The Insider Threat Prediction and Specification Language

G.Magklaras[1] and S.M.Furnell[1,2]

[1]Center for Security, Communications and Network Research,
Plymouth University, Plymouth, UK
[2]School of Computer & Security Science, Edith Cowan University, Australia
e-mail: info@cscan.org

Abstract

Various information security surveys and case studies indicate the importance and manifestation of the insider threat problem. One of the most important tools to address insider threats is to enable the researchers to build case studies and express/replay threat scenarios. The Insider Threat Prediction and Specification Language (ITPSL) is a Domain Specific Language (DSL) created to provide a systemic way to describe insider threats and misuse incidents. This paper presents the scope of creation as well as the design philosophy of the language. An early language compiler prototype and its underlying insider threat monitoring framework are presented followed by an evaluation of the language against real world insider threat scenarios. The paper concludes with a brief discussion of the future trends in insider threat monitoring and specification.

Keywords

Insider misuse, insider threat specification, logging engine, Domain Specific Languages, insider threat signature

1. Insider Threat and its specification

Information Technology (IT) security threats concern every component of the modern computing infrastructure world. Pfleeger et al., (2003) defines the term threat in an IT infrastructure context as “a set of circumstances that has the potential to cause loss or harm”. These circumstances might involve human-initiated actions (intentional IT intrusions), flaws in the design of the computer system and environment factors (natural disasters). In the Information Security literature, the term “insider” has been defined by means of highlighting different parts of the problem. However, it always refers to legitimate users, people that are trusted to access the IT infrastructure. Trust is a key issue and a good general definition of an “insider” that emphasizes this is the following Probst et al., (2009): ”An insider is a person that has been legitimately empowered with the right to access, represent, or decide about one or more assets of the organization's structure”. This definition gives a wide perspective.

Insider IT misuse threats have been documented in recent information security surveys. The ISBS 2010 (PwC, 2010) and the CSI 2010 (Richardson, 2010) are two examples that shed light in different parts of the insider misuse threat problems.

Neither the information security surveys nor the stories in the press can provide a clear picture of the mechanism with which the problem manifests itself in IT infrastructures. This clear picture should ideally display a mechanism that shows how a threat is realized into a misuse act. This is the field of Insider Threat Specification, the process of using a standardized vocabulary to describe in an abstract way how the aspects and behavior of an insider relate to a security policy (Caelli et al., 1991) based misuse scenario.

Personality, organizational role, financial status and access credentials are some examples of insider aspects. In contrast, the insider behavior refers to the actions of an individual for accessing, representing or deciding about organizational assets.

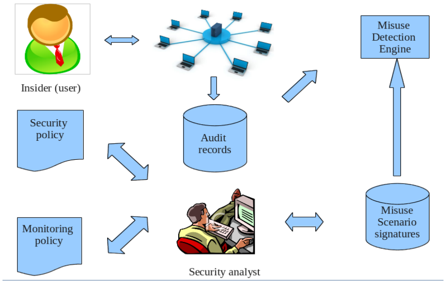


Figure 1: Misuse detection information flow

A security policy defined misuse scenario implies the existence of a monitoring policy. The security policy (Caelli et al., 1991) defines in plain language the borders between acceptable and unacceptable usage of IT resources. However, this plain language description must then be converted into suitable monitoring statements. Bace, (2001) discusses the difference between a security and a monitoring policy. Figure 1 illustrates the misuse detection information flow. The actions of an insider are monitored using a tailored logging engine (Magklaras et al., 2011) and generate audit records. The security analyst will consult the security policy and generate a suitable monitoring policy. The next step is to construct Misuse Scenario signatures expressing various insider threats. The misuse signatures are constructed so they describe certain monitored events and on the basis of the collected audit records, certain misuse incidents or insider threats (series of events that are likely to generate misuse incidents) can be detected.

2. The scope of ITPSL

The previous section defined the term threat and how it should be specified. Understanding the relationship between a threat specification language and a threat model, a common technique to study threats is vital for setting the scope of our proposed language. Figure 2 illustrates such the relationship between the Insider Threat Prediction and Specification Language (ITPSL) and an insider threat model (Magklaras et al., 2005).

