Expert Systems with Applications 37 (2010) 7138–7146

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# An ontology-based intrusion alerts correlation system

Wan Li [*], Shengfeng Tian

*School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*

ARTICLE INFO

*Keywords:*
Alert correlation
Intrusion detection
Ontology
System integration and implementation

ABSTRACT

Alert correlation techniques effectively improve the quality of alerts reported by intrusion detection systems, and are sufficient to support rapid identification of ongoing attacks or predict an intruder's next likely goal. In our previous work, an alert correlation approach based on our XSWRL ontology has been proposed. This paper focuses on how to develop the intrusion alerts correlation system according to our alert correlation approach. At first, the multi-agent system architecture consisting of agents and sensors is shown. The sensors collect security relevant information, and the agents process the information. Then we present each modules of the system in detail. The State Sensor collects information about security state and the Local State Agent and Center State Agent preprocess the security state information and convert it to ontology. The Attack Sensor collects information about attack and the Local Alert Agent and Center Alert Agent preprocess the alert information and convert it to ontology. The Attack Correlator correlates the attacks and outputs the attack sessions.

## 1. Introduction

Despite many years' efforts on intrusion detection, current intrusion detection systems (IDSs) are still not perfect (Axelsson, 2000; Ning & Xu, 2004; Tsai, Hsu, Lin, & Lin, 2009; Yu & Frincke, 2007). The criticism of the weakness of present IDSs focuses on the following points.

Firstly, both the false positive rate and false negative rate of IDSs are often not acceptable. There are rarely IDSs that are able to completely find not only known attacks but also unknown attacks (Haines, Ryder, Tinnel, & Taylor, 2003).

Secondly, most of the IDSs focus on low-level attacks or anomalies; they cannot capture the logical steps or strategies behind these attacks. In other word, the alerts of IDS are too elementary and not enough accurate to be directly managed by a security administrator (Cuppens, 2001).

The last point is that the response capabilities of the current IDSs are weak. After detecting attacks, most IDSs do nothing except sending alerts. The analysis and responses of these attacks are the job that administrators have to do manually.

Alert correlation techniques present solutions for the first and second problem and lay the foundation for solving the last problem above (Valeur, Vigna, Kruegel, & Kemmerer, 2004). Several alert correlation techniques have been proposed in recent years to facilitate the analysis of intrusion alerts. Alert correlation, informally, is the process of discovering the relationships between alerts. Recent intrusion alert correlation techniques can be roughly classified into three categories.

The first method groups intrusion alerts into different clusters based on the similarity between the alerts (Dain & Cunningham, 2001a, 2001b; Debar & Wespi, 2001; Porras & Neumann, 1997; Valdes & Skinner, 2001). This method can rapidly process large number of alerts, and can correlate unknown attacks. However, this method depends on parameters selected by human experts, and is not suitable for fully discovering causal relationships between alerts.

The second method performs correlation according to predefined attack scenarios, which are patterns of known sequences of attacks consisting of individual attack steps (Debar & Wespi, 2001; Geib & Goldman, 2001; Morin & Debar, 2003). An approach has been proposed to learn alert correlation models by applying machine learning techniques to training data sets embedded with known intrusion scenario (Dain & Cunningham, 2001a, 2001b). This approach can automatically build models for alert correlation. However, it requires training in every deployment, and the resulting models may overfit the training data, thereby missing attack scenarios not seen in the training date sets.

The third method models attacks by specifying their preconditions (prerequisites) and postconditions (consequences), and then correlates alerts (i.e., detected attacks) together when an earlier alert's postcondition (partially) matches a later one's precondition (Cuppens & Miege, 2002; Ning, Cui, & Reeves, 2002; Ning, Cui, Reeves, & Xu, 2004; Ning & Xu, 2004). To do this, we have to understand all possible attacks very well in order to know their prerequisites and consequences. Therefore, this method highly

[*] Corresponding author. Tel.: +86 10 51685922.
*E-mail addresses:* lee_wan@sina.com, 05112060@bjtu.edu.cn (W. Li).

depends on the experience of human experts and is hard to deal with new types of attacks in time.

We noticed that each of the three methods has advantages and disadvantages. The first method is complementary with the second and the third methods, they can confirm each other. Because of its high speed, the first method can be used as a filter of the second and the third methods. Combining different methods is a good idea. Zhuge, Han, Ye, and Zou (2006) and Zhuge, Xu, and Pan (2004) defines an attack knowledge model which combines the second and the third methods. However, combining in the model is halfway. We also define a hierarchical compound alert correlation knowledge model which combines the second and the third methods (Li, Zhu, & Tian, 2008). However, our model is more comprehensive.

A model must be actualized in someway. Several alert correlation languages and technologies have been proposed (Cuppens & Ortalo, 2000; Eckmann, Vigna, & Kemmerer, 2002; Zhuge, Xu, & Pan, 2004), but most are not grounded in any particular taxonomy, hence their associated classification schemes are ad hoc and localized.

To mitigate the effects of this problem, Undercoffer, Joshi, and Pinkston (2003) suggests transitioning from taxonomies to ontologies. Gruber (1993) defines ontology as an explicit specification of a conceptualization. Ontologies, unlike taxonomies, provide powerful constructs that include machine interpretable definitions of the concepts within a domain and the relations between them. Ontologies, therefore, provide software systems with the ability to share a common understanding of the information at issue.

According to our hierarchical compound alert correlation knowledge model, we proposed an intrusion alert correlation approach based on Ontology (Li et al., 2008). In this paper, we focus on the intrusion alert correlation system implement.

Section 2 introduces our previous work. Section 3 introduces the alert correlation system architecture. Section 4 expounds how to generating security state ontology. Section 5 expounds how to preprocess alert. Section 6 expounds the Attack Correlator. Section 7 discusses the related work, and Section 8 concludes the paper.

## 2. Preliminary

The system in this paper is based on our previous work: the XSWRL (Extended Semantic Web Rule Language) (Li & Tian, 2008) and the alert correlation approach based on XSWRL Ontology (Li et al., 2008). In the following, we briefly describe them.

### 2.1. XSWRL

The OWL (Web Ontology Language) (McGuinness & Harmelen, 2004), whose formalization relies directly on description logic (DL) (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003), dates back to the World Wide Web Consortium (W3C) recommendation released on 10 February, 2004. The basic elements in OWL are class, property, restriction and individual. Although the ontology languages include a relatively rich set of class constructors, it provides much weaker constructors for roles. One way to overcome this expressive limitation of ontologies would be to extend it with rules (Antoniou & Bikakis, 2007; Horrocks, Patel-Schneider, Bechhofer, & Tsarkov, 2005).

