Journal of Universal Computer Science, vol. 11, no. 8 (2005), 1367-1382
submitted: 10/3/05, accepted: 5/5/05, appeared: 28/8/05 © *J.UCS*

Network Attack Scenarios Extraction and Categorization by Mining IDS Alert Streams

Wei Yan
New Jersey Institute of Technology, USA
wy3@njit.edu

Abstract: The past few years have witnessed significant increase in DDoS attacks on Internet, prompting network security as a great concern. With the attacks getting more sophisticated, automatically reasoning the attack scenarios in real time and categorizing those scenarios become a critical challenge. However,the overwhelming flow of events generated by Intrusion Detection System (IDS) sensors make it hard for security administrators to uncover hidden attack plans. This paper presents a semantic vector space model to extract and categorize attack scenarios based on First-order Logics (FOL) and linguistics. The modified Case Grammar is introduced to formalize the heterogeneous IDS alerts into uniform structured alert streams. The attack resolution is then used to generate attack semantic network. Afterwards, mutual information is used to determine the alert semantic context range. Based on the attack ontology and alert contexts, attack scenarios are extracted and the alerts are represented as attack semantic space vectors. Finally text categorization technique are used to categorize the intrusion stages. The preliminary results show our model has better performance than the traditional alert correlations.

Key Words: network security, intrusion detection, first-order logics, resolution.
Category: I.2.4

1 Introduction

With the computer attacks have increased dramatically over the last several years, the network security became a critical issue with the development of the computer networks. Intrusion Detection System (IDS) has become an important tool to secure the networks by detecting, alerting and responding to malicious activity. Intrusion detection can be divided into misuse detection [Snort, Emerald], and anomaly detection [NADIR, NIDES]. Misuse detection works by searching for a set of known attacks that have been stored in the database. The knowledge of the attacks is encoded as a set of signatures, which are patterns that occur every time an attack takes place. Anomaly detection is based on a set of statistical details that model the behavior of traffic or host machines. It monitors the operations of network systems, and constantly compare the profiles with ones stored in its database. If it detects what it considers to be a large deviation from normal behavior, it signals an alarm. The contemporary commercial IDSs include both misuse detection and anomaly detection: ISS RealSecure [ISS], Cisco Secure IDS [CISCO], etc.,

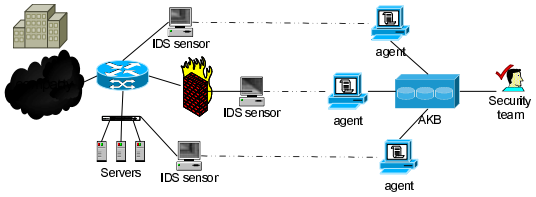


Figure 1: IDSs deployment with three layers of defense

Fig. 1 shows the IDS deployment in the network. First, traffic from the Internet enters the network through a router, where the external IDS sensor is placed. Afterwards, IDS sensor behind the firewall is used to defends the network. In [Wespi01], the aggregation and correlation component (ACC) was introduced, the purpose of which is to group the alerts into the duplication relationship and consequence relationship. Duplicate relationship is defined in the duplicate definition file. For example, to be considered a duplicate, the new alert's attributes must be equal to the previous alerts. The consequence relationships are defined in consequence definition file according to the causal relations. M2D2 [Dain01] included four information types in the alert correlation process: the monitored system, the known vulnerabilities, the security tools, and the alerts. A mapping function is used to convert the non-formal vulnerability names into the formal ones. Furthermore, the prerequisites are modeled as remote, remote user, and local. The consequences of the vulnerability are grouped as CodeExec, DoS, and Info. This kind of classification can be viewed as a preliminary attack ontology. As a result, M2D2 aggregated alerts as "caused by the same event" and "referring to the same vulnerability". In [Moring02], a new incoming alert was compared to the latest alerts in all existing scenarios, and then joined with the scenario with the highest probability score. However, this method requires the attack scenarios to be generated in advance by hands which may not be adapted to the diverse attacks and attack strategies.

However, several aspects are needed to be considered regarding current IDS technique. First, since every IDS may generate a huge volume of alert events, it is time-consuming and resource intensive for security administrator to analyze all alerts in time, which hampers the performance of the attack scenario extraction. Furthermore, when IDS alerts are collected from decentralized sensors, the major obstacle is the lack of universal alert description standard. IDS sensor may have its own alert format, making alerts correlation time-consuming and difficult. Intrusion Detection Message Exchange Format (IDMEF) was proposed to be a standard IDS alert format[IDMEF]. However, IDMEF is a syntax structure, which cannot make IDS alerts machine-understandable. Finally, nearly all of the proposed alerts correlation methods are based on syntax-oriented approaches,

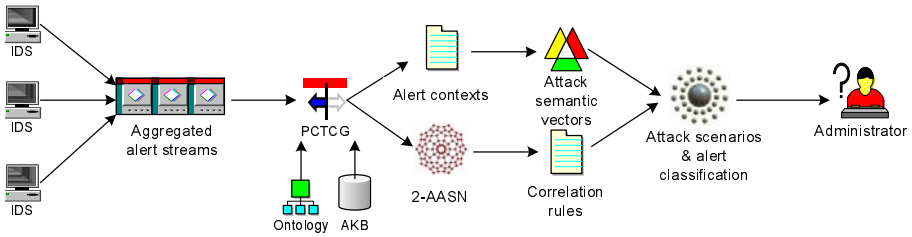


Figure 2: Processing of IDS Semantic Vector Space Model

making it impossible to infer the hidden attack scenario knowledge automatically.

