Designing IDLE: The Intrusion Data Library Enterprise*

Ulf Lindqvist[1,3] Douglas Moran[2]
Phillip A. Porras[3] Mabry Tyson[2]

[1] Dept. of Computer Engineering
Chalmers University of Technology
Göteborg, Sweden
e-mail: `ulfl@ce.chalmers.se`

[2] Artificial Intelligence Center
SRI International
Menlo Park, California
{`moran, tyson`}`@ai.sri.com`

[3] Computer Science Laboratory
SRI International
Menlo Park, California
{`ulf, porras`}`@csl.sri.com`

Abstract

High-quality, timely information on intrusions is crucial in the development, testing, tuning, and updating of intrusion detection systems (IDSs) and intrusion recovery systems. We present the Intrusion Data Library Enterprise (IDLE), a design and initial compilation of an extensible library of intrusion data that is efficiently parseable in both human-readable and platform-independent machine-readable forms. We are currently in the early stages of design and are building example entries. The IDLE library will be made available as a resource specifically for the intrusion detection community. IDLE will provide IDS developers and users with accurate field data for testing and tuning, and as new intrusion types are discovered, it will enable tools to automatically update rule-sets and parameters.

*Abstract presented at the UC Davis Intrusion Detection and Response Data Sharing Workshop, July 15, 1998, Davis, California and at RAID '98 (First International Workshop on the Recent Advances in Intrusion Detection), September 14–15, 1998, Louvain-la-Neuve, Belgium. The full paper was presented at NORDSEC '98 (Third Nordic Workshop on Secure IT Systems), November 5–6, 1998, Trondheim, Norway. Parts of this work were funded by The Swedish National Board for Industrial and Technical Development (NUTEK) under project P10435-1.

1 Background and motivation

As developers of tools for intrusion detection and diagnosis, we have identified a need for data on intrusions. Typically, we would like to have a collection of reliable, detailed records that include exploit instructions and data on vulnerable system configurations to make the intrusions repeatable in a lab environment. We would also like to be able to add various types of information, such as what signs of the intrusion we are able to observe in the system. Legitimate concerns about distributing information on intrusion schemes has hampered the growth of such databases. Ironically, this has led to the current situation where security professionals find their best sources for intrusion data on underground Internet sites.

Our experience from detailed intrusion analysis indicates that a vast amount of information is needed about the particular intrusion sample to make correct statements about an intrusion type in terms of prerequisites, impact, traces, difficulty, remedies etc. Also, because systems are continually patched to block known intrusions, it can be difficult to recreate a vulnerable system configuration for each intrusion sample when detailed vulnerability information is missing.

Typically, intrusion handling systems need to be updated when new types of intrusions are discovered. A major problem facing intrusion detection system (IDS) developers is that the intrusion databases utilized provide little leverage for automating extensions to their systems. In many systems it is assumed that the users will collect intrusion descriptions from various sources and write new rules or change parameters as new intrusions are discovered. This could be compared to modern virus detection systems, which often have automatic network-based update functionality.

As a solution to this problem, we present the Intrusion Data Library Enterprise (IDLE) which is an effort to create a standard format for describing vulnerability and exploit information for IDS purposes.

2 Related work

Most of the previously presented databases for system vulnerabilities and/or exploits fall into one of two different categories, each with their own shortcomings, respectively:

- Databases that are kept internal to an organization.

  Problems: People outside the owner organization do not know about the databases, their contents, or their structure. Sharing is discouraged or even prohibited, motivated by liability issues or the simple fact that organizations do not want to reveal how their systems could be attacked.

- Publicly available databases from "underground" sources.

  Problems: Data on such sites is often incomplete, of varying reliability, and certainly not intended for intrusion detection.

To our knowledge, the only previously published work on vulnerability database structure that talks about storing signatures for intrusion detection is that by performed by Krsul at Purdue University [3]. However, the main purpose of Krsul's database is the same as that of the often-cited Landwehr taxonomy [4], namely, vulnerability analysis and prevention rather than intrusion detection.

Vulnerability prevention was also the original goal of the vulnerability database project managed by Matt Bishop at University of California at Davis, but it has come to include intrusion signatures as well [2]. We currently seek to coordinate the efforts in IDLE with the Davis project.

At Chalmers, we have previously worked on categorization of many aspects of vulnerabilities and intrusions, including cause [9, 5], method and result [6], risk [7], traces [1], and remedy [8]. That work will serve as a foundation for the IDLE structure.