The SWRL (Semantic Web Rule Language) (Horrocks et al., 2004) specification was submitted in May 2004 to the W3C. SWRL is a proposal for a Semantic Web rules-language, combining the OWL with the Rule Markup Language.

The XSWRL (Extended Semantic Web Rule Language) (Li & Tian, 2008) which we presented is an extension to SWRL. The variables in SWRL are universally quantified variables. Universally quantified variables are indicated prefixing them with a question mark (e.g., $?x$). XSWRL introduces existentially quantified variables to rules, so XSWRL can represent some knowledge that SWRL cannot represent. Existentially quantified variables are indicated prefixing them with an exclamation mark (e.g., $!x$).

Using this human readable syntax, a XSWRL rule asserting that a left shoe and a right shoe which have same color compose a pair of shoes would be written as:

$$LeftShoe(?x) \wedge RightShoe(?y) \wedge color(?x, ?z) \wedge color(?y, ?z) \Rightarrow Shoes(!a) \wedge hasPart(!a, ?x) \wedge hasPart(!a, ?y). \quad (1)$$

### 2.2. Alerts correlation based on XSWRL ontology

We proposed a hierarchical compound alert correlation knowledge model (Li et al., 2008) which combines prerequisites and consequences of attacks and predefined attack scenarios, and introduces hierarchy to view security information from different levels. In this model prerequisites and consequences of attacks are denoted with security states. The model combines the advantages of both alert correlation methods, and is more comprehensive.

A Hierarchical Compound Alert Correlation Knowledge Model (HCACKM) $M$ is a tuple of the form $M = \langle A, S, AF, AW, AD, AP, AC, AI, SF, SW\rangle$, where:

$A$ is a set of attacks.
$S$ is a set of security states (states for short) that $S = SS \cup SA$, where $SS$ is a set of system states and $SA$ is a set of attacker states.
$AF$ is a set of subclass relationships of attacks.
$AW$ is a set of part–whole relationships of attacks.
$AD$ is a set of early–late relationships of attacks.
$AP$ is a set of prerequisite state relationships of attacks.
$AC$ is a set of consequence state relationships of attacks.
$AI$ is a set of indirect correlation relationships of attacks. $AP$ and $AC$ imply $AI$. The indirect correlation relationships of the attacks are the early–late relationships of the attacks which are inferred from the prerequisite state relationships and the consequence state relationships of the attacks.
$SF$ is a set of subclass relationships of states.
$SW$ is a set of part–whole relationships of states.

Our alert correlation ontology frame is based on our hierarchical compound alert correlation knowledge model, and is represented in OWL. The frame defines some basic classes, properties and restrictions. The alert correlation ontology frame is shown in Fig. 1.

After further extending these basic classes and properties in the ontology frame, we can get a practical alert correlation ontology knowledge base. Generally, we should extend the frame according to attack scenarios.

Following we give an example of part–whole relationships of attacks. We know that attack PortProbe is composed of attack PortRequest and attack PortConnect, where PortProbe, PortRequest, and PortConnect are subclasses of Attack. If there are an instance $a$ of PortRequest and an instance $b$ of PortConnect, and their target addresses are the same, then there must exist an instance $c$ of PortProbe, and its target is the target of $a$ (or $b$) and its sources are the source of $a$ and $b$. It is represented in a XSWRL rule as

$$PortRequest(?x) \wedge target(?x, ?a) \wedge source(?x, ?b) \wedge PortConnect(?y) \wedge target(?y, ?a) \wedge source(?y, ?c) \Rightarrow PortProbe(!z) \wedge target(!z, ?a) \wedge source(!z, ?b) \wedge source(!z, ?c). \quad (2)$$