Our current work is motivated by the needs to overcome the drawbacks mentioned above. In this paper, we exploit the semantics of attack behaviors, and presents the semantic vector space model to extract and classify the attack scenarios automatically. Section 2 describes how to use our modified Case Grammar to formalized alerts. Alert semantic network, 2-AASN, is introduced in Section 3. Section 4 describes the attack scenarios backward-chaining reasoning. Section 5 presents the simulation results, and Section 6 is the conclusion.

2 Alert Formalization

Fig.2 shows our IDS semantic vector space model. First, the number of duplicate raw alerts are decreased by aggregation according to same source IP address, target IP address, and same consecutive time slot. Afterwards, based on attack action-based semantic ontology and Attack Knowledge Bases(AKB) which store the semantic information of the alerts, Principal-subordinate Consequence Tagging Case Grammar(PCTCG) converters the aggregated alerts into uniform streams. Attack resolution is then applied to the PCTCG streams to generate 2 Atom Attack Semantic Network(2-AASN), where the correlation rules are applied to derive the attack scenarios. On the other hand, based on the alert context, the alerts are transformed into attack semantic vectors, from which text categorization techniques are applied to derive different intrusion stages. Finally, those the highly interpretable attack scenario results can be forwarded to the security administrator.

2.1 Principal-subordinate Consequence Tagging Case Grammar

The aim of PCTCG is to normalize the aggregated intrusion alerts into uniform semantic representation of attack behavioral actions. PCTCG is based on Case Grammar [Fillmore97]. The reasons for choosing Case Grammar are three

Sentence	Syntactic level		Semantic level	
	Object	**Subject**	**Agent**	**Theme**
The man moved the desk.	the man	the desk	the man	the desk
The desk was moved by the man.	the desk	the man	the man	the desk

Table 1: Syntactic vs. semantic

folds: First, Case Grammar structure specifies the semantic relations between a verb and its slots. Second, Case Grammar cab be easily represented by semantic network, which includes abundant semantic relations to express the alerts' associations. Third, unlike the syntactic level, Case Grammar theory is deep semantic, which means it does not change under the grammatical transformation. As shown in Table 2.1, *Subject* role and *Object* role change in syntactic level when the sentence switches from active to passive form. However, the *Agent* (*Agent* is what causes the action of the verb) and *Theme* (*Theme* is the object in motion or being located) roles in the semantic level remain the same.

Our assumption is that attack scenario can be regarded as a sequence of attack events, each of which includes a certain attack action. When consider two alerts (two actions), we use semantic roles to correlate them bi-literally. That is, we apply the *Principal-subordinate* relation on two alerts. When one alert is in the *principal* phase, we think it as a verb and replace the other alert with its subordinate keywords (noun phases). If the subordinate keywords is in a specific case relationship with the verb, these two alerts are correlated.

PCTCG is formally defined as $G = \{M_n, C, F, S\}$, where M_n is the alert messages set of the IDS sensor with sensor name n, C specifies the set of possible semantic roles (slots) between alerts, F is the set of case fillers (legal value for each slot), and S is subordinate keywords. In PCTCG, the semantic roles chosen should reflect the semantic logic of attack actions. We define PCTCG intrusion attack ontology, $\mathcal{O}_{\mathcal{T}}$, based on the following questions that security administrators would naturally ask: *When did the actions happened? Where did the actions happened? By which means did the actions happened? What results did the actions caused? etc.,* Fig. 3 presents the hierarchy of concepts and relations. Each concept in the ontology is described by a set of attributes. *Object* role means the receiving end of the action, and it has has object and be object of attributes. The *meronymy* (has an object) and *holonymy* (is a part of) attributes from part-whole role describe the situations that one entity contains other entity. For every alert, we define some subordinate keywords which can describe the alert background well.

In this paper, we use First-order Logic (FOL) [Brachman04] as the alert representing and reasoning language. In FOL, there are three types of logic symbols:

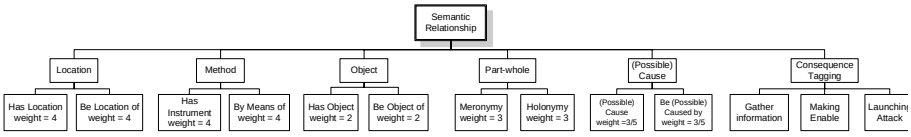


Figure 3: PCTCG Ontology

- punctuation: "(", ")", and "."
- connectives: "¬", "∀", "∃", " ∨ ", " ∧ ", *and* " ≡ ". Note that "¬" is logical negation, " ∧ " is logical conjunction, " ∨ " is logical disjunction, "∀" means "for all...", "∃" means "there exists...".
- predicate: Predicates denote the semantic roles defined in $\mathcal{O}_{\mathcal{T}}$.

