A Learning Framework for Transitioning Network Intrusion Alerts Management System to Ontology

Chunlei Fu

Information and Network Management Center, Chongqing University, China
clfu@cqu.edu.cn

Qichang Duan

School of Automation, Chongqing University, China
qichangduan@cqu.edu.cn

Li Fu

School of Software Engineering, Chongqing University, China
fuli@cqu.edu.cn

Hong Xiang

School of Software Engineering, Chongqing University, China
xianghong@cqu.edu.cn

Zhongyang Xiong

Information and Network Management Center, Chongqing University, China
zyxiong@cqu.edu.cn

Haibo Hu

School of Software Engineering, Chongqing University, China
hbhu@cqu.edu.cn

Intrusion detection is not new in the area of information security. It is crucial for the intrusion alerts management system to correlate the collected intrusion alerts to reflect the causal relationships between the attack steps and construct the attack scenarios. Most of these systems, however, have been built on the relational database logging the intrusion alerts. The relational database has been proven to be a very useful model and applied in the wide area. But their persisting limitation lies in the flat structure which is not capable of representing the complex relations. An ontology is an explicit specification of a conceptualization using an agreed vocabulary. In this paper, ontology is put into use and a learning framework is presented which depicts how the intrusion alerts ontology can be learned and further enriched exploiting both the database schema and the stored data. Moreover, we introduce the vulnerabilities database to refine the ontology hierarchy and the restriction of classes and apply the ontology design pattern to represent the sequence of a series of events. The whole transitioning process is implemented in OBNAMS, an intrusion alerts management system constructed on the learned ontology automating the consisted steps.

Keywords: Intrusion Detection, Intrusion Alerts Correlation, Ontology learning
ACM Classification: C.2.0., I.2.5., I.2.6.

Manuscript received: 17 May 2011
Communicating Editor: Rafael Valencia García

1. INTRODUCTION

With the deployment of multi-sensors at the critical nodes of the target network, NIDS(Network Intrusion Detection System) is the special cyber security device that monitors the incoming packets and tries to detect malicious activity according to the suspicious patterns known as signatures or rules. There are actually two main intrusion detection approaches: the behavioural approach, namely anomaly detection, and the signature analysis which is called misuse detection. Anomaly detection utilizes statistical description of the normal behaviour of users or applications to detect any abnormal action performed by these users or applications. And misuse detection summarizes the distinctive feature of each specific attack to detect known attack behaviour accurately. The NIDS products founded on either of the above two methods will give rise to a great volume of intrusion alerts logged into the database. Security administrators are generally provided with the network intrusion alerts management system which is an operator interface to access the intrusion database and analyze the NIDS alerts.

Few of the existing NIDS products and their associated management tool can meet the requirements of the security operators. At first, most of the intrusion alerts generated by the NIDS are too elementary. They can only identify the low-level malicious behaviour information of a single attack step notwithstanding the existence of the logical relationship between attack steps. Due to lack of the overview of the attack scenario, NIDS cannot uncover the actual intention and strategy of the attacker. Consequently, in the context of the current alerts management system constructed on the intrusion database, we can only build the query to search the database for the particular alerts matching the specific condition, such as attack signatures, detection time, address, ports and so on. Moreover, it is deemed too tedious and time-consuming for the security operator to analyze the alerts on this kind of analysis system.

In this paper, we develop the framework transitioning the intrusion alerts management system from the relational database to ontology. Our transitioning approach considers not only the database schema but the contained data. In addition, mining the relevant vulnerabilities information of each attack from the vulnerabilities knowledge bases, we depict the causal relationship between the attack stages in ontology and apply the ontology design pattern to alerts correlation.

The remainder of the paper is organized as follows. We illustrate in Section 2 the transitioning process in detail. Section 3 describes the application of ontology design pattern for intrusion alerts correlation. Section 4 provides an overview of OBNAMS, the NIDS alerts management system based on the target ontology of the transitioning procedure. Section 5 describes the experiment we conducted to evaluate the implemented method in OBNAMS. In Section 6, we review previous work on ontology learning from relational database and the related issue of intrusion alerts correlation. Then, we conclude with some directions for further research.

2. THE TRANSITIONING PROCESS

The transitioning process consists of several automated steps, which include database normalization, mapping from relational database schema to ontology, mining the class hierarchy from the stored data and ontology population. It is noticed that some of the steps involved with the process will be implemented in the OBNAMS. (See Section 4)

2.1 The Example of NIDS Intrusion Database

In the following, we refer to Snort database as an example NIDS alerts database and OWL 2 as the ontology description language (OWL 2, 2009). The Snort database schema is depicted in Figure 1.

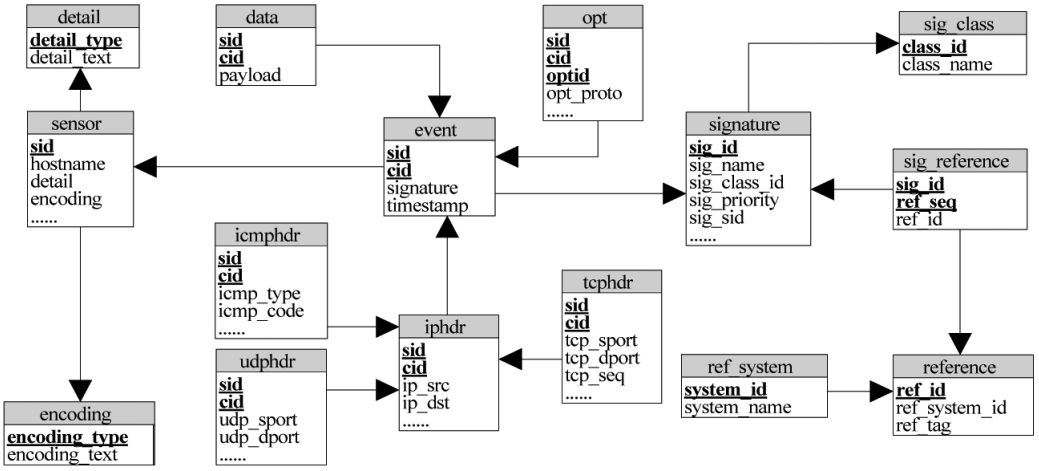


Figure 1: The Diagram of Snort Database Schema

2.2 Database Normalization

The process of organizing data to minimize redundancy is called normalization in the design of a relational database. The objective of database normalization is to analyze the relations with anomalies and decompose it so that we will give rise to well-formed relations. A series of guidelines for ensuring that databases are normalized have been developed. These are referred to as normal forms including 1NF, 2NF, 3NF and so forth. It is obvious the above mentioned Snort database schema meets the requirements of 3NF. As shown in Figure 1, each tuple of the relations contained in the Snort database, consists of a primary key value that identifies some entity, together with a set of mutually independent attribute values that describe that entity in some way.

