890123" category="dns">
        <name>fileserver.bigcompany.com</name>
      </Node>
    </Target>
  </Alert>
</IDMEF-Message>
```

The proposed XML implementation of the IDMEF is intended to operate in the tell-only mode, in which all information relevant to an alert is sent to the manager in a single message. With appropriate support

from the communications layer, the implementation could be extended to support the tell-and-ask mode.

3. Comparison criteria

The authors have identified a variety of criteria against which the two implementations should be evaluated. These are presented below together with descriptions of how each criterion is met by the individual implementations.

3.1 Representation issues

Representation issues are those factors of the implementations that involve how intrusion detection alerts are represented: naming of alerts, fields in alerts, and alert-related information.

3.1.1 Naming

Naming is important to enable querying for object values. Examples include:

- what is the complete list of hosts that were the target of a scan?

- how many TCP-SYNs were sent to host foo.bar.com from network aaa.bbb.ccc.ddd during the attack?

There are three characteristics of naming that are important to us: flexibility, granularity, and ease of use:

- How flexible is the naming scheme used by the implementation? For example, can objects in lists and tables be named even when the length/size of the list or table is not known?

- How granular is the naming scheme used by each implementation? Can individual objects in lists and tables be named? Can entire lists and tables be named?

- How easy is it to use the naming scheme in the above cases?

3.1.1.1 SMI

The naming offered by SMI is flexible to the extent that all objects of interest can be named. Rows in a table are indexed. Indices are comprised of the values of one or more objects. The number of rows in a table need not be fixed or even known before hand.

By using lexicographic ordering and a mechanism to address the (lexicographically) next object an application may discover the

objects that are serviced by an agent. This means that the application can also discover the indices of the rows in a table.

There is no restriction on granularity. Thus, it is straightforward to name an object which represents the nth bit of the mth packet of a certain protocol that passed through the ith interface of a router.

It is necessary that both managers and agents know the names, syntax and other relevant attributes of the corresponding objects. MIB definitions need to be published for that purpose.

3.1.1.2 XML

The XML IDMEF DTD specifies that each semantic element of an alert will be individually tagged or contained in an attribute. As shown in the example in the previous section, tags within an alert are organized in a more or less hierarchical structure. There are no limits to the depth or breadth of the hierarchy (other than those imposed by the IDMEF data model itself).

The Document Object Model (DOM), defined by the W3C, is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of XML documents. Through the DOM, an application that processes XML IDMEF messages can treat an XML document (IDMEF alert) as a "database," and extract individual elements from it based on tag or attribute name.

The DOM allows on-the-fly manipulation of individual XML IDMEF messages by an application. The DOM does not, however, replace relational or object-oriented database management systems. An application that has a need to process historical alert data (e.g., for correlation or analysis purposes) would presumably use an RDBMS or OODBMS for this purpose. The XML IDMEF DTD's requirement that each semantic element of an alert be tagged or contained in an attribute, and the forthcoming XML Schema standard will make the conversion between the XML IDMEF message format and the database record format simple.

3.1.2 Data model

Debar, Huang, and Donahoo [6] have proposed that intrusion detection events in the IDMEF be represented by a class hierarchy. The working group has adopted this model as the one that should be followed by any data format implementation.

In this section, we examine how well the implementations match the data model, and any significant differences between the implementation and the model.

3.1.2.1 SMI

An Alert-MIB can be designed to fit the data model neatly. A draft Alert-MIB has been published [4]. In the Alert-MIB the messages themselves are indexed by a unique message-id. The manager uses this message-id to obtain further information about the event that caused the message e.g., the destinations that were targeted by the attack and/or the packet-trace that contained the signature of the attack.

3.1.2.2 XML

The proposed XML DTD [5] implements the IDMEF data model easily. Although it does not support classing and inheritance, XML does force a more or less hierarchical structure on documents, which fits nicely with the class hierarchy.

The XML DTD provides three extension mechanisms that will allow users of the DTD to encode their own data within defined elements or add their own attributes and elements. This handles non-standard extensions, and also makes it easy to evaluate an extension before adopting it as a standard.

