Extracting Attack Knowledge Using Principal-subordinate Consequence Tagging Case Grammar and Alerts Semantic Networks

Wei Yan, Edwin Hou, Nirwan Ansari
Advanced Networking Laboratory
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07102
Email: {wy3, hou, nirwan.ansari}@njit.edu

Abstract---**The increasing use of Intrusion Detection System and a relatively high false alarm rate can lead to a huge volume of alerts. This makes it very difficult for security administrators to analyze and detect network attacks. Our solution for this problem is to make the alerts machine understandable. In this paper, we propose a novel way to convert the raw alerts into machine understandable uniform streams, correlate the streams, and extract the attack scenario knowledge. The modified case grammar Principal-subordinate Consequence Tagging Case Grammar and the 2-Atom Alert Semantic Network are used to generate the attack scenario classes. Alert mutual information is also applied to calculate the alert semantic context window size. Based on the alert context, the attack scenario instances are extracted and the attack scenario descriptions are forwarded to the security administrator.**

I. INTRODUCTION

As more and more network facilities are connected to the internet, preventing networks from Distributed Denial of Service (DDoS) attacks has become a critical issue that must be tackled by security administrators. Intrusion Detection Systems (IDSs) are used to protect computer networks. However, IDS can generate a huge volume of alerts due to elementary and/or false alarm alerts. Furthermore, the overwhelming volume of alerts makes it difficult for security administrators to analyze and extract the attack knowledge, therefore hampering network attack detection.

Alerts corrrelation was proposed to resolve the above problem. Alerts corrrelation refers to the combination and analysis of attack events from IDS for the purpose of extracting attack strategies. Alerts correlation can enhance the detection rate and provice more accurate attack stategies. In [10], the correlation system used the hyper alert. Hyper alerts can be correlated if the consequence of a hyper alert fulfills the prerequisites of the second hyper alert. This alert correlation method is largely based on the causal relation, one of the most frequently used relations. However, attacks are lauched in several steps, such as gathering related user account information, finding applications' vulnerabilities, and trying to build up the unauthorized connections to the victims. All these actions can generate alerts and they cannot be correlated with the causal relation. SRI International introduced a probabilistic approach of alert correlation [1, 12]. MIT Lincoln Laboratory [5] used a similar approach to perform correlation. New incoming alerts can be added to the existing scenarios after the evaluation of its probability. However, this method requires the attack scenarios to be manually generated in advance. In [7], the alerts correlation was extended to include attack plan recognition. But the technique did not mention how to pre-process the raw alert data and extract the related information from heterogeneous alert reporting formats.

This work is motivated by the fact that raw alerts are mostly inaccessible to the host machines (most alert reporting formats of IDS sensors are based on structured indexing). At present, the heterogeneous IDS sensors deployed have different alert description formats. If we can convert these formats into a uniform structure, the problem of alert fusion can be resloved. Our assumption is that the attack scenario is a sequence of atomic events and attack actions, $S = \{(e_1, a_1), (e_2, a_2), (e_3, a_3), ...(e_n, a_n)\}$ where the 2-tuple (e_i, a_i), $1 \le i \le n$, denotes attack action a_i is the only action performed during the attack event e_i. These attack actions can be converted into the

Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)
0742-1303/04 $ 20.00 IEEE

semantic role structures. Principal-subordinate Consequence Tagging Case Grammar (PCTCG) is used to convert the alerts into machine-understandable uniform PCTCG streams. Afterwards, 2-Atom Alert Semantic Network (2-AASN) is generated from PCTCG streams. Then correlation rules are applied to the 2-AASN to derive the attack scenario classes for the security administrator, who may make further modifications. The attack scenario class includes all possible attack strategies the attacker may take. Finally, based on the alerts within the alert context window range, the specific attack scenario instances are generated.

The rest of the paper is organized as follows. In section 2, we describe the attack knowledge extraction semantic scheme. The PCTCG is proposed in section 3. Semantic network 2-AASN is introduced in section 4. Section 5 describes the alert context window size and attack scenario instances. In section 6, we explain the simulation results, and section 7 is the conclusion.

II. Attack Scenarios Extraction Functional Architecture

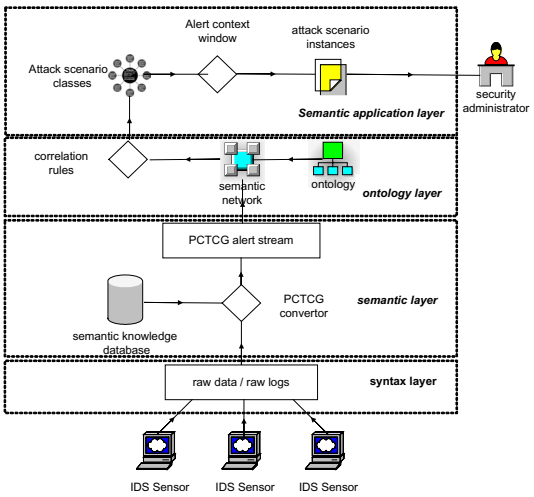


Figure 1. Attack Knowledge Extraction Semantic Scheme

Fig. 1 shows the semantic scheme of attack knowledge extraction. It has four layers: the syntax layer, the semantic layer, the ontology layer, and the semantic application layer. In the syntax layer, sensors generate the alert files. These sensors should be heterogeneous so that different views of the attacks can be observed. Although IDMEF (Intrusion Detection Message Exchange Format) was proposed to be a standard alert format for IDS to exchange alerts [4]. One principle problem is its use of XML (syntax structure) which limits the semantic representation.