2.3 Mapping from relational Databases Schema to Ontology

We proposed a mapping model that is a tuple of the form $RTO=\{\Delta_S , O_d ,\Sigma_R \}$, where:

Δ_S is the schema pattern of the database. It is comprised by:

- A finite set R is called relations each of which is characterized by its finite set of attributes $\{ R_1, R_2, \ldots\ldots ,R_m\}$.
- A finite set A is called Attributes $\{A_1, A_2, A_3, \ldots , A_n\}$.
- A function *attr:* $R \rightarrow 2^A$ that defines the attributes contained in R such that $attr(R_i) \subseteq R_i$.
- A function *pkey:* $R \rightarrow 2^A$ that associates to each relation its primary key which is a set of attributes such that $pkey(R_i) \subseteq R_i$.
- A function *fkey:* $R \rightarrow 2^A$ that defines the attributes in a relation which matches the primary key of another relation.
- A set T of atomic data types.
- A function *dt:* $A \rightarrow T$ that gives the type of each attribute.

O_d is the finite set of the ontology elements, namely *class*, *object property*, *datatype property*, *domain and range of the property*, *individual* and so on. Σ_R is the set of mapping rules which are applied in the migration from database to ontology.

We depict some of the rules with First-Order Logic as follows with use of the auxiliary predicate *Type(x,y)* denoting the relation with which type a object *x* is associated. These rules have been employed in several existing approaches.

Rule1: learning class from relation

$(\forall x)(Type(x,R)\rightarrow|pkey(x)|=1)\Rightarrow(\forall x)(Type(x,Class))$

$(\forall x)(\exists y)(Type(x,R)\rightarrow(|pkey(x)|>1\wedge Type(y,A)\wedge y\in pkey(x)\wedge y\notin fkey(x)))\Rightarrow(\forall x)(Type(x,Class))$

An object *x* of which the type is Relation R can be mapped to an ontological class if there are not any relations that can be incorporated with the Relation R. Obviously, the relation R is used to describe an entity, instead of a relationship between relations, then it can be mapped into one ontological class.

Rule2: learning object property from the relationship

$(\forall x)(\forall y)((Type(x,A)\wedge Type(y,A))\rightarrow(x\subset R_i\wedge y\subset R_j\wedge x\subset y\wedge x\not\subset pkey(R_i)))\Rightarrow$
$(\forall x)(ObjectProperty(P_R)\wedge Type(x,Domain(P_R))\wedge Type(y,Range(P_R)))$

For the object x and y of which type are all attributes of the Relation R_i and R_j respectively, if $x\subset y$ and $x\not\subset pkey(R_i)$ are satisfied, then an object property P can be created based on x. And the domain and range of P are x and y.

Rule3: learning Datatype property from the attribute

$(\exists x)(\forall y)(Type(x,A)\wedge x\subset R_i\wedge\neg(y\subset R_j\wedge x\subset y))\Rightarrow DatatypeProperty(P_R)\wedge(dt(P_R)\in T)$

If an object x of which the type is an attribute of Relation R_i, and it cannot be used to generate object property by using Rule 2, then it can be used to create datatype property of x.

Applying the mapping rules we can generate the initial ontology automatically. The extracted ontology from Snort database schema is shown in Figure 2 which illustrates the object properties.

2.4 Mining the Stored Data for the Refinement of the Class Hierarchy

With the help of the above mapping rules, we obtain the ontology with flat structure that simply mirrors the schema from the intrusion database. NIDS users attracted by the powerful expressiveness of the ontology would not be satisfied with the results at all. It is understood for an analysis of the data stored in the existing intrusion database in depth that ontology elements can further be learned from the data to significantly enrich the ontology structure. A depiction of the refining procedure is given below. And then with the given method, we discuss the generation of the subclasses in the above ontology obtained with the mapping rules.

2.4.1 Categorizing Attributes Selection

Categorizing attributes selection are involved with our background knowledge of the particular intrusion database. Each of the attributes plays its specific role in relation to the database. In our context, we exemplify the Snort database schema given in Figure 1. Every tuple of relation *event* symbolizes the event generated by Snort. For each event, the attack signature attribute, namely *signature*, serves as the foreign key to link the relation *signature* together. The attribute *sig_priority* of the relation *signature* indicates the severity level of each event and the *sig_class_id* is the foreign key which matches the primary key *class_id* of *sig_class*. *Sig_class* is the relation which stored the data describing the attack classification. The attribute *ip_src* and *ip_dst* which means the address of the attacker and victim respectively in the relation *iphdr* that is related to the relation *event* with the foreign key *cid*.

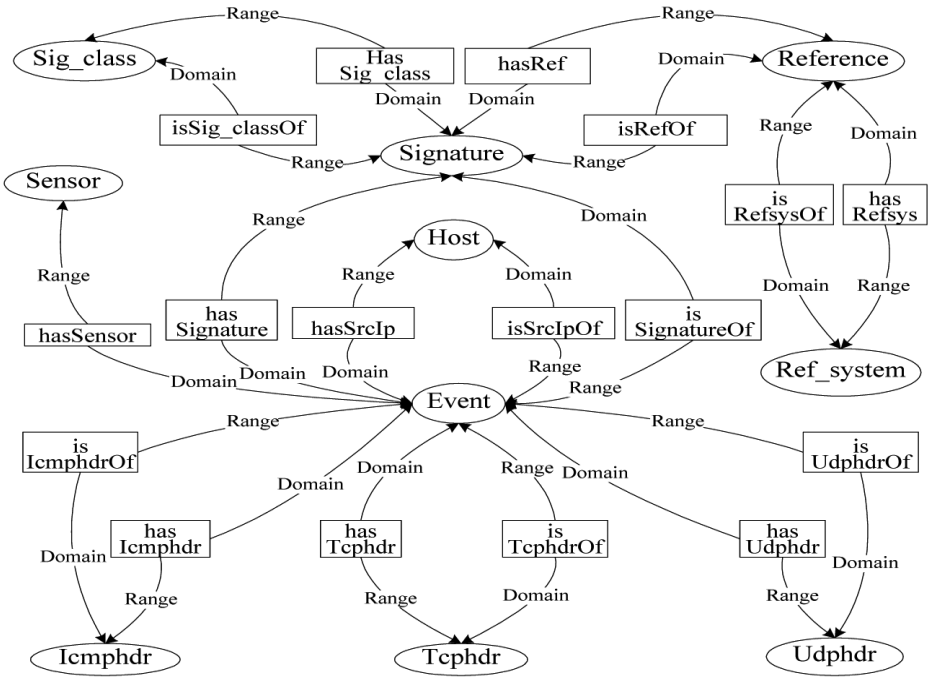


Figure 2: The Basic Ontology Mapped from Snort Database Schema

Based on our security knowledge, we will collect the security-related attribute to define the list of categorizing attributes candidate. It is crucial for us to be aware that not all of the candidates can be viewed as the categorizing attributes for subclasses generation. For example, given a candidate attribute which is the primary key of a relation *R*, we cannot generate subclasses from the tuples projection on it in spite of its security-related interest.

