A SECURITY INCIDENT DATA MODEL

Meletis A. Belsis, Nick Godwin, Leon Smalov

Data Knowledge Engineering Research Group, Coventry University, Coventry, U.K.

Tel. +44 24 76 631313, Fax: +44 24 76 888052, e-mail: Belsis@coventry.ac.uk

Abstract

This paper presents a security incident data model. The model uses many of the existing security incident taxonomies to organise its security incident related information. Where characteristic of incidents is not covered by existing taxonomies some progress has been made in developing one. The proposed structure stores both management and technical information. Indication of further development for the presented core structure has also been given.

Keywords: Security Incidents, Incident Data Model, Incident Analysis

1. ISSUES ON REPORTING AND RETRIEVAL

Experience has shown that computer security incidents can be disastrous for an organization. Analysis of past security incidents provides a valuable information [1]. The need to maintain information from these experiences in order to develop a defensive strategy is described in [2, 22].

The first step in delivering a structured security incident library is to identify common ways to organised the information store in it. Researchers propose different approaches that try to categorize security incidents, attacks and tools and some even create special languages to describe security incidents. We consider the term of an incident, as it was explained by NIST [3]. Each one of these taxonomies could be used to organise a data structure that stores security incidents but would be incomplete. Examples are:

- categories related to the technique used [4];

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35586-3_46](https://doi.org/10.1007/978-0-387-35586-3_46)

- categories related to the type of attack (i.e. Social Engineering, Denial of Service and exploits) [5];
- categories related to the type of attack (i.e. stealing passwords, backdoors and Protocol Failures) [6];
- the possible adversaries along with the potential effects are organized in a two dimensional matrix [7];
- proposal of a complete common language to describe security incidents [8];
- Approaches that are not mentioned in this paper can be found in [9-11].

Security incident libraries are important when security is needed. Currently work has been done in developing inventories of security incidents and/or attacks oriented for specific applications [23]. These models may or may not use the previous described taxonomies to organise their structure. An example of this is the Intrusion Detection Exchange Format Data Model (IDEFDM) [12]. This model is to assist intrusion detection sensors exchange information between them and the management console. The model allows its implementers to either select a classification of vulnerabilities from two recognized ones, or develop their own classifications according to their needs. Although this data model was created for intrusion detection systems, its developers provided some indication how to extend the model.

The European project S2003 proposes a simple data structure and gives guidelines how to use this structure in order to develop a comprehensive library of security incidents [13]. The proposed model can be used by European Computer Security Incident Response Teams (CSIRT) as the means of storing data collected from security incidents.

An interesting approach is the Incident Object Description and Exchange Format (IODEF) developed by the Incident Taxonomy and Description Working Group (TF-CSIRT) [14]. The model was created to assist CSIRT to exchange incident data. The model is based on the IDEFDM and so it can automatically read incident information from any intrusion detection sensor equipped with the IDEFDM model. Although the model was first created as a mechanism for information exchange it can be also used to develop a CSIRT DB (DB).

Currently a number of CSIRT exists and each one of them is using its own data model to store the reported security incidents. Examples of such are the CERT, CIAC and the CIRDB from CERIAS Laboratory. Most of the CSIRT are interested in the technical details of a security incident.

The variety of incident data models that exist makes the process of handling an incident even more difficult. CSIRTs cannot communicate easily to exchange incident information. They need to use semiautomatic ways to exchange information like faxes and emails. The form in [15] presents the incident reporting form that is used by the CERT organization as the means of recording security incidents.

Security experts need to use a number of CSIRTs to find the needed security information. In addition to the previous the existent data structures are interested in storing only the technical information coming from security incidents. This makes the use of those structures inappropriate for the management teams that are responsible to take strategic decisions on the enterprise's security requirements and security budget.