Standard extensions are made by releasing updated versions of the DTD. Provided the update only adds new attributes/elements, existing systems would not have to be modified, save for adding code to handle the new attributes/elements if so desired (new modifications could be ignored).

3.1.3 Data format

What data representation format or formats are used by the implementation? For example, is everything "ASCII text" or "binary"? In addition to impacting ease-of-use, data format may affect the overall performance of an implementation.

3.1.3.1 SMI

SMI just specifies the data model. It allows the ASN.1 basic data types: INTEGER, OCTETSTRING, NULL and OBJECTIDENTIFIER. Several Application specific datatypes are defined, too. "ASCII text" as well as binary data are easily represented.

SMI does not specify how the data will be encoded, stored and/or transported over the network. It will be left to the application designers to choose the appropriate encodings and transports depending on the application requirements.

3.1.3.2 XML

The XML standard specifies that all XML documents (and therefore all XML IDMEF messages) be encoded in either the UTF-8 or UTF-16 encodings of ISO 10646 (Unicode).

All data in an XML IDMEF message will be encoded as text; there will be no "binary" content. Numbers will be encoded as their formatted output equivalents (e.g., the number 123.45 might be represented by the six characters, '1,' '2,' '3,' '.,' '4,' and '5'). Note that it may be decided to use a more efficient encoding than decimal, for example, base 64.

XML is capable of representing non-printable "binary" data, although the representation is not very efficient. Any arbitrary value can be encoded as decimal one-byte quantites (e.g., "<") or hexadecimal two-byte quantities (e.g., "<").

3.2 Operational issues

Operational issues are those factors of system and network operation that will be impacted by use of the implementation: network bandwidth consumed, memory and processor resources used, etc.

3.2.1 Bits on the wire

Intrusion detection systems are capable of sending many alerts in a very short period of time. Each of these alerts consumes some bandwidth on the network; if the number of alerts is too high, the network could become swamped, impacting not only the delivery of alerts, but all other applications using the network as well.

The following sample alert is used in the discussion below:

```
idMesg.version                        = 1
idMesg.priority                       = 1
idMesg.confidence                     = 100
idMesg.severity                       = 100
idMesg.name                           = bugtraqid.33
idMesg.signature                      = loadmodule forking shell
idMesg.method                         = 1
idMesg.time.date                      = 1999/10/21
idMesg.time.time                      = 08:12:32
idMesg.analyzer.ident                 = 12345678
idMesg.target[0].host.name            = machine.domain.com
idMesg.target[0].host.address.type    = 11
idMesg.target[0].host.address.value   = 123.234.345.456
idMesg.source[0].process.name         = /usr/openwin/bin/loadmodule
idMesg.source[0].user.name            = joe
idMesg.source[0].user.uid             = 13243
```


3.2.1.1 SMI

Assuming that the ID MIB module will be a subtree under mib-2, it will have an identifier of 1.3.6.2.1.2.1.1.idMesg, where "idMesg" will be some number assigned by IANA. All identifiers of objects in the ID MIB will have the prefix 1.3.6.2.1.2.1.1.idMesg.

Assuming that the data will be transmitted as a list of (Object Name,Value) pairs the payload will comprise a minimum of 337 bytes.

If the BER encoding associated with ASN.1 is employed then the payload will be roughly 454 bytes.

The application level protocol may or may not have some form of compression but there is probably no straightforward way of saying whether the SMI payload is "more compressible" than the XML payload.

3.2.1.2 XML

The XML IDMEF encoding of the sample alert shown in Section 3.2.1, above, is shown in Section 2.2. When formatted as it would be sent over an XML IDMEF message channel (no newlines or indentation), the encoding of this alert required 1018 bytes.

XML, because of the open-tag/close-tag syntax, has a relatively high overhead percentage. For the example above, XML tagging makes up about 70% of the total message.

However, XML is also readily compressed. Using Lempel-Ziv coding, the example above compresses to 562 bytes, a savings of 45%. By sending multiple alerts in the same message, compression results can be improved still further; savings of 80-90% are easily achieved.

3.2.2 Load on the CPU

Encoding data on the analyzer for transmission to the manager will put additional load on the analyzer's processor. Likewise, decoding and parsing the data on the manager will put additional load on the manager's processor. Depending on the processing resources needed, this additional load may impact the ability of the analyzer/manager to perform its other tasks.