2.4.2 Candidate Categorizing Attributes Evaluation

Our candidates evaluation method relies on a simple attribute selection measure for which the Information Entropy lays the foundation. The Entropy describes the uncertainty of a data source. It is assumed that an appropriate candidate for tuples categorization may to some extent represent the information diversity hold in the relation projection on the attribute (Mehmed, 2002). Attributes with highly repetitive values will be characterized by a low entropy. On the contrary, among attributes of a given relation, the primary key attribute will have the highest entropy since all values held in them are distinct.

If A is an attribute of a relation schema R instantiated with relation r, the diversity in A is estimated by:

$$Info(A) = \sum_{v \in prj_A(r)} P_A(v) \cdot log P_A(v)$$

At first, we define an arbitrary tuple in relation *r* as a function, namely *tuple(r)*, and $prj_A(r)$ as the function that represents the projection of *r* on *A*. And $P_A(V)$ is just the proportion of tuples of each value category which will be mapped to a subclass in the ontology. A log function to the base

2 is used, because the information is encoded in bits. As the result, *Info(A)* is the average amount of information which estimates the data diversity held in the $prj_A(r)$.

As stated earlier, it has been found that the entropy of the primary key attributes is always the highest among the attributes of the relations. The highest entropy, $Info_{max}(A)$, is shown as follows.

$$Info_{max}(A) = log|tuple(r)|$$

So the evaluation criteria to filter the candidate is formally given by:

$$C = \{A \mid Info(A) \in [\lambda, \mu \cdot Info_{\max}]\} \quad \lambda, \mu \in [0, 1]$$

2.4.3 Generation of Subclasses

It is natural that subclasses are generated and populated from an identified categorizing attribute. A subclass is derived from each value partition of the projection on the candidate attribute. As a result, the data category can be extracted from the database using the SQL statement. For example, in the data contained in the Snort database, we can obtain the particular classification of each attack signature from the *signature* and *sig_class* table.

select c.sig_class_name from signature s, sig_class c where s.sig_class_id = c.sig_class_id group by c.sig_class_name

For each value *v* of *s.sig_class_name* which denotes the classification of each attack signature, the corresponding subclass of class *sig_class* is given, of which the name is the string within the *v*. Figure 3 illustrates the process.

And the priority category of each attack signature can be extracted from the signature table.

select s.sig_priority from signature s group by s.sig_priority

For each value *v* of *s.sig_priority*, the corresponding subclass of *Priority* is generated, and its name is just the actual string within the value *v*. The process is given in Figure 4.

The relevant attack signature information of each event may be further depicted in detail. As specific attacks often target specific vulnerabilities, each attack has the corresponding vulnerability information. There are several popular vulnerability knowledge bases, such as CVE, NVD, BugTraq, CERT and so on(CVE, NVD, BugTraq, and CERT/CC). NVD is the one which assigns to each vulnerability a unique number and a description. The vulnerability number is separated into

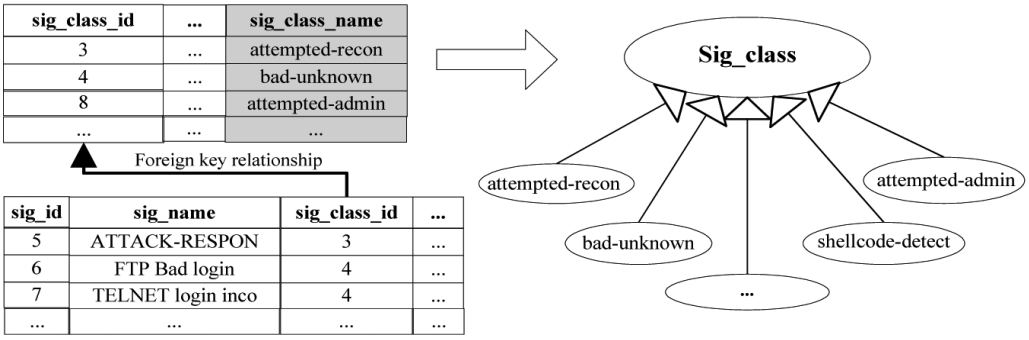


Figure 3: The Subclass Hierarchy of the Class Sig_class

sig_id	sig_name	...	sig_priority
5	ATTACK-RESPON	...	2
6	FTP Bad login	...	2
9	ICMP PING	...	3
16	RSERVICES rsh root	...	1

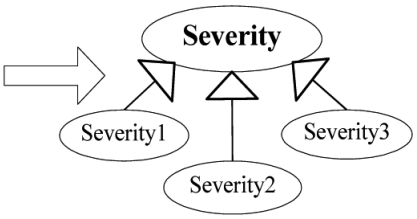


Figure 4: The Subclass Hierarchy of the Class Severity

ref_id	...	ref_tag
3	...	www.microsoft.com
5	...	2003-0722
7	...	2005-0068
...	...	...

Foreign key relationship

sig_id	sig_name	ref_id	...
5	ATTACK-RESPON	3	...
6	FTP Bad login	4	...
7	TELNET login inco	4	...
...	...	...	...

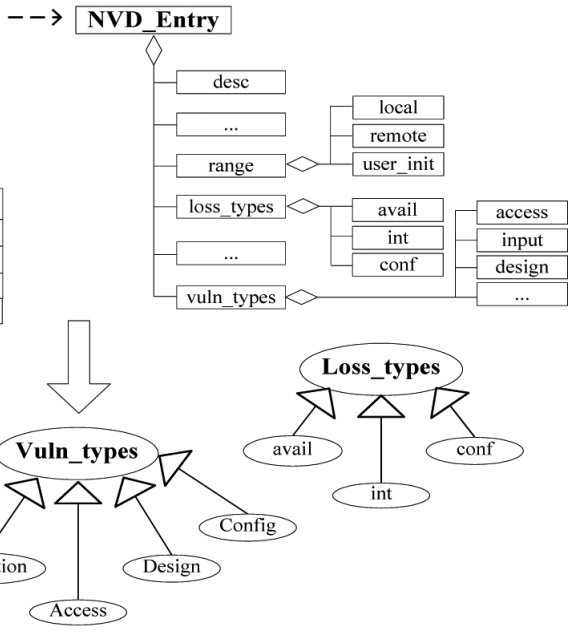


Figure 5: The Subclass Hierarchy of the Specific Classes modelling the associated vulnerabilities

three parts. The first part is the prefix CAN or CVE for candidate and confirmed vulnerabilities respectively. The second part is the year that the vulnerability was discovered. In addition, the NVD database does provide involved attack signatures specification of the range of exploitability, of the vulnerability type and of the loss type of the attack, as shown in Figure 5.

2.5 Ontology Population

Ontology population aims at not only generating instances of classes and properties from the data stored in intrusion database but also further defining classes. It is noticed that the whole process will be conducted in a fully automated way.

2.5.1 Create Existential Restriction on the Signature Class

Create an existential restriction on subclasses of *Signature* that acts along the property *hasSig_class* with a filler of a certain subclass of *Sig_class*, as shown in Table 1.

