Common Ground - A Discussion of Standards in Network Security and How to Extend Them into the Network Assessment Arena

Timothy Politowicz
Practical Version 1.4b, Option 1

Abstract

Effective communication between existing security tools, products and groups has been hampered by a lack of common ground. Standards, whether de facto standards or formally developed, can facilitate collaboration, simplify integration of heterogeneous systems, and foster revolutionary solutions in new fields. A discussion is provided of several entrenched and emerging standardization efforts in the security field. Based on some of these technologies, a data model suitable for use in the network assessment arena is proposed.

Introduction

In order to secure a network an understanding of the threats against it and the vulnerabilities is has is required. With this understanding, appropriate defense and response tactics can be used. Typical defensive tactics include the use of packet filtering mechanisms such as firewalls and router access control lists. The defensive strategy may also include preventative measures. Periodic network assessments, used to uncover and correct vulnerabilities, are a common intrusion prevention technique.

This paper discusses areas of “common ground”, or standardization, that have improved or have the potential to improve communication between various network security components and groups. The paper is divided into three main sections. The first section addresses efforts related to vulnerabilities. Overviews of the Common Vulnerabilities and Exposures (CVE) list, ICAT metabase and the Open Vulnerability Assessment Language are provided. The second section addresses standardization efforts in the defense and response area. Two closely related IETF drafts, the Intrusion Detection Message Exchange Format (IDMEF) and Incident Object Description and Exchange Format (IODEF) are covered. The third section then takes a closer look at two commonly used tools, SARA™ and Nessus, used for network assessments and how some of the standardization efforts covered in this paper can be used to improve network assessment tools and processes. Finally, a summary is provided along with an acknowledgements and references section.

Vulnerabilities

The Merriam-Webster dictionary defines a vulnerability as something that makes one “open to attack or damage” [16]. In the context of computer networks most people, and this paper, identify vulnerabilities as specific flaws in software packages, operating systems, networking hardware and other related components that leave a system open to attack or damage.

There are many companies and groups that discover, organize and provide vulnerability information to the public. Some of these groups include [11]:

- CERT® Coordination Center
- ISS X-Force
- Security Focus/Bugtraq/NT Bugtraq

Until recently, most of the information obtained by these groups was provided by the vendors and open disclosure by members of the computer community. Not surprisingly, some companies are now openly paying hackers for vulnerability information that has been previously "unknown" so that they can bolster their portfolio and use it as a competitive advantage by providing clients with advanced warning [13]. Regardless of the methods of obtaining the information, each group has its own format for representing and reporting vulnerability information.

To a certain extent, end-users of vulnerability information, don't care where the data comes from, they just want to use it to improve the security of their networks. Until a standard representation of this information was available, users were left with a few choices including: use only one source, manually slog through multiple sources and try to correlate items, or try to write scripts. Fortunately, work has been done by the MITRE Corporation and Computer Security Resource Center (CSRC) of the National Institute of Standards and Technology (NIST) to help address issues related to standardizing vulnerability related information.

Common Vulnerabilities and Exposures (CVE)

In 1998/1999, work was performed for the MITRE Corporation which resulted in the introduction of the concept of a Common Vulnerabilities Enumeration (CVE) [14]. As the work progressed, the "E" in CVE was changed to represent Exposures. CVE is a publicly available dictionary of standardized names that can be used to cross reference data from different vulnerability databases and security tools [1]. The security community, including vendors and open source projects, has embraced CVE. Over 70 organizations participate in the effort, and there are well over 100 "CVE-compatible" products and services on the market and freely available [4].

Show in Figure 1 is a typical CVE entry. The ID for the entry is shown at the top, as "CVE-2002-0055". The ID or name for an entry is based on the year and a counter (reset annually) when the entry was given "candidate" status. For example, the entry shown in Figure 1 was the 55th entry added in 2002. The only other parts contained in a CVE entry include a brief description, in this case related to an SMTP denial-of-service attack, and a list of related references.

CVE-2002-0055

SMTP service in Microsoft Windows 2000, Windows XP Professional, and Exchange 2000 to cause a denial of service via a command with a malformed data transfer (BDAT) request.

Reference: BUGTRAQ:20020306 Vulnerability Details for MS02-012
Reference: MS:MS02-012
Reference: XF:ms-smtp-data-transfer-dos(8307)
Reference: BID:4204

Figure 1 Example CVE Entry [5]

By itself, CVE does not help end-users solve security. It does not include technical details, patch, classification, or impact information [8]. For example, it is not designed to allow users to search based on operating system or vendor, it is simply meant to help identify specific vulnerabilities that have been reported from various sources. One may wonder how a standardized set of names can help solve problems if it isn't designed to answer the questions that need to be asked? The answer is that other work has been done which builds on CVE to provide this functionality.

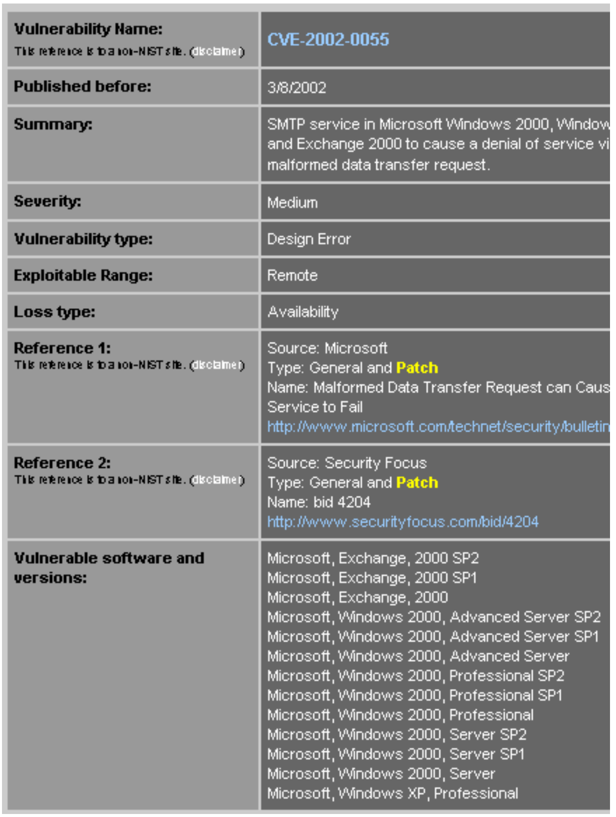
ICAT

There are many vulnerability databases and security tools, and CVE provides a common link between these resources. In order to have a common interface to, and not just a link between, the information provided by these vulnerability databases, ICAT was developed by a group at the National Institute for Standards and Technology (NIST). ICAT is a metabase, not a database. Metadata is data that describes data. The ICAT metabase contains information that describes the data contained in other databases, specifically vulnerability databases.

