AN XML-BASED DATA MODEL FOR VULNERABILITY ASSESSMENT REPORTS

George Valvis[1] and Despina Polemi[2]
[1] *University of Pireaus, Informatics Department, Karaoli & Dimitriou 80 Pireaus 18534, Greece gvalvis@bankofgreece.gr;* [2] *University of Pireaus, Informatics Department, Karaoli & Dimitriou 80 Pireaus 18534, Greece dpolemi@unipi.gr*

Abstract: Periodic vulnerability assessment (VA), used to uncover and correct vulnerabilities, is a common intrusion prevention technique. Although the VA tools that perform those assessments, report similar information, there are tool specific differences. Unfortunately, trying to combine the output of these tools would require separate parsing tools to address the significant low-level differences. A new data model (Vulnerability Assessment Report Format - VARF) is presented in this paper in order to define data formats for sharing information of interest to VA and to facilitate the interaction with the risk management process. As a proof of concept a set of XSLT transformations was built in order to transform the results of an open source VA tool to a VARF compliant report enabling further processing of the results.

Key words: Vulnerability assessment, XML modeling

1. INTRODUCTION

Although organizations understand the consequences of being insecure, their response time to the security challenges, by finding and applying appropriate security fixes or configurations, is not prompt. With the increasing sophistication and scalability of attacks [1], such an approach will provide diminishing returns.

A major challenge when performing a vulnerability analysis of a business-critical system today is the fact that the surrounding IT infrastructure undergoes continuous changes, the risk analysis and remediation planning are based primarily on human resources and the correlation of the technical vulnerabilities and exposures to business impact is complex.

Vulnerability assessment tools are a valuable aid in this area, as they automate the process of identifying vulnerabilities. However, vulnerability assessment tools reveal a large volume of known weaknesses, the majority of which are not critical. This leaves the task to prioritize the mitigation of the vulnerabilities to the security personnel; a task that takes time to deliver, producing a long window of exposure.

Also, various vendors of vulnerability assessment tools use different terminology to explain the same issue and there is a lack of machine-readable information. Thus, although the data provided by vulnerability assessment reports are detailed, are presented in an ambiguous textual form or in a proprietary data format. The effect is that a vulnerability report has become tightly coupled to specific tool and cannot easily be shared across different tools. This lack of common ground has hampered the ability to integrate diverse sources of available security data more effectively. The result is twofold: the security personnel are overloaded with redundant data and it is not feasible to fully utilize all possibly available data in making the most accurate diagnosis. On the other hand the development of commonly agreed or standardised and extensible exchange formats will enable interoperability between commercial, open source, and research systems, allowing users to deploy their preferred systems according to their advantages in order to obtain an optimal implementation with improved response time. In general, a standardised format in vulnerability assessment will provide increased efficiency of vulnerability assessment results:

- *Extensibility* and *flexibility*: providing data in extensible and flexible format could facilitate the collaboration and simplify the integration of information generated by heterogeneous sources.
- *Broader analysis*: the co-use and combination of more than one complementary vulnerability assessment tools increases the likelihood that more reported vulnerabilities are taken into consideration.

Our goal in this paper is to propose a "common ground" in the vulnerability assessment area providing the Vulnerability Assessment Report Format (VARF) data model. VARF based on emerging standardization efforts in the security field, outputs assessment information in XML. It is designed to facilitate the extraction of meaningful, tool-independent information to address the vulnerability assessment needs. This paper has been organised as follows: In section 2 we present a condensed overview of standardization efforts on vulnerability schemes and description languages. Section 3 describes the VARF model and subsequently section 4 demonstrates some applications of the model. The last section concludes with some experience statements and a rough outlook on future directions.

2. STANDARDIZATION ACTIVITIES

This section introduces the main standardization and research efforts on XML-based vulnerability schemes and description languages.