MISC-ATTACK_Signature	⊑ Signature ⊓	∃hasSig_class.MISC-ATTACK_Sig_class
ATTEMPTED-DOS_Signature	⊑ Signature ⊓	∃hasSig_class.ATTEMPTED-DOS_Sig_class
ATTEMPTED-RECON_Signature	⊑ Signature ⊓	∃hasSig_class.ATTEMPTED-RECON_Sig_class
SHELLCODE-DETECT_Signature	⊑ Signature ⊓	∃hasSig_class.SHELLCODE-DETECT_Sig_class
...		...

Table 1: The Restrictions on the Subclasses of Signature Acting along the Property hasSig_class

Create an existential restriction on subclasses of *Signature* that acts along the property *hasSeverity* with a filler of a certain subclass of *Severity*, as shown in Table 2.

Severity1_Signature	⊑ Signature ⊓	∃hasSeverity.Severity1
Severity2_Signature	⊑ Signature ⊓	∃hasSeverity.Severity2
Severity3_Signature	⊑ Signature ⊓	∃hasSeverity.Severity3

Table 2: The Restrictions on the Subclasses of Signature Acting along the Property hasSeverity

Create an existential restriction on subclasses of *Signature* that acts along the property *hasVul* with a filler of a certain subclass of *Vul_type*, as shown in Table 3.

Input_Vul_Signature	⊑ Signature ⊓	∃hasVul.Input_Vul
Access_Vul_Signature	⊑ Signature ⊓	∃hasVul.Access_Vul
Config_Vul_Signature	⊑ Signature ⊓	∃hasVul.Config_Vul
Design_Vul_Signature	⊑ Signature ⊓	∃hasVul.Design_Vul
Exception_Vul_Signature	⊑ Signature ⊓	∃hasVul.Exception_Vul

Table 3: The Restrictions on the Subclasses of Signature Acting along the Property hasVul

Create an existential restriction on subclasses of *Signature* that acts along the property *hasRange* with a filler of a certain subclass of *Range*, as shown in Table 4.

Local_Ran_Signature	⊑ Signature ⊓	∃hasRange.Local_Range
Remote_Ran_Signature	⊑ Signature ⊓	∃hasRange.Remote_Range
User_init_Ran_Signature	⊑ Signature ⊓	∃hasRange.User_init_Range

Table 4: The Restrictions on the Subclasses of Signature Acting along the Property hasRange

Create an existential restriction on subclasses of *Signature* that acts along the property *hasLosstype* with a filler of a certain subclass of *Losstype*, as shown in Table 5.

Conf_Loss_Signature	⊑ Signature ⊓	∃hasLosstype.Conf_Loss
Avail_Loss_Signature	⊑ Signature ⊓	∃hasLosstype.Avail_Loss
Int_Loss_Signature	⊑ Signature ⊓	∃hasLosstype.Int_Loss

Table 5: The Restrictions on the Subclasses of Signature Acting along the Property hasLosstype

2.5.2 Convert the Necessary Condition into the Necessary and Sufficient Conditions

Given our current description of class *Signature* this knowledge is not sufficient to determine that the individual is a member of a certain subclass of *Signature*. To make this possible we need to change the conditions for *Signature* from *necessary conditions* to *necessary AND sufficient conditions*, as shown in Table 6.

Severity1_Signature	≡	Signature ⊓ ∃hasSeverity.Severity1
Severity2_Signature	≡	Signature ⊓ ∃hasSeverity.Severity2
Severity3_Signature	≡	Signature ⊓ ∃hasSeverity.Severity3
Input_Vul_Signature	≡	Signature ⊓ ∃hasVul.Input_Vul
Access_Vul_Signature	≡	Signature ⊓ ∃hasVul.Access_Vul
Config_Vul_Signature	≡	Signature ⊓ ∃hasVul.Config_Vul
Design_Vul_Signature	≡	Signature ⊓ ∃hasVul.Design_Vul
Exception_Vul_Signature	≡	Signature ⊓ ∃hasVul.Exception_Vul
Local_Ran_Signature	≡	Signature ⊓ ∃hasRange.Local_Range
Remote_Ran_Signature	≡	Signature ⊓ ∃hasRange.Remote_Range
User_init_Ran_Signature	≡	Signature ⊓ ∃hasRange.User_init_Range
Conf_Loss_Signature	≡	Signature ⊓ ∃hasLosstype.Conf_Loss
Avail_Loss_Signature	≡	Signature ⊓ ∃hasLosstype.Avail_Loss
Int_Loss_Signature	≡	Signature ⊓ ∃hasLosstype.Int_Loss
MISC-ATTACK_Signature	≡	Signature ⊓ ∃hasSig_class.MISC-ATTACK_Sig_class
ATTEMPTED-DOS_Signature	≡	Signature ⊓ ∃hasSig_class.ATTEMPTED-DOS_Sig_class
ATTEMPTED-RECON_Signature	≡	Signature ⊓ ∃hasSig_class.ATTEMPTED-RECON_Sig_class
SHELLCODE-DETECT_Signature	≡	Signature ⊓ ∃hasSig_class.SHELLCODE-DETECT_Sig_class
...		...

Table 6: The Necessary AND Sufficient Conditions for the Subclasses of Class Signature

Along with the above *equivalent class* definition, we can create an existential restriction on subclasses of *Event* that acts along the property *hasSignature* with a filler of a certain equivalent class of subclass of *Signature*, and convert it to the *equivalent classes*, as shown in Table 7.

Severity1_Event	≡	Event ⊓ ∃hasSignature.Severity1_Signature
Severity2_Event	≡	Event ⊓ ∃hasSignature.Severity2_Signature
...	≡	...
Input_Vul_Event	≡	Event ⊓ ∃Signature.Input_Vul_Signature
Access_Vul_Event	≡	Event ⊓ ∃hasSignature.Access_Vul_Signature
...	≡	...
Local_Ran_Event	≡	Event ⊓ ∃hasSignature.Local_Ran_Signature
Remote_Ran_Event	≡	Event ⊓ ∃hasSignature.Remote_Ran_Signature
...	≡	...
Conf_Loss_Event	≡	Event ⊓ ∃hasSignature.Conf_Loss_Signature
Avail_Loss_Event	≡	Event ⊓ ∃hasSignature.Avail_Loss_Signature
...	≡	...
MISC-ATTACK_Event	≡	Event ⊓ ∃hasSignature.MISC-ATTACK_Signature
ATTEMPTED-DOS_Event	≡	Event ⊓ ∃hasSignature.ATTEMPTED-DOS_Signature
...		...

Table 7: The Necessary AND Sufficient Conditions for the Subclasses of Event

2.5.3 Import Instances and Classifty

Initially we have constructed the basic ontology which does not contain any individual because we intended to apply it in various NIDS deployment. We are then able to import instances of the intrusion database to enrich the basic ontology with the information contained in the specific instance which is derived from each tuple of the source relation. Every instance differs from each other in terms of the set of signatures used and their classification. Moreover, if refinement into subclasses has been successfully applied on the class, the instances need to be further partitioned into the various subclass respectively. Finally, we will renew the ontology by importing events recorded and their relative information.