For people interested in accessing computer vulnerability information, ICAT provides a normalized, searchable index of the sources previously listed in the Vulnerabilities section, in addition to the following data sources [10]:

- Neohapsis
- Microsoft

The metabase is freely available for download in several formats, and is also accessible through the ICAT website. Figure 2 shows the ICAT entry for CVE-2002-0055 provided by the ICAT website. Note that ICAT adds a great deal of additional information based on the CVE references. This includes patch information if available.



Vulnerability Name: This reference is to a non-NIST site. (disclaimer)	CVE-2002-0055
Published before:	3/8/2002
Summary:	SMTP service in Microsoft Windows 2000, Window and Exchange 2000 to cause a denial of service vi malformed data transfer request.
Severity:	Medium
Vulnerability type:	Design Error
Exploitable Range:	Remote
Loss type:	Availability
Reference 1: This reference is to a non-NIST site. (disclaimer)	Source: Microsoft Type: General and **Patch** Name: Malformed Data Transfer Request can Caus Service to Fail http://www.microsoft.com/technet/security/bulletin
Reference 2: This reference is to a non-NIST site. (disclaimer)	Source: Security Focus Type: General and **Patch** Name: bid 4204 http://www.securityfocus.com/bid/4204
Vulnerable software and versions:	Microsoft, Exchange, 2000 SP2 Microsoft, Exchange, 2000 SP1 Microsoft, Exchange, 2000 Microsoft, Windows 2000, Advanced Server SP2 Microsoft, Windows 2000, Advanced Server SP1 Microsoft, Windows 2000, Advanced Server Microsoft, Windows 2000, Professional SP2 Microsoft, Windows 2000, Professional SP1 Microsoft, Windows 2000, Professional Microsoft, Windows 2000, Server SP2 Microsoft, Windows 2000, Server SP1 Microsoft, Windows 2000, Server Microsoft, Windows XP, Professional

Figure 2 Example ICAT Entry [9]

The download section of the website also includes a documentation file discussing the schema. An examination of the schema will provide the interested reader with an exact understanding of the different dimensions which can used to search for vulnerability information. Searches can be based on, but are not limited to, keyword, source of information, vulnerability type (e.g. buffer overflow or race condition), vulnerable software, loss type (e.g. confidentiality or availability), and operating system [12].

In summary, ICAT provides a single interface to many vulnerability information sources that can be used to search based on various attributes. The entries provide classification, severity/impact statements, patch details, and other descriptive information. With the combination of CVE and ICAT, vulnerabilities can be openly discussed and understood with little or no confusion. One issue not addressed by these technologies is a standardized way to test for the existence vulnerabilities.

Open Vulnerability Assessment Language (OVAL)

The Open Vulnerability Assessment Language (OVAL) is another product of the MITRE Corporation announced in late 2002. An OVAL query specifies how to test for, and can potentially be used to test for, the existence of vulnerabilities based on system configuration information. Just as CVE has standardized the names of vulnerabilities, OVAL standardizes how to test for them. It is meant to help avoid some of the issues that arise when using assessment tools, such as conflicting results due to the use of different testing techniques for the same vulnerability [18].

For a given vulnerability with a CVE number, an OVAL query is written along with a synopsis and submitted by anyone who cares to join the "OVAL Community Forum". The synopsis section should include information related to two main items: the vulnerable software and the vulnerable configuration(s). The query itself should use the OVAL schema and be written using the Structured Query Language (SQL) with comments. For those that aren't aware, SQL is the language of choice for interacting with relational databases (a discussion of SQL details is outside the scope of this paper), and a schema defines the organization of the data contained in a database. The OVAL schema is currently designed to support the following operating systems [17]:

- Microsoft Windows 2000/NT
- Sun Solaris 7 and 8
- Red Hat Linux

Below are the contents of the "Query Synopsis" section for OVAL query number 30.

Vulnerable Software

- Windows 2000
- Affected smtpsvc.dll versions
- Patch Q313450_W2K_SP3_X86_EN.exe not installed
- Windows 2000 Service Pack 3 (or later) not installed

Vulnerable Configuration

- SMTP service enabled

The reader should have a basic understanding of what the OVAL query will be testing for from the synopsis section, and then be able to use the details provided in the SQL query to carry out the test.