3.2.2.1 SMI

There is an associated load with encoding the data represented in the SMI format. The sender does the encoding and the receiver the decoding.

For example, if the BER encoding of ASN.1 is used the integer 389360048 is represented as

```
41 04 17 35 29 B0
```

Where the first byte (41) represents the datatype, the second byte (04) represents the length in bytes of the following contents, the remaining 4 bytes (17 35 29 B0) represent the contents the hexadecimal representation of 389360048.

Incidentally if a plain ASCII text representation was used, the string would just be represented by

```
63 68 69 63 66 60 60 64 68
```

The time taken to parse an average BER encoded SMI IDMEF message using a publicly available software package on a 300MHz CPU machine is roughly 60 microseconds.

3.2.2.2 XML

The load placed on an analyzer to generate XML IDMEF messages is minimal. The message itself is simply character string data, usually generated with a formatting function such as sprintf() from C/C++. Only basic control structures are needed to create the format of the message (e.g., "if value known then print else don't").

The load placed on a manager to process XML IDMEF messages is somewhat higher, since the manager must parse (and optionally validate) the XML document that represents an IDMEF message.

The following times to parse an "average" XML IDMEF message were measured on a 360 MHz UltraSPARC II with 128MB of memory:

Package	Language	Validation	Time
IBM XML4J 3.0 SAXCount	Java	On	5.52 ms
IBM XML4J 3.0 SAXCount	Java	Off	4.95 ms
IBM XML4C 2.3.1 SAXCount	C++	On	4.40 ms
IBM XML4C 2.3.1 SAXCount	C++	Off	4.31 ms

Timings for other processors, other parser implementations, and other IDMEF messages will of course vary.

3.3 Implementation issues

Implementation issues are those factors of the implementations that will impact vendors and authors of intrusion detection systems - i.e., how easy will it be to add support for IDMEF to existing and new intrusion detection systems?

3.3.1 Size of code

Adding support for IDMEF will require adding code to both the analyzer and the manager.

3.3.1.1 SMI

The parser of SMI-MIB objects is essentially an ASN.1 parser. With reference to a publicly available implementation the size of the parser written in C is Source code = 55Kbytes, object code = 75 Kbytes.

3.3.1.2 XML

Code size on the analyzer to generate XML IDMEF messages is so small as to be insignificant. 1-2 KB of string storage space (for the tag and attribute names), and a few hundred bytes of control structures.

Code size on the manager to parse the XML IDMEF message is also not large. The binary object or script file sizes for several freely available XML parsers is shown below:

Package	Language	Validating	Program Size
Expat 1.1 (size of EXE and DLL files)	C	No	164 KB
IBM XML4J 3.0 (includes multiple parsers and document object model support) (size of Java JAR file)	Java	Yes	865 KB
IBM XML4C 2.3.1 (size of text+data+stack as reported by UNIX "size" command)	C++	Yes	18 KB
TclXML 1.2 (size of scripts)	Tcl	No	58 KB
xmlproc 0.61 (size of scripts)	Python	Yes	156 KB
XP 0.5 (size of Java JAR file)	Java	No	166 KB

The manager will also need space for the Java, Tcl, or Python runtime

environments, if those are used, plus memory for the parser's data structures.

3.3.2 Availability of code

Availability of code addresses how easy it is for a vendor or author of an intrusion detection system to obtain code to implement the IDMEF.

3.3.2.1 SMI

Software packages that incorporate the code for handling SMI-MIBs are freely available for a large range of platforms. The reader is referred to http://wwwsnmp.cs.utwente.nl/software/ for further information.

3.3.2.2 XML

Software tools for processing XML documents are widely available, in both commercial and open source forms. A variety of tools and APIs for parsing and/or validating XML are available in Java, C, C++, Tcl, Perl, Python, and GNU Emacs Lisp. For a comprehensive list of both commercial and open source XML tools, see

http://www.oasis-open.org/cover/publicSW.html#xmlTools

3.4 End point issues

End point issues are those factors of the implementations that impact how alert data will be handled once it reaches the manager.