Our proposed scheme adds the semantic layer, the ontology layer, and the semantic application layer on the existing alert syntax platform. The semantic layer consists of various semantic knowledge databases that are coupled to the IDS sensors. Each semantic knowledge database contains the semantic information of all the alert messages that can be generated by the particular sensor. The alert file and the sensor type are the inputs to the PCTCG converter. Using the semantic information in the corresponding semantic knowledge database, the PCTCG converter transfers the incoming alert messages into uniform PCTCG stream. Note that only dissimilar alerts in the alert log are converted into PCTCG stream. For example, the data set 2000 DARPA Intrusion Detection Scenario (MIT Lincoln Laboratory) [14], Snort was used as the IDS sensor and *home net* was set to *172.16.112.0* and *172.16.115.0*. The alert log file yeilded 9 different alerts. In the ontology layer, we defined action-based semantic ontology. The semantic ontology is applied to PCTCG streams coming from the semantic layer, and the 2-AASN semantic networks are generated. In the semantic application layer, the correlation rules are used to build up the attack scenario classes from the 2-AASN. Based on the alert semantic context, the attack scenario instances are extracted, and the attack scenario knowledge is forwarded to the security administrator.

III. Principal-Subordinate Consequence Tagging Case Grammar

In this section, we proposed using PCTCG as the standard semantic format to represent the alert format. Here we assumed that DDoS attacks are composed of discrete attacker behavioral action units. Thus the attack processes can be viewed as a behavior-action streams. If every attack action is viewed as a verb, formal linguistic model can be applied. We modified the traditional case grammar theory to PCTCG, which is used as standard semantic alert format. Case grammar is proposed by Fillmore [3, 6] and is widely used in the field of linguistics. The reasons for adopting case grammar are:

1) Case grammar describes the relation between a verb and other components. The verb is the focus of a sentence, and its case frame structure specifies the slots from the sentences containing the types of relationship (semantic roles) between verbal and other entities [6].
2) Case grammar can be easily represented by semantic networks, which has abundant semantic relations to express the alert correlations.

3) The attack actions are much more unambiguous than the alert logs. For example, two sensors with heterogeneous alert formats can generate two different types of alert record for the same attack action. However, the action which caused the alerts will be the same.

Table 1. Syntactic vs. Semantic.

Sentences	Syntactic level		Semantic level	
	Subject	Object	Agent	Theme
The man moved the desk.	Man	Desk	Man	Desk
The desk was moved by the man.	Desk	Man	Man	Desk

As shown in Table 1, *subject* and *object* roles change in syntactic level when the sentence changes from active to passive form. However, the *Agent* (what causes the action of the verb) and *Theme* (the object in motion or being located) on the semantic level remain the same. Unlike at the syntactic level, case grammar theory is of deep or semantic case [3], and it does not change under the grammatical transformation of sentences, i.e., from passive to active.

PCTCG is formally defined as:

$$G = \{M_n,\ C,\ F,\ S\}$$

where M_n is the set of alert messages from the sensor n, C specifies the set of possible semantic roles between the alerts, F is the set of arguments (case fillers), and S is the set of *subordinate keywords*. As there are no universally agreed set of semantic roles in case grammar, the choice of C depends on the specific application. In PCTCG, the set of chosen semantic roles should reflect the semantic logic of the attacker's actions. Since an attack action may give rise to a certain consequence, it is important for the security experts to clearly know what consequence the current attack may cause, and also predict what the next step should be. Thus we added the *consequence tagging* into C to model this feature. The *consequence tagging* is composed of three alert consequence entities: *gather information*, *making enable*, and *lauching attack*.