The SQL for OVAL30 is shown [19]. The reader will notice that the test details for elements listed in the synopsis are included in the various "EXISTS" clauses. For example, the first EXISTS clause test for the appropriate version of Windows.

```
SELECT 'CVE-2002-0055' FROM Placeholder WHERE EXISTS
-- ### BEGIN VULNERABLE SOFTWARE EXISTS
--
-- Windows 2000
-- This query is for Windows 2000 Professional, Server, and Advanced Server.
    (SELECT 'Windows 2000 Installed' FROM Win2K_RegistryKeys WHERE
    RegistryKey = 'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
    NT\CurrentVersion' AND
    EntryName = 'CurrentVersion' AND
    EntryValue = '5.0')
AND EXISTS
-- Affected smtpsvc.dll versions
    -- Build the FilePath for smtpsvc.dll by retrieving the value of
    -- SystemRoot from the registry, and concatenating it with
    -- '\System32\inetsrv\smtpsvc.dll' (using || concat. operator):
            (SELECT 'File %windir%\System32\inetsrv\smtpsvc.dll version <
            6.0.2600.28' FROM Win2K_FileAttributes WHERE
            FilePath = (SELECT EntryValue || '\System32\inetsrv\smtpsvc.dll' FROM
            Win2K_RegistryKeys WHERE
            RegistryKey =
            'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
            NT\CurrentVersion' AND
            EntryName = 'SystemRoot') AND
        -- To avoid lexical (string) comparisons of file versions, the
        -- version string (e.g. '6.0.2600.28') is broken into its
        -- components, stored as numbers.
            (Version1 < 6 OR
            (Version1 = 6 AND Version2 = 0 AND
            (Version3 < 2600 OR
            (Version3 = 2600 AND Version4 < 28)))))
AND NOT EXISTS
-- Patch Q313450_W2K_SP3_X86_EN.exe installed
    (SELECT 'Patch Q313450' FROM Win2K_RegistryKeys WHERE
    RegistryKey = 'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
    NT\CurrentVersion\Hotfix\Q313450' AND
    EntryName = 'Installed' AND
    EntryValue = '1')
AND NOT EXISTS
-- Windows 2000 Service Pack 3 (or later) installed
    (SELECT 'Windows 2000 SP3 Installed' FROM Win2K_RegistryKeys WHERE
    RegistryKey = 'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
    NT\CurrentVersion' AND
    EntryName = 'CSDVersion' AND
    EntryValue >= 'Service Pack 3')
-- ### END VULNERABLE SOFTWARE EXISTS
--
-- ### BEGIN VULNERABLE CONFIGURATION
AND EXISTS
-- SMTP service enabled
    (SELECT 'SMTP Enabled' FROM Win2K_RegistryKeys WHERE
    RegistryKey =
    'HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SMTP' AND
    EntryName = 'Start' AND
    EntryValue != '4')
-- ### END VULNERABLE CONFIGURATION
;
```

[19]

As can be seen from the example, the queries are typically formatted and commented such that the vulnerable software and vulnerable configuration sections are easily distinguishable. In addition, the items listed in the synopsis are usually included in the relevant comment sections and expanded upon. By combining this common format with knowledge of SQL (a fairly common language), many people are able to easily and unambiguously understand how testing of the given vulnerability should be performed.

It should be noted that the use of SQL to describe testing procedures does not limit users of OVAL to SQL based solutions. An obvious application of OVAL is to directly apply the queries to an existing asset management database system, but OVAL is meant to solve a more fundamental problem. It helps to solve the same problem CVE and ICAT address, effective communication in the security field through standardization. The last section of this paper will return to the issue of testing for vulnerabilities and show how providing one more point of common ground can help create more effective and interoperable tools, processes and products.

Defense and Response

Threats, such as crackers, use vulnerabilities to compromise systems. In order to minimize the chances of being compromised defensive tactics must be used, and when a compromise does occur there must be a response. This section highlights two efforts currently in progress in Internet Engineering Task Force (IETF) working groups. Both of these efforts are currently in draft status. They are also both based on the Extensible Markup Language (XML). A brief introduction to XML will be provided.

The first working group is the Intrusion Detection Exchange Format or "idwg" and the work being discussed in this paper is the "Intrusion Detection Message Exchange Format Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition." The second working group is the Extended Incident Handling or "inch" working group. The work related to the "Incident Object Description and Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition" will be discussed.

Extensible Markup Language (XML)

The Extensible Markup Language (XML) is a subset of an ISO standardized language called Standard Generalized Markup Language (SGML), which is much larger and more complex. XML is technically a metalanguage, or a language to describe languages [23].

XML should not be confused with HTML. The Hypertext Markup Language (HTML) is an application of SGML. It uses a predefined set of tags such as "<title>" and "</title>" to mark the content that falls between them as a title. For example, given the HTML snippet "<title>Common Ground</title>" a web browser knows that "Common Ground" is the title of this document, and can use it accordingly, because the title tags are part of the HTML specification. When XML is applied, tags are also used, but one of the major differences is that the set of tags available is defined by the user and can represent anything, not just document structure and formatting elements.

An application of XML is shown below that could be used to describe someone's home.

```
<RESIDENCE>
      <GARAGE>1 car</GARAGE>
      <ROOMS>
            <KITCHEN floor="first"></KITCHEN>
            <BEDROOM floor="first">
                  <FURNITURE>bed</FURNITURE>
                  <FURNITURE>nightstand</FURNITURE>
            </BEDROOM>
      </ROOMS>
</RESIDENCE>
```

For the purposes of this paper, you need to understand a few basic aspects of XML. The first has been mentioned; the ability to define the set of tags (also referred to as elements and at times classes) will be. Also, it is important to understand that an XML element can have attributes. In the above example, the KITCHEN element and BEDROOM element have an attribute called “floor” both with the same value “first”. The XML schema could be changed to have a generic element called ROOM with two attributes “floor” and “type” where the value for “type” could be “kitchen” or “bedroom”. The last aspect of XML that will be useful is an understanding of what a Document Type Definition (DTD) is.

A DTD formally defines what elements will exist, what attributes those elements will have and how elements can be combined. The DTD for the example shown would indicate that a RESIDENCE can consist of a GARAGE element and a ROOMS element, a BEDROOM element has a floor attribute and can have zero or more FURNITURE elements … and so on. A DTD is not required. Without a DTD, an XML document can still be used and processed, but it cannot be formally verified against a given data model. In those cases where the DTD is not available, users who want to understand the underlying data model are left to inspection methods such as source code inspection or reverse engineering the data model from sample XML documents. When working with sample XML documents, the derived data model may be incomplete since attributes and elements may be optional and the sample dataset may not provide proper coverage.

Intrusion Detection Message Exchange Format (IDMEF)

The Intrusion Detection Message Exchange Format (IDMEF) Data Model currently in draft format uses XML and a DTD to provide a data model to be used by automated intrusion detection systems for reporting alerts. It is hoped a standard format:

> will enable interoperability among commercial, open source, and research systems, allowing users to mix-and-match the [security products] according to their strong and weak points to obtain an optimal implementation [6].

Figure 3 shows a typical network configuration. There are multiple segments protected by filters on the access router, the firewall, and an intrusion detection system. The intrusion detection system will typically consist of network and/or host probes and a management console.