3.4.1 Data display aspects

Data display aspects are those features of the data format that impact how data can be displayed to users, on graphical displays as well as in printed documents.

3.4.1.1 SMI

The MIB representation does not aid or hinder display at the end point. Textual Conventions have been defined for the SMI wherein provisions are made for specifying "DISPLAY HINT". The "DISPLAY HINT" gives a hint as to how the value of an instance of an object with the syntax defined using the textual convention might be displayed.

There are numerous MIB browsers (Perl, TCL. JAVA, HTML based).

3.4.1.2 XML

One of the principal advantages of semantic tagging, such as that used by XML, is that the same document can be used for a variety of applications without having to translate it to another format.

The Extensible Stylesheet Language (XSL), defined by the W3C, is a language for expressing stylesheets. Given a class of structured documents or data files in XML, designers use an XSL stylesheet to express their intentions about how that structured content should be presented; that is, how the source content should be styled, laid out and paginated onto some presentation medium such as a window in a Web browser or a set of physical pages in a book, report, pamphlet, or memo.

To display an XML IDMEF message on a graphical display, all that is needed is a viewing program (such as a web browser) that supports XML, and a style sheet that tells the program how to display the content of the various tags. The Microsoft Internet Explorer and Mozilla (not Netscape) web browsers have support for XML and XSL. There are also several free and commercial XML browsing tools available.

To display an XML IDMEF message on the printed page, all that is needed is a formatting program that supports XML, and a style sheet (different from the browser style sheet) that tells the program how to print the content of the various tags.

3.4.2 Data transfer aspects

Data transfer aspects are those features of the data format that impact how efficiently the implementation can transfer data.

3.4.2.1 SMI

SMI-MIBs are not good for representing bulk data as there is an SMI-specified size limitation of 65535 for a single object.

3.4.2.2 XML

The amount of data to be transferred is not a problem for XML, since the DTD just defines tags that identify that markup -- everything is treated as a simple stream of bytes. In situations where most of the data elements are small, however, XML may impose a lot of overhead, resulting in messages that require more bytes to represent the data tags than to represent the data itself. This overhead can be

partially offset by XML's easy compressibility, but only if compression is available.

XML is primarily intended to represent "printable" data (UTF-8 or UTF-16). Although it is capable of representing arbitrary "binary" data, its method for doing so is both cumbersome and inefficient, and should be avoided if at all possible.

3.5 Deployment issues

Deployment issues address how easy it will be to actually "get IDMEF out there" once it has been standardized (and adopted by vendors). These issues affect existing deployed systems (which may have to be upgraded or replaced), and existing products (which may have to be modified).

3.5.1 SMI

Most network devices have an SNMP agent in them, and thus the code to generate and handle SMI compliant (SNMP) messages is already there.

3.5.2 XML

To the authors' knowledge, XML is not currently supported by any existing intrusion detection products, commercial or otherwise.

However, some existing products already make use of the Java runtime environment to implement their management console functionality; this may make the integration of a Java-based XML parser somewhat easier than starting from scratch.

3.6 Transport issues

Transport issues are those factors of the implementations that impact how IDMEF messages can be transmitted via the network.

3.6.1 TCP/UDP

Can TCP-based protocols, UDP-based protocols, or both be used?

3.6.1.1 SMI

The SMI representation by itself does not have any bearing on the transport protocol. The application protocol designers can base their choice of transport protocol on the requirements of the application.

3.6.1.2 XML

XML IDMEF messages, since they are simply a stream of bytes, can be sent over TCP without problems.

XML IDMEF messages can, in general, be sent over UDP too, although if messages are split across UDP datagrams, message reassembly would have to be performed on the receiving end.

3.6.2 Intrusion Alert Protocol (IAP)

The Intrusion Alert Protocol (IAP) has been selected by the working group as the protocol to be used for sending and receiving IDMEF alerts. IAP is an HTTP-like protocol over TCP that uses the Transport Layer Security protocol [11] (an Internet Standard protocol based on Netscape's Secure Sockets Layer, SSL) for security and authentication.

3.6.2.1 SMI

The SMI-represented payload will have no problems being transported over IAP.

3.6.2.2 XML

As XML is simply a stream of bytes, it can be transported over the IAP without problems.