Consider using the semantic roles to correlate two different alerts. We applied the *Principal-subordinate relation* to the two alerts. When one alert is in the *principal* state, it can be viewed as a *verb* and the other alert is replaced with its *subordinate keywords* (the *subordinate keywords* is viewed as *noun phrases*). If the *subordinate keyword* is in a specific case relationship with the *verb*, we treat these two alerts as correlated. From this point of view, we added S into PCTCG definition. The *subordinate keywords* are defined so that they can describe the alert subject appropriately. In PCTCG, "{}" is used for grouping and "+" means "gathering information" or "gaining privileges." For example, consider the Snort alert message RPC *sadmind UDP PING*, its PCTCG format is:

{{PRC sadmind UDP PING}$_{\text{Snort}}$,
{has object, possible cause, by means of, consequence tagging},
{Sadmind RPC service, {+information, +priveledge}, ping, gather information},
{Sadmind, ping}}

Here, *has object*, *possible cause*, *by means of*, *consequence tagging* are the semantic roles. *Sadmind PRC service,* {*+information, +previledge*}*, ping,* and *gather information* fill in the slots of the above semantic cases respectively, and *Sadmind*, and *ping* are the *subordinate keywords*.

IV. ALERT SEMANTIC NETWORK

In the field of linguistics, the semantic ontology is related to the lexical entities (such as verb, noun, etc.). In this paper, our goal is to find a well-defined semantic ontology, which can be used to extract attack scenarios. We need to focus on the following questions that security administrators would naturally ask about the attack actions: *When did the actions happened? Where did the actions happened? By which means the actions happen? What results did the actions caused?* etc., Fig. 2 shows the two-level semantic ontology of the semantic roles and the semantic attributes of PCTCG. The first level is the semantic roles and it includes *Object*, *Location*, *Method*, (*Possible*) *Cause*, *Part-Whole*, and *Consequence Tagging*. The second level shows the semantic attributes and their weights. The *Object* role is the receiving end of the action and it has *has object* and *be object of* attributes. Since an attack action can be either successful or is just an attempt that failed, the *possible cause* and *cause* semantic roles are used to model the above situation. The *meronymy* (has an object) and *holonymy* (is a part of) attributes from the *part-whole* role describe the situations that one entity contains another entity. *Consequence tagging* role indicates at which stage the attack may be located (*gather information*, *making enable* or *launching attacks*).

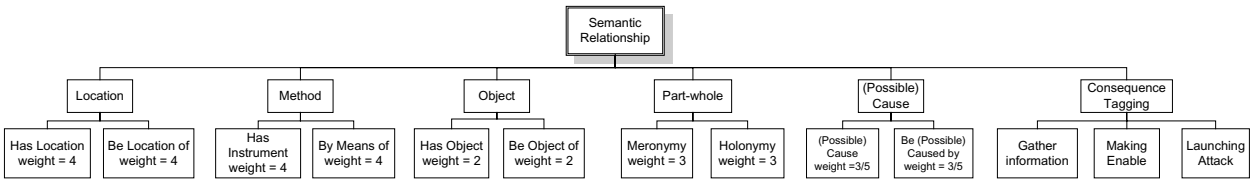


Figure 2. Ontology of Semantic Roles and Attributes

Based on [13], 2-AASN is proposed as the semantic correlation representation and can be viewed as a weighed graph. The edge of the graph presents PCTCG semantic attribute or *subordinate*. The set of nodes includes two parent nodes (two atomic alerts), and their child nodes (case filler or *subordinate keyword*). The formal representation of 2-AASN is based on slots. Each slot is a 2-tuple: <*semantic attributes, case filler*>, or <*subordinate, subordinate keyword*>, which describes the semantic role and its case filler. The format of 2-AASN is:

SN (node1, node2) = {
node 1: < subordinate, subordinate keyword>$^{+)}$
node 2: < semantic attribute, case filler >$^{+)}$ or
node 1: < semantic attribute, case filler >$^{+)}$
node 2: < subordinate, subordinate keyword>$^{+)}$,
node2: case filler < semantic attribute, node 1: subordinate keyword >$^{+)}$or
node1: case filler < semantic attribute, node 2: subordinate keyword >$^{+)}$ }

Consider two Snort alerts: *RPC Portmap Sadmind request UDP* and *RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt*. Their semantic attributes and case fillers are shown in Table 2:

Table 2. Semantic Attributes and Case Fillers.

RPC portmap Sadmind request UDP	
Semantic roles	Semantic case fillers
Has object	Port
Possible cause	Buffer overflow, gain privilege
By means of	Portmap GETPORT request
Consequence tagging	Vulnerabilities collection
Subordinate keywords	Sadmind, port

RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt	
Semantic roles	Semantic case fillers
Has object	Buffer overflow
Possible cause	Buffer overflow, gain privilege, root
By means of	NETMGT_PROC_SERVICE request, improper bounds checking
Consequence tagging	Vulnerabilities collection
Subordinate keywords	Sadmind, NETMGT_PROC_SERVICE, buffer overflow, root

In order to construct the semantic 2-AASN, we first generate the PCTCG format stream using the semantic information from the corresponding semantic knowledge database. The PCTCG format stream of the above two alerts are:

{{PRC Portmap Sadmind request UDP}$_{Snort}$,
{has object, possible cause, by means of, consequence tagging},
{port, {buffer overflow, +priveledge},
portmap GETPORT request, information collection},
{Sadmind, port}}
{{PRC Sadmind UDP NETMGT PROC SERVICE overflow attempt}$_{Snort}$,
{has object, possible cause, by means of, consequence tagging},
{buffer overflow, {buffer overflow, +priveledge, root},
{NETMGT_PROC_SERVICE, improper bounds checking}, gather information},
{Sadmind, NETMGT_PROC_SERVICE, buffer overflow, root}}

Secondly, put alert *RPC Portmap Sadmind request UDP* in *principle* state, and *RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt* in *subordinate* state. When an alert is in the *principle* state, 2-AASN use its semantic case fillers as the nodes, whereas if an alert is in *subordinate* state, 2-AASN uses its *subordinate keywords* as the nodes. If there are semantic attributes matching between case fillers and the *subordinate keywords*, 2-AASN fills the slot:

$\{node1 : case\ filler < semantic\ role, node2 : keyword >\}$ or

$\{node2 : case\ filler < semantic\ role, node1 : keyword >\}$.

The 2-AASN representation format is:

node 1 = RPC Sadmind UDP overflow attempt
node 2 = RPC Portmap Sadmind request UDP
SN (node1, node2) = {
node1: < subordinate, Sadmind > ,
node 1: < subordinate, root>
node 1: < subordinate, buffer overflow>
node 2: < location at, port > ,
node 2: < cause, buffer overflow > ,
node 2: port < has object, node1: Sadmind > ,
node 2: buffer overflow < cause, node2: root > ,
node 2 < cause, node1: buffer overflow > }

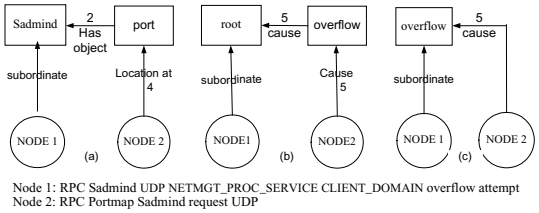


Figure 3. An Example of 2-AASN

The format can also be represented by the semantic weighed network graph as shown in Fig. 3. Finally, we put the alert *RPC Portmap Sadmind request UDP* in *subordinate* state, and the alert *RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN*

Table 3. Correlation rules.

Correlation rules	Representation	Match phrases	Match process
(Possible) Cause relation	$A \xrightarrow{C} B$, or $C(A, B)$	Primary→ Secondary	*A::filler* matches "(possible) cause" to *B::keyword*: *C(A, B).* *A::filler* matches "(possible) be caused by" to *B::keyword*: *C(B, A).*
		Secondary← Primary	*B::filler* matches "(possible) cause" to *A::keyword*: *C(B, A).* *B::filler* matches "(possible) be caused by" to *A::keyword*: *C(A, B).*
Enable relation	$A \xrightarrow{E} B$, or $E(A, B)$	Primary→ Secondary	*A::filler* matches "enable" to *B::keyword*: *E(A, B)* *A::filler* matches "be enabled by" to *B::keyword*: *E(B, A).*
		Secondary← Primary	*B::filler* matches "enable" to *A::keyword*: *E(B, A)* *B::filler* matches "be enabled by" to *A::keyword*: *E(A, B).*
Instrument relation	$A \xrightarrow{I} B$, or $I(A, B)$	Primary→ Secondary	*A::filler* matches "has instrument" *B::keyword*: *I(A, B).* *A::filler* matches "be means of" to *B::keyword*: *I(B, A).*
		Secondary← Primary	*B::filler* matches "has instrument" *A::keyword*: *I(B, A).* *B::filler* matches "be means of" to *A::keyword*: *I(A, B).*
Object relation	$A \xrightarrow{O} B$, or $O(A, B)$	Primary→ Secondary	*A::filler* matches "has object" to *B::keyword*. *O(A, B).* *A::filler* matches "be object of" to *B::keyword*. *O(B, A).*
		Secondary← Primary	*B::filler* matches "has object" to *A::keyword*. *O(B, A).* *B::filler* matches "be object of" to *A::keyword*. *O(A, B).*
Part-whole relation	$A \xrightarrow{P} B$, or $P(A, B)$	Primary→ Secondary	*A::filler* matches "meronymy" to *B::keyword*: *P(A, B).* *A::filler* matches "meronymy" or "holonymy" to *B::keyword*: *P(B, A).*
		Secondary← Primary	*B::filler* matches "meronymy" to *A::keyword*: *P(B, A).* *B::filler* matches "meronymy" or "holonymy" to *A::keyword*: *P(A, B).*
Spatial relation	$A \xrightarrow{S} B$, or $S(A, B)$	Primary→ Secondary	*A::filler* matches "has location" to *B::keyword*: *S(A, B).* *A::filler* matches "be located of" to *B::keyword*: *S(B, A).*
		Secondary← Primary	*B::filler* matches "has location" to *A::keyword*: *S(B, A).* *B::filler* matches "be located of" to *A::keyword*: *S(A, B).*

overflow attempt in the *principle* state and perform semantic attributes matching.