In this paper we present a conceptual data model that can be used to organize information collected from security incidents. This is the first step in the development of comprehensive common data architecture able to store detailed descriptions of computer security incidents. This new data structure will differ from existent databases in that it will include both technical and managerial data related to an incident. This will be effective for both security related personnel and corporate managers.

2. PROPOSED MODEL

Security incidents frequently comprise several *steps*. Each one of the steps may include one or more operational actors, one attack technique and one target. Every incident may have one. These elements have to be organized and grouped according to their properties. This grouping provides us with the basis for creating a comprehensive data structure.

Figure 1 shows the entities of the proposed structure and the way these are interrelated. We have decided to use an ERD diagram to represent the model at this stage due to the fact that it is widely known in the computing community. In later stages the model is going to be implemented using XML in order to provide even more interoperability and assist the information exchange between CSIRTs.

Figure 1. Entity Relational Model.

It would be ineffective to simple combine basic taxonomies into an orthogonal structure. Due to this a number of taxonomies have to be used in order to correctly categorise and organise the properties that security

incidents have. As an example of this are the Trojan Horses and Worms. Both of these tools belong to the Malicious Code category and may be used as different steps in the same incident (I.e. the worm gains access to a system and then installs a Trojan horse).

This proposal will develop a description framework that uses both taxonomies and structural ideas from previous publish work. Instead of using a single taxonomy we are going to use a combination of these to organise the security incidents stored in the proposed data model.

The following sections describe the structure of the proposed data model. Using known taxonomies to organise the incident information stored in the model will enhance the ability of sharing information among administrators and specialists due to the fact that it will provide a common terminology for important elements in the description of a security incident. In addition, these fields are going to provide effectiveness when searching the DB and support for useful statistical analyses (i.e. *what percentage of incidents involves a hacker?*).

2.1 Registered Organisations

Before discussing the data structure of storing security incidents we need to propose a structure that stores the details of the organizations that record the security incidents. This structure holds details on the name, type and contact details for every registered organization. Additional fields include a name of the security manager (if available), the type of connection the organization has, the number of hosts, the type of the local network and the operating system environment the hosts run on (Figure 2).

```
Registered_Company = (#Organization_ID (Integer), Organization_Name (String),
    Business_Type (String),
    Type_Of_service_Provided (String), Size_Of_Organization (String),
    General_Contact_Details (Memo),
    IT_Security_Manager_Contact_Details (Memo),
    Domain (string), Subnet_Masking (String), Number_Of_Subnets (String),
    URL (string), Internet_Connection_Type (String), Bandwidth (String),
    No_Of_Hosts (Integer), Type_of_Local_Network (String),
    Operating_System_Enviroment (String), Firewall_Characteristics (Memo),
    ISP (String));
```

Figure 2. Registered Organisations

Some of the fields in Figure 1 could be analysed further to produce a more detailed data-structure. For the current version it is sufficient to note the following:

- The Business type could be one of the following:
Government, Academic, Commercial, and Voluntary.
- The size of the enterprise could be one of the following: Large, Medium, and Small.

2.2 Actors

One kind of ‘thing’ to be identified when recording an incident is the set of actors that were participating in it. Identifying the actors could also help us to identify the resources (both money and time) that were used to break the system. At an initial stage in the incident, the actor may be difficult to identify. There are cases where the organizing actor is never identified or only identified after several years. For these cases we will have to use the *unknown* category until the organizing actor can be identified. At a later stage, when most of the incident details have been recorded, it would be easier to identify him/her. Every incident may have one or more actors. The main actor, the *Organizing Actor* (Figure 3), is the one that initiates and/or organizes the attack. Along with the main actor, others called *Operational Actors* (Figure 3) will be the ones that actually perform the steps that the attack includes. The operational actor could be a user or a process running on a host. In addition the incident may include a lot of compromised hosts (operational actors) that run malicious processes (i.e. TFN attack). To cover these cases the relation of operational actor and incident step must be normalized. By normalizing this relation a new table called *Operational_Actor-Incident_Step* is created (Figure 3). In cases where hosts from a specific domain are participating in an incident (i.e. Flooding attacks or stealth probing attacks) instead of recording every host as an operational actor it would be good practice to deal with the whole network as one operational actor.

```

Organising_Actor=(#Organising_Actor_Id (Integer),Actor_Type (String),
                  No_Of_Incidents (Integer), First_Time_Reported (Date),
                  Actual_Name_or_Code_Name (String));
Operational_Actor=(#Operational_Actor_ID (Integer), IP_Address (String),
                  Company_Responsible_For_Domain (String),
                  Contact_Details (Memo),Account_Name(String),
                  ComputerName (String), OS_and_ver (String));
Operational_Actors-Incident_Step = (#Operational_Actor_ID (Integer),
                                   #Incident_Code(String),Step_No (Integer),
                                   CSIRT_Involved (String),
                                   CSIRT Reference No (String));