3. APPLYING ONTOLOGY DESIGN PATTERN FOR ALERTS CORRELATION

Traditional NIDS products raised a large amount of the events independently, while there may be inherent logical connections between them. Then it is fundamental to correlate the alerts and facilitate attack scenario analysis. Several approaches have been proposed to try to automate this procedure, although a definitive solution is far from being found. And we try to point out the complete sequence of the malicious actions performed. The Sequence relation are a natural part of the world to be modeled in ontologies and many applications require this kind of expression. Although OWL do not contain specific primitives for sequence relations, OWL do support sufficient machinery to express much of what one may want to represent about sequence relations (Presutti and Gangemi, 2008).

Generally speaking, our idea is to employ the constructed ontology to describe the patterns of known attacks scenario composed by temporal sequences of single events. And then we have to require new class and property definitions and add the additional restriction on the specific class for the temporal sequence. And the properties, including *hasPrior*, *hasPrior_immediate*, *hasNext* and *hasNext_immediate*, are the main components that will assign a temporal sequence relationship to a set of events. These properties, *hasPrior_immediate* and *hasNext_immediate* represent the direct relation in which the event happens one after the other in a sequence, and they are subproperties of *hasPrior* and *hasNext* respectively which just refer to the comparison of timestamp associated with the events. These four properties are defined as follows.

$$hasPrior\_immediate \equiv hasNext\_immediate^{-}$$
$$hasPrior \equiv hasNext^{-}$$
$$hasPrior\_immediate \sqsubseteq hasPrior$$
$$hasNext\_immediate \sqsubseteq hasNext$$

The *Event* class is both *domain and range* of these four properties, shown in Figure 6.

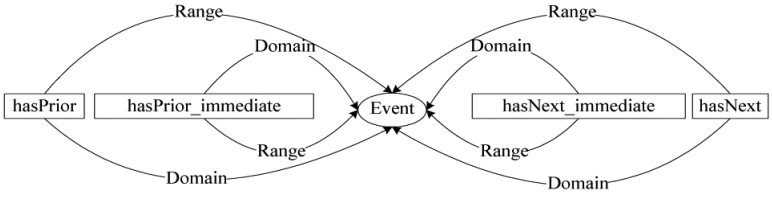


Figure 6: The Domain and Range of These Properties

hasPrior_immediate and *hasNext_immediate* are defined as functional properties, while *hasPrior* and *hasNext* are transitive.

$$Tr(hasPrior)$$
$$Tr(hasNext)$$
$$Event \sqsubseteq \leq 1 hasPrior\_immediate.Event$$
$$Event \sqsubseteq \leq 1 hasNext\_immediate.Event$$

Now we will add restrictions on the properties *hasPrior_immediate* and *hasNext_immediate* for each phase of the specific sequence, as shown in Table 8.

MISC-ACTIVITY_Event ≡	∃hasNext_immediate.ATTEMPTED-RECON_Event
ATTEMPTED-RECON_Event ≡	∃hasPrior_immediate.MISC-ACTIVITY_Event ⊓ ∃hasNext_immediate.SUCCESSFUL-RECON_Event
SUCCESSFUL-RECON_Event ≡	∃hasPrior_immediate.ATTEMPTED-RECON_Event ⊓ ∃hasNext_immediate.ATTEMPTED-ADMIN_Event
ATTEMPTED-ADMIN_Event ≡	∃hasPrior_immediate.SUCCESSFUL-RECON_Event ⊓ ∃hasNext_immediate.SUCCESSFUL-ADMIN_Event
SUCCESSFUL-ADMIN_Event ≡	∃hasPrior_immediate.ATTEMPTED-ADMIN_Event ⊓ ∃hasNext_immediate.ATTEMPTED-DOS_Event
ATTEMPTED-DOS_Event ≡	∃hasPrior_immediate.SUCCESSFUL-ADMIN_Event ⊓ ∃hasNext_immediate.SUCCESSFUL-DOS_Event
...	...

Table 8: The Temporal Sequence Restrictions on the Subclasses of Event

Furthermore we add the new class definition, *Alert* class, which expresses the actual intrusion alerts caused by the intrusion event. Accordingly the causal relationship between class *Alert* and class *Event* are defined as property *caused_by* of which the domain and range are class *Alert* and class *Event* respectively. The restrictions added for the alerts on the properties *caused_by* for each phase of the specific sequence is listed below in Table 9.

ATTEMPTED-RECON_Alert ⊑	∃caused_by.ATTEMPTED-RECON_Event
SUCCESSFUL-RECON_Alert ⊑	∃caused_by.SUCCESSFUL-RECON_Event
ATTEMPTED-ADMIN_Alert ⊑	∃caused_by.ATTEMPTED-ADMIN_Event
SUCCESSFUL-ADMIN_Alert ⊑	∃caused_by.SUCCESSFUL-ADMIN_Event
...	...

Table 9. The Restrictions on the Subclasses of Alert Acting along the Property caused_by

Benefiting from the sequence pattern, we have constructed the structure that can be applied in the user queries. The description is modeled in Figure 7.

For example, if we want any alerts caused by the events that happens before the *SUCCESSFUL-DOS_Event*, we define the equivalent class *ALERTQUERY* as follow.

$$ALERTQUERY \equiv Thing \sqcap \exists caused\_by.\ \exists hasNext.SUCCESSFUL\text{-}DOS\_Event$$

We will use the reasoner to automatically infer the subclasses of the class *ALERTQUERY* shown in Figure 8.