Based on the alert semantic information, PCTCG streams of alerts: *FINGER 0 query* and *FINGER redirection attempt* are represented by predicate logic format:

$$E[M_n{:}(FINGER\ 0\ query)_{snort}]=$$

$$\lambda e.[\exists v[command(C{::}has\ object(FINGER\ daemon),third\ party,v)]\wedge C{::}possible\ cause$$

$$(User\ account,\ password)\wedge C{::}cause\ (FINGER\ command\ with\ username\ '0')$$

$$\wedge C{::}consequence\ tagging(launching\ attack)\wedge S{:}(Finger\ query,\ third\ party)].$$

$$E[M_n{:}(FINGER\ redirection\ attempt)_{snort}]=$$

$$\lambda e.[\exists v[forward(C{::}has\ object(FINGER\ query),third\ party,v)]\wedge C{::}possible\ cause\ (gain\ info)$$

$$\wedge C{::}cause\ (DDos,\ indirect\ connection)\wedge S{:}(Finger\ query,\ third\ party)].$$

where E is an entity described as "the event in which Finger daemon forward the query to the third party". Here, *has object, possible cause, cause, consequence tagging* are the semantic roles,*finger query, +info,DDoS, indirect connection, launching attack* fill the slots of the above roles respectively, and *FINGER query* and *thirty party* are the subordinate keywords. Predict logic describe the conjunction of the action predicate with other predicates described in the event.

2.2 Forward or Backward Chaining

A statement is called a Horn clause if it only contains at most one positive literal [Brachman04]. When there is only one positive literal in the clause, it is called the *positive* Horn clause. When there is no positive literal in the clause, it is called the *negative* Horn clause. Here, we are only interested in the *positive* Horn clause, because *positive* Horn clauses can be expressed as "if-then" statements. For example, a *positive* Horn clause $[\neg x_1, \neg x_2, y]$ can be thought as $x_1 \wedge x_2 \Rightarrow y$

or "if both x_1 and x_2 happen, then y must happen". By this way, we can write the production rules (correlation rules) with *positive* Horn clauses.

Backward or forward chaining are two primary methods for reasoning the knowledge from AKBs. Backward chaining starts with the hypothesis, and it works backward from the hypothesis to the facts which support the hypothesis. Forward chaining travels from the facts to the conclusions which follow from the facts. To determine whether backward or forward chaining should be used depends on the specific problems. If the facts in AKB can reach a large number of entailments, and few of which you are interested, then backward chaining should be adopted. Otherwise, forward chaining should be used. For the security administrator, what are known to him/her are the vulnerabilities of the networks and hosts, the malicious attacks being happening, and the generated alerts; he/she also wants to know if some specific attack attempts happened. Equivalently, our system stores the vulnerabilities of the networks and hosts in AKB in advance, and converts the generated alerts into streams, and then into the *positive* Horn clause facts. To extracting the attack scenarios, the system needs to check whether or not the AKB together with the production rules can satisfy WH-question related attack attempts: when, where, by which means,what results did the attack actions? With the semantic roles defined in $\mathcal{O}_{\mathcal{T}}$, we can pre-define the reasoning goals. Furthermore, to guarantee real-time efficiency, reasoning should be done without many unrelated conclusions. Therefore, we choose backward chaining.

In our simulations, the backward chaining language Prolog is used for logic programming in predicate logics. By backtracking, the inference engine provides Prolog with powerful reasoning capacity [Teft89]. A Prolog program $\mathcal{P}$ is a set of Horn clauses containing the facts $\mathcal{F}$, production rules $\mathcal{R}$, and goals $\mathcal{G}$. The general view of Prolog programs is that we are given a collection of $\mathcal{F}$ and $\mathcal{R}$ and wish to deduce $\mathcal{G}$ from them [Nerode97].