```

Figure 3. Incident Actors

The actor can be classified as any of the following, e.g. [8]: *Hacker; User; Criminal; Government agent; Terrorist; Industry agent, and, Unknown.*

As already indicated, to represent the actors that are involved in a security incident two tables are used. The first table (Figure 3), the *Organizing Actor*, represents what is known of the actor that organized the attack. More specifically it includes fields to store an actor id, and actor type (one of the above classification), and if known, the actor's real or nick name. The

organizing actor could be one or a group of persons. This table provides the ability to track the actions that different hacking groups perform.

The data about the operational actors are stored in the table *Operational Actor* (Figure 3). This table includes fields for: an operational actor id; the IP or Network address that the actor used; the company responsible for the operational actor's domain; the user name that was used, and, the relationship to the organizing actor. Note that an attack may include one or more operational actors. Example of such an attack is a distributed denial of service (DDOS) attack.

2.3 Attack Technique

The second important aspect that must be identified is the types of attacks an incident involved. The attack methodology that has been used may include one or more types of attack. Each one of the methods that are used can be described by using the one of the following terms [1, 16]: *Malicious code; Flooding; Communication Eavesdropping; Probing and Scanning; System or Protocol Flaw Exploitation; User Deception; Impersonation, and, Miscellaneous.*

To record the techniques that adversaries employ in an incident, the *Attack Technique* table (Figure 4) is used. This table stores information specific of every attack technique. These are: a Technique Id; a Technique Name; the Attack Type (from the above list) and a technical description field which is going to hold a technical description of the specific technique. This table also provides the *Common Vulnerabilities and Exposures* (CVE) classification field. This field assists security analysts to use the CVE list as a way to provide a more widely accepted naming scheme of the attacks. The CVE_Url field specifies the place where the user could access the CVE information on the specified attack. As a further development, the DB includes a more detailed technical analysis of an attack technique. The further analysis of an attack technique is used by the security technical personnel. Security analysts are able to identify how adversaries are using the Internet protocols to perform and scale their attacks. In addition to that this information can be used as inputs to Intrusion Detection systems, firewalls and System Scanning tools. This analysis of an attack is supported by two structured tables (Figure 4). The first table the *Attack Signatures* describes low level details of every attack. It stores information on the protocol used the source and destination port, the service attacked important data that packets contain and the flags that the packet uses. The second table, *Attack Tools* stores details on the attack tool that was used. It records the tool name, the OS and version, and the author details.

```

Attack_Technique=(#Attack_Code (String), Attack_Name (String),
                  CVE_classification (String), CVE_Url (String), Attack_Type (String),
                  Technical_Description (Memo), Defence_Available (Boolean),
                  Defence_Code (String));
Attack_Signature=(#Attack_Code (Integer), Protocols (String), SRC_ports (String),
                  Dest_Ports (String), Service_Names (String),
                  Packet_Contains (Memo),
                  Flags (String), Packet_Size (number));
Attack_Tool = (#Attack_Tool_Id (Integer), Tool_Name (String), Tool_Description (Memo),
              OS_and_ver (String), Freeware (Boolean), Author_Details (Memo),
              URL (String));