We defined the correlation rules and apply these rules to extract the alert correlations from 2-AASN. In order to extract the correlation from 2-AASN, we define the semantic attribute operation ⊗ : *principle alert: <semantic attribute, case filler>* ⊗ *subordinate alert: <subordinate, subordinate keyword>*. Table 4 defines the correlation operation between two semantic attributes.

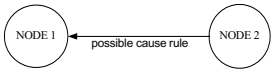
Table 4. Correlation operation.

	1	2	3	4	5	6	7	8	9	10
1	1	×	3	4	5	6	11	8	1	1
2	×	2	3	4	5	6	7	12	2	2
3	3	×	3	×	5	3,6	11	12	3	3
4	4	4	×	4	4,5	6	11	12	4	4
5	1	×	5,3	5,4	5	×	7	12	5	5
6	×	2	6,3	6,4	×	6	11	8	6	6
7	7	11	7	11	7	11	7	×	7	7
8	8	8	12	8	12	12	×	8	8	8
9	1	2	3	4	5	6	7	8	9	×
10	1	2	3	4	5	6	7	8	×	10

× means two semantic attributes can not be operated, not considered by the correlation rules.
1. has object, 2. be object of, 3. has location, 4. be location of
5. has instrument, 6. by means of, 7. (possible) cause,
8. be (possible) caused of, 9. meronymy, 10. Holonymy.
11. enable 12. be enabled by.

Not all the semantic attributes can be operated with each other (Those cannot are marked by ×, meaning the two semantic attributes will not be considered by the correlation rules). On the other hand, some semantic attributes cannot be fused, so the operation result is their union. The operation ⊗ may generate new semantic attributes, which can describe the semantic roles more precisely. For example, (*possible*) *cause* ⊗ *be object of* = *enable.* The correlation rules try to test the semantic attributes matching between the *principle* alert and *subordinate* alert. For examples, suppose we have two alerts: *A* and *B*. We denote alert *A*'s principle case filler and *subordinate keyword* as $A::filler$ and $A::keyword$ respectively and similarly for alert *B*. We then defined the (*possible*) *cause*, *enable*, *instrument*, *object*, *part-whole*, and *spatial* relations as follows. The *enable* relation takes place when one entity facilitates the other's attack process. The *spatial* relation describes the situation where one entity is surrounded by another entity but is not part of that entity. The (*possible*) *cause*, *enable*, *instrument*, and *object* relations are concerned with attack action "time" domain whereas the *part-whole* and *spatial* relations are related to "space" domain. In general, any correlation rule (see Table 3) can be represented as $A \xrightarrow{R} B$ or $R(A, B)$, where R is the name of the correlation rule. When extracting correlations, we first add up the weights of the semantic attributes. If the sum of the weights is more than the weight threshold (set to 5 for the simulations), the semantic attributes are operated according to Table 4. Afterwards, the correlation rules are applied to the

operation result to acquire the correlation. The correlation from 2-AASN is shown in Fig. 4:



Node 1: RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt

Node 2: RPC Portmap Sadmind request UDP

Figure 4. An Example of Correlation Rule

Because the attack scenario classes contain all possible attack correlation combinations above the semantic weight threshold, they may include some inaccurate semantic correlations. If they are not eliminated in time, a chain of errors can occur and give rise to inaccurate attack scenario instances. On the other hand, when establishing the attack scenario instances, the whole aggregated alert file will be reviewed. The aggregated file is generated based on the same source or destination IP addresses, and the time stamps within a certain time slot. Removing the inaccurate correlations in advance will reduce the computation load. Therefore, after the attack scenario classes are extracted, the security administrators may make further modifications using their practical experiences.

V. ATTACK SCENARIO INSTANCES

Since the attack scenario classes include all the possible combinations of attack strategies but the attackers may only adopt a subset of the attack strategies to lauch the attacks. The alert context of specific alerts have to be considered and the alert context window (ACW) size has to be determined. The correlations generated within the context window range will build up the attack scenario instances. In natural language processing, the context is used to determine the pronunciation, words collocation and words unambiguity [8, 11]. The ACW size (which is the number of alerts before and after an interested alert) is an important parameter of the alert context. If the ACW size is too small, the correlated alerts would be absent. On the other hand, if the ACW size is too large, unnecessary computations and unrelated alerts will be added. In natural language processing, small window size can identify the fixed expressions and word collocations, which hold over short range. In [2, 9], the context window size was set to be 5. Here our interest lies in the semantic correlation between the alerts. ACW size should be much larger to cover the semantic knowledge. The mutual information method [8] was applied to three alert data sets: DARPA 2000 LLDOS 1.0, LLDoS 2.0.2, and 1999 week 2 Friday data sets of MIT Lincoln Laboratory. Mutual information is defined to be:

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

$$I(a,c,d) = \log_2 \frac{p(a,c,d)}{p(a)p(c)}$$

where $I(a,c,d)$ $(a \neq c)$ is the association ratio of two alerts a and c, and $p(a)$ and $p(c)$ are the probabilities of a and c, and $p(a,c,d)$ is the probability that a occurs before or after c at the distance d.