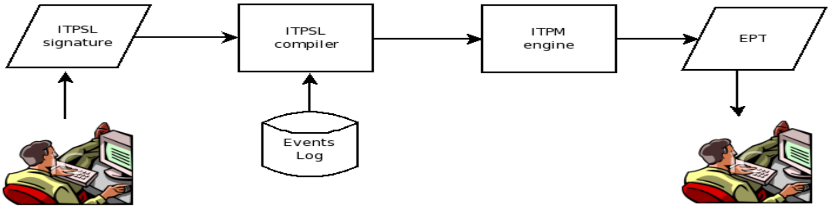


Figure 2: The relationship between ITPSL and a threat model

The flow of information starts with a security analyst writing a description of the particular insider misuse scenario, using the ITPSL semantics. The signature is validated by a compiler that translates the signature directives to query commands and makes use of the logging infrastructure, in order to examine whether the criteria and metrics the signature mentions exist in the system. The Evaluated Potential Threat (EPT) is a score indicating the likelihood of a threat occurring given the detected conditions.

The emphasis is on the ability to make insider case repositories. An early report outlining aspects of the insider threat to the US government information systems published by NSTISSAM, (1999) considers the absence of case repositories as one of the limiting factors in the field of insider IT misuse mitigation research. In addition, Capelli et al., (2006) state clearly the need to keep detailed records of employee actions in relation to file access, application usage and network connection matters. Brancik (2008) mentions the importance of suitable tools to produce Key Fraud Signatures (KFS) to aid insider threat mitigation and thus signifies the overlap between insider misuse and the field of digital forensics.

As a result, ITPSL should be viewed as a specialized language that is able to encode system level data that concern legitimate user actions, in order to aid the process of misuse threat prediction and assist computer forensic officers in the process of examining insider misuse incidents. As such, ITPSL's target audience is the security analyst/expert, as well as the seasoned IT administrator in charge of system operation and security issues. Both of these types of domain experts should be able to express insider misuse scenarios by using the language semantics to construct signatures of threat scenarios.

3. The design philosophy of ITPSL

The ITPSL scope defines clearly a specific task of expressing insider threat metrics. This paves the way for the selection of a mechanism that allows the language designer to focus on the problem in question. A Domain Specific Language (DSL) is a semantic mechanism tailored specifically for describing the details of a particular task. The main goal is the usage of appropriate semantics to reduce the effort required to reference and manipulate elements of that particular domain.

Spinellis (2001) defines a Domain Specific Language as "programming language tailored specifically to an application domain: rather than being for a general purpose, it captures precisely the domain's semantics". DSL schemata have been employed successfully in a number of different areas. Consel (2004) discusses the range of applications that have employed a DSL that includes device driver construction, active networking and operating system process scheduling.

DSLs are categorized as external and internal ones. An internal DSL is implemented as an extension of the semantics of a generic programming language. ITPSL follows the external DSL approach allowing for freedom to create the semantics from scratch with commonly changed parameters to be altered without recompilation issues and no dependence on host language idiosyncrasies. This approach has been followed by a number of security related research DSLs such as CISL by Feirtag et al., (1999) and Panoptis by Spinellis et al., (2002).

The ITPSL semantics is the second important aspect of the design philosophy. Software language engineers relate semantics to the term language by using the following definition: “A description of the semantics of a language L is a means to communicate a subjective understanding of the linguistic utterances of L to another person or persons.” (Kleppe, 2009).

