A Novel Framework for Alert Correlation and Understanding

Dong Yu[1,2] and Deborah Frincke[1]

[1] Center for Secure and Dependable Software, University of Idaho, USA
dongyu@csds.uidaho.edu
[2] Microsoft Research / Redmond, USA, frincke@cs.uidaho.edu

Abstract. We propose a novel framework named Hidden Colored Petri-Net for Alert Correlation and Understanding (HCPN-ACU) in intrusion detection system. This model is based upon the premise that intrusion detection may be viewed as an inference problem – in other words, we seek to show that system misusers are carrying out a sequence of steps to violate system security policies in some way, with earlier steps preparing for the later ones. In contrast with prior arts, we separate actions from observations and assume that the attacker's actions themselves are unknown, but the attacker's behavior may result in alerts. These alerts are then used to infer the attacker's actions. We evaluate the model with DARPA evaluation database. We conclude that HCPN-ACU can conduct alert fusion and intention recognition at the same time, reduce false positives and negatives, and provide better understanding of the intrusion progress by introducing confidence scores.

1 Introduction

Intrusion detection system (IDS) is originated as a mechanism for managing the detection of system misuse through the analysis of activity [3]. A typical state-of-the-art IDS detects intrusions by analyzing audit data from various sources (hosts and networks) and alert users or defense systems automatically when possible intrusive behaviors are observed. A key factor in determining an effective IDS is its ability to properly correlate information drawn from appropriately placed IDS sensors due to the following three reasons. First, IDS sensors can generate massive amount of alerts [17], if they have a high sensitivity to potential misuse; examining these alerts is costly and not all of this information leads to good decisions. Second, the false positive rate is one of the most serious problems with current IDSs [2,4]. Third, false negatives are another problem – those intrusions missed by the IDS may later result in damage to the system. Given these, intelligent analysis of activity is critical to the overall success of the IDS. Alert Correlation and Understanding (ACU) can improve the effectiveness of the IDS by examining how the outputs of IDS sensors (the alerts) may be used to better identify misuse and develop response plans.

Current approaches in ACU can be classified into two primary categories: alert fusion and intention recognition. *Alert fusion*, also known as aggregation,

M. Jakobsson, M. Yung, J. Zhou (Eds.): ACNS 2004, LNCS 3089, pp. 452–466, 2004.

or clustering, is to aggregate similar alerts from multi-sensors into so called meta-alerts (or hyper-alerts) based on feature similarities, with the hope to enhance the quality of the resulting information [35,33,9,10,20,6,17,29]. The fusion process usually involves the merging of the features of the two alerts. For example, alerts from the same sensor and belong to the same attack (identified by the same source and target IP address) are considered similar alerts [33]. In alert fusion, alerts are first classified into alert clusters that correspond to the same occurrence of an attack based on similarity. Each cluster is then merged and a new, global alert is generated to represent the whole cluster [9,29]. The main purpose of the alert fusion is to reduce the number of alerts to be provided to the administrators and reduce the false positives to some extent [10].

In contrast, *intention recognition* (or attack plan recognition) [16,32,13,12,7, 8,26,27] seeks to recognize an attacker's intention from the alerts. The emphasis here is to give administrators and active reactors better understanding of ongoing activities so that they can make appropriate responses. The importance of intention recognition is not so much in the "average" generic attack on a system, but for instances where it is important to more fully identify complex, multi-stage scenarios. Detecting an attacker's plan at an early stage would make it easier to prevent the attacker from achieving his/her goal. Intention recognition is also aimed to reduce some false positives during correlation; further, it should be possible to increase true positives (therefore reducing false negatives) by inferring the existence of attacks during correlation.

Some of these technologies have already been implemented in Commercial Off The Shelf (COTS) intrusion detection tools from companies such as Netforensics, Q1, Object neworks, and Arcsight, to name a few. However, current ACU approaches have several limitations:

- Alert fusion and intention recognition are usually two separate steps. Intention recognition approaches are applied on the result of alert fusion [9].
- Uncertainty information is usually not used in the ACU process. For example, the rate of false positives and false negatives would provide some hint on whether a conclusion that an attacker did take some action can be drawn reliably when an alert was observed. Other sources of uncertainties include trustworthiness of alerts gathered from different sensors.
- No confidence score is associated with the ACU's outputs.