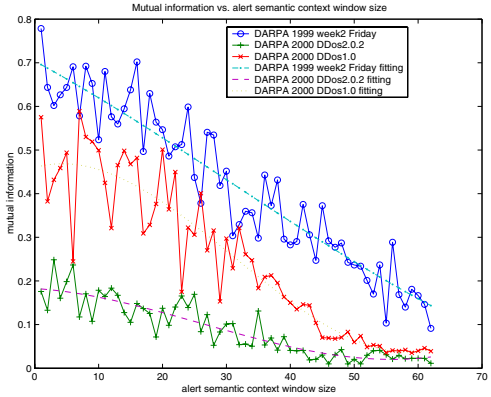


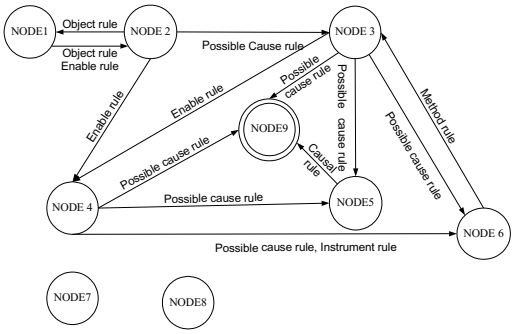
Figure 5. Mutual Information at Various ACW Size

Fig. 5 shows as ACW increases, the degree of mutual information decreases. Beyond a threshold, the association becomes very small and do not decrease significantly, which means there are almost no associations between them. In the simulation, we chose 60 as the ACW size. The alerts in the attack scenario classes are called focus alerts. If a focus alert and the alerts within its ACW range are used to construct the sub-classes of the attack scenario classes, they will build up the attack scenario instances. The format of the nodes in the attack scenario instances includes the alert message name and 3-tuple < source IP, dest. IP, timestamp > :

Node = alert message name,
{Node: < source IP, dest. IP, timestamp > $^{+)}$ }

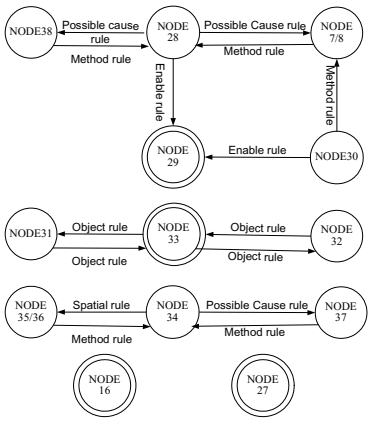
The scenario description is defined based on attack sub-objectives: *gather information control* (try to break into the target), *get control* (get control of the target), and *launching attacks*. These sub-objectives are derived from the consequence tagging of PCTCG. The scenario description structure is defined as:

AS (attack scenario name) = {
objective name:
gather information: < alert message, source IP, dest. IP, timestamp > $^{+)}$
get control: < alert message, source IP, dest. IP, timestamp > $^{+)}$,
launching attacks: < alert message, source IP, dest. IP, timestamp > }



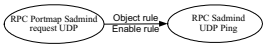
Node 1: RPC Portmap Sadmind request UDP
Node 2: RPC Sadmind UDP Ping
Node 3: RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt
Node 4: Reservices rsh root
Node 5: Attack response directory list
Node 6: TELNET access/ TELNET login incorrect
Node 7: Netbios NT null session
Node 8: Web MISC doc/access
Node 9: Bad-traffic loopback traffic

Figure 6. Alert Scenario class of DARPA 2000 Data set



Node 7: Telnet access login
Node 8: Telent incorrect login
Node 16: ATTACK-RESPONSES Invalid URL
Node 27: WEB-FRONTPAGE shtml.dll access
Node 28: FINGER 0 query
Node 29: FINGER redirection attempt
Node 30: FINGER root query
Node 31: SNMP request tcp
Node 32: SNMP trap tcp
Node 33: SNMP AgentX/tcp request
Node 34: SCAN SOCKS Proxy attempt
Node 35: SCAN Squid Proxy attempt
Node 36: SCAN Proxy \(8080\) attempt
Node 37: FTP satan scan

Figure 7. Alert Scenario class of DARPA 99 week 2 Data set



Node1 = { RPC Portmap Sadmind request UDP,
< 202.77.162.213, 172.16.115.20, 10:08:07.354091>}
Node2 = { RPC Sadmind UDP Ping,
< 202.77.162.213, 172.16.115.20, 10:08:07.359636>}