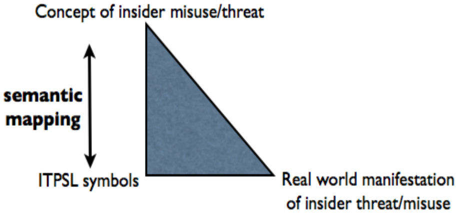
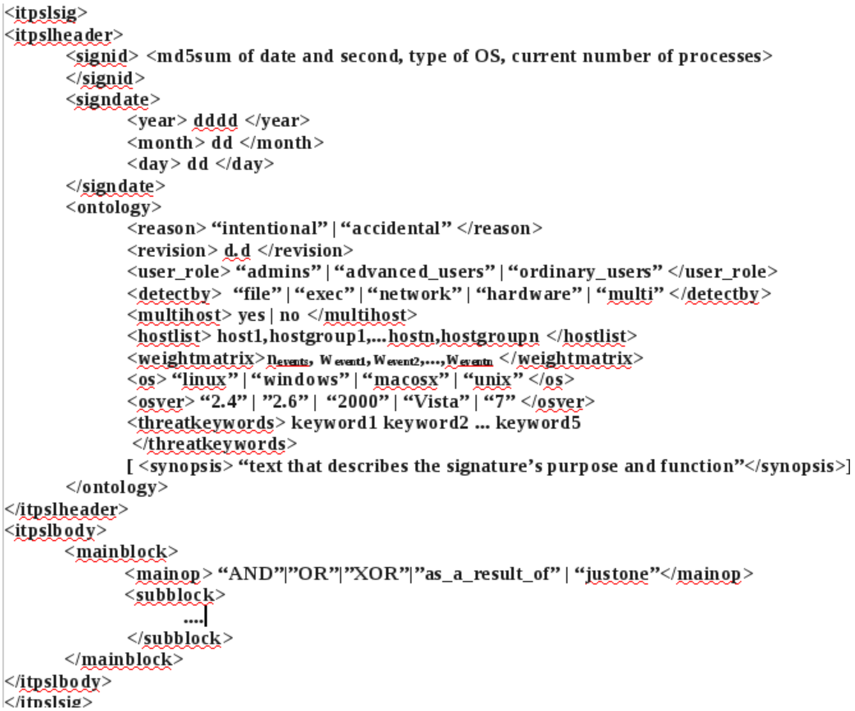

Concept of insider misuse/threat
semantic mapping
ITPSL symbols
Real world manifestation of insider threat/misuse

Figure 3: Semantic mapping in ITPSL

Figure 3 illustrates the role of semantics in ITPSL. The semantic triangle mapping (Kleppe, 2009) is all about connecting the symbols of ITPSL to the human expert's view of how insider misuse incidents or threats occur in a computer system. On the right hand side of the triangle resides the real world system view. Thus, the hypotenuse of the triangle symbolizes the differences between the concept and the real world manifestations of threats and misuse incidents. Finally, ITSPSL is an XML DSL markup. XML is a universal data exchange language. All ITPSL signature sections conform to the XML well formed rules (Goldberg, 2009) and a specific XML schema against which are validated.

4. The ITPSL markup

At the heart of the semantic framework lies the form of the ITPSL signature, a semantic structure that represents the encoding of an insider threat. Figure 4 shows the general structure of an ITPSL signature. It consists of a header section followed by the main body of the signature where sub-blocks of file, exec, network and hardware statements are encapsulated, in accordance to the different types of the LUARM audit log data in (Magklaras et al. (2011).