```

Figure 4. Attack Details

2.4 Actor's Aim

Every security incident, no matter how many steps it contains, can be characterized by an overall aim. This aim states the motive that prompts the incident. Subsequent actors (the actors that perform the steps of the incident) may have a different aim but this model is interested only in the aim that the initiator of the incident has. The overall aim of the incident can be identified either by investigating the results of a successful attack or by investigating system and data files after the attack has been prevented. The aim can be described as being in one of the following categories. The terminology used here is widely used by the computer security experts, a similar list can be found in [17]: *Information Stealing; Destruction of System Integrity; Destruction of Public Image; Service Disruption; Network/Host misuse; Trespass*, and, *Accidental*.

The previous data about the incident can be represented in a table called *Aims* (Figure 5). There the DB user selects one of the previous categories as the aim of the incident and has the option of providing an additional free-text explanation. This helps the DB provide a better view of the aims of those that cause security incidents. The table includes three fields: the aim ID; the aim type (one of the above categories), and, additional information on the diagnosed aims of the specific incident.

```

Aim= (#Aim_ID (Integer), Aim_Type (String), Aim (Memo),
      Aim_Succeeded (Boolean));

```

Figure 5. Aim

2.5 Targets.

In every step of an attack the adversary(s) can target a different object in the Network and/or the System. Examples of such are, the WWW Server, and, the Network Router. For the incident model the targets are the parts of a network that can be attacked in order to fulfil the aims of the attack. The

analysis of security incidents and current Internet technologies shows that potential targets of an attack can belong in any of the following categories: *Public enterprise server; private enterprise server; Network Host; Network bandwidth; Data/Configuration/System Files; Network Traffic*, and, *Miscellaneous*.

Each of these categories of network and/or system elements is a potential target of break-in. Instances of any one of them could give access for further attacks in the internal network. Experience has shown that it is not unusual for attackers to break into a Web Server and from there exploit trust relations to gain access to other hosts in the network.

To store the details of the parts of a system that were targeted a separate table called *Target* (Figure 6) is used. It organizes the records using a unique record id and the previously described categories. For each one of the targets that are involved in an incident this table stores information on: *the type of target; the name of the target (i.e. object class); the company that developed it; the version of the target, the IP or network address of the target and additional information on the target*. For system files, like password files, the ‘company that developed the object’ is considered to be the company that created the specific operating system.

```
Target= (#Target_Id (Integer), Asset_Type (String),
        Object_Class (String), Object_Developer (String),
        Object_Name (String), Object_Version (String), Object_IP (String)
        Additional_Information (Memo));
```

Figure 6. Targets

2.6 Incident Detection

The method that was used to detect an incident is vital to the overall incident handling process. Recording the way the incident was detected as well as providing a copy of the actual detection record can assist professionals with setting up their intrusion detection tools. The incident detection methods that exists can be organised using the following list: *Anomaly Detection; Signature Based Detection; Attacker Deception (Honey pots)*, and, *General*. The techniques belonging to these categories could be applied in automatic and/or manual ways. Security analysts can investigate the network traffic or can apply tools that automatically inspect traffic.

To organize the description of the detection method the *Incident Detection* table (Figure 7) is used as the means of recording the details of the way the incident was detected. The fields for this table include: Incident Detection Id, Tool Type for the tool that detected the incident, tool name, OS and the version, the actual log record that the tool produced, the time and date stamps, and, some additional information.


```
Incident_Detection=(# Incident_Detection_ID (Integer), Tool_Type (String),
    Tool_Name (String), OS_and_ver (String), Log_Record (Memo),
    Additional_Info (Memo), Time_Date_Of_Log (DateTime));
```

Figure 7. Incident Detection

2.7 Incident Control

To record a security incident one has to identify the different steps that the actors took in pursuit of their aim. Some general information has to be stored such as, who reported the incident first and the time of the incident.