2.1 Common Vulnerabilities and Exposures (CVE)

CVE [2] is a dictionary of information security vulnerabilities and exposures that aims to provide common names for publicly known weaknesses. The goal of CVE is to standardize the names of vulnerabilities and make it easier to share data across separate vulnerability databases and security tools.

2.2 Open Vulnerability Assessment Language (OVAL)

OVAL [3] standardizes how to test for the existence of those vulnerabilities, based on system configuration information. The vulnerabilities are identified with a CVE number by OVAL queries, which perform the checks. A synopsis section accompanies the OVAL query that should include information related to two main items: the vulnerable software and the vulnerable configuration(s). Currently individual platform-specific XML schemas have been developed. In order to use OVAL, an operator should install the OVAL definition interpreter and use the specific OVAL schema that is defined for her platform.

2.3 The Common Vulnerability Scoring System (CVSS)

The CVSS [4] is designed to provide a composite score representing the overall security and risk a vulnerability represents. Using CVSS, the security personnel will have the basis for a common language with witch to discuss vulnerability severity. It is a modular system with three distinct groups that combine the characteristics of vulnerability. Each of these qualities or "metrics" has a specific way of being measured and each group has a unique formula for combining and weighing each metric. The three groups are the base, the temporal and the environmental group. The base group contains all of the qualities that are intrinsic to any given vulnerability that does not change over time or in different environments. The temporal group contains the characteristics of vulnerabilities that are time-dependant and change as the vulnerability ages. Finally, the environmental group contains the characteristics of vulnerabilities that are tied to implementation and are specific to a user's environment.

2.4 Extensible Configuration Checklist description Format (XCCDF)

The XCCDF specification [5] defines a data model and format for storing results of benchmark compliance testing. An XCCDF document is a structured collection of security configuration rules for a specific set of target systems. The model and its XML representation are intended to be platform-independent and portable, to foster broad adoption and sharing of rules and support information interchange, document generation, environmental tailoring, compliance testing and scoring.

2.5 Open Security Organization Advisory and Notification Markup Language (ANML)

The Open Security Organization (OpenSec) [6] is developing a framework of XML-based technologies to aid system management. The technologies planned for development include:

- Advisory and Notification Markup Language (ANML) to describe security advisories and other types of notifications in consistent and machine-readable way.
- System Information Markup Language (SIML), for describing a system's properties and providing a detailed inventory of software, hardware, and configuration information that will allow management software to assess the status of a system.
- Software Description Markup Language (SDML), for describing the properties of software and its environment.

3. VULNERABILITY ASSESSMENT REPORT FORMAT DATA MODEL (VARF)

The Vulnerability Assessment Report Format data model is designed, based on the emerging standardization efforts of Intrusion Detection Working Group [7] and Incident Handling Working Group [8], to provide a standard representation of vulnerability assessment report in an unambiguous fashion. This could permit the association between reports generated by vulnerability assessment tools that address system and network level vulnerabilities.

The top-level class for all VARF messages is *VulnerabilityReport*; each type of message is a subclass of this top-level class. There are presently two types of messages defined; *Reports* and *ScanAlerts*. Within each message, subclasses of the message class are used to provide the detailed information

carried in the VARF message. It is important to note that the data model does not specify how vulnerabilities in system and network level should be classified or identified. However, once a vulnerability assessment tool has determined the type of vulnerability that exists, the model dictates how that vulnerability information could be formatted. VARF does not deal with the formatting of application level vulnerabilities (like the schemas presented in section 2).

In this section, the individual components of the VARF data model are explained in detail. Some UML diagrams of the model are provided to present how the components are related to each other, and the relevant sections of the XML DTD are presented to indicate how the model is translated into XML. Figure 1 depicts the relationship between the principal components of the data model (due to limited space occurrence indicators and attributes are omitted).