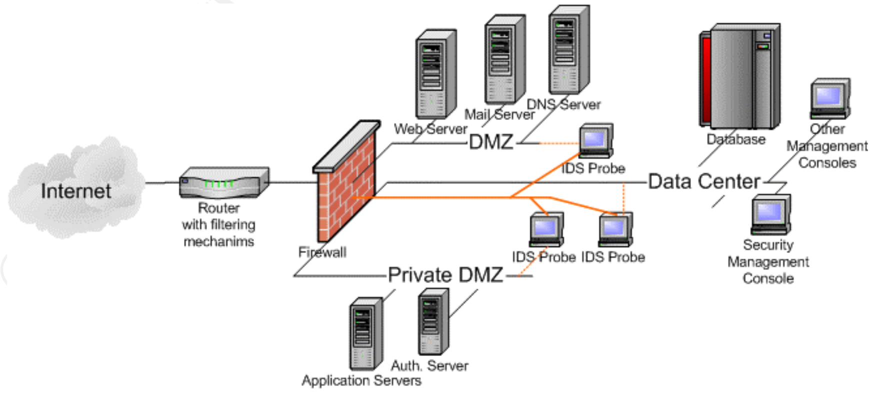


Figure 3 Sample Network Configuration

Figure 4 shows an overview of the data model presented in the current IMDEF draft. There are currently five message types defined (shown directly connected to “IDMEF-Message”). In the sample network configuration shown the “IDS Probes” and any host based probes would send these messages to the IDS/Security management system(s). If firewall and router vendors choose to implement IDMEF functionality, those network elements could also issue these messages, allowing an IDMEF aware management system to correlate information from a variety of security devices from various vendors.

The Heartbeat message is used by an analyzer to communicate its current status. Referring to the IDMEF data model, there are four main elements associated with a Heartbeat message: Analyzer, CreateTime, AnalyzerTime, and AdditionalData. Many of the elements in the data model are self-explanatory. For example, the Analyzer element is used to identify the source of the Heartbeat message, and the CreateTime indicates the time the message was created. Note that not all elements are required and some can be included multiple times. For example, the AnalyzerTime element is optional, and if included is limited to one entry, while the AdditionalData element is an example of an element that can be included zero or more times. The AdditionalData element is provided to permit the inclusion of information not directly supported by the data model. As certain added elements become commonplace, they will be considered for inclusion in the standard model. The details of which elements are optional or can be multiply included can be found in the draft, and will not be further discussed unless necessary [6].

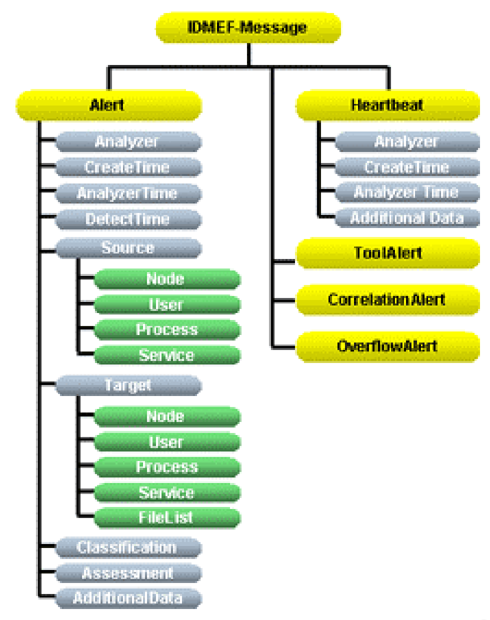


Figure 4 IDMEF Data Model Overview [6]

The IDMEF Alert message contains all of the elements of a Heartbeat message and adds five more: DetectTime, Source, Target, Classification, and Assesment. The purpose of the message is to relay all pertinent information related to what the analyzer considers an "event". Some of the core *classes* defined in the IDMEF standard are shown in Figure 4 under the Source and Target elements. The first of these core classes shown, the Node class includes a free form text location value, zero or more Address entries, a name entry and the category entry (which indicates one of 13 defined values such as "unknown", "DNS" or "Windows NT domain") that has been used as the basis of the Node information. The design of the Address class is fairly robust. It can be used to represent a wide range of address types. Support for VLAN information is also built in. Some of the addressing schemes supported include IPv4, IPv6, ATM, hardware (e.g. MAC), and application (e.g. e-0mail addresses). Another high-level class used by the Source and Target elements is the User class. This class is basically a container for UserId entries, which contain username, ID-number and type (e.g NIS) information. The Process class as expected describes processes being executed and includes such items as the name, process ID, path and arguments [6].

Like the Node and many other IDMEF classes, the Service class will prove to be very useful for a variety of applications discussed later in this paper. The basic Service class generically describes network services, where a network service is defined by a name, a port, the combination of a name and port, or set of ports (an optional protocol can also be specified). It has already been extended in the draft to include two sub-classes, the SNMPService and WebService classes, to provide more focused information such as an SNMP object identifier or a URL [6].

The two remaining high-level elements included in Alert messages are the Classification and Assessment elements. The purpose of the Classification element is to allow the manager who receives the Alert messages to be able to obtain additional information. To accommodate this, a name, URL and the source of the name are included. The current list of valid source values includes "unknown", "vendor-specific", and not surprisingly, CVE. The Assessment class provides information related to the analyzer's assessment of the impact, action(s) taken, and confidence level in relation to the observed event [6].

Below is an example based on one in the IDMEF draft. This shows what an Alert message would look like when an analyzer detects a Ping-of-death attack (CVE-1999-128) from IP address 192.0.2.200 targeting two hosts, 192.0.2.50 and Cisco.router.b10:

```
<IDMEF-Message version="1.0">
  <Alert ident="abc123456789">
    <Analyzer analyzerid="bc-sensor01">
      <Node category="dns">
        <name>sensor.example.com</name>
      </Node>
    </Analyzer>
    <CreateTime ntpstamp="0xbc71f4f5.0xef449129">
      2000-03-09T10:01:25.93464Z
    </CreateTime>
    <Source ident="a1a2" spoofed="yes">
      <Node ident="a1a2-1">
        <Address ident="a1a2-2" category="ipv4-addr">
          <address>192.0.2.200</address>
        </Address>
      </Node>
    </Source>
    <Target ident="b3b4">
      <Node>
        <Address ident="b3b4-1" category="ipv4-addr">
          <address>192.0.2.50</address>
        </Address>
      </Node>
    </Target>
    <Target ident="d7d8">
      <Node ident="d7d8-1">
        <location>Cabinet B10</location>
        <name>Cisco.router.b10</name>
      </Node>
    </Target>
    <Classification origin="cve">
      <name>CVE-1999-128</name>
      <url>http://www.cve.mitre.org/</url>
    </Classification>
  </Alert>
</IDMEF-Message>
```
[6]

Referring back to figure 4, three special messages, which provide information about other Alert messages, are available for use. The CorrelationAlert message provides a mechanism for grouping previous alerts together by providing the name/reason for grouping them along with the list of alert identifiers. The ToolAlert is basically a special type of CorrelationAlert where the name is typically the name of a specific tool and additional "command" details can be provided. The last message type is the OverflowAlert. This message type allows analyzers to provide details of the buffer overflow attacks, such as the targeted program and overflow data [6].