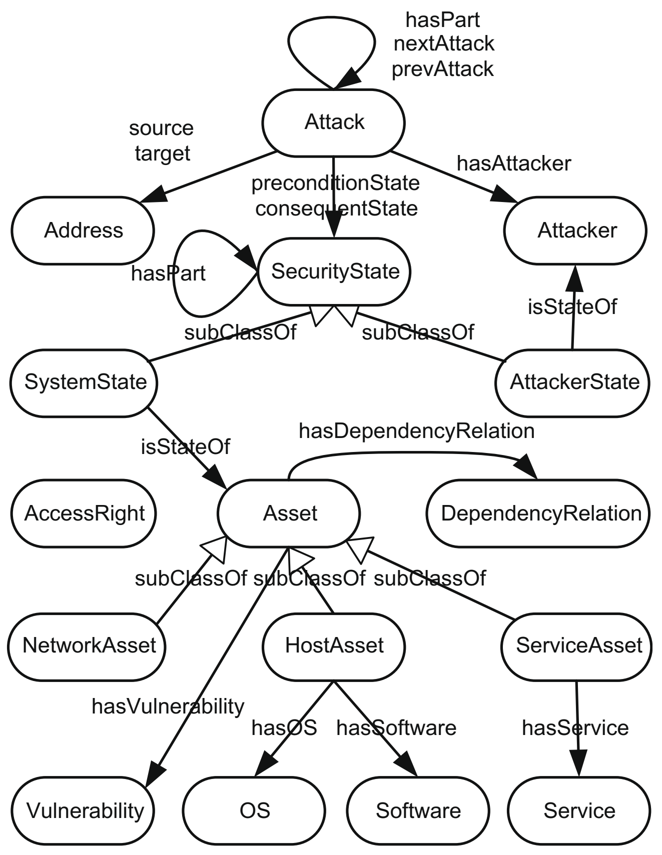

**Fig. 1.** Alert correlation ontology frame.

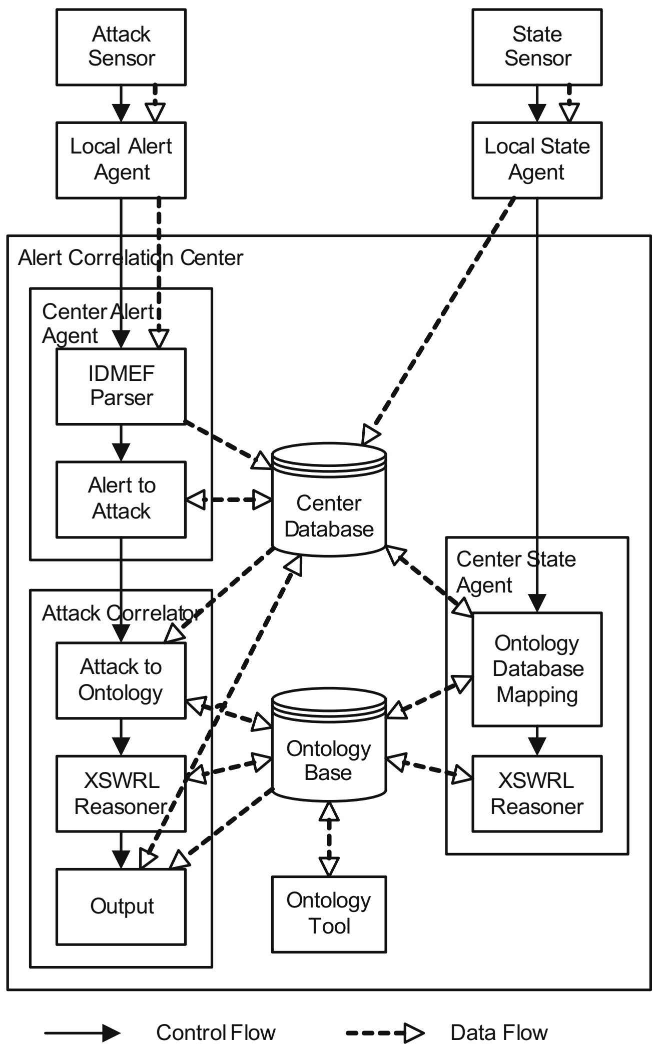

**Fig. 2.** Ontology-based intrusion alerts correlation system architecture.

## 3. System architecture

According to the alert correlation approach based on XSWRL Ontology, we have developed the ontology-based intrusion alerts correlation system. The system architecture is shown in Fig. 2.

We assume a multi-agent system architecture consisting of agents and sensors. A sensor typically refers to an IDS or a tool which can gather information about security state, such as vulnerability scanner, but it could be any information gathering program, device or somebody capable of collecting security relevant data. An agent is responsible for preprocessing security information or correlating alert based on data collected from a number of sensors. The multi-agent architecture has been chosen for its flexibility and scalability, in order to support future applications, such as distributed automated response.

State Sensor gathers information about security state. Local State Agent preprocesses the information and converts the information to the medium model data which will be conveniently converted to ontology by the Center State Agent. Each Local State Agent corresponds to a State Sensor.

Attack Sensor typically refers to IDS. Generally, there is at least one NIDS (Network-based IDS) in the system, and one HIDS (Host-based IDS) in each protected hosts. Each Local Alert Agent corresponds to an Attack Sensor. Local Alert Agent preprocesses the alert information generated by IDS and sends the reduced IDMEF (Debar, Curry, & Feinstein, 2007) alerts to the Center Alert Agent.

The Alert Correlation Center is composed of the Center Alert Agent, Attack Correlator, Center State Agent, ontology tools, and Center Database and Ontology Base. These modules of the Alert Correlation Center can be installed on one host or be distributed on different hosts.

Center State Agent converts the security state information in the medium model to ontology. It has two sub-modules: Ontology Database Mapping and XSWRL Reasoner. The Ontology Database Mapping converts data between the medium model and ontology. The XSWRL Reasoner is used to infer the security state from the security state relationship.

Center Alert Agent further preprocesses the alert information. It has two sub-modules: IDMEF Parser and Alert to Attack. The IDMEF Parser converts the alert information in IDMEF to data in the Center Database. The Alert to Attack converts alert information in the center database to attack information, and at the same time eliminates the redundant alerts produced by different IDS.

Attack Correlator has three sub-modules: Attack to Ontology, XSWRL Reasoner and Output. The Attack to Ontology converts the attack information in the Center Database to attack individuals in the ontology. The XSWRL Reasoner is used to infer attack process. An attack process is a series of attack steps which correspond to the attack individuals. The Output converts attack process from ontology to database for the use of follow-up, such as risk assessment.

The ontology tools are used to build the ontology and support reasoning. The ontology tools used in the system include Protégé (2009), a description logic reasoner (Baader et al., 2003) which support the DIG interface (DIG, 2009) (e.g., Racer (2009) and FaCT++ (2009)) and Jess (a rule engine) (Jess, 2009). Protégé is used to build and operate ontology (manual or with its API). The description logic reasoner and Jess are used by Protégé and our XSWRL reasoner.

Following we expound these modules in more detail.

## 4. Generating security state

A State Sensor typically refers to a tool which can gather information about security state, such as vulnerability scanner, but it could be any information gathering program, device or somebody capable of collecting security state relevant data. There are many types of State Sensors. They produce a wide variety of information formats. These formats are structured (e.g., databases), semi-structured (e.g., XML) and unstructured (e.g., plain text).

It is an important problem which must be resolved that how to convert the wide variety of information formats produced by the State Sensors to the ontology form. Considering the ontology technique has not yet been familiarized and used in a wide range, in order to make a person who knows less knowledge about ontology can do this converting work, we designed a relational database schema as a medium model. Through the medium model, the original information of the state sensors can be converted to the ontology form conveniently.

To build security state ontology includes the six steps, as shown in Fig. 3.

Following we expound these steps in more detail.

### 4.1. Extending security state ontology

Based on the ontology frame in Section 2, we extend SecurityState. Fig. 4 illustrates the class SecurityState and some of its subclasses.

The SecurityState class and its subclasses correlate to other classes by the isStateOf property and stateValue property. Subclasses inherit the restriction of its superclasses. The restrictions of the SecurityState class and some of its subclasses are shown in Table 1.

### 4.2. Mapping between ontology and relational database medium model

As we mentioned above, the purpose of relational database medium model is to conveniently convert the wide variety of information formats produced by the State Sensors to the ontology form. Therefore, in addition to as far as possible a high degree of

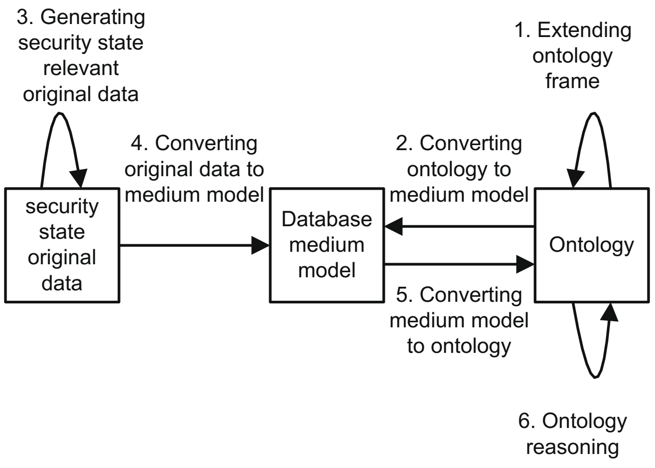
3. Generating security state relevant original data
1. Extending ontology frame
4. Converting original data to medium model
2. Converting ontology to medium model
security state original data
Database medium model
Ontology
5. Converting medium model to ontology
6. Ontology reasoning