3 IDLE: A standard format for data sharing

In IDLE, we attempt to create a standard format that will facilitate rapid distribution of information among IDS developers and related groups in order to achieve "critical mass" in the coverage. Although the breadth of such exchange and the access controls are outside the scope of this work, we would like to point out that IDLE supports fine-grained data filtering that can be used to implement various policies for data sharing and sanitization.

3.1 IDLE design highlights

The emphasis of our design is to include information that ensures automated repeatability, detection, and diagnosis of each intrusion sample. What makes IDLE different from current intrusion databases is that it is designed to serve the IDS community by coordinating detailed information on vulnerable configurations and exploit instructions with documented observable dynamic and static traces (signs) of the intrusion type. The IDLE trace information is structured in a form that will support an IDS downloading a new description and extracting the information needed to automatically generate new rules (signatures, parameters, etc.) to identify the new intrusion.

IDLE is designed to store technical data about intrusion *types*, where the primary distinction between two types is that their observable traces differ in a significant way. Another distinction between types is information that concern the vulnerable configurations, for example, operating system, applications etc. However, IDLE is not meant to be a database of intrusion *incidents* where evidence concerning attack cases should be stored. Such a database would have other concerns and goals, but could use IDLE for the technical description of the types of intrusions occurring in the recorded incidents.

Support for partial information is a core part of the design of IDLE. Information about an intrusion type will typically be initially incomplete, and different groups may tend to populate only chosen subsets of a record. Incremental population is especially important in the observables, because different developers monitor system activity from different perspectives: network traffic, audit logs, application logs, filesystem traces, etc. IDLE must also be easily extensible to support the aspects relevant to new and different target platforms, tools, and IDSs. In summary, there are three aspects of incremental population:

- New entries (intrusion types) in the repository
- New data for certain fields of an existing entry
- New types of fields for new and old entries

It should also be noted that published records may be incomplete due to sharing policies requiring certain parts of the database to be suppressed.

3.2 Implementation

We have chosen to use the Extensible Markup Language (XML)[10], successor to HTML, for the intrusion database. There is an intense ongoing development of tools for authoring, displaying, browsing and handling XML documents, and we expect substantial leverage from this activity. XML provides a number of features such as platform-independence, naturally hierarchical structure, customizable field display filtering, the possibility to mix human-readable free-text fields and machine-readable fields in the same record, and easy addition of new types of fields. Those features enable IDLE

Identification	Title			
	Related item *			
	RCS			
	Alias *			
Summary	Description			
Severity	Impact	Immediate		
		Potential		
	Difficulty			
Preconditions	Privileges			
	Affected platforms	Confirmed vulnerable ?	Platform description +	OS name
		Confirmed not vulnerable ?		OS rev
		Suspected vulnerable ?		App name +
		Suspected not vulnerable ?		App rev +
		Unknown vulnerable status ?		Pf details
Exploit +	Local files	URL +		
	Sources	URL +		
	Procedure	Step +	Description	
			Command	
			Result	
			Observables	Network +
				Audit trail +
				App logs +
				Filesystem +
Solution +	Procedure	Step +	Description	
			Command	
Further info ?				

Table 1: Intrusion element hierarchy.

to evolve both in terms of the content of individual records and of the structure of the library. This makes us confident that for IDLE, XML is a far better choice than an existing proprietary database format.

We will now give some examples of what the first draft structure of IDLE looks like, both in terms of element hierarchy and some pieces of XML code examples. Starting with the element hierarchy in Table 1, we have borrowed a few conventions from XML Document Type Definitions (DTDs). A plus sign (+) after an element means that it may be repeated more than once but must occur at least once. A question mark (?) means that the element is optional but may occur at most once, while an asterisk (*) means optional but may occur multiple times.

Table 1 does not show the entire depth of the structure, but that is illustrated in Figure 1 which shows the XML code for a platform description. Such an element belongs below preconditions, affected platforms and any of the vulnerability status tags as shown in Table 1. Figure 2 shows a severity element and illustrates how information can be conveyed with so-called empty tags, of which the `<LOCALROOT/>` and `<READYEXPLOIT/>` constructions are examples. Empty tags can be parsed and interpreted by a program, while the free text is more suited for human readers.

```
<PLATFORM_DESC><OSNAME>Solaris</OSNAME> <OSREV>2.6</OSREV>
  <PFDETAILS>
    <SYSINFO command="uname -a">
      SunOS machine 5.6 Generic sun4m sparc SUNW,SPARCstation-5
    </SYSINFO>
    <PATCHES command="showrev -p">No patches are installed</PATCHES>
    <FILES>
       <FSENTRY>
          <PATHNAME><DIRNAME>/usr/bin/</DIRNAME>
          <FILENAME>eject</FILENAME></PATHNAME>
          <CHECKSUM algorithm="MD5">
            75fe78587c3b465c60ac28c7cb966906
          </CHECKSUM>
          <FILESIZE unit="bytes">13144</FILESIZE>
          <FILEPERMISSIONS>-r-sr-xr-x</FILEPERMISSIONS>
          <FILEOWNER uid="0">root</FILEOWNER>
          <FILEGROUP gid="2">bin</FILEGROUP>
       </FSENTRY>
    </FILES>
  </PFDETAILS>
</PLATFORM_DESC>
```