4. Selected Implementation

On Februrary 1-2, 2000, an interim meeting of the Intrusion Detection Working Group was held at Harvey Mudd College. At this meeting, a tentative decision was made to use the XML IDMEF implementation. This recommendation was put forth to the working group (via the mailing list), with strong comments requested. A final decision on the matter was made at the March, 2000 IETF meeting in Adelaide, Australia.

4.1 Selection Rationale

The following points, taken from the minutes of the February, 2000 interim meeting, were given as the rationale for this decision:

- The IDMEF must support both network-based and host-based intrusion detection systems ([3], Requirement 7.1). While both XML and SMI make sense in network- based systems, XML is much more "natural" in

  host-based systems. This is especially true when considering the representation of text-based log formats such as UNIX "syslog."

- XML is more easily extended to other, related activities such as the command and control of intrusion detection systems. While such activities are outside the scope of the IDWG, selection of a data format that is compatible with them is viewed as beneficial ([3], Section 3.1, paragraph 4).

- The SMI implementation of the tell-only mode is awkward (i.e., when all data must be communicated via trap/inform). The idea of trap-directed polling (tell-and-ask) for intrusion detection alerts is not seen as the primary IDS communications mode by the group.

- Some of the vendor representatives to the group indicated that they could/would not support the idea of querying their analyzers for more data (i.e., tell-and-ask mode). While this idea is interesting from a research and correlation perspective, the vendors are more concerned with fast, lightweight analyzers that operate only in a tell-only mode.

- XML is attractive to these same vendors, since it requires only a basic "print" function to produce XML-formatted IDMEF messages on the analyzer.

- The Intrusion Alert Protocol (IAP) [8], adopted by the group, is an HTTP-like protocol. XML, because it is some sense "designed" for protocols of this type, is a "natural" for use with IAP.

5. Security Considerations

This Internet-Draft compares two data formats that have been proposed for the exchange of security-related data between security product implementations. There are no security considerations directly applicable to the format of this data. There may, however, be security considerations associated with the transport protocol chosen to move this data between communicating entities.

6. Acknowledgements

The authors would like to thank Mike Erlinger, Stuart Staniford-Chen, Jurgen Schoenwaelder, Dipankar Gupta, John White, Herve Debar, and the other members of the idwg-public mailing list for their comments and suggestions.

7. References

[1] Bradner, S., "The Internet Standards Process -- Revision 3," BCP 9, RFC 2026, October, 1996.

[2] Internet Engineering Task Force, "Intrusion Detection Exchange Format (idwg) Charter."

[3] Wood, M., "Intrusion Detection Message Exchange Requirements," draft-ietf-idwg-requirements-02.txt, October 21, 1999, work in progress.

[4] Mansfield, G. and D. Gupta, "Intrusion Detection Message MIB," draft-glenn-id-notification-mib-02.txt, January 29, 2000, work in progress.

[5] Curry, D, "Intrusion Detection Message Exchange Format Extensible Markup Language (XML) Implementation," draft-curry-idmef-xml-01.txt, March 15, 2000, work in progress.

[6] Debar, H., M. Huang, and D. Donahoo, "Intrusion Detection Exchange Format Data Model," draft-ietf-idwg-data-model-02.txt, March 7, 2000, work in progress.

[7] Case, J., R. Mundy, D. Partain, and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework," RFC 2570, April, 1999.

[8] Gupta, D., "IAP: Intrusion Alert Protocol," draft-ietf-idwg-iap-01.txt, January 27, 2000, work in progress.

[9] McCloghrie, K., Perkins, D. and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[10] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824, (December, 1987).

[11] Dierks, T., and C. Allen, "The TLS Protocol - Version 1.0," RFC 2246, January 1999.

8. Authors' Addresses

Glenn Mansfield
Cyber Solutions, Inc.
6-6-3 Minami Yoshinari
Aoba-ku, Sendai 989-3204
Japan
Phone: +81 22-303-4012
Email: glenn@cysols.com

David A. Curry
Internet Security Systems
345 Route 17 South

Upper Saddle River, NJ 07458
USA
Phone: +1 201 934-4207
Email: davy@iss.net

Full Copyright Statement