**Fig. 3.** Steps of building security state ontology.

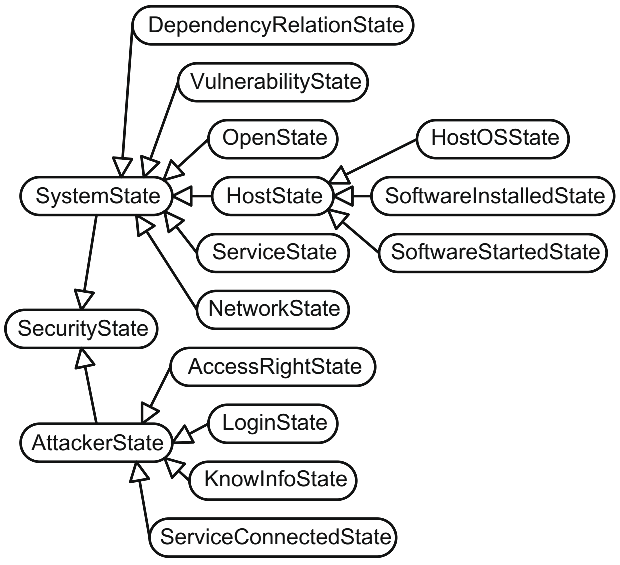

**Fig. 4.** SecurityState and its subclasses.

**Table 1**
Restrictions of SecurityState and subclasses.

| Class | isStateOf | stateValue |
|---|---|---|
| SecurityState | | |
| SystemState | Asset | |
| AttackerState | Attacker | |
| AccessRightState | | AccessRight |
| KnowInfoState | | SystemState |
| LoginState | | LoginInfo |
| ServiceConnectedState | | Service |
| DependencyRelationState | Asset | DependencyRelation |
| OpenState | | |
| VulnerabilityState | Asset | Vulnerability |
| ServiceState | ServiceAsset | Service |
| HostState | HostAsset | |
| NetworkState | NetworkAsset | |
| SoftwareInstalledState | | Software |
| SoftwareStartedState | | Software |
| HostOSState | HostAsset | OS |

standardization (for example, to meet 3NF or BCNF), the ideal medium model should also meet the following principles.

(1) *Easier to understand: if* the model is not intuitive, it will bring into difficulties.
(2) *Structural stability: that* is to say that change in ontology will not cause change in the database schema.
(3) *Query efficiency: it* is an important indicator to evaluate storage models.

The description logic is the logic basis of OWL. A description logic knowledge base (KB) comprises two components, the TBox and the ABox (Baader et al., 2003). The TBox introduces the terminology, i.e., the vocabulary of an application domain, while the ABox contains assertions about named individuals in terms of this vocabulary. The vocabulary consists of concepts, which denote sets of individuals, and roles, which denote binary relationships between individuals. In addition to atomic concepts and roles (concept and role names), all DL systems allow their users to build complex descriptions of concepts and roles.

The basic elements of OWL ontology are class (i.e., concept), two kinds of properties (i.e., role), i.e., object property and datatype property, and individual (instances of classes are individuals). In addition, OWL provides a number of language elements to enhance its power of expression. However, not all of the language elements must be used in the medium model. There are only seven tables in the medium model schema. These tables respectively correspond to the four basic elements of OWL, i.e., named classes, object properties, datatype properties and named individuals, and the three kinds of assertions, i.e., class assertions (i.e., instances of class), object property assertions and datatype property assertions. The medium model schema is shown in Fig. 5.

The medium model is easy to be understood, and has high structural stability. Whenever to modify some information in the ontology, it is only need to modify the corresponding records in the tables, but not need to modify the schema itself.

Since the medium model does not change with the ontology, so we can program to automatically convert data between the medium model and the ontology. When converting the ontology to the medium model, the four basic elements of OWL and the three kinds of assertions are all converted to the database. When converting the medium model to the ontology, only the three kinds of assertions are converted to the database, because the four basic elements of OWL are still in the ontology and need not be converted back.

### 4.3. Local State Agent

Each State Sensor needs special Local State Agent. The Local State Agent converts the original information produced by the State Sensor to the medium model.

The possible converting methods utilized by these Local State Agents can be divided into manual, semi-automatic and automatic methods. The manual methods are inefficient, but need not program; the automatic methods are efficient, but have to program; the semi-automatic methods are in the middle. From another perspective, the converting methods can be divided into general and special methods. The general methods are difficult, but have good adaptation; the special methods are just the opposite. It is lying on its corresponding State Sensor that a Local State Agent adopts which converting methods.

### 4.4. Infer security state from security state relationship

Some of the security state relevant original information generated by State Sensor (but not all of the information) has the form of relationship. The security state relationship is defined to represent it. Security state relationships and their respective counterparts of the security state are one-to-one. For example, the security state relationship hasVulnerability corresponds to the security state VulnerabilityState. The hasVulnerability is a subproperty of the securityStateRelationship, and the VulnerabilityState is a subclass of the SecurityState. With VulnerabilityState as an example, Fig. 6 illustrates the relationship between the security state relationship and the security state.

The securityStateRelationship is a binary relation, and the SecurityState is a class. We utilize the reasoning ability of XSWRL to infer the existence of an instance of SecurityState from an assertion of securityStateRelationship. It is written in the following XSWRL rule:

$$\begin{aligned} &\mathit{securityStateRelationship}(?x, ?y) \Rightarrow \mathit{SecurityState}(!z) \\ &\quad \wedge \mathit{isStateOf}(!z, ?x) \wedge \mathit{stateValue}(!z, ?y) \end{aligned} \tag{3}$$

The instance of SecurityState, which is inferred from the rule (3), needs to be reasoned to an instance of the more specialized subclass of SecurityState by the description logic reasoner. To this end, the property restrictions should be added to the subclasses of SecurityState. Taking VulnerabilityState as the example, the property restriction shows as follows:

$$\mathit{VulnerabilityState} \equiv \exists \mathit{isStateOf}.\mathit{Asset} \wedge \exists \mathit{stateValue}.\mathit{Vulnerability} \tag{4}$$

## 5. Alert preprocessing

### 5.1. Local Alert Agent

Local Alert Agent is an important link in distributed intrusion detection system environment. It provides the preprocessed alert data to other modules in the system. Its performance directly impacts on the effectiveness and timeliness of the whole system.