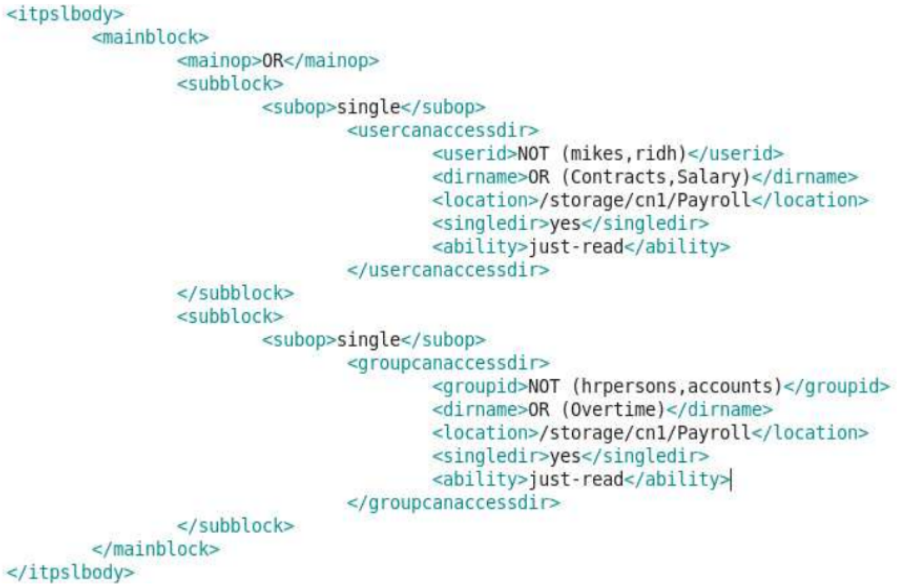
```
<itpslsig>
<itpslheader>
        <signid> <md5sum of date and second, type of OS, current number of processes>
        </signid>
        <signdate>
                <year> dddd </year>
                <month> dd </month>
                <day> dd </day>
        </signdate>
        <ontology>
                <reason> “intentional” | “accidental” </reason>
                <revision> d.d </revision>
                <user_role> “admins” | “advanced_users” | “ordinary_users” </user_role>
                <detectby>  “file” | “exec” | “network” | “hardware” | “multi” </detectby>
                <multihost> yes | no </multihost>
                <hostlist> host1,hostgroup1,...hostn,hostgroupn </hostlist>
                <weightmatrix>n_events, W_event1, W_event2,...,W_eventn </weightmatrix>
                <os> “linux” | “windows” | “macosx” | “unix” </os>
                <osver> “2.4” | ”2.6” |  “2000” | “Vista” | “7” </osver>
                <threatkeywords> keyword1 keyword2 ... keyword5
                 </threatkeywords>
                [ <synopsis> “text that describes the signature’s purpose and function”</synopsis>]
        </ontology>
</itpslheader>
<itpslbody>
        <mainblock>
                <mainop> “AND”|”OR”|”XOR”|”as_a_result_of” | “justone”</mainop>
                <subblock>
                        ....
                </subblock>
        </mainblock>
</itpslbody>
</itpslsig>
```

Figure 4: The ITPSL signature structure

The ontology header subsection is of particular importance for the creation of signature repositories. An ontology is a data model that represents a set of concepts within a domain (or formally known as domain of discourse in linguistic terms and the relationships between those concepts. A variation of the threat model published by (Magklaras et al., 2005) and the audit engine data (Magklaras et al., 2011) constitute the data model and the domain that the proposed markup language addresses.

As discussed in previous sections, ITPSL is trying to address the lack of insider threat scenario repositories. When such a repository is constructed, facilities to search and relate signatures (descriptions of threat scenarios) will be important and thus the language must have both semantic and data identifiers to allow security specialists to locate a class of signatures. For example, one could select all signatures that use network detection criteria, or all signatures that target p2p client installation and detect their presence at multiple levels ('multi' refers to a combination of employing 'file', 'exec', 'network' and 'hardware' detection statements). A third example could be the last two revisions of a particular set of signatures or a set of signatures whose weight matrix places more emphasis on network detection criteria.

The <reason> tag specifies whether we are searching for a threat that is a result of deliberate actions (intentional) or accidental mistakes (accidental). The <revision> tag makes possible to trace signatures whose detection criteria are modified to improve the accuracy or to examine slightly different aspects of the target problem. In that case, the ‘signid’ identifier remains the same amongst the related signatures.



```
<itpslbody>
        <mainblock>
                <mainop>OR</mainop>
                <subblock>
                        <subop>single</subop>
                                <usercanaccessdir>
                                        <userid>NOT (mikes,ridh)</userid>
                                        <dirname>OR (Contracts,Salary)</dirname>
                                        <location>/storage/cn1/Payroll</location>
                                        <singledir>yes</singledir>
                                        <ability>just-read</ability>
                                </usercanaccessdir>
                </subblock>
                <subblock>
                        <subop>single</subop>
                                <groupcanaccessdir>
                                        <groupid>NOT (hrpersons,accounts)</groupid>
                                        <dirname>OR (Overtime)</dirname>
                                        <location>/storage/cn1/Payroll</location>
                                        <singledir>yes</singledir>
                                        <ability>just-read</ability>
                                </groupcanaccessdir>
                </subblock>
        </mainblock>
</itpslbody>
```

Figure 5: An example ITPSL signature body

Figure 5 illustrates an example of an ITPSL signature in an information leakage detection scenario. In particular, we are interested to check the accessibility of the Payroll directory folder (/storage/cn1/Payroll). The main-block consists of two sub-blocks. The first one contains a 'usercanaccessdir' directive as the access control policy mentions that only certain user names (mikes,ridh) should access the 'Salary' and 'Contracts' sub-folders. The <userid>NOT (mikes,ridh)</userid> tag is expanded to all other userids of the host and the directory access check is performed for each one of them.