The model stores some high level details of the incident in the *Security Incidents* table (Figure 8). In addition, the time the company needed to recover from the incident and the estimate economic damage that the incident had. To record details about the incidents another table, called *Incident Steps* (Figure 8) is used. This table is related to the *Security Incidents* table through an *Incident Code*. One should note that every security incident may include one or more steps. For example: “The hacker exploits a vulnerability to gain access to the Web server and from there it uses a packet sniffer to capture passwords and log in to the File Server.” The table stores details of every step that an incident includes.

```
Security_Incident=(#Incident_Code (String), Organization_ID (Integer),
    Aim_ID (Integer), Time_To_Recover (Time),
    Risk_Type (String),Organizing_Actor_Id (Integer),
    Estimate_Damage (Currency),Additional_Info (Memo),
    Date_and_Time (DateTime), Date_CSIRT_Notified (Date),
    Incident_Record_Complete (Boolean));
Incident_Step=(#Incident_Code (String),#Step_No (Integer),Target_Id (Integer),
    Time_Started (Time), Time_Lastest (Time),
    Incident_detection_ID (Integer),Time_Zone (String),
    Source_Address_Spoofed (Boolean),Attack_Code (String),
    Attack_ToolID (Integer));
```

Figure 8. Security Incidents

To show the currently available security countermeasures related to the incidents recorded in the DB additional tables are included. This requires a structured description of such measures. An example could be found in [18]. The widely used terminology from the computer security literature is used. More specifically the security countermeasures can be organized into the following: *Firewalls; Secure Communications; Secure Storage; Antivirus; Software patches/ updates; User and Software Management; Security Policy; Load Balancing; Antisniffing; IDS, and, System Penetration testing.* Most of the existing CSIRTs are concentrated in providing advice on software patches that need to be installed. In this structure the ability to

provide advice on a variety of security software or processes is given (i.e. security policy).

The *Defence Advice* table (*Figure 9*) stores the descriptions of the methods that are appropriate to defend the system against a specified attack technique. The security methods are categorised using the titles described earlier. This table includes a memo field that is used as the means of performing a detailed description on how to apply the method (i.e. the firewall must block all incoming ICM packets, the password should be longer than eight characters). The *Problems* field describes possible drawbacks that the specified protection method may bring to the system. To enhance support another table called *Defence Tools* (*Figure 9*) is used. This table holds the details of the hardware and/or the software that can help implement the proposed defence advice. Every piece of advice may or may not have one or more software/hardware associated with. Except from the typical fields that describe the product the *Certification* fields assist users to select the appropriate product depending on the security requirements their system has.

```
Defence_Advice= (#Defence_Code (String),Advice_Type (String)
                Advice (Memo), Problems (Memo), Time_Needed_To_Apply (Time),
                Defence_Tool_ID (Integer));
Defence_Tool= (#Defence_Tool_Id (Integer),Tool_Type (String),
               Version (string),OS (String), Tool_Name (String),
               Tool_Description (Memo),
               Company_Produces (String), Contact_Details (Memo),
               Certification_Type (String),Certification_Level (String),
               URL (String));
```

Figure 9. Defence Strategy

3. DATABASE EVALUATION

To evaluate the usability and the efficiency of the proposed designed; a prototype implementation has been created. The prototype database has been used to record invented security incidents. For the purposes of the evaluation, the prototype DB includes also a simple graphical user interface that was used to access and record data in the tables. This implementation has been given to an appropriate department for testing and evaluation.