In this paper, we propose a novel framework named Hidden Colored Petri-Net for Alert Correlation and Understanding (HCPN-ACU). This model is based upon the premise that intrusion detection may be viewed as an inference problem – in other words, we seek to show that system misusers are carrying out a sequence of steps to violate system security policies in some way, with earlier steps preparing for the later ones. We assume that the attacker's actions themselves are unknown, but the attacker's behavior may result in alerts. These alerts are then used to infer the attacker's actions. In this paper, we discuss how HCPN can model the attacker's behaviors, intrusion's prerequisites and consequences, security policies, and the alerts. We argue that HCPN-ACU can conduct alert

fusion and intention recognition at the same time, reduce false positives and negatives, and provide better understanding of the intrusion progress.

The remainder of the paper is organized as follows. In section 2, we introduce the background and motivation of this research. Specifically, we discuss the task of ACU and the limitations in current ACU approaches. In section 3, we propose the HCPN- framework to model the ACU inference process, present basic theories related to the inference process, and describe how HCPN-ACU works. We introduce the inference and learning algorithms in section 4, and evaluate our system with the DARPA intrusion detection evaluation database in section 5. In section 6, we conclude this paper.

2 Background and Motivation

ACU is increasingly gaining attention as an area of research due to the following two reasons: the potential to improve efficiency by reducing the number of alerts that an IDS would generate to more manageable levels while still retaining strong detection capacities, and the potential to improve IDS correctness by reducing the false positives and negatives in the alerts generated by the IDS sensors and/or low level heterogeneous IDSs.

Fig. 1 depicts the architecture of an IDS that contains the ACU component. In this architecture, audit data are first analyzed and alerts are generated. These alerts are then fed as the observations into the ACU component. We can consider ACU as a second level analyzer or booster that uses the first level analyzers' results as inputs.

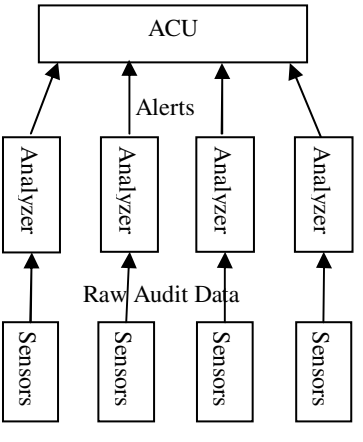

Fig. 1. The Architecture of an IDS that contains ACU

Three tasks are associated with ACU: aggregating alerts to reduce the total number of alerts presenting to the administrators and active reactors; reducing

false positives and negatives; and understanding the attacker's intrusion behavior and plan.

ACU is usually conducted as two steps: First, similar alerts from multi-sensors are aggregated into so called meta-alerts (or hyper-alerts) based on feature similarities. These meta-alerts are then correlated based on the prerequisite-consequence relationships [7,8,26,27].

Let's examine how current approaches work using the Distributed Denial of Service (DDoS) attack as an example. Assume that an intruder needs to conduct the following five steps to launch a DDoS attack:

1. IPsweep the hosts from a remote site;
2. Probe (SadmindPing) live IPs to look for the sadmind daemon running on Solaris hosts;
3. Break into some of the hosts via the sadmind vulnerability (SadmindBOF);
4. Install the Trojan mstream DDoS software on some of the hosts;
5. Launch the DDoS.

To correlate alerts with current ACU approaches, security experts build a set of rules that describe each action's prerequisites and consequences. In other words, each action is associated with a set of prerequisites that must be met before the attacker can take the action, and a set of consequences that the action would lead to. Table 1 lists the prerequisites and consequences of the SadmindPing action and the sadmindBOF action. From the table, we can see that the intruder can conduct the SadmindPing action only if he/she already knows that the host exists. As a result of probing, the attacker would know whether the sadmind daemon is running on the host. Similarly, an intruder usually launches the sadmind attack only if he/she already knows that the sadmind daemon is running on the host. After launching the attack, the intruder compromises the host.

Table 1. Prerequisites and consequences of actions SadmindPing and sadmindBOF

Action	**Prerequisites**	**Consequences**
SadmindPing	Knowledge that the host exists	Knowledge that the sadmind daemon is running on the host
SadmindBOF	Knowledge that the sadmind daemon is running on the host	The host is compromised

Let us consider the following three ACU scenarios. To make discussion easier, we assume that SadmindPing's prerequisites are always met and no other action other than SadmindPing would provide the prerequisites for SadmindBOF.

Scenario 1: alert_SadmindPing and alert_SadmindBOF are both issued by the low level analyzers. A typical ACU component will correlate these two alerts since the set of consequences of SadmindPing contains all the prerequisites of SadmindBOF.

Scenario 2: alert_SadmindBOF is issued by the low level analyzers but alert_SadmindPing is not. A typical ACU component will consider alert_Sadmind-BOF as a false positive since the prerequisites of SadmindBOF are not met.