To Summarize, the IDMEF data model provides a standardized mechanism for intrusion detection devices to deliver alert and heartbeat information. The classes and relationships that have been defined are well designed for their intended purpose and many of the core classes are generic enough to be applied in other areas. In the next section, the reader will see how the IDMEF work, even in draft format, has been leveraged to attempt to improve organizational communication issues at the organizational level. A discussion of how this work can be applied to network assessments will then be presented.

Incident Object Description and Exchange Format (IODEF)

Computer Security Incident Response Teams are established to perform functions necessary to protect and secure critical assets for an organization [3]. The organization can be, but is not limited to, a company, country, or geographic region. The Incident Object Description and Exchange Format (IODEF) are currently in draft status, being worked on by the INCH working group. The purpose of the work is to provide:

> a format for Computer Security Incident Response Teams (CSIRTs) to exchange operational and statistical incident information among themselves, their constituency, and their collaborators. It can also provide the basis for the development of interoperable tools and procedures for incident reporting [15].

By no coincidence, IODEF shares many similarities with IDMEF. Compatibility with IDMEF is one of the goals of the IODEF work. As previously described, IDMEF messages contain information from intrusion systems. Incident response teams, for investigation and documentation purposes, use this same information and can include it in an IODEF document. In some cases, the core IDMEF classes have been enhanced; these will be discussed as needed [15].

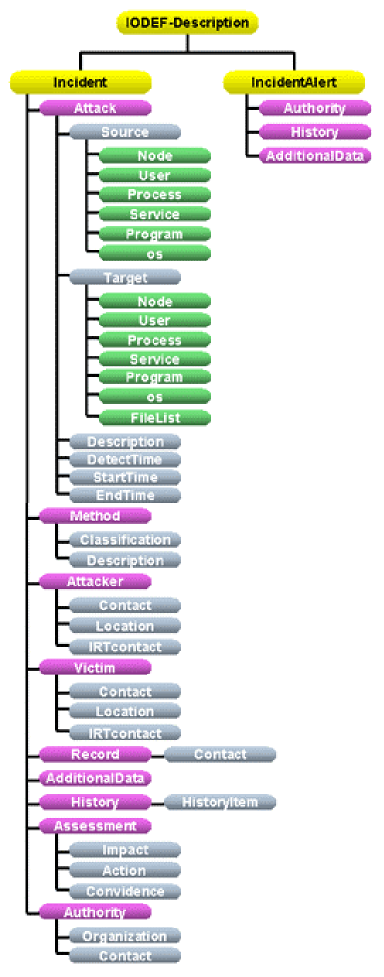


Figure 5 IODEF Data Model Overview [15]

Figure 5 provides an overview of the IODEF data model. There are two classes currently defined: IncidentAlert and Incident. The IncidentAlert class is basically a wrapper for IDMEF messages. The IDMEF messages generated by analyzers are encapsulated as AdditionalData elements and appropriate Authority and History information is included.

Additional details have been shown for the Incident class in Figure 5. Notice that the Attack class contains Source and Target elements. The

IODEF versions have two additional elements, “OS” and “Program”. The OS element indicates the operating system. The Program element indicates the program used to perform the attack when used with the Source class and represents the program being attacked when used with the Target class. Both of these additional elements are optional. The other elements of the Attack class include a free form text description and three time based values indicating when the attack started, stopped and was detected [15].

Information related to real world entity(ies) or people is specified via the Attacker and Victim classes. These classes correspond to the Target and Source elements. Information such as personal names, postal addresses and Internet registries (e.g. ARIN or APNIC) can be encapsulated. The method(s) for determining this information is not defined by IODEF and rests in the hands of the individual CSIRTs [15].

The Method class refers to the technique used to perform the attack. It is another ‘wrapper’ class, combining a description with an IDMEF Classification element. Recall that IDMEF Classification elements are based on well-known vulnerabilities from sources such as Bugtraq and CVE (vendor-specific entries can also be used) [15].

The Assessment class is modeled on the IDMEF format, but is semantically different. The assessment contained in an IDMEF message is based on the viewpoint of the analyzer sending the message; the assessment in an IODEF Incident message is based on the CSIRT’s viewpoint.

The CSIRT, specified through the Authority element, which created the incident report may wish to include information that is not supported by any of the classes discussed so far. This information includes actions taken, significant events that occurred, exported data from monitoring tools, or the results of forensics investigations. The Record, History, and AdditionalData elements are used to detail this supportive information.

An interesting part of the IODEF design that differentiates it from IDMEF is support for “restriction” tagging. Supported by all elements, the restriction attribute allows the creator of the IODEF description to indicate that data is sensitive and should be handled accordingly. This is an optional attribute. The current values defined for this attribute are default, public, internal, and restricted. The purpose of IODEF is to facilitate communication between organizations. Restriction tagging should encourage more organizations to share their information since it allows them to easily sanitize information.

Prevention through Network Assessments

Most of the discussion to this point has focused on standardization efforts with only minor references made to the actual use of those standards in the real world. The OVAL section covered a language used to specify exactly how to test for vulnerabilities. It was mentioned how OVAL entries could be used along with detailed information about network resources to determine if the exhibit a known vulnerability or exposure. Some may see this as a great alternative to actively probing systems with scanning tools, but it is probably better to view OVAL as a complimentary technology since the configuration information may be stale or incorrect. This section provides a detailed discussion of two popular CVE enabled vulnerability scanners SARA™ and Nessus. A detailed discussion of the tools' capabilities, options and functionality will not be provided since there are many excellent papers that address these aspects of the tools, and the documentation for both is quite good and readily available. The main focus is on the XML reports and how some of the standardization technologies already discussed are being used, and how others could be applied to improve their usefulness of this class of tools.