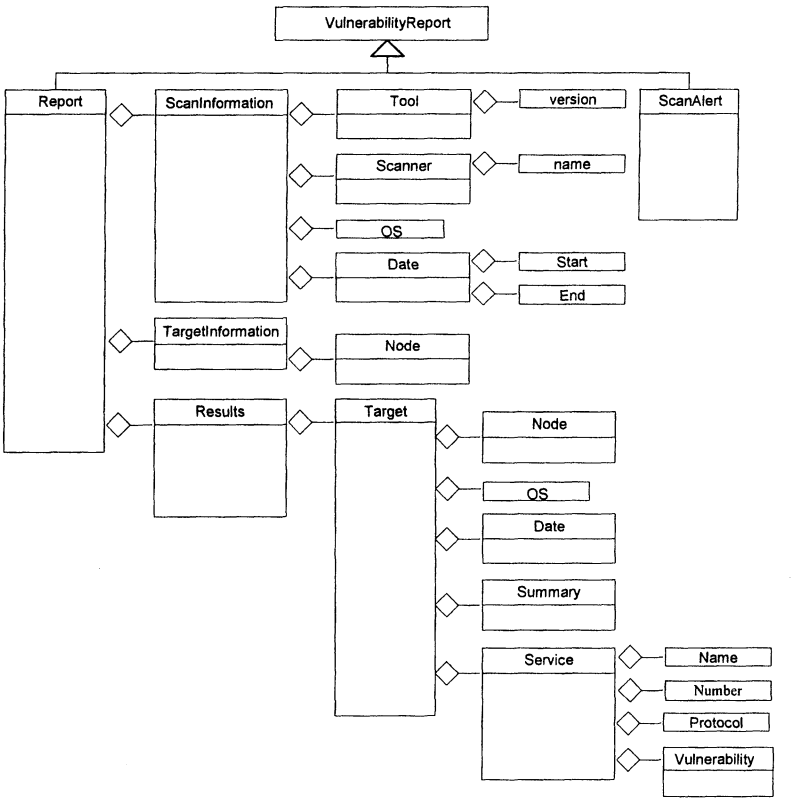

Figure 1. Data model overview.

The VulnerabilityReport Class: All VARF reports are instances of the VulnerabilityReport class; it is the top-level class of the VARF data model, as well as the VARF DTD. There are currently two types (subclasses) of *VulnerabilityReport* the *Report* and the *ScanAlert*. Because DTDs do not support sub classing, the inheritance relationship between *VulnerabilityReport* and the *Report* and *ScanAlert* subclasses have been replaced with an aggregate relationship. This is declared in the VARF DTD as follows:

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

<!ELEMENT VulnerabilityReport ( Report , ScanAlert
)>

<!ATTLIST VulnerabilityReport %attlist.varf;>
```

The *VulnerabilityReport* class has the single attribute Version, which is the version of the *VulnerabilityReport* specification this message conforms to.

Report class: The *Report* is generated every time a vulnerability assessment tool is deployed. The *Report* is composed of three aggregate classes which are:

- *ScanInformation* (exactly one), which contains identification information for the vulnerability tool that generated the report.
- *TargetInformation* (exactly one), that contains identification information for the target of evaluation.
- *Results* (exactly one) that contains all the useful assessment results.

ScanAlert class (Figure 2): The *ScanAlert* (exactly one) is modeled on the IODEF *IncidentAlert*, but it provides a different type of functionality. In the same manner, that the *IncidentAlert* is used to simply alert the occurrence of an incident and provide relevant information (such as raw IDMEF messages), the *ScanAlert* may alert an intrusion detection management system that a scan is going to be performed against the hosts specified in the *ScanAlert*. By taking this information into account, false positives could be suppressed by a correlation engine. As part of this alert, the scanner would provide *ScanInformation* and *TargetInformation*.

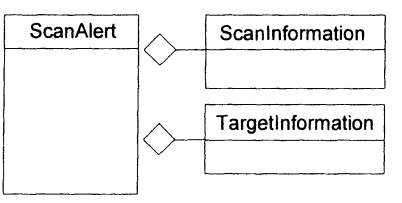

Figure 2. The ScanAlert Class.

The *ScanInformation* class carries additional information related to the conducted vulnerability assessment. The aggregate classes that make up *ScanInformation* are the *Tool*, *Scanner*, *OS* and *Date*:

- The *Tool* class (exactly one) includes the name attribute, which is the name of the tool that performed the assessment and the *Version* class (zero or one), which is the version of the deployed VA tool.
- *Scanner* class (exactly one): The aggregate class that makes up Scanner is Name, which is the host name of the machine where the VA tool is installed.
- *OS* (zero or one): The name of the OS of the host where the VA tool is installed, e.g. Linux.
- *Date*: (exactly one): Information about the time that the assessment occurred.

The aggregate classes that make up *Date* are *Start* (exactly one) and *End* (exactly one). *Start* and *End* are date time strings that identify the particular instant in time that the vulnerability assessment session was started and ended respectively.

The *TargetInformation* Class provides basic information about the assessed nodes. The aggregate class that make up *TargetInformation* class is *Node*. The *Node* class is composed by the *Name* (exactly one), which provides the host name of the target of evaluation and *Address* (exactly one), which is the IP address of the node that is the target of evaluation.

The *Results* Class: The *Results* element is meant to take the place of SARA vulnerability assessment tool [9] *Details* and Nessus [10] *Results*. It is closely tied to the IODEF *Attack* class, which in turn shares a great deal of structure with IDMEF *Alerts*. The aggregate class that makes up *Results* class is *Target* (figure 3).

Target (one or more): It includes the full assessment information discovered by the tool. The aggregate class that make up Target class are:

- *Node* (exactly one): It includes basic information about the node.
- *OS* (zero or one): It includes information about the operating system.
- *Date* (exactly one): It includes information about the time that the assessment occurred of the specific target.
- TargetSummary (zero or one): The elements that are included are the Number of services, Number of security holes, Number of security warnings, Number of security notes.
- *Service* class (zero or more), which is made up by the classes *Name* (zero or one), Number (zero or one), Protocol (exactly one), Vulnerability (zero or more).

By using the IDMEF/IODEF *Target* class, a related format for representing the 'host' specific information is achieved. This includes support for the standard types of address that Nessus and SARA vulnerability

assessment tools support. Other different types of addresses and names, as defined in the IDMEF draft could be included. 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.

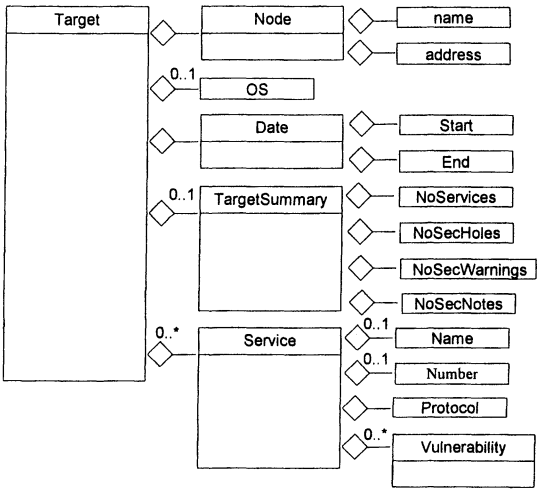

Figure 3. The Target class.

This is declared in the VARF DTD as follows:

```
<!ELEMENT Results (Target+)>
<!ELEMENT Target (
  Node, OS?, Date, Summary?, Service*)>
<!ELEMENT Service (
  Name?, Number?, Protocol, Vulnerability*)>
```

The aggregate classes that make up *Vulnerability* class (Figure 4) are *Name* (exactly one), *Family* (zero or one), *Summary* (zero or one), *Category* (zero or one), *Classification* (zero or more), *Assessment* (zero or more), *Data* (zero or one).

The *Data* element contains additional textual information related to the reported vulnerability and provides a catchall rule to accommodate items that have not been directly addressed by the data model.

The *Classification* class provides the name of vulnerability, or other information allowing the analyst to determine what it is. The purpose of the *Classification* element is to allow the analyst who receives the Vulnerability

message to be able to obtain additional information. To accommodate this, a name and an URL are included. The current list of valid source values includes "unknown", "vendor-specific", and CVE. The Classification class is composed of two aggregate classes. The aggregate classes that make up *Classification* are:

- *Name* (exactly one): The name of the vulnerability (e.g. CAN-2002-1165), from one of the origins listed below.
- *URL* (exactly one): A URL (string) at which the risk analyst may find additional information about the vulnerability.

The document pointed to by the URL may include an in-depth description of the vulnerability, appropriate countermeasures, or other information deemed relevant by the vendor.

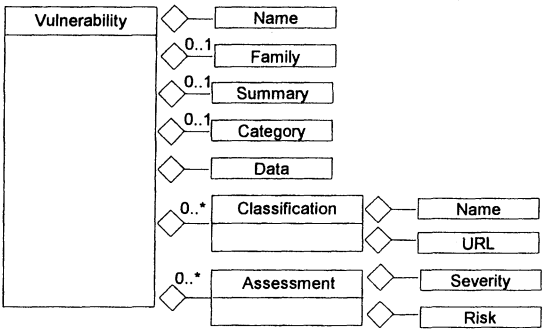

Figure 4. The Vulnerability class.

This is declared in the VARF DTD as follows.

```
<!ELEMENT Vulnerability (
  Name, Family?, Summary?, Category?, Data?,
  Classification*, Assessment*)>
```

Table 1. The values of the Origin attribute

Origin attribute	Explanation
Unknown	Origin of the name is not known
Bugtraqid	The Security Focus vulnerability database identifier [11]
CVE	The common Vulnerabilities and Exposures name
Vendor-specific	A vendor-specific name and hence, URL; it can be used to provide product specific information

The *Classification* class has the attribute *Origin* (required), which provides the source from which the name of the alert originates. The attribute's values and their explanations are given in table 1. The default value is "unknown".

The *Assessment* class provides information related to the VA tool's assessment of a discovered vulnerability, its severity and the related risk. The *Assessment* class is composed of two aggregate classes:

Severity (exactly one): The VA tool's assessment of the impact that the vulnerability might have.

Risk (exactly one): The level of risk that the discovered vulnerability poses. This is declared in the VARF DTD as follows.