Production rules are expressed as conditional sentences in the form:

`IF semantic roles match THEN semantic correlation.`

The semantic attack knowledge, alert's frame structures, and production rules are the input clauses to Prolog, and the reasoning goal is equivalent to procedure call. For example, the related facts of alert *FINGER requery* can be represented in Prolog as clauses:

1. `object("finger_requery", "finger_daemon").`
2. `method("user_account", "finger_requery").`
3. `idsEvent("finger_0_query", "finger_daemon", "finger_daemon", "empty", "user_account", "empty", "empty", "finger_with_username_0" ).`

```
judge_object(PAlert, SlotFiller, SAlert, Keyword, SlotName) :-
writef("(%, %,%)", SlotName, SlotFiller, Keyword),
object(SlotFiller, Keyword).
```

3 First-order Semantic Correlation Resolution

3.1 Conjunctive Normal Form

In this section, we will expound in detail how to automate a attack reasoning procedure. In order to extract 2-AASN from alerts, we propose First-order Semantic Correlation Resolution First-order Semantic Correlation Resolution is based on the resolution for propositional Logic [Brachman04], which works on a normalized representation called Conjunctive Normal Form (CNF). CNF is a conjunctive of disjunctions of literals. For example, a CNF has the following form:

$$(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge (x_3 \vee y_3)$$

where is propositional logic clause

Because PCTCG streams are the predicate logic literals of conjunction format, they must be transformed to CNF before resolution. According to the DeMorgan's law, $(x_1 \wedge y_1) \vee (x_2 \wedge y_2) = (x_1 \wedge x_2) \vee (y_1 \wedge x_2) \vee (x_1 \wedge y_2) \vee (y_1 \wedge y_2)$, we distribute conjunction over disjunction. For alert 1 *FINGER 0 query* and alert 2 *FINGER redirection attempt*, their CNF are following:

$CNF(alert1,\ alert2)=$

$(has\ object(alert1,\ FINGER\ daemon) \wedge\ possible\ cause(alert1,\ User\ account)$

$cause(alert1y,\ FINGER\ command\ with\ username\ '0') \wedge S(alert1,\ Finger\ query))$

$\vee(has\ object(alert2,\ FINGER\ query) \wedge possible\ cause\ (alert2,\ gain\ info)$

$\wedge cause\ (alert2, DDos\ \&\ indirect\ connection) \wedge S{:}(alert2,\ Finger\ query\ \&\ third\ party))$

$=(has\ object(alert1,\ FINGER\ daemon) \wedge\ has\ object(alert2,\ FINGER\ query)) \vee \ldots,$

$\vee(S(alert1,\ Finger\ query)) \wedge\ S{:}(alert2,\ Finger\ query\ \&\ third\ party)).$

3.2 First-order Attack Resolution

In this section, First-order attack resolution is proposed to generate 2-AASN from alerts. First-order attack resolution works under the Principal-subordinate relation. When one alert is in the subordinate phase, if its subordinate keywords is in a specific relationship with the principle alert, these two alerts are correlated. The process that subordinate alert is replaced with subordinate keyword is called substitution.

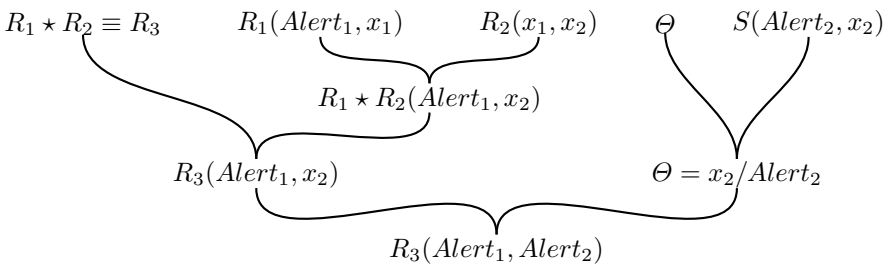


Figure 4: Resolution tree

Definition 3.1 *A substitution Θ is a set of pairs $\{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$, where $x_i, 1 \leq i \leq n$, is the subordinate keyword of alert t_i, and x_i/t_i means that keyword term x_i is substituted by its alert t_i throughout the resolution.*

Definition 3.2 *If there exists a fact $R_2(x_1, x_2)$, First-order Attack Resolution of two frame slots, $R_1(Alert1, x_1)$ and $S(Alert2, x_2)$, is defined as following:*

$$R_1(Alert1, x_1) \wedge S(Alert2, x_2) \models R_3(Alert1, Alert2)$$

where $R_1 R_2 \equiv R_3$, and the substitution Θ is $x_2/Alert2$.

Fig. 4 shows the process of First-order attack resolution on Alert1 and Alert2. Initially there are no connections between the case filler variables. When doing resolution between them, we generally rename variables so that they have on variables in common. Afterwards, we try to find if there exists semantic matching among case fillers, whose semantic roles R_1 and R_2 can be fused into R_3. Here $\star$ is semantic role operator. Table 3.2 shows results of these operations. Some semantic attributes cannot be operated, which are marked by $\emptyset$. If matching, the substitution Θ is set to replace subordinate keyword to the alert, because subordinate keyword describes the alert's attack background well. Fig.5 represents an example of resolution tree.