In contrast, the second sub-block contains a 'groupcanaccessdir' because the other part of the policy defines two distinct groups (hrpersons, accounts) that should access the 'Overtime' folder. The <groupid>NOT(hrpersons,accounts)</groupid> tag is expanded to all non system groups (root, wheel) and then to all resulting usernames that belong to these groups, in order for the check to be performed for each user account member.

5. Implementing and evaluating ITPSL

Figure 6 illustrates the components of the ITPSL compiler prototype system. The prototype system consists of a number of relational tables and Perl scripts, as well as an XML Schema file (validate.xsd). This file describes the syntax of the ITPSL signature and helps the various tools ensure that the processed signature is consistent with the syntax and format of the language. On the left hand side of Figure 5, the 'submitsig.pl' script helps the user submit a syntactically correct signature to the

ITPSL signature repository ('signatures_repository.sql'). This repository implements the ITPSL signature ontology (ITPSL signature header). In contrast, the 'searchsig.pl' and 'getsig.pl' are used to search and retrieve one or more signatures from the ITPSL signature repository.

The fetched signatures are then fed to the main ITPSL compiler module ('itpslc.pl'). This module has the vital job of interpreting LUARM data by turning the ITPSL signature semantics into misuse detection and prediction output, informing the analyst on whether something is happening (misuse detection) or is about to happen (threat prediction).

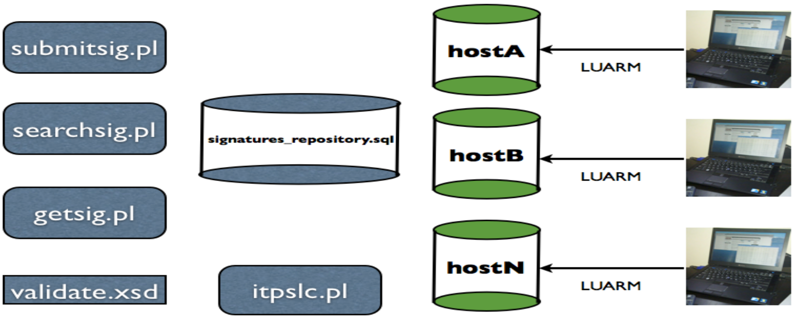


Figure 6: The ITPSL prototype system

Evaluating a DSL language like ITPSL requires a carefully designed process. DSLs like general purpose programming languages can be evaluated empirically. While this empirical process is still valuable, today the tasks of building and evaluating DSLs have become part of the Software Language Engineering (SLE) discipline (White et al., 2009). SLE improves dramatically the quality of DSL evaluation. However, it still leaves important gaps in the process of measuring vital language attributes such as expressiveness, effectiveness, usability and maintainability.

For these reasons, a controlled experiment (Magklaras, 2011) on insider IT misuse detection and prediction scenarios was chosen to evaluate the language. The scenarios were derived from real world incident data collected by testers of the LUARM audit engine (Magklaras et al., 2011) and the misuse game had three important entities:

1. **The users**: The people that are associated to a particular scenario and have unique user-id and authentication credentials.
2. **The analyst**: The person who was responsible for examining the logs and using ITPSL to detect/predict the threats and the associated users (security officer or a third party security auditor).
3. **The IT infrastructure**: A number of Linux workstations with shared filesystems that simulated an IT infrastructure.

There were four scenarios that represented a range of common IT misuse incidents. Scenario 1 was a typical Intellectual Property theft scenario with the added complexity of a masquerade attack. Scenario 2 re-enacted the detection of accessing pornographic material. Scenario 3 is an example of an insider IT misuse prediction task. In essence, it aims to demonstrate the decision theoretic information features of ITPSL and help the analyst predict the installation of a dangerous DoS attack tool by an ordinary user. The fourth and last scenario of the game is also demonstrating a predictive operation of file access control settings that could produce accidental (non intentional) information leak. The scenarios were discussed during an initial briefing amongst the analyst and three IT specialists that would re-enact these incidents by playing the role of the misuser for each scenario. After the initial briefing, the users met amongst them to discuss a role allocation for each scenario without the knowledge of the analyst. At that point, LUARM was activated and logging commenced for a period of 4 weeks. After this audit period, results were collected and verified with the users by the analyst.