Node1 = { RPC Portmap Sadmind request UDP,
< 202.77.162.213, 172.16.112.10, 10:15:10.023115>}
Node2 = { RPC Sadmind UDP Ping,
< 202.77.162.213, 172.16.112.10, 10:15:10.026586>}

Node1 = { RPC Portmap Sadmind request UDP,
< 202.77.162.213, 172.16.115.50, 10:15:10.098496>}
Node2 = { RPC Sadmind UDP Ping,
< 202.77.162.213, 172.16.115.50, 10:15:10.102257>}

Node1 = { Telnet access,
< 172.16.115.20, 202.77.162.213, 10:50:01.819752>}
Node2 = { RSERVICES rsh root,
< 172.16.115.20, 202.77.162.213, 10:50:04.146207>}

Node1 = { Telnet access,
< 172.16.112.10, 202.77.162.213, 10:50:21.064056>}
Node2 = { RSERVICES rsh root,
< 172.16.112.10, 202.77.162.213, 10:50:22.146207>}

Node1 = { Telnet access,
< 172.16.115.50, 202.77.162.213, 10:50:37.923074>}
Node2 = { RSERVICES rsh root,
< 172.16.115.20, 202.77.162.213, 10:50:38.176538>}

Node1 = { RPC Portmap Sadmind request UDP,
< 202.77.162.213, 172.16.115.20, 10:33:10.611612>
< 202.77.162.213, 172.16.115.20, 10:33:12.642958>
< 202.77.162.213, 172.16.115.20, 10:33:18.875888>
< 202.77.162.213, 172.16.115.20, 10:33:20.913357>
< 202.77.162.213, 172.16.115.20, 10:33:27.155926>
< 202.77.162.213, 172.16.115.20, 10:33:29.205551>}

Node2 = { RPC Sadmind UDP NETMGT_PROC_SERVICE overflow ,
< 202.77.162.213, 172.16.115.20, 10:33:10.621429>
< 202.77.162.213, 172.16.115.20, 10:33:12.652687>
< 202.77.162.213, 172.16.115.20, 10:33:18.885651>
< 202.77.162.213, 172.16.115.20, 10:33:20.923039>
< 202.77.162.213, 172.16.115.20, 10:33:27.165722>
< 202.77.162.213, 172.16.115.20, 10:33:29.223090>}

Node3 = { Telnet access,
< 202.77.162.213, 172.16.115.20, 10:33:14.728748>
< 202.77.162.213, 172.16.115.20, 10:33:23.011892 >
< 202.77.162.213, 172.16.115.20, 10:33:32.470221>}

Figure 8. Alert Scenarios Instances of DARPA 2000 Data set

AS (DARPA 2000) = {
objective name: attack 172.16.115.20, 172.16.112.10, 172.16.115.50

gather information

RPC Portmap Sadmind request UDP,	*enable*	RPC Sadmind UDP Ping,
< 202.77.162.213, 172.16.115.20, 10:08:07.354091>		< 202.77.162.213, 172.16.115.20, 10:08:07.359636>
< 202.77.162.213, 172.16.112.10, 10:15:10.023115>		< 202.77.162.213, 172.16.112.10, 10:15:10.026586>
< 202.77.162.213, 172.16.115.50, 10:15:10.098496>		< 202.77.162.213, 172.16.115.50, 10:15:10.102257>

RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow	*cause*	Telnet access,
< 202.77.162.213, 172.16.115.20, 10:33:10.621429>		< 202.77.162.213, 172.16.115.20, 10:33:14.728748>
< 202.77.162.213, 172.16.115.20, 10:33:12.652687>		
< 202.77.162.213, 172.16.115.20, 10:33:18.885651>		
< 202.77.162.213, 172.16.115.20, 10:33:20.923039>		< 202.77.162.213, 172.16.115.20, 10:33:23.011892 >
< 202.77.162.213, 172.16.115.20, 10:33:27.165722>		< 202.77.162.213, 172.16.115.20, 10:33:32.470221>}

get control

RPC Portmap Sadmind request UDP	*cause*	RPC Sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow
< 202.77.162.213, 172.16.115.20, 10:33:10.611612>		< 202.77.162.213, 172.16.115.20, 10:33:10.621429>
< 202.77.162.213, 172.16.115.20, 10:33:12.642958>		< 202.77.162.213, 172.16.115.20, 10:33:12.652687>
< 202.77.162.213, 172.16.115.20, 10:33:18.875888>		< 202.77.162.213, 172.16.115.20, 10:33:18.885651>
< 202.77.162.213, 172.16.115.20, 10:33:20.913357>		< 202.77.162.213, 172.16.115.20, 10:33:20.923039>
< 202.77.162.213, 172.16.115.20, 10:33:27.155926>		< 202.77.162.213, 172.16.115.20, 10:33:27.165722>
< 202.77.162.213, 172.16.115.20, 10:33:29.205551>		< 202.77.162.213, 172.16.115.20, 10:33:29.223090>