4 Ontology-based Alert Categorization

In Natural Language Processing (NLP), text categorization means the assignment of free text documents to one or more predefined categories based on their contents. A number of statistical classification and machine learning techniques has been applied to text categorization [Sebastiani02, Tong01]. In this section, mutual information is used to determine the alert semantic context range. Based

X	$X \star C = C$	$X \star C = X$	$X \star C = EE$	$X \star C = EB$
OH	OH,LH,LB,MH,MB,PB,CB	WM,WH	PC,CC	$\emptyset$
OB	OB,LH,LB,MH,MB,PC,CC	$\emptyset$	$\emptyset$	PB,CB
LH	OH,LH,LB,MH,MB,WM,WH	MB,WM,WH	PC,CC	$\emptyset$
LB	LB,MH	MH,WM,WH	PC,CC	PB,CB
MH	LH,LB,MH,PC,CC	LH,LB,WM,WH	$\emptyset$	PB,BB
MB	OB,LH,LB,MB,PB,BB	LH,LB,WM,WH	PC,CC	$\emptyset$
PC	PC,CC	OH,LH,MH,WM,WH	OB,LB,MB	$\emptyset$
PB	PB,BB	OH,OB,LB,PB,CB, WM,WH	$\emptyset$	LH,MH,MB
WM/WH	OH,OB,LH,LB,MH,MB,PC, CC,PB,CB,WM,WH	$\emptyset$	$\emptyset$	$\emptyset$

where OH: has object, OB: be object of, LH: has location, LB: be location of, MH: has instrument, MB: by means of, PC: (possible) cause, PB: be (possible) caused of, CC: cause, CB: be caused of, WM: meronymy, WH: Holonymy, EE:. Enable, EB: be enabled by.

Table 2: Semantic operations

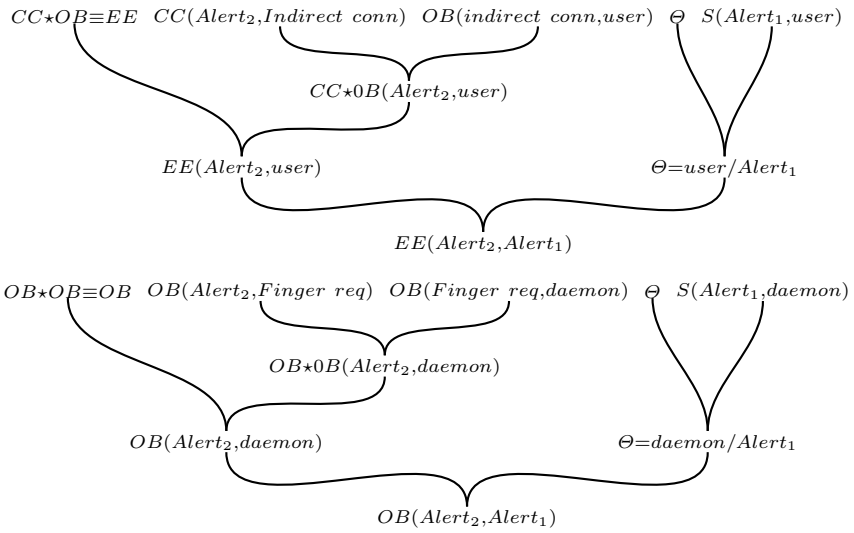


Figure 5: Example of resolution tree

on the attack ontology and alert contexts, alerts are represented as attack semantic space vectors. Text categorization technique are then applied to categorize the intrusion stages.

4.1 Alert Semantic Context Window Size

To guarantee reasoning the attack scenarios in real time, we need to determine the alert semantic context range, the size of which is Alert Semantic Context Window Size (ASCWS). Optimal ASCWS should provide enough semantic in-

formation and restrain the correlation noise at the same time. If ASCWS is too small, the correlated alerts would be absent. On the other hand, if it is too large, the unnecessary computation and the correlation noise (unrelated alerts) will be added. In this paper, mutual information is used to determine ASCWS [Smadja93].

Definition 4.1 *Suppose A and C are the sets of interested alerts and context alerts respectively. They have values according to a probability distribution $p(\alpha)$ and $p(c)$ where $\alpha \in A$ and $c \in C$. Mutual information between $\alpha \in A$ and $c \in C$ at the distance d is defined to be:*

$$MI(A, C, d) = \sum_{c \in C} \sum_{\alpha \in A} p(a, c, d) log_2 \frac{p(a, c, d)}{p(a)p(c)}$$

where $a \neq c$, and $p(a, c, d)$ is the probability that α occurs before or after c at the distance d.

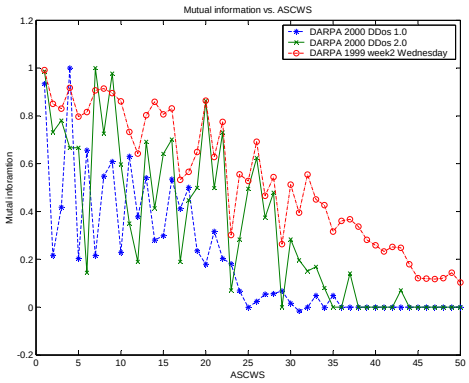


Figure 6: Normalized Mutual Information vs. ASCWS

In Fig.6, three datasets, DDoS 1.0, DDoS 2.0, and DARPA 1999 week 2 Wednesday are simulated [MIT]. As shown in Fig. 6, the alert context window size increases, the degree of the normalized mutual information decreases. At some distances, the associations are very small and do not decrease significantly, implying that there are almost no semantic associations between them. In our simulations, we chose ASCWS as 35.