6. Evaluation of results and conclusions

The game scenario (Magklaras, 2011) results revealed that ITPSL is a useful framework that can help researchers and practitioners profile a large range of common misuse incidents, expressing both threat detection and prediction. The language and underlying audit engine provided useful clues for all four scenarios.

The submitsig.pl and searchsig.pl (Figure 6) ITPSL utilities can create insider misuse incident signature repositories, combining the description of static and live forensic data under one common semantic framework. ITPSL is the first misuse detection language to support decision theoretic information. The association of weights and events facilitates insider threat prediction based on the analyst's view of how the threat precursors occur at file, network, process execution and hardware device level. Despite the fact that none of the scenarios provided a true opportunity to test event correlation across multiple hosts, ITPSL is equipped with operators such as the <onhost> tag, which could bind certain events to specific hosts and permit cross-host event correlation.

However, there are certain areas where ITPSL has weaknesses. One important issue to consider is the expressive granularity of the events. ITPSL (and its underlying audit record) capture file, network endpoint and process creation data. All applications generate these types of events in an operating system. However, not all of these events are meaningful to insider misuse detection and prediction at that level. There is a category of applications whose file, network and process execution operations are not easy to interpret. A great example is that of database applications where the observation of file access patterns do not reveal any clues about what kind of information is accessed and in what manner. In order to address that problem, the Intrusion Detection System (IDS)/Intrusion Prevention System (IPS) community suggests application-integrated monitoring (Phyo et al., 2003), a technique where audit records are generated by monitoring routines internal to the application itself. These routines know which internal events are of interest and can export relevant activities in specific audit log formats.

ITPSL (and its underlying logging mechanism) clearly needs certain semantic extensions to monitor applications such as databases, social networking sites and other applications that organize information using internal mechanisms. Everything else can be described by the proposed file, network, process and hardware semantics.

A last but equally important issue to consider is that of the monitoring framework scalability. The LUARM audit engine enforces a relational model (Codd, 1990) and the SQL interface (ISO/IEC 9075, 2008). This was a core design choice aiming to enhance the correlation versatility of the audit log structure. This goal was achieved, however, it imposes certain scalability limitations.

Relational databases have been at the forefront of massive data storage and organization for several decades. A number of different approaches enable relational databases to scale well, so that they can handle concurrently a large number of operations. Despite the success of employing relational database scalability measures, every practitioner agrees that the measures can become a cumbersome process and they have limits (Strandell, 2010). As the Relational Database scales horizontally (spread in various nodes), the complexity of managing software and hardware aspects in relation to interprocess communication increases.

The previous complexity and transaction volume requirements created a new generation of database products that are collectively referred to as 'NoSQL databases' (Zawodny, 2009). The 'NoSQL' term emphasizes the departure of these products from the traditional relational model, in an attempt to balance the need to scale and the need to preserve some of the properties of relational consistency . These products deviate from traditional RDBMS requirements such as data normalization and create simpler but faster key lookup mechanisms, in order to achieve massive concurrency and scalability. LUARM's SQL table schema is simple and does not require data normalization or embody any relational key constraints amongst the various client tables (audit levels). Consequently, it should be possible to port the audit log structure into a 'NoSQL' product and take advantage of its speed and scalability.

7. References

Bace R. (2000), “Intrusion Detection”, First Edition, Macmillan Technical Publishing, Indianapolis, USA: 25. Page 227 discusses the distinction between the security and monitoring policies.

Brancik K.C. (2008), ”Insider Computer Fraud An in-depth Framework for Detecting and Defending Against Insider IT Attacks”, Auerbach Publications, Taylor & Francis Group, ISBN 1-4200-4659-4.

Caelli W., Longley D. and Shain M. (1991), Information Security Handbook, Stockton Press.