```
<!ENTITY % attvals.origin "
 ( unknown | bugtraqid | cve | vendor-specific )">
<!ELEMENT Classification ( name, url )>
<!ATTLIST Classification %attvals.origin;
'unknown'>
<!ELEMENT Assessment ( Severity, Risk )>
```

By introducing the VARF model, an end user of vulnerability assessment tools would have an extended ability to process vulnerability information. The next section tries to demonstrate this enhancement.

4. PROOF OF CONCEPT

The previous section detailed an approach to represent infrastructure level vulnerability data as XLM documents. In this section we will discuss some applications that could use these representations.

4.1 Vulnerability reports

The most obvious place to implement VARF is in the data path between a vulnerability assessment tool and the database that collects all the risk analysis information. A set of scripts was built in order to transform the output of Nessus open source vulnerability assessment tool to a VARF compliant report (Figure 5). The outcome of the transformation extracts the asset information and groups together the associated vulnerabilities.

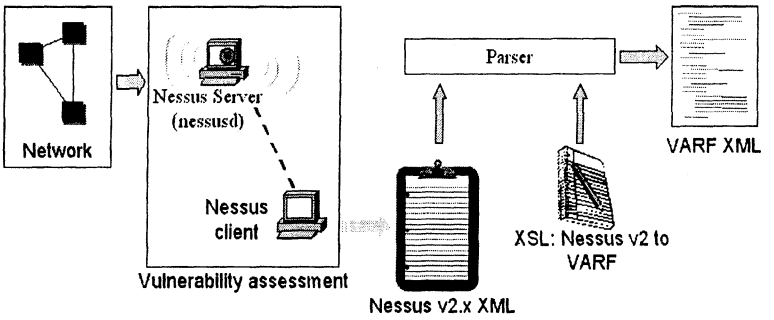

Figure 5. Nessus to VAFR tranformation.

By transforming Nessus data in VARF format, the data are better structured and further transformations to other formats (e.g. HTML) that integrate to the IT needs of an organization become feasible. This enhanced structure allows for further transformations to be delivered in order to satisfy organization needs for other data formats (for example to transform a VARF report to an HTML page is quite trivial).

4.2 Vulnerability diagrams

A visualisation of the reported vulnerabilities can be achieved by producing a tree-structured diagram. Figure 6 depicts the direct association between network assets and vulnerabilities. This diagram could increase the readability of the results and help to ensure that fewer discovered vulnerabilities go unnoticed. Figure 7 shows a process that enables us to generate vulnerability diagrams. VARF based vulnerability reports, obtained with the method explained in the previous paragraph, were feed to the Graphviz open source graph drawing software [12] that produced the diagrams in image format (GIF, JPEG, etc). Since Graphviz only accepts input complied with a particular format, the VARF based reports need to be processed by additional XSLT transformation before presented to the drawing software. Being in XML format this in not complicated.

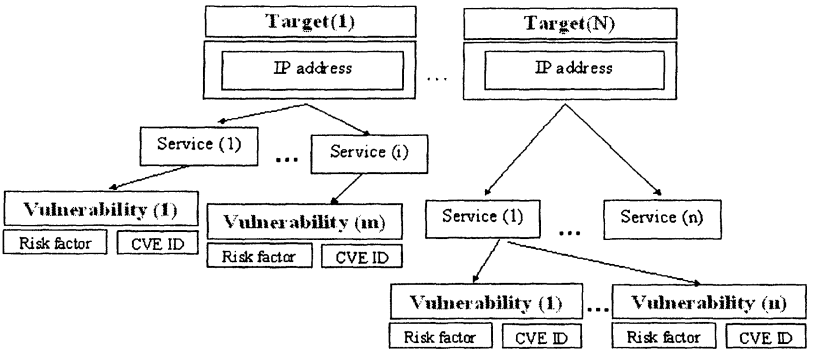

Figure 6. Vulnerability diagram.

Also, by deploying the Graphotron open source tool [13] the development of the Graphviz XSLT transformation can be further simplified.

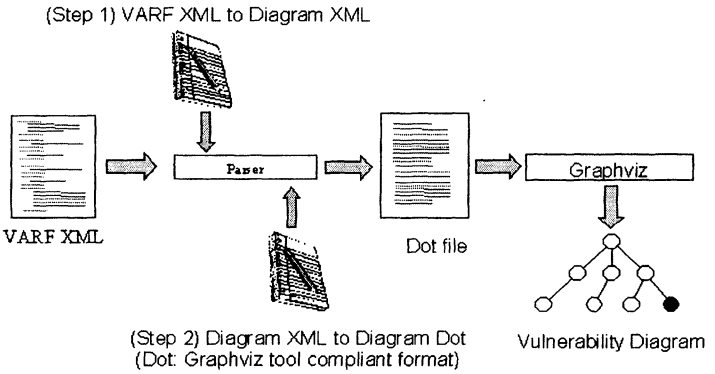

Figure 7. Producing a vulnerability diagram

We actually perform vulnerability assessment on a number of segments of our campus switched network. By processing the results using our VARF based transformations their visibility was increased and the administration task was simplified.

Other places where the VARF could be useful is an event correlation system that would accept vulnerability assessment reports and IDS alerts from a variety of vulnerability assessment and intrusion detection tools and perform

cross-correlation and cross-confirmation computations in order to provide more focused assessments of high-priority threats. The *Report(s)* will detail vulnerable systems and have the potential to be easily used for automatic validation of perimeter defenses, by comparing *Results* and IDMEF's *Alerts* since both are XML based datasets.

5. CONCLUSIONS