Figure 1: An example of the platform description element in XML.

```
  <SEVERITY>
    <IMPACT>
       <IMMEDIATE><LOCALROOT/>
         When the exploit program is executed, an interactive shell
         with effective userid 0 is started on the attacker's terminal.
       </IMMEDIATE>
       <POTENTIAL><LOCALROOT/>
       </POTENTIAL>
    </IMPACT>
    <DIFFICULTY><READYEXPLOIT/></DIFFICULTY>
  </SEVERITY>
```

Figure 2: An example of the severity element in XML.

3.3 Current status

Currently, we are populating the database with example entries to evaluate and refine the first draft structure. For presentation of the database contents, automatic generation of HTML controlled by mapping rules written in the Extensible Style Language (XSL) has been implemented. This conversion illustrates how data stored in XML easily can be filtered, split, hyperlinked and interpreted.

4 Open issues and future work

IDLE is based on results the authors' previous research on intrusion analysis, detection and diagnosis. However, there are still issues in the IDLE design that will require further research, for example:

- How should observable traces be represented to efficiently serve different needs and tools?

There are also other problems of a political rather than technical kind:

- Should there be a central IDLE data repository? If so:
  - Who can be trusted to handle such a repository?
  - Who has the resources to maintain the repository?
  - What organization is willing to expose itself to the risks of publishing high-quality exploit information, considering the possible legal and public relations problems?
  - What restrictions on access to the data should there be?

We hope that the benefits for the intrusion detection community of having a resource such as IDLE will motivate collaborative efforts to solve the remaining problems.

5 Conclusion

We have presented IDLE which is an effort to create a common format for describing intrusion data for the IDS community, something that is clearly needed and in great demand. We have shown how we can use XML to get built-in support for partial information and incremental population while allowing both human-readable and machine-readable fields to be stored in a platform-independent format. Because there are difficult political problems with a repository, we are currently concentrated on developing the format rather than populating the database.

References

[1] S. Axelsson, U. Lindqvist, U. Gustafson, and E. Jonsson. An approach to UNIX security logging. In *Proceedings of the 21st National Information Systems Security Conference*, pages 62–75, Arlington, Virginia, Oct. 5–8, 1998. National Institute of Standards and Technology/National Computer Security Center.

[2] M. Bishop. The UC Davis vulnerabilities project, Aug. 26, 1998. http://seclab.cs.ucdavis.edu/projects/vulnerabilities/.

[3] I. V. Krsul. *Software Vulnerability Analysis.* PhD thesis, Purdue University, West Lafayette, Indiana, May 1998.

[4] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3):211–254, Sept. 1994.

[5] U. Lindqvist, U. Gustafson, and E. Jonsson. Analysis of selected computer security intrusions: In search of the vulnerability. Technical Report 275, Department of Computer Engineering, Chalmers University of Technology, Göteborg, Sweden, 1996. Presented at NORDSEC – Nordic Workshop on Secure Computer Systems, Göteborg, Sweden, Nov. 7–8, 1996.

[6] U. Lindqvist and E. Jonsson. How to systematically classify computer security intrusions. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 154–163, Oakland, California, May 4–7, 1997. IEEE Computer Society Press, Los Alamitos, California.

[7] U. Lindqvist and E. Jonsson. A map of security risks associated with using COTS. *Computer*, 31(6):60–66, June 1998.

[8] U. Lindqvist, P. Kaijser, and E. Jonsson. The remedy dimension of vulnerability analysis. In *Proceedings of the 21st National Information Systems Security Conference*, pages 91–98, Arlington, Virginia, Oct. 5–8, 1998. National Institute of Standards and Technology/National Computer Security Center.

[9] U. Lindqvist, T. Olovsson, and E. Jonsson. An analysis of a secure system based on trusted components. In *Proceedings of the Eleventh Annual Conference on Computer Assurance (COMPASS '96)*, pages 213–223, Gaithersburg, Maryland, June 17–21, 1996. IEEE, Piscataway, New Jersey.

[10] The World Wide Web Consortium. *Extensible Markup Language (XML) 1.0*, Feb. 10, 1998. http://www.w3.org/TR/REC-xml.