Security Auditor's Research Assistant (SARA™)

SARA™ is a free, modular vulnerability scanning tool based on the Security Administrator's Tool for Analyzing Networks (SATAN). It is designed to interface with existing tools (e.g. nmap) rather than recreating them. The integration and reporting capabilities provided by this tool have made it a popular assessment tool [22].

The "SARA Report Writer" generates well-organized, useful reports in several formats. By choosing the comma separated value (CSV) report format, an XML based report is also generated. Figure 6 shows an interpretation of the data model based on an analysis of the report writer scripts [21] and an examination of a few sample reports (an official DTD could not be found).

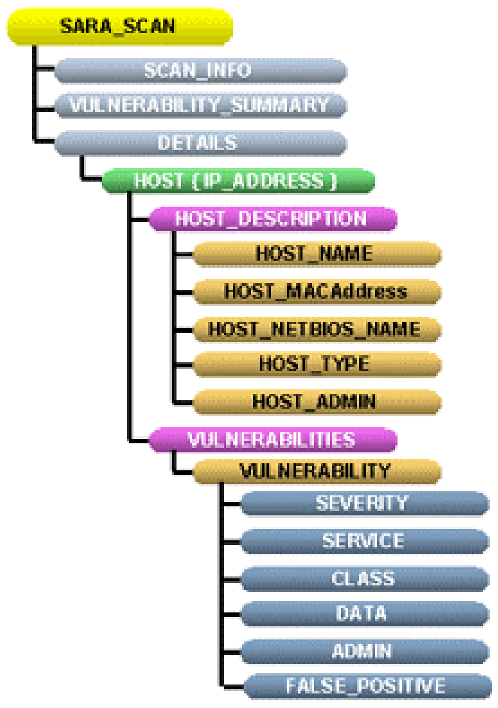


Figure 6 SARA™ XML Output Overview

As far as I have been able to determine, only the highest-level element, SARA_SCAN, and HOST element (under DETAILS) make use of XML attributes. The SARA_SCAN attributes consist of an identifier, date, version, and attack level. The HOST element has only one attribute, IP_ADDRESS (shown in braces in the figure).

With the exception of details that have been omitted from the SCAN_INFO and VULNERABILITY_SUMMARY elements, the figure shows all of the information that is presented as output. A typical XML report contains one entry for the SCAN_INFO, VULNERABILITY_SUMMARY, and DETAILS elements, with the DETAILS containing many HOST entries. Depending on the results of the scan, there could be multiple VULNERABILITY entries for each HOST entry.

HOST entries contain a description section and potentially vulnerability entries. A VULNERABILITY entry contains up to 6 types of elements. The SEVERITY element provides a color value such as brown, yellow, red, grey or green. The SERVICE element consists of only a name and port. The CLASS element contains a text description such as "possible backdoor found" or "possible http cgi access found." The DATA section provides additional information such as a detailed description. The FALSE_POSITIVE element is used to mark discovered information accordingly after the vulnerability has been investigated.

Nessus

Nessus, like SARA™, is a modular vulnerability scanning tool. For those wondering what the name means, don't worry it's not an acronym and doesn't really have any significance other than being the name of one of the most popular free vulnerability scanners.

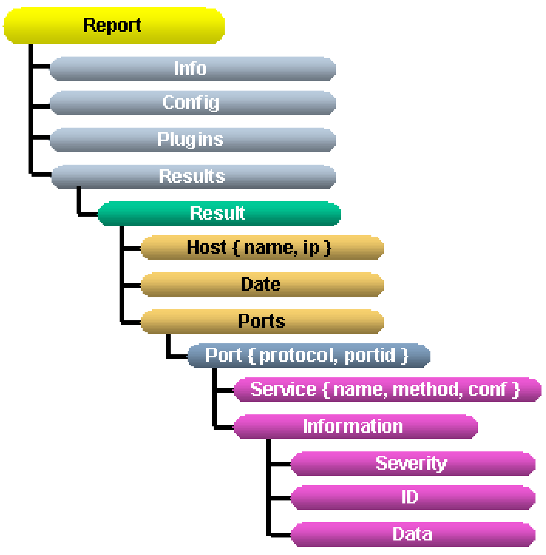


Figure 7 Nessus XML Output Overview

At the time of writing, the latest stable release of Nessus available is 2.0.1. There is a DTD provided with the nesses source code. The DTD version is 0.2 and comments indicate that it was last modified 2000-11-21. This version of Nessus provides two versions of XML output, the standard version and the "old" version. The DTD provided appears to match the old style and one specifically for the new version could not be found. Figure 7 is based on an examination of reports generated using the "new" version of the XML report, all further discussion will be based on this style.

The first two elements for a Nessus XML report, Info and Config, contain the same type of information represented by the SARA SCAN_INFO element. The Plugins element contains individual Plugin entries, each with a Nessus specific integer-based identifier. The Plugin entries also specify information such as a name, version, family (e.g. CGI abuses), CVE-ID, Bugtraq-ID, category, risk, and summary. Examples of the category values include "attack", "infos", "denial", "scanner" and "mixed" [2]. The risk elements have values such as "Serious", "High", "Medium" and "Low" rather than the colors used for SARA severity levels. The values for Plugin entries are established in the plugin scripts.

New modules, or plugins, are continually added to Nessus to test for new vulnerabilities and add other functionality. Many of these plugins are written in the Nessus Attack Scripting Language (NASL). NASL plugins designed to test for a CVE vulnerability are supposed to call the "script_cve_id()" function to enable all of CVE related functionality provided by the tool (for example, in the reports) rather than simply putting the CVE-id in the description. The "script_id()" function sets the internal Nessus identifier. The following table shows a code snippet (some lines have been omitted or truncated to conserve space) taken from the plugin that tests for CVE-2002-0055, the vulnerability discussed in previous sections. Also shown is how the XML report presents this information. For example, the "`script_cve_id("CVE-2002-0055");`" and "`script_bugtraq_id(4204);`" lines in the plugin map to the "<cve_id>" and "<bugtraq_id>" elements of the XML document respectively. With the current XML design, the Plugin entries contain the CVE information and the plugin-id is used in other elements to link back to the CVE information rather than directly inserting it. For additional details on NASL and creating plugins, refer to reference [2].