The main functions of Local Alert Agent are as follows:

(1) unifying different IDS alerts message format to IDMEF for data standardization;
(2) clustering alerts, merging duplicated alerts;
(3) sending reduced IDMEF alerts to the Center Alert Agent.

The flow diagram of Local Alert Agent is shown in Fig. 7.

The basic idea of reducing duplicated alerts is that when sensor reports alerts, the agent firstly puts the alerts in the buffer, then clusters alerts and merges duplicated alerts in the buffer, finally sends reduced alerts to the center. The quantity of the same alerts is more, the time which the alert stays in the buffer is longer (Mu, 2006).

The algorithm with self-adapting characteristics is shown in Fig. 8, where $St_i$ is the time which the alert $Alert_i$ (its alert type is $i$) stays in the buffer, $St_{ithreshold}$ is the time threshold which the alert type $i$ stays in the buffer, $St_{ithreshold} \in [St_{min}, St_{max}]$, $St_{min}$ and $St_{max}$ are the minimum and maximum of $St_{ithreshold}$ respectively. $St_{min}$, $St_{max}$ and $St_{ithreshold}$ have default value. $St_{ithreshold}$ is modified in the course of system operation, this makes the system self-adapting. $St_{max}$ can be used to control the amount of the alerts staying in the buffer. The two parameters of the alerts, i.e., the alert type $i$ and the source IP address of $Alert_i$, are used to judge whether the alerts are duplicated. If the two parameters of alerts are all the same, then these alerts are duplicated.

### 5.2. IDMEF parser

The purpose of the Intrusion Detection Message Exchange Format (IDMEF) (Debar et al., 2007) is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to the management systems that may need to interact with them. The most obvious place to implement the IDMEF is in the data channel between an intrusion detection analyzer (or "sensor") and the manager (or "console") to which it sends alerts. IDMEF document describes a data model to represent information exported by intrusion detection systems and explains the rationale for using this model. An implementation of the data model in the Extensible Markup Language (XML) is presented. The IDMEF data model is an object-oriented representation of the alert data. The data model of IDMEF is shown in Fig. 9.

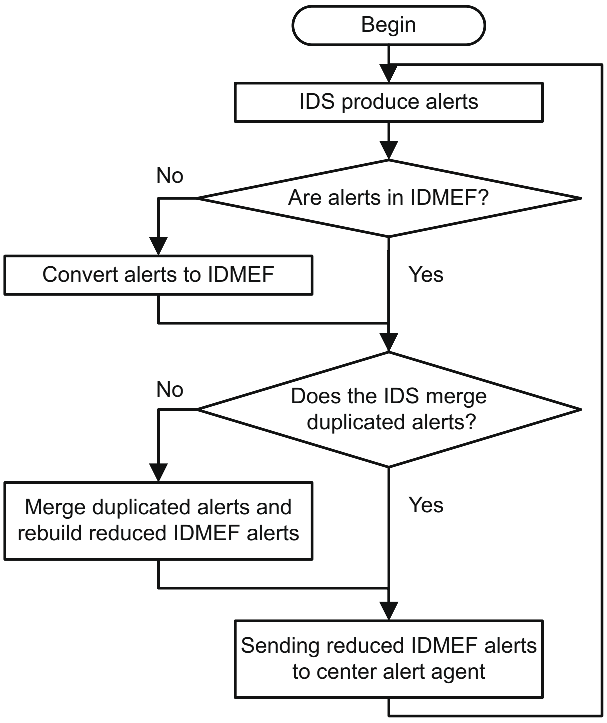
Begin
IDS produce alerts
Are alerts in IDMEF?
No
Yes
Convert alerts to IDMEF
Does the IDS merge duplicated alerts?
No
Yes
Merge duplicated alerts and rebuild reduced IDMEF alerts
Sending reduced IDMEF alerts to center alert agent

**Fig. 7.** Flow diagram of Local Alert Agent.

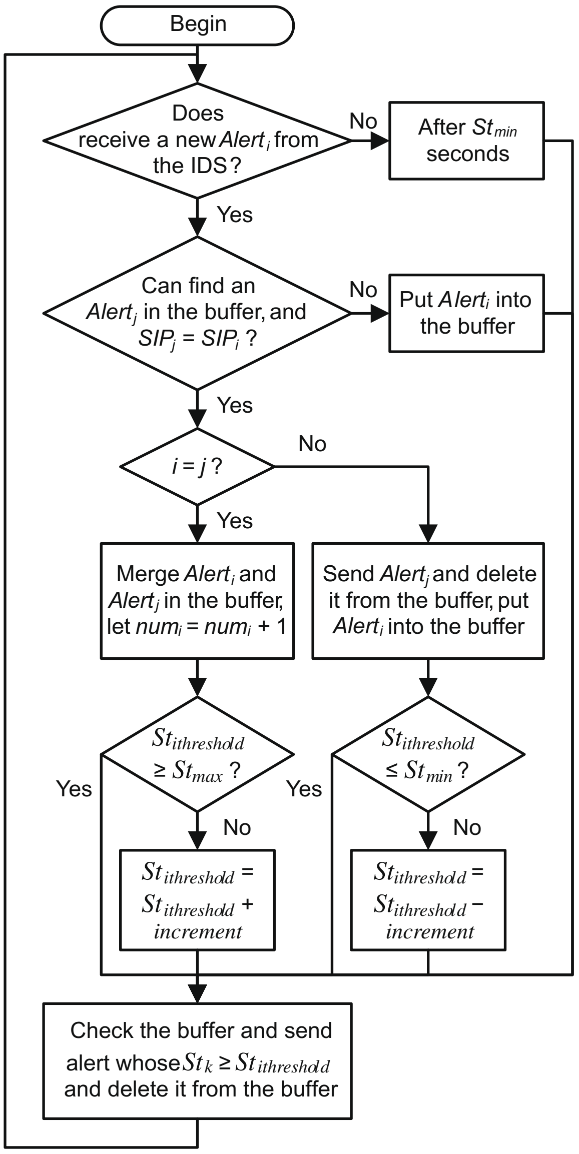
Begin
Does receive a new $Alert_i$ from the IDS?
No
After $St_{min}$ seconds
Yes
Can find an $Alert_j$ in the buffer, and $SIP_j = SIP_i$ ?
No
Put $Alert_i$ into the buffer
Yes
$i = j$ ?
No
Yes
Merge $Alert_i$ and $Alert_j$ in the buffer, let $num_i = num_i + 1$
Send $Alert_j$ and delete it from the buffer, put $Alert_i$ into the buffer
$St_{ithreshold} \geq St_{max}$ ?
Yes
No
$St_{ithreshold} = St_{ithreshold} + increment$
$St_{ithreshold} \leq St_{min}$ ?
Yes
No
$St_{ithreshold} = St_{ithreshold} - increment$
Check the buffer and send alert whose $St_k \geq St_{ithreshold}$ and delete it from the buffer

**Fig. 8.** Algorithm of reducing duplicated alerts.

IDMEF is in XML format. In order to better storage and use IDMEF alert data, we convert IDMEF data into the relational database format data. The primary problem is to design a relational database which must cover all contents in IDMEF. A high degree of standardization (for example, to meet 3NF or BCNF) and query efficiency are also considered in the database design.