Due to the lack of space it is impossible to describe all incident examples that were used to test the proposed model. The following example illustrates the mapping of a security incident to the fields of the proposed structure. This sample incident has been taken from [12]. Some “fictional” data have been added: *The network administrator of a commercial marketing company (AX Ltd) discovered that one of the routers was down. He discovered that by noticing that the second router was slowing down. So he run the tcpdump program and discovered that all of the hosts were using the second router although the network had two routers. He checked the log files of the first*

router (123.234.1.21 Cisco.router.b10) to discover that someone with source address 222.121.111.112 send an ICMP packet with size 66,535 at 12/02/1999 10:01:25.93464. The domain belonged to a University. He contacted the admin services of the university and after collaboration they discovered that the ICMP message came probably from a hacked account with account name sxxx01 in a SuSe ver. 7.1 Linux Host. The university administrator discovered a program similar to the ping with the name hackping in the account. The damage to the network was small due to the fact that employees of AX Ltd could still use the network but it still took 30 minutes for the first router to recover.

Figure 10 shows an instance of the model which includes the details of the previously described incident. Each one of the boxes represents one table of the proposed structure. The data inside the boxes represent the fields of each table and the value that these have. The symbol --/-- denotes that additional details are omitted for the purpose of this paper only. As it can be seen, the *Organising Actor* is unknown in the above example and so in this specific example is empty.

Figure 10. Evaluation

4. CONCLUSIONS

This paper has described the core structure of a DB that can be used for several purposes in a security context. The construction of the DB and the population of it have provided a useful focus for the discussion of a number of aspects of the use of security data.

The core structure in itself is a useful base for general queries about security incidents. Organization managers are not interested in the detailed technical characteristics of the attack. They are mostly interested in the damage that an incident can inflict on the business and the cost of securing against that. The population of the structure presented in this paper with data from some of the existing lists of incidents would help in this respect. Without reading all the details of all the incidents a few queries relevant to the current management issue put to the proposed structure would focus on the most useful data.

Deciding on a common agreed data structure is not a complete solution. The common structure will solve only some of the problems that CSIRTs have. Other issues are concern with the way this structure will be used. In particular the way data are acquired, protected and retrieved.

Due to the nature of the data stored inside, such DB can be also an invaluable tool to all sorts of hackers and criminals. Adversaries will be able to identify the hardware and software that a company uses. The DB could be used as an on line hacking tutorial to advice adversaries on how to break into specific systems. The problem becomes more apparent when the DB is going to be accessed via the Web. In order to trust the DB, there must be security procedures in place to protect data stored in it. There must be a security infrastructure that is capable of providing confidentiality, integrity, availability, authentication and access control. There are a number of ways to provide this functionality by using digital certificates, encryption, and security protocols.

Incident reporting, processing and retrieval are vital for the correct operation of the CSIRT. Security incidents may include large amount of attributes. It is extremely difficult for the administration to be able to know all the attributes of an incident immediately after it occurred. An administrator must have the ability to store part of the incident and complete the rest records afterwards. The best way will be to use the web based interface. Email or Fax forms although sometimes more secure usually discourage the use of the DB. On the other hand it would be confusing to display a huge amount of information on the screen when data incidents are retrieved. The software behind the DB must create views of the data targeting at different people. Examples views could include: *Management view* and *Technical view*.

To provide friendly access a Natural Language Interface DB (NLIDB) system e.g. [19, 20] could potentially be used. These systems provide the ability to use regular English expressions to search the DB instead of SQL queries. The system is responsible to translate the English expression into an appropriate SQL statement and to format the search output into a given form (i.e. How many incidents involve attacking an Apache server?)

Initial designs for the MS SQL Server and MS English Query have been constructed and the above examples have been shown to be relevant to the proposed DB structure. The system will be based on the Web. By using secure HTML pages users will be able to access the NLIDB/SQL server which is going to hold the DB. Another method will be to use the CORBA model and allow users to download CORBA applets through the Web. More detailed description of this approach can be found in [21].