Description Section of Nessus Plugin [7]	Nessus XML Report Element

```
if(description)
{
 script_id(10885);
 script_cve_id("CVE-2002-0055");
 script_bugtraq_id(4204);
 script_version ("$Revision: 1.13 $");

 name["english"] = "MS SMTP DoS";
 script_name(english:name["english"]);

 desc["english"] = "
It is possible to make the remote SMTP
server fail
…
Solution :
http://www.microsoft.com/technet/security/b
ulletin/MS02-012.asp
Risk factor : Medium";

script_description(english:desc["english"])
;


 summary["english"] = "Checks if the remote
SMTP server can be restarted";

script_summary(english:summary["english"]);

 script_category(ACT_ATTACK);
 script_copyright(english:"This script is
Copyright (C) 2002 Renaud Deraison");
 family["english"] = "SMTP problems";
 script_require_ports("Services/smtp", 25);
}
```
	<plugin id="10885">
	    <name>**MS SMTP DoS**</name>
	    <version>**$Revision: 1.11$**</version>
	    <family>**SMTP problems**</family>
	    <cve_id>**CVE-2002-0055**</cve_id>
	    <bugtraq_id>**4204**</bugtraq_id>
	    <category>**attack**</category>
	    <risk>**Medium**</risk>
	    <summary>**Checks if the remote SMTP server can be restarted**</summary>
	    <copyright>**This script is Copyright (C) 2002 Renaud Deraison**</copyright>
	</plugin>

Referring back to Figure 7, the last portion of a Nessus XML report is the Results section. The Result entries contain practically the same information provided by the Host entries in a SARA™ report. For example, a Host element with two attributes, "name" and "ip", is used in Nessus rather than the SARA™ style Host element, with an IP_Address attribute and a Host_Description sub-element with the name buried in it. Similarly, SARA™ uses a list of Vulnerability entries for each Host element and Nessus uses a list of Port entries (which typically contain the related vulnerability information) for each Result (which are essentially per-host). Basically, both of these tools report the same basic information, with some tool specific differences, in XML formats that share very similar high-level structure. Unfortunately, trying to combine output from these tools would require separate parsing tools to address the significant low-level differences. A standard format for representing assessment information in XML would bring with it the same types of benefits to the network assessment area that work like IDMEF and IODEF hope to bring to the intrusion detection and incident handling areas.

Opportunity for Improvement

An overview has been provided of some standardization efforts that have had an impact, such as CVE, and some that are still under development, such as IDMEF. It has been shown how CVE is tied into two of the more popular free network vulnerability assessment tools, and how these tools present their results in XML format. In this section, the discussion turns to how IDMEF and IODEF could be applied to network assessment tools and processes. A data model is suggested (this model is merely a suggestion meant to stimulate discussion of this topic (there wasn't enough time to start a new IETF working group and churn out a requirements document and a few 100+ page drafts to come up with a solid design).

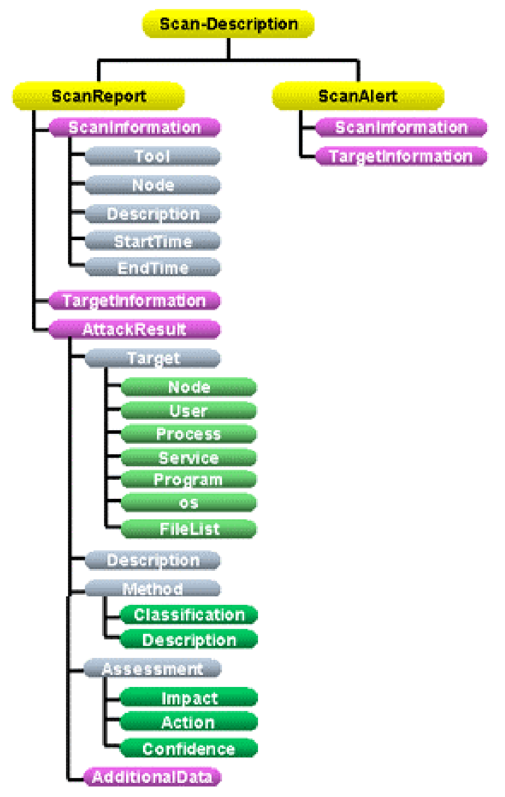


Figure 8 Suggested XML Assessment Data

Figure 8 shows the cursory design of the suggested data model for representing network assessment data. Two main elements are provided, the ScanAlert and ScanReport. This model is based on a comparison of the latest versions of SARA™ and Nessus XML reports, and the latest IDMEF and IODEF drafts. As previously described, IODEF reuses many of the IDMEF elements either directly or by sub-classing them to add functionality. The two top-level elements in the IODEF specification, Incident and IncidentAlert serve as models for the suggested ScanReport and ScanAlert respectively.

The ScanAlert is modeled on the IncidentAlert, but provides a different type of 'service.' The IODEF IncidentAlert is used to simply alert someone/something to the occurrence of an incident and provide relevant information (such as raw IDMEF messages). The ScanAlert would alert an intrusion detection management system or other network/security management system that a scan is going to be performed. As part of this alert, the scanner would provide ScanInformation and TargetInformation (detailed below).

If implemented by a tool such as Nessus, the model provided by the ScanReport element would either replace the current XML output format, or more likely be an alternative. The ScanInformation section is meant to encapsulate information such as the tool that is performing the scan, information about the network node(s) that are being used to launch the scan, time information for documenting scan duration(s), and a general description. If a common format such as the one suggested here were developed, the design of the ScanInformation class would be the main area of new work since the core classes provided by IDMEF and IODEF cover most of the required elements. The TargetInformation element would be used to document the target of the scan and would contain items such as address ranges, network names, and/or network domains. The addition of the AttackResult element(s), differentiates the ScanReport from the ScanAlert.