We use the dom4j (2009) to parse the IDMEF alerts documents. The dom4j is an easy to use, open source library for working with XML. The workflow of the IDMEF parser is as follows:

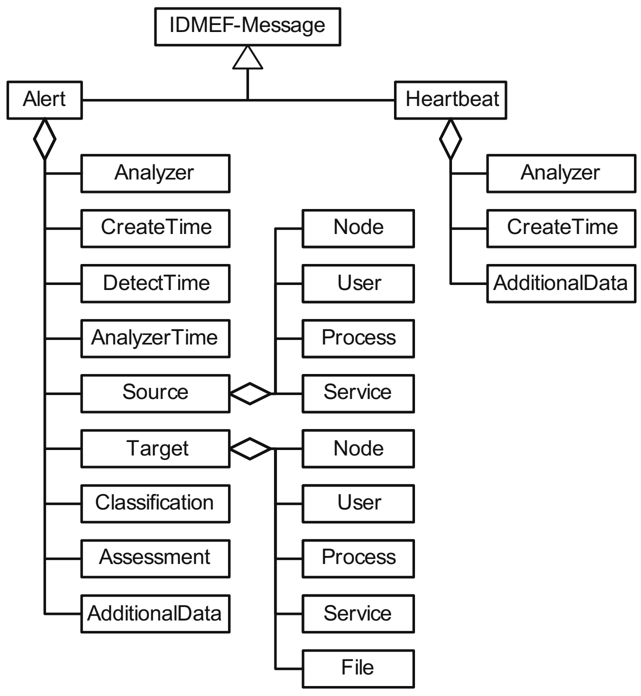

**Fig. 9.** IDMEF data model overview.

(1) Read IDMEF alerts.
(2) Parse IDMEF alerts, and build corresponding data objects according to the data model of IDMEF.
(3) Save data into the Center Database.

### 5.3. Alert to attack

This module converts alerts information in the Center Database to attacks information. Alerts are observations of attack actions. Different kind of IDS may produce different kind of alerts for the same attack, that is to say different kind of IDS may produce redundant alerts for the same attack. The main function of the module is mapping observational alerts to attack actions, at the same time eliminate the redundant alerts produced by different IDS.

The mapping alerts to attack may be predefined with expert knowledge; it may also be automatically built by vulnerability. Specific attacks often target specific vulnerabilities. Some alerts have vulnerability information and the vulnerability CVE (Common Vulnerabilities and Exposures) (CVE, 2009) identifiers are changeless. So the mapping alerts to attack may be automatically built by CVE identifiers.

CVE is a well-known vulnerability database. International in scope and free for public use, CVE is a dictionary of publicly known information security vulnerabilities and exposures. CVE's common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services.

The basic idea of the alert to attack algorithm is that

(1) if there is not alert type in an alert then automatically mapping the alert to attack by CVE identifiers;
(2) if there is alert type in an alert and there is mapping between alert type and attack type in knowledge base then mapping alerts to attack by expert knowledge;
(3) if there is alert type in an alert but there is not mapping between alert type and attack type in knowledge base then automatically mapping the alert to attack by CVE identifiers.

The information involved in the mapping mainly includes the type name, the CVE identifier, the source and target address, and the start and end time of alert and attack.

In the algorithm we design a self-learning switch. If self-learning switch is turned on, then the program can remember those mappings which are automatically built by CVE identifiers. These mappings may be analyzed and verified by experts and continued to use later.

## 6. Attack Correlator

### 6.1. Attack to ontology

At first, the Attack class is extended to add known attack scenario knowledge.

As an example, we take one of the 2000 DARPA intrusion detection scenario specific data sets, LLDOS 1.0 (DARPA, 2000), to illustrate attack ontology. LLDOS 1.0 contains a series of attacks, these attacks have been grouped into five attack phases, over the course of which the attacker probes the network, breaks into a host by exploiting the Solaris sadmind vulnerability, installs trojan mstream DDoS software, and launches a DDoS attack at an off-site server from the compromised host.

We extract attack types from LLDOS 1.0 scenario to build OWL classes of attacks. The hierarchy of attack ontology is shown in Fig. 10.

The input of this module is the output of the Alert to attack module. The output of this module is the attack ontology which is the input of the attack correlation module.

The format of attack data is standard (due to the IDMEF). In order to obtain a higher operating efficiency, the Attack to Ontology

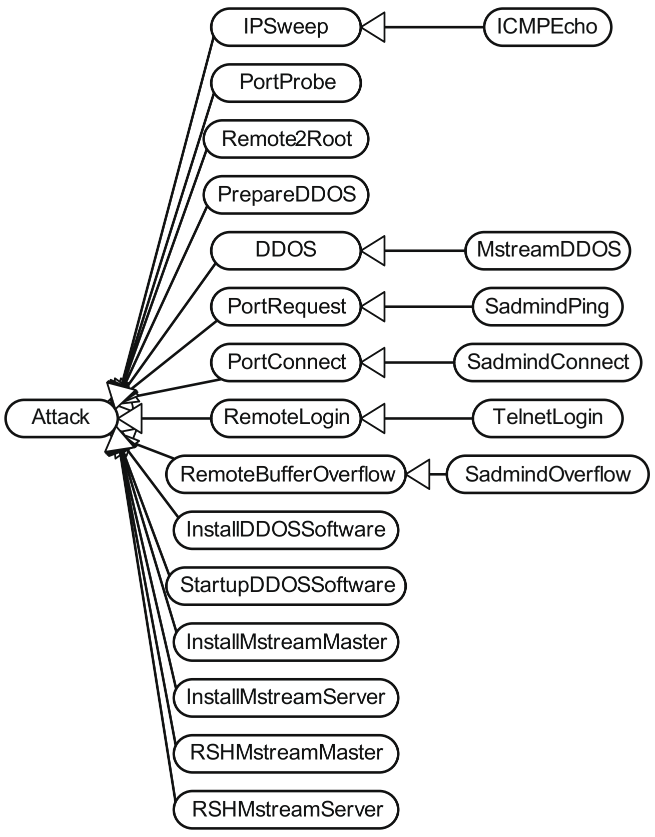

**Fig. 10.** LLDOS 1.0 attack classes hierarchy.

module has not used the medium model in Section 4, instead, it converts the attack data to ontology in a special procedure.