The increasing complexity of today's IT dependent systems urges the improvement of existing methods for analysing systems in order to increase the likelihood that all possible threats and vulnerabilities are taken into consideration. Many current tools address vulnerabilities in the context of a single host. However, due to their proprietary nature and lack of standardisation, the effectiveness of their large reports, is constrained. In order to reduce the window of exposure, the security personnel need a way to set priorities and reduce the volume of vulnerabilities generated by the tools down to the few critical risks that matters. A short overview of the standardisation efforts that focus on the information systems security was given on the section 2.

An area of potential improvement, which could enhance communication between existing security tools, products and groups, has been identified and the Vulnerability Assessment Report Format XML data model, based on the IDMEF work, was proposed in sections 3.

By introducing the VARF model, an end user of vulnerability assessment tools would have a standards based format to describe vulnerabilities that would allow sharing information easier and combining it with other data sets from a variety of compliant tools and systems. If XML is chosen as an implementation it is possible to immediately combine the assessment information with intrusion detection information (assuming that the latter is IDMEF compliant). For example, local statistics, from the intrusion alerts, could be used to perform focused assessments of high-priority threats.

In addition, if there were reasons for sharing 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. It is established that having a common format or framework encourages vendors to invest in new or improved products, services and approaches.

As proof of concept XML technologies like XPath and XSLT were deployed to process vulnerability assessment data reported by Nessus open source vulnerability assessment tool and produce vulnerability diagrams, in

order to indicate how VARF data model may contribute to the optimisation of the vulnerability management effort. We plan to scale up the vulnerability assessments on our campus network including more segments and increasing the number of the assessed hosts. Finally, it is our intention to extend the assessment operations to web services by deploying WSDL and SOAP [14].

6. REFERENCES

1. CERT Coordination Centre, Overview of Attack Trends, (June 4, 2004); http://www.cert.org/archive/pdf/attack_trends.pdf
2. D.Mann and S.Christey, Towards a Common Enumeration of Vulnerabilities, (January 8, 1999); http://www.cve.mitre.org/docs/cerias.html
3. Open Vulnerability Assessment Language XML specification page, (June 4, 2004); http://oval.mitre.org/oval/xml_specification.html
4. Common Vulnerability Scoring System (CVSS), (April 14, 2005); http://www.first.org/cvss/
5. Extensible Configuration Checklist Description Format (XCCDF) specification, (April 18, 2005); http://csrc.nist.gov/checklists/xccdf.html
6. OpenSec, The Open Security Project, (January 7, 2005); http://www.opensec.org
7. IETF Intrusion Detection Working Group, (January 7, 2005); http://www.ietf.org/html.charters/idwg-charter.html
8. Incident Object Description and Exchange Format Working Group (IODEF WG), (January 7, 2005); http://www.terena.nl/tech/task-forces/tf-csirt/iodef
9. The Security Auditor's Research Assistant (SARA), (March 7, 2005); http://www-arc.com/sara
10. Nessus vulnerability assessment tool, (March 7, 2005); http://www.nessus.org
11. SecurityFocus Bugtraq Searchable Database by Keyword. (January 12, 2004); http://www.securityfocus.com/bid/keyword
12. Graphviz, an Open source graph drawing software, (September 20, 2004); http://www.research.att.com/sw/tools/graphviz
13. Graphotron (September 20, 2004); http://www.zvon.org/ZvonSW/ZvonGraphotron
14. F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S.Weerawarana, Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI, *IEEE Internet Computing*, Vol. 6, No. 2, Mar.-Apr. 2002, pp.86-93.