The AttackResult element is meant to take the place of SARA Details and Nessus Results. It is closely tied to the IODEF Attack class, which in turn shares a great deal of structure with IDMEF Alerts. By using the IDMEF/IODEF Target class, a standard format for representing the 'host' specific information is achieved. This includes support for the many different types of addresses and names defined in the IDMEF draft. By using the IODEF version, it is also possible to accommodate the type of operating system for a target, useful for tools make use of stack fingerprinting and other OS detection techniques. The reuse of the Method and Assessment classes provides a uniform specification of how the attack was performed and an evaluation of the result. If tightly defined, this would help address the issue of different risk scales (e.g. color levels vs. HIGH/LOW). Finally, the ever-popular AdditionalData element provides a catch-all to accommodate items that have not been directly addressed by the data model.

So what does this buy us? First, as the user of these tools you would have a standards based format that allows you to easily share information and combine it other datasets from a variety of compliant tools and systems. If XML is chosen as an implementation it is possible to immediately combine your assessment information with your intrusion detection information (assuming it is IDMEF compliant). Local statistics, from the intrusion alerts, could be used to perform focused scans of high-priority threats. The ScanReport(s) will detail vulnerable systems and have the potential to be easily used for automatic validation of perimeter defenses, by comparing AttackResults and Alerts since both datasets are based on IDMEF. In addition, there may be reasons for sharing scrubbed assessment information in a manner similar to the sharing of IDMEF messages via IODEF in order to gather statistics, the suggested data format would simplify this exchange of information. Finally, having a common interface or framework encourages others to invest in new or improved products, services and approaches.

Summary

By providing areas of common ground, more effective communication is possible. Various standardization efforts in the security area have been covered which have, or have the potential to, improve communication between existing security tools, products and groups. These include CVE, ICAT, OVAL, IDMEF, and IODEF. In addition an area of potential improvement in the network assessment arena has been identified and a data model based on the IDMEF/IODEF work proposed.

Acknowledgements
I would like to thank the following people who took a few minutes to respond to my inquiries including:

- Robin Cover, OASIS/xml.org
- Renaud Deraison, nessus.org
- Dale Waldt, OASIS/xml.org

References
[1] "About CVE." 18 June 2002. URL: http://www.cve.mitre.org/about/ (09 Mar. 2003).
[2] Arboi, Michel. "NASL2 reference manual." 06 Feb. 2002. URL: http://www.nessus.org/doc/nasl_doc/ (09 Mar. 2003).
[3] "Computer Security Incident Response Team (CSIRT) Frequently Asked Questions (FAQ)." 02 July 2002. URL: http://www.cert.org/csirts/csirt_faq.html (09 Mar. 2003).
[4] "CVE-Compatible Products and Services." 03 Mar. 2003. URL: http://www.cve.mitre.org/compatible/ (09 Mar. 2003).
[5] "CVE (version 20020625)." 25 June 2002. URL: http://www.cve.mitre.org/cve/downloads/full-cve.html (09 Mar. 2003).
[6] Curry, D. and Debar, H. "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition." 30 Jan. 2003. URL: http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-10.txt (09 Mar. 2003).
[7] Deraison, Renaud. "MS SMTP DoS." Revision: 1.13. URL: http://cvs.nessus.org/cgi-bin/cvsweb.cgi/~checkout~/nessus-plugins/scripts/mssmtp_dos.nasl (09 Mar. 2003).
[8] "Frequently Asked Questions." 21 Mar. 2002. URL: http://www.cve.mitre.org/about/faq.html (09 Mar. 2003).
[9] "ICAT Metabase: A CVE Based Vulnerability Database." 01 Apr. 2002. URL: http://icat.nist.gov/icat.cfm?cvename=CVE-2002-0055 (09 Mar. 2003).
[10] "ICAT Metabase: A CVE Based Vulnerability Database." 01 Apr. 2002. URL: http://icat.nist.gov/icat.cfm (09 Mar. 2003).
[11] "ICAT Metabase Documentation." URL: http://icat.nist.gov/icat_documentation.htm (09 Mar. 2003).
[12] "ICAT Text File Download Documentation." URL: http://icat.nist.gov/icat_textfile_doc.htm (09 Mar. 2003).
[13] McAlearney, Shawna. "Vulnerable Commodity." Feb. 2003. URL: http://www.infosecuritymag.com/2003/feb/news.shtml#top (09 Mar. 2003).
[14] Mann, David E. and Christey, Steven M. "Towards a Common Enumeration of Vulnerabilities." 8 Jan.1999. URL: http://www.cve.mitre.org/docs/cerias.html (09 Mar. 2003).
[15] Meijer, J. J., Danyliw, R. and Demchenko, Y. "Incident Object Description and Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition." Oct. 2002. URL: http://www.ietf.org/internet-drafts/draft-ietf-inch-iodef-00.txt (09 Mar. 2003).
[16] "Merriam-Webster OnLine." URL: http://www.webster.com/ (09 Mar. 2003).

[17] "Official OVAL Schema." 04 Nov. 2002. URL: http://oval.mitre.org/oval/schema.html (09 Mar. 2003).

[18] "Open Vulnerability Assessment Language (OVAL)." 04 Nov. 2002. URL: http://oval.mitre.org/about (09 Mar. 2003).

[19] "OVAL30." 05 Dec. 2002. URL: http://oval.mitre.org/oval/review/queries/OVAL30.html (09 Mar. 2003).

[20] Regan, Keith. "Groups Develop Granular Security Info." Feb. 2003. URL: http://www.infosecuritymag.com/2003/feb/news.shtml#8 (09 Mar. 2003).

[21] SARA™ source code. sara_info_database_csv.pl and sara_report_svc.pl. URL: http://www-arc.com/sara/downloads/sara-4.1.4b.tgz (09 Mar. 2003).

[22] "Security Auditors Research Assistant." 15 Aug. 2002. URL http://www-arc.com/sara/ (09 Mar. 2003).

[23] "The XML FAQ." v.3.01. 14 Jan. 2003. URL: http://www.ucc.ie:8080/cocoon/xmlfaq (09 Mar. 2003).