Cappelli D., Moore A., Shimeall T.J., Trzeciak R. (2006). “Common Sense Guide to Prevention and Detection of Insider Threats”, 2nd Edition Version 2.1, Carnegie Mellon University Cylab,http://www.cert.org/archive/pdf/CommonSenseInsiderThreatsV2.1-1-070118.pdf

Consel C. (2004), “From A Program Family To A Domain-Specific Language”, in Lengauer, C.; Batory, D.; Consel, C.; Odersky, M. (Eds.), Domain-Specific Program Generation, LNCS 3016, Springer-Verlag, pp. 19-29.

Codd E. (1990), “The Relational Model for Database Management”, Addison-Wesley Publishing Company, 1990, ISBN 0-201-14192-2.

Feiertag R., Kahn C., Porras P., Schnackenberg D., Staniford-Chen S., Tung B. (1999), “A Common Intrusion Specification Language (CISL)”, June 1999 revision, URL: http://gost.isi.edu/cidf/drafts/language.txt

Goldberg K. (2009), “XML: Learn XML the Quick and Easy Way”, Second Edition, Peachpit Press, ISBN-13:978-0-321-55967-88, pp. 5-6 explain the rules of well formed XML.

ISO/IEC 9075(1-4,9-11,13,14) (2008) Information Technology-- Database Languages – SQL:, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45498

Kleppe A. (2009), “Software Language Engineering – Creating Domain Specific Languages Using Metamodels”, Addison-Wesley/Pearson Education, ISBN: 978-0321553454.

Magklaras G. (2011), “An Insider Misuse Threat Detection and Prediction Language”, PhD Thesis, School of Computing and Mathematics, University of Plymouth, UK: Chapter 6 (pp. 113- 201) describes the ITPSL Language in detail. Chapters 7 and 8 (pp. 202-265) discuss the implementation of the language http://folk.uio.no/georgios/MagklarasPhDThesisv3.pdf

Magklaras G., Furnell S. (2005), “A Preliminary Model of End User Sophistication for Insider Threat Prediction in IT Systems”, Computers & Security, Volume 24, Issue 5, August 2005, pp. 371-380.

Magklaras G., Furnell S., and Papadaki M. (2011), “LUARM: An Audit Engine for Insider Misuse Detection”, International Journal of Digital Crime and Forensics, (IJDCF), pp. 37-49.

NSTISSAM. (1999), “The Insider Threat To US Government Information Systems”, NSTISSAM INFOSEC /1-99, U.S. National Security Telecommunications And Information Systems Security Committee, http://www.cnss.gov/Assets/pdf/nstissam_infosec_1-99.pdf

Pfleeger C., Pfleeger S. (2003), “Security in Computing”, 3rd edition, Prentice Hall, Englewood Cliffs, NJ, 1. ISBN:0130355488: Page 6 contains the definition of the term “threat” in an information security context.

Phyo A., Furnell S. (2003), “Data Gathering for Insider Misuse Monitoring”, Proceedings of the 2nd European Conference on Information Warfare and Security, Reading, UK, 30 June - 1 July, pp247-254.

Probst C., Hunker J., Bishop M., Gollman D. (2009), “Countering Insider Threats”, ENISA Quarterly Review Vol. 5, No. 2, June 2009, pp. 13-14.

PwC (2010), “INFORMATION SECURITY BREACHES SURVEY 2010 | technical report”, http://www.pwc.co.uk/eng/publications/isbs_survey_2010.html

Richardson R. (2010), “15th Annual 2010/2011 Computer Crime And Security Survey”, http://gocsi.com/2010_survey_purchase

Spinellis D. (2001), “Notable design patterns for domain-specific languages”, The Journal of Systems and Software Volume 56, Issue 1 (2001), pp. 91-99.

Strandell T. (2010), “Open Source Database Systems: Systems Study, Performance and Scalability”, VDM Verlag Dr. Müller, ISBN: 978-3639093506.

White J., Hill J.H., Tambe S., Gokhale A., Schmidt D.C. (2009), “Improving domain- specific language reuse with software product line techniques”, IEEE Software Vol. 26, No. 4, pp.47–53.

Zawodny J. (2009), “NoSQL: Distributed and Scalable Non-Relational Database Systems”, Linux Magazine web portal, http://www.linux-mag.com/id/7579/