The algorithm of converting attack to ontology is relatively simple. The algorithm directly converts the attacks information (the type name, the source and target address, and the start and end time, etc.) in database into the instances of OWL classes and the property values according to attack type name.

In this module we also design a self-learning switch. If the switch is turned on, and if there is not a certain attack class in the ontology knowledge base, then the program automatically create the attack class. The attack class may be analyzed and verified by experts and continued to use later.

### 6.2. Attack Correlation

The Attack Correlator invokes the XSWRL Reasoner (Li & Tian, 2008) to automatically correlate attack individuals and security state individuals. The input of the XSWRL Reasoner is the established ontology. The ontology consists of two parts. The first part is the alert correlation ontology knowledge base established according to the method in Section 2, and is independent of concrete environment. The second part consists of the security state individuals and attack individuals, and is dependent of concrete environment. The output of the XSWRL Reasoner is the new ontology added the correlation information. The attacks in the same attack process are correlated to each other with part–whole relationships and early–late relationships. The attack individuals correlated to each other form an *attack session*.

### 6.3. Output

The result of attack correlation will be used by the following steps (such as risk assessment). The result in the ontology form is not convenient for the following steps to use. Therefore, the Output module converts the result in the ontology form to the relational database form to facilitate the use of the following steps.

Our risk assessment only need to use which attack individuals are in an attack session, but does not care about the relationships between the attack individuals. In order to simplify, the outputted attack sessions only contain the information which our risk assessment needs, namely an outputted attack session is a sets of the attack individuals which correlated to each other.

The method to obtain the outputted attack sessions from the attack correlation result is as follows: all attack individual as well as the relationships (i.e., part–whole relationships and early–late relationships) between them form a directed graph; ignore the direction of the arcs, the directed graph becomes an undirected graph; the vertex set of each connected component of the undirected graph is an outputted attack session. There is a classic algorithm to get the vertex set of each connected component of the undirected graph (Horowitz, Sahni, & Mehta, 1995). The Output module uses this classic algorithm.

## 7. Related work

There are several works on applying ontology to intrusion alert correlation. Raskin, Hempelmann, Triezenberg, and Nirenburg (2001) advocate the use of ontologies for information security. Contrary to ontologies, taxonomies lack the necessary and sufficient constructs needed to reason over instances of the modeled domain. Following Kemmerer and Vigna's suggestion that additional effort is needed to provide a common ontology that lets IDS sensors agree on what they observe Kemmerer and Vigna (2002), Undercoffer, Joshi, Finin, and Pinkston (2003) and Undercoffer et al. (2003)) proposed a target centric ontology for intrusion detection. Authors argue that an ontology should only model properties that are observable and measurable by the target of an attack. Yan, Hou, and Ansari (2006) propose an IDS autonomic event analysis system represented by description logics, which allows inferring the attack scenarios and enabling the attack knowledge semantic queries. However, these works do not consider the secure state which is important to judge false positive alerts and successful possibility of attacks.

## 8. Conclusion and future work

The paper shows the implement of our ontology-based intrusion alerts correlation system. The system is based on our hierarchical compound alert correlation knowledge model and XSWRL ontology technique.

We assume a multi-agent system architecture consisting of agents and sensors. The sensors collect security relevant information, and the agents process the information. The State Sensors and Attack Sensors collect information about security state and attack respectively. The Local State Agent and Center State Agent preprocess the security state information, and the Local Alert Agent and Center Alert Agent preprocess the alert information. The Attack Correlator correlates the attacks and outputs the attack sessions.

In our future work, how to get the wide variety of intrusion scenario knowledge will be considered. It is a significant idea that applying machine learning techniques to learn intrusion scenario knowledge. But by now, the most effective method is still depending on expert.

The ontology-based intrusion alerts correlation method is hard to deal with new types of attacks in real time, because the ontology knowledge base has to be updated to reflect the new types of attacks. However, the alert correlation method based on the similarity of alerts can rapidly correlate new types of attacks. How to further combine these alert correlation methods will be considered in our future work.

Because intrusion detection systems still usually produce a large number of false positive and false negative alerts, how to deal with false positive and false negative alerts will also be considered in our future work.

Moreover, in the future work, we are going to develop a bigger system, its function including the security information collection and preprocessing, attack correlate, risk assessment and automatic response.

## Acknowledgment

This work was supported in part by the Science and Technology Foundation of Beijing Jiaotong University under Grant No. 2006XM007, the Science and Technology Project of Beijing under Grant No. Z09050600910902 and the Science and Technology Foundation of Beijing Jiaotong University under Grant No. 2007RC066.

## References

Antoniou, G., & Bikakis, A. (2007). DR-Prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Transactions on Knowledge and Data Engineering, 19*(2), 233–245.

Axelsson, S. (2000). The base-rate fallacy and its implication for the difficulty of intrusion detection. *ACM Transactions on Information and System Security, 3*(3), 186–205.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2003). *The description logic handbook: Theory, implementation, and applications*. Cambridge University Press.

Cuppens, F. (2001). Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th annual computer security applications conference* (pp. 22–31). New Orleans, Louisiana, USA.

Cuppens, F., & Miege, A. (2002). Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE symposium on security and privacy* (pp. 202–215). Oakland, California, USA.

Cuppens, F., & Ortalo, R. (2000). LAMBDA: A language to model a database for detection of attacks. *Proceedings of the 3rd international workshop on the recent advances in intrusion detection, RAID 2000, Lecture notes in computer science* (Vol. 1907, pp. 197–216). Springer-Verlag.

CVE (2009). Common vulnerabilities and exposures (CVE). MITRE Corporation. <http://cve.mitre.org/> Accessed 25.06.09.

Dain, O. M., & Cunningham, R. K. (2001a). Building scenarios from a heterogeneous alert stream. In *Proceedings of the 2001 IEEE workshop on information assurance and security* (pp. 231–235). West Point, NY, USA.