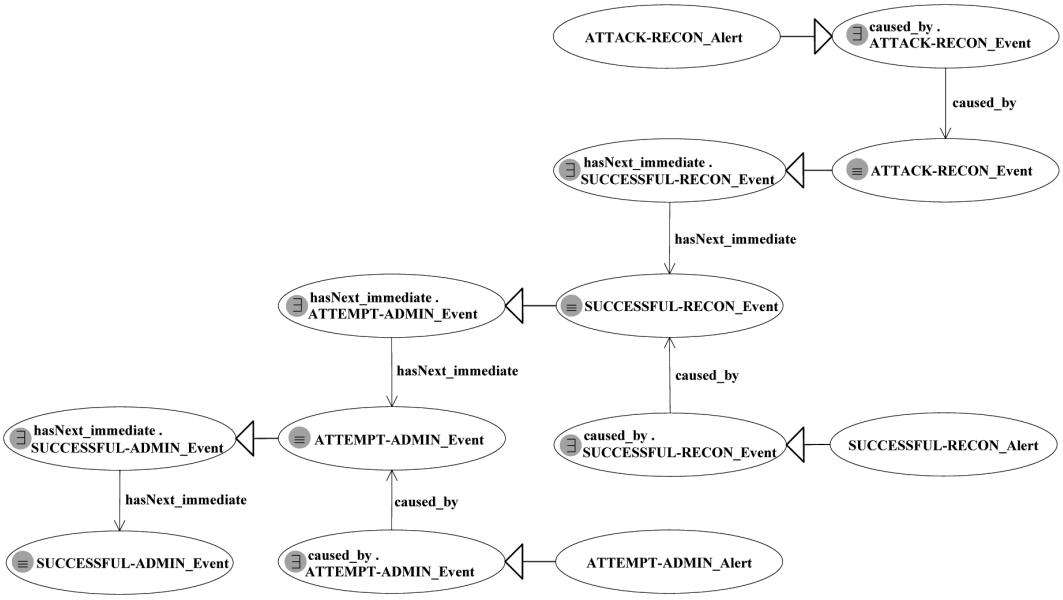


Figure 7: The Sequence Pattern of Ontology Design

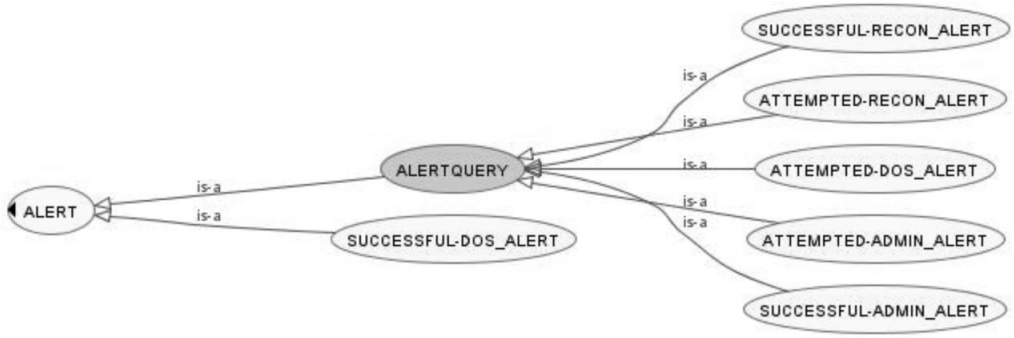


Figure 8: The Subsumed Class Hierarchy of ALERTQUERY

4. IMPLEMENTATION

After finishing the transitioning the intrusion database to ontology, we have implemented the ontology based NIDS management system in Java with the Jena API framework. The Jena API framework is an open source package to facilitate other applications to access the ontology. It is essential for the application to deduce the subsumed security facts from the knowledge base depending on the third party OWL reasoners which all provides DIG interfaces, such as pellet, racer or fact++(DIG).

The developed ontology based NIDS management system, namely OBNAMS, consists of four main layers, as shown in Figure 9.

- *User interface layer*

  It provides an interface where a user can submit a customized query and contains a Web application where the user can interact with the system. The user request will be sent to the business logic layer and the layer emphasizes on the representation of the analysis result.

- *Business logic layer*

  The developed Java Servlets are the principal components of this layer. Using the Jena Framework, it has been built up to access the knowledge bases and respond to the user query request. We import the appropriate reasoner by the DIG interface accompanied with them so as to infer the new security facts from the model.

- *Ontology layer*

  A basic principle in the design of this layer is to allow the derivation of an enriched ontology in a fully automated way. The user can get a populated ontology by simply updating the knowledge base with the relevant NIDS source data. The knowledge base is mainly composed of *TBOX* and *ABOX*. *TBOX*, known as Terminological Box, contains the formal representation of the conceptual mode. And *ABOX*, namely Assertion Box, contains the formal representation of the concrete model (Baader and Calvanese, 2003).

- *Data layer*

  The relational database is the main source data generated by the specific configured NIDS sensors. The information contained in the every specific instance of the database differs from each other in the context of the attack signatures classification. Apart from the relational database, the external vulnerabilities database with XML or text format, such as NVD, BUGTRAQ and so on, are always parsed to refine the classification of attack signatures class.

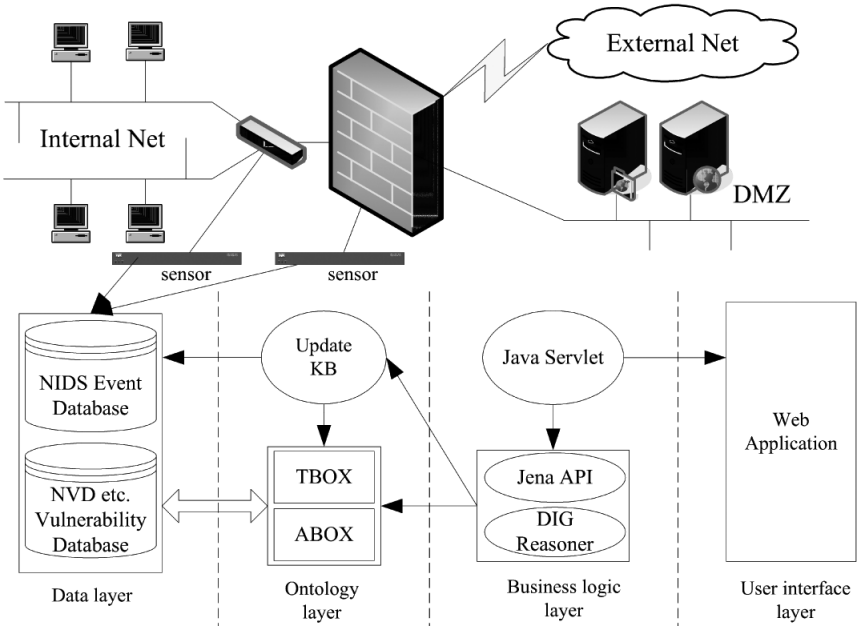


Figure 9: The Infrastructure of OBNAMS

5. EVALUATION

In this section, we show the experiment to evaluate the usefulness of the OBNAMS and its ability to perform semantic query. The experiment was conducted with the dataset which is generated upon the DARPA 2000 intrusion detection scenario-specific datasets (DARPA, 2000).

The 2000 DARPA intrusion detection scenario-specific datasets include LLDOS 1.0 and LLDOS 2.0.2. Each of them includes the network traffic collected from both the DMZ and the inside part of the evaluation network. LLDOS 2.0.2 includes a similar sequence of attacks run by an attacker who is a bit more stealthy than the first one in LLDOS 1.0. We chose LLDOS 2.0.2 for the experiment and downloaded the tcpdump file of LLDOS 2.0.2 from the website. We uncompressed it and gave Snort the package capture to read. And the Snort was appropriately configured to save all the reported alerts into the database and our dataset is obtained.

5.1 Effectiveness of OBNAMS

Our first aim of the experiment is to evaluate the effectiveness of the method incorporated in our framework. Firstly, we will report the results of OBNAMS learning ontology from the dataset contained in the Snort database. The Snort database obtained from the LLDOS 2.0.2 is strongly structured containing 14 relations and 2412 security alerts in the relation Event. Table 10. shows the outcome of applying OBNAMS to Snort database schema in company with its content. There are 10 classes mapped from the relations, some of which can be furthermore mined to generate the subclasses, such as class *Signature,* class *Sig_class*, class *Event* and so on. And security knowledge base, NVD vulnerabilities database, has been analyzed for the learning process. The extracted classes, including class *Severity*, class *Range*, class *Loss_types* and class *Vuln_types*, are the results of parsing the NVDCVE 2.0 with XML format (NVD). In addition, totally 30 Object Properties has been extracted in the process, which contain the inverse properties of themselves. And conveniently we obtained 55 Datatype Properties based on Rule 3. There are 2486 generated instances which have been dispatched into a variety of classes and its subclasses.