4.2 Alert Semantic Vector

Ontology-based text categorization is the classification of documents by using ontology as category definition. In our approach, the process of alert categorization is following: First, IDS alerts are converted into attack semantic vector

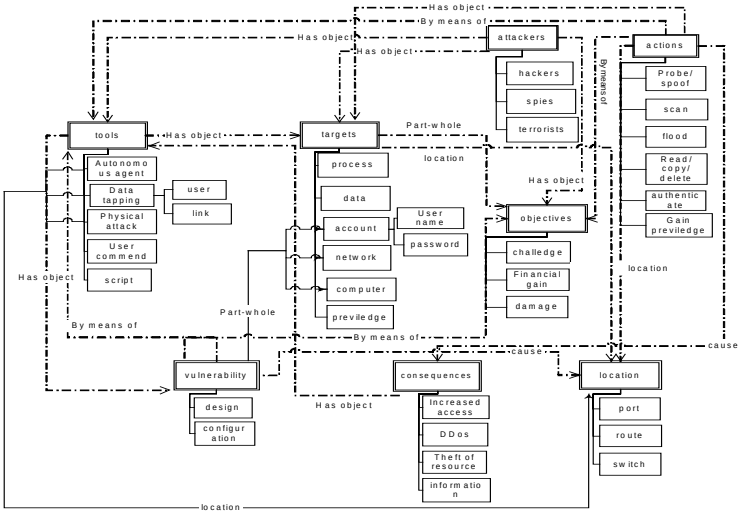


Figure 7: A segment of attack ontology

by attack ontologies. Second, to classify the alert categorizations by calculating similarity between an alert's vector and a category vector. A semantic vector is a vector which represents alert context of a log segment, while a category vector is a vector which represents the characteristic of a whole intrusion category. In this paper, three intrusion categories are divided: *gain informaton*, *making enable*, and *launching attacks*. The category vector is calculated from the feature vectors of the document assigned to the category. The features of vectors are picked from the attack ontology. Fig. 7 shows a segment of attack ontology. The feature's weight is calculated by term frequency and the inverse document frequency (TF-IDF) method [Buckley90].

TF-IDF multiplies the raw Term Frequency (TF) of a feature term in a alert segment by the term's Inverse Document Frequency (IDF) weight:

$$w_{kd} = f_{kd} \cdot idf_k = f_{kd} \cdot \log\left(\frac{N}{D_k}\right)$$

where tf_{kd} is the frequency with which feature $k, 1 \leq k \leq n$, occurs in alert segment d; N is the total number of alert segments in the log corpus; and D_k is the number of segments containing feature k. Afterwards, the vector is normalized by:

$$W_k = \frac{w_{kd}}{\sqrt{\sum_{k=1}^{n} w_{kd}^2}}$$

Cosine-based Similarity between two n dimensional semantic space vectors is

measured by computing the cosine of the angle between these two vectors:

$$sim(\overrightarrow{i}, \overrightarrow{j}) = \cos(\overrightarrow{i}, \overrightarrow{j}) = \frac{\overrightarrow{i} \cdot \overrightarrow{j}}{\| \overrightarrow{i} \|_2 * \| \overrightarrow{j} \|_2}$$

In order to classify which intrusion category an alert a belongs to, the similarities of a's semantic vector and all category vectors are measured, and a belongs to the category with the highest value.

5 Simulations

The datasets in our simulations are from the DARPA LLDOS 1.0 and 1999 week 2 Wednesday from MIT Lincoln Laboratory [MIT]. We use Snort [Snort] as IDS sensor, and set the home net as 172.16.112.0, 172.16.115.0 for LLDOS 1.0 dataset. First, we replayed the tcpdump dataset and aggregated the generated alerts according to the source IP address, target IP address, and the consecutive time slot. Afterwards, 2-AASN of the alerts is built up, and the correlation between them is extracted by the semantic attribute operation to form the attack scenarios. Our simulation results showed that there were three attack scenarios in LLDOS(attacker 202.77.162.213 → victim 172.16.115.20, 202.77.162.213 → victim 172.16.112.10, and 202.77.162.213 → victim 172.16.112.50). In table 5, for dataset LLDOS 1.0, after aggregation and scenario extraction, the number of alerts had decreased to 29.1%, and 12.2% respectively. The correct classification rates for three intrusion stages: *gather information*, *making enable*, and *launching attack* are also shown. The attack scenarios of two datasets are shown in Fig. 8.