Dain, O. M., & Cunningham, R. K. (2001b). Fusing a heterogeneous alert stream into scenarios. In *Proceedings of the 2001 ACM workshop on data mining for security application* (pp. 1–13). Philadelphia, Pennsylvania, USA.

DARPA (2000). 2000 DARPA intrusion detection scenario specific data sets, Lincoln laboratory scenario (DDoS) 1.0. <http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000/LLS_DDOS_1.0.html> Accessed 25.06.09.

Debar, H., & Wespi, A. (2001). Aggregation and correlation of intrusion-detection alerts. In *Proceedings of the 4th international symposium on recent advances in intrusion detection (RAID 2001). Lecture Notes in Computer Science* (Vol. 2212, pp. 85–103). Springer-Verlag.

Debar, H., Curry, D., Feinstein, B. (2007). RFC4765: The intrusion detection message exchange format (IDMEF). <http://www.ietf.org/rfc/rfc4765.txt>.

DIG (2009). <http://dl.kr.org/dig/> Accessed 25.06.09.

dom4j (2009). <http://www.dom4j.org/> Accessed 25.06.09.

Eckmann, S. T., Vigna, G., & Kemmerer, R. A. (2002). STATL: An attack language for state-based intrusion detection. *Journal of Computer Security, 10*(1), 71–104.

FaCT++, (2009). <http://owl.man.ac.uk/factplusplus/> Accessed 25.06.09.

Geib, C. W., & Goldman, R. P. (2001). Plan recognition in intrusion detection systems. In *Proceedings of the DARPA information survivability conference and exposition II. (DISCEX '01)* (Vol. 1, pp. 46–55). Anaheim, California, USA.

Gruber, T. F. (1993). A translation approach to portable ontologies. *Knowledge Acquisition, 5*(2), 99–220.

Haines, J., Ryder, D. K., Tinnel, L., & Taylor, S. (2003). Validation of sensor alert correlators. *IEEE Security and Privacy, 1*(1), 46–56.

Horowitz, E., Sahni, S., & Mehta, D. (1995). *Fundamentals of data structures in C++*. New York, NY, USA: W.H. Freeman.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. W3C member submission 21 May 2004. Latest version is available at <http://www.w3.org/Submission/SWRL/>.

Horrocks, I., Patel-Schneider, P. F., Bechhofer, S., & Tsarkov, D. (2005). OWL rules: a proposal and prototype implementation. *Journal of Web Semantics, 3*(1), 23–40.

Jess (2009). <http://herzberg.ca.sandia.gov/jess/> Accessed 25.06.09.

Kemmerer, R., & Vigna, G. (2002). Intrusion detection: A brief history and overview". *IEEE Computer (Supplement to Computer Magazine on Security and Privacy), 35*(4), 27–30.

Li, W., & Tian, S. (2008). XSWRL, an extended semantic web rule language. In *Proceedings of the 2008 international symposium on intelligent information technology application (IITA2008)* (Vol. 1, pp. 437–441). Shanghai, China.

Li, W., Zhu, Y., & Tian, S. (2008). Intrusion alerts correlation model based on XSWRL ontology. In *Proceedings of the 2008 international symposium on intelligent information technology application (IITA2008)* (Vol. 1, pp. 894–898). Shanghai, China.

McGuinness, D. L., & Harmelen, F. V. (2004). OWL web ontology language overview. W3C recommendation 10 February 2004. Latest version is available at <http://www.w3.org/TR/owl-features/>.

Morin, B., & Debar, H. (2003). Correlation of intrusion symptoms: An application of chronicles. In *Proceedings of the 6th international symposium on recent advances in intrusion detection (RAID2003). Lecture Notes in Computer Science* (Vol. 2820, pp. 94–112). Springer-Verlag.

Mu, C. (2006). Research on automated intrusion response system. Ph.D. dissertation, Beijing Jiaotong University, Beijing, China [in Chinese].

Ning, P., Cui, Y., Reeves, D. S., & Xu, D. (2004). Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security, 7*(2), 274–318.

Ning, P., Cui, Y., & Reeves, D. S. (2002). Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM conference on computer and communications security* (pp. 245–254). Washington, DC, USA.

Ning, P., & Xu, D. (2004). Hypothesizing and reasoning about attacks missed by intrusion detection systems. *ACM Transactions on Information and System Security, 7*(4), 1–37.

Porras, P. A., & Neumann, P. G. (1997). EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th national information systems security conference* (pp. 353–365). Baltimore, Maryland, USA.

Protégé (2009). <http://protege.stanford.edu/> Accessed 25.06.09.

Racer (2009). <http://www.racer-systems.com/> Accessed 25.06.09.

Raskin, V., Hempelmann, C. F., Triezenberg, K. E., & Nirenburg, S. (2001). Ontology in information security: A useful theoretical foundation and methodological tool. In *Proceedings of the new security paradigms workshop 2001* (pp. 53–59). Cloudcroft, New Mexico, USA.

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., & Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications, 36*(10), 11994–12000.

Undercoffer, J., Joshi, A., & Pinkston, J. (2003). Modeling computer attacks: An ontology for intrusion detection. In *Proceedings of the 6th international symposium on recent advances in intrusion detection (RAID 2003). Lecture notes in computer science* (Vol. 2820, pp. 113–135). Springer-Verlag.

Undercoffer, J., Joshi, A., Finin, T., & Pinkston, J. (2003). A target-centric ontology for intrusion detection. In *Proceedings of the 18th international joint conference on artificial intelligence* (pp. 47–58). Acapulco, Mexico.

Valdes, A., & Skinner, K. (2001). Probabilistic alert correlation. In *Proceedings of the 4th international symposium on recent advances in intrusion detection (RAID 2001). Lecture notes in computer science* (Vol. 2212, pp. 54–68). Springer-Verlag.

Valeur, F., Vigna, G., Kruegel, C., & Kemmerer, R. A. (2004). A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing, 1*(3), 146–169.

Yan, W., Hou, E., & Ansari, N. (2006). Description logics for an autonomic IDS event analysis system. *Computer Communications, 29*(15), 2841–2852.

Yu, D., & Frincke, D. (2007). Improving the quality of alerts and predicting intruder's next goal with hidden colored petri-net. *Computer Networks, 51*(3), 632–654.

Zhuge, J., Han, X., Ye, Z., & Zou, W. (2006). A network attack plan recognition algorithm based on the extended goal graph. *Chinese Journal of Computers, 29*(8), 1356–1366 [in Chinese].

Zhuge, J., Xu, H., & Pan, A. (2004). An attack knowledge model based on object-oriented technology. *Journal of Computer Research and Development, 41*(7), 1110–1116 [in Chinese].