Class Name	Subclass Count	Instance Count
Event	17	2412
Signature	17	19
Sig_class	6	6
Severity	3	3
Range	3	1
Loss_types	4	4
Vuln_types	5	3

Table 10: Some Statistical Information about the OBNAMS Effectiveness

5.2 Performance of OBNAMS

Afterwards, we tested the performance of all the modules which perform the learning task in the OBNAMS and made some time measurements (in ms). These results are summarized in Table 11. We used the computer system, DELL PowerEdge 840 on an Intel Pentium D 3.0 GHz processor with 2GB of RAM, as the application server. Given that the Pellet is an open-source Java based OWL DL reasoner, we chose Pellet 2.0 as the reasoner running locally on the server (Pellet).

Almost all the modules had a good performance, as shown in Table 11. But there is a significant time expenditure on the occasion of updating class *Event* and its instances, as observed by us. The

population of the subclasses and instances of the class *Event* from the initial dataset is time-consuming, because the initial obtained dataset has a large quantity of security event records . Since the incremental update has been implemented in the system, the performance of module synEvent is satisfactory. Apart from this, the classifiction of the ontology and query on the inferred knowledge will be done with excellent performance.

Module Name	The Time	Description
synSensors	16 ms	Update the sensor class and its instances
synSig_class	109 ms	Update the Sig_class Class and its instances
synSignature	4985 ms	Update the Signature Class and its instances
synEvent	283937 ms	Update the Event Class and its instances
synRef_system	15 ms	Update the Ref_system Class and its instances
synReference	47 ms	Update the Reference Class and its instances
classifyOntology	1562 ms	Perform the classification on the updated ontology
queryInferredAlerts	17 ms	Query the defined security alerts on the classified ontology

Table 11: Time Measurements for Various Learning Module of OBNAMS

6. RELATED WORK

A large proportion of the early work on reverse engineering of the relational database has been done on extracting *entity-relationship* and object models from them (Chiang,1994; Premerlani and Blaha, 1994; Vermeer, 1995; Ramanathan and Hodges, 1997). They are still beneficial to ontology learning from relational database which viewed the ontology as the goal of the reverse engineering. To facilitate migration of data-intensive web sites into the semantic web, an approach was given mapping relational database schemas into ontology that can form the RDF(S) ontology (Stojanovic, 2002). The method for ontology population is proposed to automate the process of generating the instances and property values of an ontology in accordance with the extracted relational data (Rubin and Hewett, 2002). The transformation model was presented, describing how the conceptual data model can be transformed to ontology using graph formalism (Justas and Olegas, 2007). Within a mapping process, the method was illustrated to build OWL ontology from conceptual database schemas. As is known to us, OWL is a more expressive ontology language founded on Description Logics (Li, 2005). In the above method, much effort is put into the mapping rules from the relational database schema to ontology. Data and attribute correlations, however, are considered rarely and have received little analysis. Then a novel approach, which is based on an analysis of key, data and attribute correlations, as well as their combinations, is proposed to resolve the problems (Astrova, 2004). A learning method that shows how the content of the databases can be exploited to identify categorization patterns from which class hierarchies can be generated. This fully formalized method combines a classical database schema analysis with hierarchy mining in the stored data (Cerbah, 2010).

With the growing deployment of host and network intrusion detection systems, managing reports from these systems becomes critically important. A probabilistic approach to alerts correlation provides a unified mathematical framework for correlating alerts (Valdes and Skinner, 2001). And the alerts clustering methods correlates alerts based on the similarities between alert attributes (Cuppens, 2001). They cannot fully discover the causal relationships between related alerts. The attack description language is described, which is based on logic and uses a declarative approach. The various steps of the attack process are associated to events, which may be combined

using specific algebraic operators (Cuppens, 2000). An algorithm is proposed to combine the alerts into scenarios which is composed of a sequence of alerts (Dain and Cunningham, 2001). A limitation of these methods is that they are restricted to known attack scenarios. Rather than think of attacks as a series of events, JIGSAW view attacks as a set of capabilities that provide support for abstract attack concepts that in turn provide new capabilities to support other concepts (Templeton and Levitt, 2000). However JIGSAW requires all required capabilities be satisfied, and the method which allows partial satisfaction of prerequisites is presented (Ning, 2004). A tool using the presented general correlation model has been applied to a number of well-known intrusion detection datasets to identify how each component contributes to the overall goals of correlation (Fredrik, 2004). A formal description was given about attack scenarios and the system architecture was proposed to generate an attack scenario database correctly and completely (Cheng, 2007). With the advent of semantic web and semantic technologies, ontologies have increasingly been employed to the study of the intrusion detection. To extract semantic relations between computer attacks and intrusions in a Distributed Intrusion Detection System, an ontology is presented and used through the process (Abdoli, 2009). In order to achieve the collaborative intrusion detection several security ontologies were developed and applied in the applicability of our approach with a case study illustrating the Mitnick attack (Artem, 2010).

Our work is built upon the above mentioned method. The learning framework largely employed the database reverse engineering approach which proposes a set of mapping rules from relational database to ontology. The approach mining the content for the refinement of the ontology structure also plays an important role in the transitioning process, and the measure evaluating the Candidate Categorizing Attributes is mainly adapted from the work of RTAXON learning method (Cerbah, 2010). The intrusion alerts correlation is not new in the area of intrusion detection research. Under the guidance of the previous alerts correlation work, we apply the ontology design pattern to represent the sequence of a series of events.

7. CONCLUSION AND FUTURE WORK

Our contribution lies in the learning framework which migrates the application from the relational database to ontology. The framework employed several kinds of method comprehensively, which takes into account both the database schema and content contained in the database. Moreover, the information from the vulnerabilities database has been parsed and imported into the ontology to refine the ontology hierarchy and enrich the restriction of classes on specific properties. The intrusion alerts correlation is not new in the area of intrusion detection research. Under the guidance of the previous work, we apply the ontology design pattern to represent the sequence of a series of events. Finally, we implement the *OBNAMS*, an intrusion alerts management system constructed on the learned ontology fulfilling the whole process in an automated way.

Future research will be conducted to investigate how to acquire the attack scenario knowledge and import them into our ontology in association with the well-known attack scenario knowledge base. These attack scenario knowledge has been considerably represented as the First-order logic with the XML or TEXT format to denote the causal relationship between the attack steps.

In our future work, moreover, Description Logic learning will be introduced and applied to learn the implicit concept, restriction even axiom so that we can express the security interesting pattern in an automated way.

ACKNOWLEDGEMENTS

This work is supported by Natural Science Foundation of Chongqing City in China under Grant No.2011BA2022, the NSF of China under Grant No. 60803027, and the National S&T Major Project of China under Grant No. 2009ZX07315-006.

We also thank the anonymous reviewers for their helpful comments.