Telnet access,	*instrument*	RSERVICES rsh root,
< 172.16.115.20, 202.77.162.213, 10:50:01.819752>		< 172.16.115.20, 202.77.162.213, 10:50:04.146207>
< 172.16.112.10, 202.77.162.213, 10:50:21.064056>		< 172.16.112.10, 202.77.162.213, 10:50:22.146207>
< 172.16.115.50, 202.77.162.213, 10:50:37.923074>		< 172.16.115.20, 202.77.162.213, 10:50:38.176538>

launching attacks
bad traffic loopback traffic
< 202.77.162.213, 172.16.115.20, 10:33:29.223090>}

Figure 9. Alert Scenario Description of DARPA 2000 Data set

VI. SIMULATIONS

The data sets used in our simulation are from the DARPA 2000 and 1999 week 2 from MIT Lincoln Laboratory [14]. We used Snort as the IDS sensor. Snort is a lightweight network intrusion detection system capable of logging every possible trace of intrusion attempts [15]. The semantic information of the alert messages is stored using MySQL database. First, we aggregated the alerts according to the same source IP address and target IP address, and the same consecutive timeslot. After the aggregation, the DARPA 2000 alert file included 9 different alerts and DARPA 1999 week 2 alert file included 38 different alerts. Then we generated the PCTCG format streams of the two data sets using the semantic information. Afterwards, we built up 2-AASN of the first two different alerts and tried to extract the correlation between them by applying the correlation rules. If they had any correlation, new attack scenario are generated. The new incoming alert are then correlated with the existing alerts in the attack scenario one by one and the whole attack scenario is build up. The attack scenario classes of DARPA 2000 and DARPA 1999 week 2 data sets are shown in Fig. 6 and 7 (The double circle in the figures indicates DDoS attacks have occurred). For the DARPA 2000 alert file, the attack scenario instances and the attack descriptions are also extracted and are shown in Fig. 8 and 9.

VII. CONCLUSION

In this paper, we proposed a novel method to extract the attack knowledge using Principal-subordinate Consequence Tagging Case Grammar and 2-Atom Alert Semantic Network. By PCTCG, the raw alerts are converted into machine-understandable uniform PCTCG streams. Then the correlation rules are applied to the 2-AASN to derive the attack knowledge for the security administrator. By making the raw alert data computer understandable will resolve the problems caused by large volume of alerts. Our future work is to use semantic query model to allow attack reasoning and have inference capabilities.

REFERENCES

[1] D. Andersson, M. Fong, and A. Valdes, "Heterogeneous Sensor Correlation: A Case Study of Live Traffic Analysis," *Proceedings of IEEE Information Assurance Workshop* (United States Military Academy, West Point, NY, June 2002).

[2] K. Church, "Word association norms, mutual information, and lexicography," *Computational Linguistics*, Vol. 16, No. 1, pp. 22-29, 1990.

[3] W. Cook, *Case Grammar Theory*, Washington, DC: Georgetown University Press, 1989.

[4] D. Curry, and H. Debar, "Intrusion Detection Message Exchange Format," http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-03.txt.

[5] O. M. Dain and R. K. Cunningham, "Building Scenarios from a Heterogeneous Alert Stream," *Proceedings of IEEE Workshop on Information Assurance and Security* (United States Military Academy, West Point, NY, June 2001).

[6] C. J. Fillmore, "The Case for Case Reopened." *Studies in Syntax and Semantics*, 8, 59-81, 1977.

[7] C. W. Geib, and R. P. Goldman, "Plan recognition in intrusion detection systems," *DARPA Information Survivability Conference*, vol. 1, pp. 46-55, 2001.

[8] J. M. Lucassen and R. L. Mercer, "An information theoretic approach to the automatic determination of phonemic baseforms," *Proceedings of ICASSP*, Vol.3, pp. 42.5.1-42.5.4, 1984.

[9] W. Martin and P. Sterkenburg, *Processing of text corpus, Lexicography: Principles and Practice*, R. Hartmann, New York, 1983, pp. 56-64.

[10] N. Peng, C. Yun, and S. R. Douglas, "Constructing Attack Scenarios through Correlation of Intrusion Alerts," *Proceedings of the 9th ACM Conference on Computer & Communications Security*, pp. 245-254, 2002.

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

[12] A. Valdes, and K. Skinner, "Probabilistic alert correlation," *Workshop on Recent Advances in Intrusion Detection*, pp. 54-68, 2001.

[13] L. Vanderwende, "The analysis of noun sequences using semantic information extracted from on-line dictionaries," Ph.D. dissertation, Georgetown University, Washington, DC., 1996.

[14] http://www.ll.mit.edu/IST/ideval/data/2000/LLS_DDOS_1.0.html

[15] http://www.snort.org