Reasoning time(s)	0.14	0.55	0.58	0.31	0.33	0.32	0.58	0.35	0.69
Inter-arrival time(s)	0.02	113.85	2.03	0.34	34.64	1.29	16.31	5.71	95.67

Table 3: Reasoning time of DARPA DDos1.0

Table 3 and Table 4 show the reasoning time and the alert inter-arrival time of these two datasets. It is clear that the reasoning time is far less than the alert inter-arrival time. There are very few alerts whose reasoning time is larger than the inter-arrival time. The reasons are two folds:

1. Some attack actions can generate more than one alerts. For example, alerts *FINGER 0 query* and *FINGER requery* are caused by the same *FINGER* action.

2. A number of alerts may be generated due to the port scanning. In that case, the inter-arrival time of these alerts are extremely small. However, we found

	0.06	0.1	0.08	0.12	0.14	0.17	0.21	0.22	0.02
	0.28	0.3	0.59	0.19	0.21	0.68	0.5	0.23	0.25
Reasoning time(s)	0.26	0.25	0.03	0.26	0.29	0.57	0.28	0.29	0.86
	0.93	1	0.62	0.67	1.08	0.46	0.45	2.23	-
	58.47	0.01	0.02	415.39	0.01	249.22	6.98	0.01	0.15
	285.79	170.7	258.6	41.7	230.17	85.47	1.21	1.17	160.1
Inter-arrival time(s)	139.7	69.5	92.19	57.9	1.62	292.09	8.16	40.2	206.1
	1.12	1.21	9.8	0.27	0.23	1.2	1.62	113.5	-

Table 4: Reasoning time of DARPA 1999 week 2 Wednesday

Data set	Aggregated alert	Alerts in scenario	Gather information	Making enable	Launch attack
LLDOS 1.0	29.1%	12.2%	74%	71%	91%
99 week2 Wed.	45.9%	6.8%	70%	63%	85%

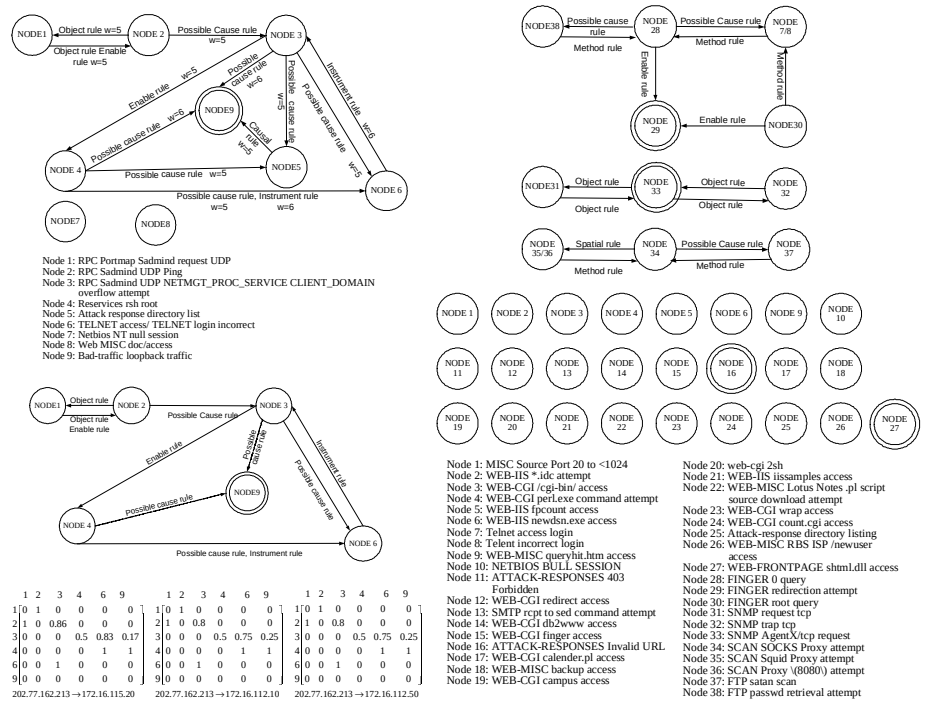
Table 5: Simulation results of alerts number in two alert datasets

the alerts generated by such port scanning action are normally not reasoned to be part of the attack process, because they are uncorrelated with each other.

Therefore, the system can efficiently automatically reason the attack plan.

We compared the performance of extracting attack scenario between FAR-FAR and the Apriori method [Agrawal94]. Apriori is a method to extract the highest possible association rules. Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items, and let D be a set of the transactions where each transaction T is a set of the items such that $T \subset I$. An association rule is defined as: $X \rightarrow Y$, where $X \in I, Y \in I$ and $X \cap Y = \emptyset$. The association rule $X \rightarrow Y$ has the confidence degree $c\%$ if $c\%$ of transactions in D containing X also contain Y. The association rule $X \rightarrow Y$ has the support degree $s\%$ if $s\%$ of the transactions in D contain $X \cup Y$ [Agrawal94]. Given a set of transactions D, the problem of mining association rules is to generate all the association rules that have s and c greater than the user-specified minimum support and minimum confidence degrees, respectively.