REFERENCES

ABDOLI, F. and KAHANI, M. (2009): Ontology-based distributed intrusion detection system, 14th International CSI Computer Conference (2009), 65–70, Tehran, Iran.

ARTEM V. (2010): An ontology-driven approach applied to information security, *Journal of data and knowledge engineering, Journal of Research and Practice in Information Technology*, 42(1): 61–76.

ASTROVA, I. (2004): Reverse engineering of relational databases to ontologies. *1st European Semantic Web Symposium*, LNCS 3053:37-341. Heraklion, Crete, Greece.

BAADER, F., CALVANESE, D., McGUINNESS, D.L., NARDI, D. and PATEL-SCHNEIDER, P. (2003): *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press.

BUGTRAQ (1993): http://www.securityfocus.com/

CERBAH, F. (2010): Learning ontologies with deep class hierarchies by mining the content of relational databases. *Advances In Knowledge Discovery and Management*, 292: 271–286.

CERT/CC (1999): Computer Emergency Response Team/Coordination Center, http://www.cert.org

CHENG, Y., CHEN, C., CHIANG, C., WANG, J. and LAIH, C. (2007): Generating attack scenarios with causal relationship. *IEEE International Conference on Granular Computing (2007)*, 368–373, San Jose, CA, USA.

CHIANG, R., BARRON, T. and STOREY, V. (1994): Reverse engineering of relational databases: Extraction of an EER model from a relational database. *Journal of data and knowledge engineering*, 12(2): 107–142.

CUPPENS, F. and ORTALO, R. (2000): LAMBDA: A language to model a database for detection of attacks. *The Recent Advances in Intrusion Detection (RAID 2000)*: 197–216, Toulouse, France.

CUPPENS, F. (2001): Managing alerts in a multi-intrusion detection environment. *17th Annual Computer Security Applications Conference*, New Orleans, Louisiana.

CVE (1999): Common Vulnerabilities and Exposures, http://cve.mitre.org

DAIN, O. and CUNNINGHAM, R. (2001): Fusing a heterogeneous alert stream into scenarios. *ACM Workshop on Data Mining for Security Applications (2001)*, Philadelphia, PA, USA.

DARPA 2000 (2000): DARPA Intrusion Detection Scenario-Specific Data Sets, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000data.html

DIG (2004): DL Implementation Group, http://dl.kr.org/dig/

FREDRIK V., GIOVANNI V., CHRISTOPHER K. and RICHARD A.K. (2004): A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169.

JUSTAS, T. and OLEGAS, V. (2007): Building ontologies from relational databases using reverse engineering methods. *International Conference on Computer Systems and Technologies (CompSysTech'07)*, Rousse, Bulgaria.

LI, M., DU, X. and WANG, S. (2005): Learning ontology from relational database. *4th International Conference on Machine Learning and Cybernetics*, Guangzhou, China.

MEHMED K. (2002): *Data Mining: Concepts, Models, Methods, and Algorithms*. Wiley-IEEE Press, 360.

NING, P., CUI, Y., REEVES, D.S. and XU, D. (2004): Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security*, 7(2): 274–318.

NVD (2002): National Vulnerability Database, http://nvd.nist.gov/

OWL 2 (2009): OWL 2 Web Ontology Language, http://www.w3.org/TR/owl2-overview/

PELLET (2008): OWL 2 Reasoner for Java., http://pellet.owldl.com

PREMERLANI, W. and BLAHA, M. (1994): An approach for reverse engineering of relational databases. *Communications of the ACM*, 37(5): 42–49.

PRESUTTI, V. and GANGEMI, A. (2008): Content ontology design patterns as practical building blocks for web ontologies. *27th International Conference on Conceptual Modeling (ER2008)*, LNCS 5231, 128–141, Barcelona, Catalonia, Spain.

RAMANATHAN, S. and HODGES, J. (1997): Extraction of object-oriented structures from existing relational databases. *ACM SIGMOD 1997*, 26(1): 59–64, Tucson, Arizona.

RUBIN, D.L., HEWETT, M., OLIVER, D.E., KLEIN, T.E. and ALTMAN, R.B. (2002): Automatic data acquisition into ontologies from pharmacogenetics relational data sources using declarative object definitions and XML. *7th Pacific Symposium on Biocomputing*, 7:88–99. Lihue, Hawaii, USA.

STOJANOVIC, L., STOJANOVIC, N. and VOLZ, R. (2002): Migrating data-intensive Web Sites into the Semantic Web. *ACM Symposium on Applied Computing (SAC 2002)*, 1100-1107, Madrid, Spain.

TEMPLETON, S. and LEVITT, K. (2000): A requires/provides model for computer attacks. *New Security Paradigms Workshop (2000)*, 31–38, Cork, Ireland.

VALDES, A. and SKINNER, K. (2001): Probabilistic alert correlation. *4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*: 54–68, Davis, CA, USA.

VERMEER, M. and APERS, P. (1995): Object-oriented views of relational databases incorporating behaviour. *4th International Conference on Database Systems for Advanced Applications*, 26-35, Singapore.

BIOGRAPHICAL NOTES

Chunlei Fu

Chunlei Fu has received a BS in computer science from Henan Institute of Science and Technology, China, in 2003. He received his MS (by research) in computer science from Chongqing University, China, in 2007. He is currently pursuing a PhD in Chonqing University. His major research interests include ontology learning, semantic web technology, and information security. Chunlei Fu is also a researcher at Information and Network Management Center of Chongqing University that is dedicated into the research of information security and network management.

Qichang Duan

Qichang Duan is currently a full professor and the Director of Chongqing University-Rockwell Automation Lab at the school of Automation, Chongqing University, China. He is the senior member of China Automation Association and the member of China Automation Association Expert Consultation Committee. His research interests include Web-based synthesis intelligent control and information management, intelligent data mining and semantic web technology.

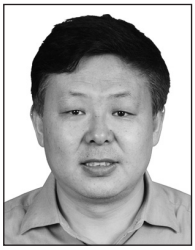
Li Fu

Li Fu is currently a full professor and the Chairman of Professors Committee at School of Software Engineering, Chongqing University. He is also Vice Chairman of Chongqing Institute of Electronics and the member of the Society for Industrial and Applied Mathematics. He is also the technical director of Information Security Technology Center in Chongqing City. His research interests include software engineering, information security, and semantic web technology.

Hong Xiang

Hong Xiang is currently a full professor at School of Software Engineering, Chongqing University. He is the Dean of Network Security Center at the School and the director of Information Technology Center of Chongqing City. He is also the member of Information Security Consultation Committee of Chongqing Science and Technology Committee. His research interests include network attack model, intrusion detection, and system survivability.

Zhongyang Xiong is the dean of Information and Network Management Center of Chongqing University. He is also the director of Chongqing Computer Federation and a member of the expert group of National Applied Information Technology Certificate Test in Chongqing. His research interests include data warehouse, data mining, and information security.



Zhongyang Xiong

Haibo Hu is currently a full lecturer at school of Software Engineering, Chongqing University. He is currently pursuing a PhD at Chonqing University. His major research interests include verification of software requirements and information security.

Haibo Hu