There are a number of aspects that can be refined further. For day-to-day management the security personnel would need a reference on the new attack trends and the defence measures. Such a refined data model would assist also in the provision of statistical data. To provide for the design of security systems the data model must be extended to allow a model of the current system to be stored. The *Target* table must be refined and some estimates of value of assets must be represented. Of course the existence of such a representation itself poses a security risk and emphasizes the need to further consider at some stage the security aspects of the deployment of a DB based on the proposed structure.

5. REFERENCES

- [1] Power R. *2000 CSI/ FBI Computer Crime and Security Survey*. Computer Security Issues and Trends. Vol. VI, No. 1. Spring 2000
- [2] Anderson R. *Why Cryptosystems fail*. Technical Report. University Laboratory, Cambridge. January 1994
- [3] NIST 800-3. *Establishing a computer security incident response capability*. November 1991
- [4] Icove D., Seger K. and VonStorch W. *Computer Crime: A Crimefighter's Handbook*, O'Reilly Inc., 1995
- [5] Ranum M. J. *Taxonomy of Internet Attacks*. Presentation. 1995
- [6] Cheswick W., Bellovin S. *Firewalls and Internet Security*. Addison and Wesley. 1994
- [7] Perry T., Wallich P. *Can Computer Crime Be Stopped?* *IEEE Spectrum*, Vol.21, No. 5, pp. 34-39, 1984
- [8] Howard J.D., Longstaff T. A. *A Common Language for Computer Security Incidents*. Sandia National Laboratories. 1998
- [9] Aslam T., Krsul I., and Spafford E.H. *Use of Taxonomy of Security Faults*, TR96-051, Dept. of Computer Science, Perdue University, 1996
- [10] Debar H., Dacier M., and Wespi A. *Towards a Taxonomy of Intrusion-detection Systems*. *Computer Networks* 31, p.805-22, 1999
- [11] Landwehr C.E., Bull A.R., McDermott J.P. and Choi W.S. *A Taxonomy of Computer Security Flaws*, *ACM Computing Surveys*, Vol.26, No.3, pp.211-254, 1994.
- [12] IDWG. *Intrusion Detection Exchange Format Data Model (IDEFDM)*. 15 June 2000

- [13] Commission of the European Communities Security Investigations Projects. Project S2003-Incident Reporting a European Structure "Final Feasibility and Strategy Report". Report No19733. Version 1.0. 1992
- [14] Demchenko Y. *Incident Object Description and Exchange Format Data Model and Extensible Markup Language (XML)*. Internet Draft. July 2001
- [15] CERT Coordination Centre. *Incident Reporting Form*. CERT 2000
- [16] Brandi W. *Security Attacks and Techniques employed by Hackers*. 1999
- [17] Cohen F.B. *Information System Attacks: A preliminary Classification Scheme*. Computers and Security. Vol.16, No.1, pp.29-46, 1997.
- [18] Olovsson T. *A structure Approach to Computer Security*. Chalmers University of Technology. Sweden. Technical Report No 122. 1992
- [19] Androutsopoulos I., Ritchie G.D., and Thanisch P. *Natural Language Interfaces to DBs - An Introduction*. Natural Language Engineering, vol. 1, part 1. Cambridge University Press, pp.29-81, 1995.
- [20] Ott N. *Aspects of the automatic Generation of SQL Statements in a Natural Language Query Interface*. Information Systems, 17(2), pp.147-159, 1992
- [21] Belsis M., Godwin N., and Smalov L. *Delivering Secure Manufacturing IT Systems within the CORBA Security Framework*. Proc. of XIV International Conference on Systems Engineering (ICSE), pp.67-72, 2000
- [22] Landwehr C., et al., *A Taxonomy of Computer Program Security Flaws with Examples*, US Naval Research Laboratory, NRL/FR/5542-93-9591, November 19, 1993
- [23] Gritzalis D., et al., *A classification of health information systems security flaws*, in Proc. of the 11th International Information Security Conference (SEC '95), Eloff J., von Solms S. (Eds.), pp. 453-464, Chapman & Hall 1995.