However, in this paper, the Apriori method is not used for two reasons. First, for those duplicated alerts indicating the DDoS attacks, the aggregation process usually eliminates them causing very low support degree, which in terms causes those alerts missing in the attack scenario. Second, for the very common alert, such as “telnet” or “scan” alerts, since they can be associated with a number of alerts, the association rules containing them will have very low “confidence” degree, which leads to high “missing focus alerts” and “missing attack links”.



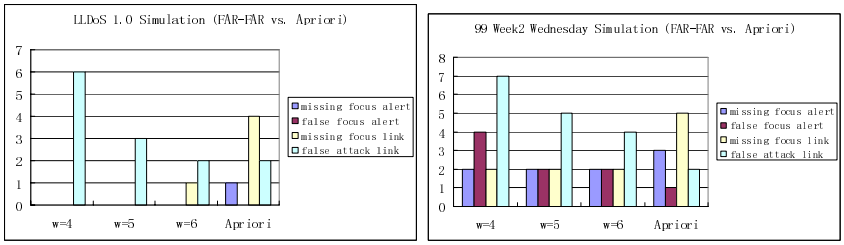
(a) Attack Scenario of LLDOS 1.0 (b) Attack Scenario Class of 99 week 2 Wed

Figure 8: Simulations of DARPA LLDOS 1.0 and 99 week 2 Wednesday dataset

In Figure 9, we compared the performance of extracting the attack scenario between FAR-FAR and the Apriori method. It is clear that the Apriori method had much higher "missing focus alerts" and "missing attack links" value. For the false attack links, since Apriori extracted much less attack correlations than FAR-FAR, it produced a lower value.

6 Conclusion

In this paper, we look to the semantics of attack behaviors for inspirations, and propose FAR-FAR using linguistics approach. By PCTCG, the raw alerts were converted into machine-readable uniform PCTCG streams. Then, the attack scenarios were extracted from 2-AASN. Based on the alert context, the alerts were transformed into attack space vectors and were classified into three intrusion categories automatically. Our simulation results show the scheme not only performs as well as the traditional alert correlation technique, but also facilities the intelligent semantic reasoning.



(a) LLDoS 1.0 Simulation (FAR-FAR vs. Apriori) (b) 99 Week2 Wednesday Simulation (FAR-FAR vs. Apriori)

Figure 9: Simulation comparison between FAR-FAR and Apriori method.

References

[Brachman04] R.J. Brachman, and H.J. Levesque, *Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, 2004.

[Buckley90] G. Salton, and C. Buckley, "Improving retrival performance by relevance feedback," in *Journal of the American Society for Information Science*, 1990, 41(4): pp.288-297.

[Wespi01] H. Debar, A. Wespi, Aggregation and correlation of intrusion-detection alerts, in *Workshop on Recent Advances in Intrusion Detection*, 2001, 85–103.

[Fillmore97] C. Fillmore, *Syntax and Semantics 8: Grammatical Relations*, (Academic Press, New York, 1997), 59–81.

[Dain01] O. M. Dain, and R.K. Cunningham, "Building Scenarios from a Heterogeneous Alert Stream," *Proc. of IEEE Workshop on Information Assurance and Security*, pp.5-6, West Point, NY, 2001.

[Moring02] B. Moring, and L. Me, and H. Debar, and M. Ducass, "A formal Data Model for IDS Alert Correlation", *Proc. of the 5th International Symposium, Recent Advances in Intrusion Detection*, 2002.

[Nerode97] A. Nerode, R. Shore, "Logic for Application"; Springer, Berlin, 68-70.

[Sebastiani02] F. Sebastiani, "Machine Learning in Automated Text Categorizaition," in *ACM Computing Surveys*, Vol. 34 ,No. 1 ,March 2002 ,pp. 147.

[Smadja93] F. Smadja, "Retrieving collocations from text: Xtract," in *Computational Linguistics*, Vol. 19, No. 1, 1993, pp.143-177.

[Teft89] L. Teft, *Programming in Turbo Prolog with an Introduction to Knowledge-based Systems*, Prentice Hall, 1989.

[Tong01] S. Tong, and D. Koller, "Support Vector Machine Active Learning with Applications to Text Classification," in *Journal of Machine Learning Research*, 2001 ,pp. 4566.

[Valdes01] A. Valdes, K. Skinner, Probabilistic alert correlation, in *Workshop on Recent Advances in Intrusion Detection*, 2001, 54–68.

[Agrawal94] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," In *Proceedings of 20th International Conference of Very Large Data Bases*, pp. 487-499, 1994.

[MIT] www.ll.mit.edu/IST/ideval/data/.

[Snort] http://www. Snort.org.
[Emerald] http://www.sdl.sri.com/emerald
[NADIR] http://nadir.lanl.gov/
[NIDES] http://www.csl.sri.com/nides
[ISS] http://www.iss.net
[CISCO] http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/
[IDMEF] http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml- 03.txt.