Scenario 3: alert_SadmindBOF is issued by the low level analyzers ten times but alert_SadmindPing is not issued. A typical ACU component will first aggregate the ten alert_SadmindBOF into one hyper alert_SadmindBOF and then correlate the hyper alert_SadmindBOF with other alerts. Since the prerequisites of SadmindBOF are not met, alert_SadmindBOF is considered a false positive.

From these scenarios, we can observe three issues in current ACU approaches:

First, in current approaches, IDS's observations (alerts) are not distinguished from an attacker's real actions. This can be easily noticed when we examine the correlation process – **alerts** are correlated based on **actions**' prerequisites and consequences directly. An action is assumed to have happened iff the corresponding alert is issued and the prerequisites of the action are met.

However, alerts and actions are not one to one mapped. Due to false positives, the low level analyzers may issue alert_SadmindBOF while no Sadmind-BOF action is actually conducted. This suggests that the co-occurrence of alert_Sadmind-Ping and alert_SadmindBOF in scenario 1 does not necessarily mean that SadmindBOF is really carried out by the intruder. Similarly, due to false negatives, SadmindPing might be missed by the low level analyzers and no alert is issued. This suggests that issuing alert_SadmindBOF alone, as what happened in scenario 2, does not necessarily mean that SadmindBOF is not taken by the attacker. Table 2 summarizes these two error conditions.

Table 2. Conditions in which the current ACU approaches may generate errors

Scenario	**Alerts Issued**	**ACU Result**	**Failure Condition**
Scenario 1	alert_SadmindPing, alert_SadmindBOF	Both SadmindPing and SadmindBOF have happened	When alert_SadmindBOF is a false positive
Scenario 2	alert_SadmindBOF	SadmindBOF did not happen	When alert_SadmindPing is a false negative

Note that the reason ACU errors occur here is that alerts are treated the same as actions during the correlation process. Information such as false negative rate and false positive rate of that action is not used in the correlation process.

Second, the number of the occurrence of the same alert is not used in the correlation process due to the two-step strategy in current ACU approaches. The drawback of this limitation can be observed when comparing scenario 3 with scenario 2. We would guess that the action SadmindBOF very likely has happened in scenario 3 since the alert_SadmindBOF is issued ten times; while it less likely has happened in scenario 2, where alert_SadmindBOF is issued only once. However, as we already discovered, current ACU approaches typically

generate exactly the same result in both scenarios. The number of the occurrence of alert_SadmindBOF does not affect the correlation result.

Third, no confidence scores are provided in the current ACU approaches. Alerts are either correlated and should be delivered to the administrators, or not correlated, considered as false positives, and discarded. Using scenarios 2 and 3 as examples, confidence scores would aid administrators and active reactors to better understand the whole attack picture.

3 The Hidden Colored Petri-Net Framework

In this section, we propose a novel framework named Hidden Colored Petri-Net for Alert Correlation and Understanding (HCPN-ACU). HCPN is our extension to Colored Petri-Net (CPN) [19]. CPN has been used in modeling Discrete Event Dynamic Systems (DEDS) such as "communication protocols, operating systems, hardware designs, embedded systems, software system designs, and business process re-engineering" [21]. It has also been introduced to model the intruder's misuse behaviors [11,22,23].

An *HCPN-ACU* is an 11-tuple $HCPN = (\Sigma, Q, D, A, O, G, E, \Pi_0, \Delta, \Gamma, \Theta)$, where:

1. Σ (*color set*) is a non-empty finite set of *agents*;
2. Q (*place set*) is a finite set of *resources*;
3. D (*transition set*) is a finite set of *actions* agents might take;
4. A (*arc set*) is a finite set that $A = A_1 \cup A_2$, where $A_1 \subseteq (Q \times D)$, and $A_2 \subseteq (D \times Q)$;
5. O (*observation set*) is a set of *observations*. It can be alerts or raw audit and traffic data;
6. G (*guard function set*) is a set of *guard functions* associated with arcs A_1, such that $G = \{g : A_1 \to S_{M(\Sigma)}\}$. Guard functions represent the conditions to be met before an action can be conducted by the agents.
7. E (*effect function set*) is a set of *effect functions* associated with arcs A_2, such that $E = \{e : A_2 \to S_{M(\Sigma)}\}$. Effect functions represent the agent-resource relationship change due to an action.
8. Π_0 (initial marking distribution) is the initial agent-resource ownership probability distribution $\Pi_0 = P_0\left(Q, S_{M(\Sigma)}\right) = \{\pi : \left(Q, S_{M(\Sigma)}\right) \to [0,1]\}$.
9. Δ (transition probability) is the probability that actions might be conducted next: $\Delta = P\left(D \text{ will be fired next} | D \text{ is enabled}\right) = \{\delta : D \to [0,1]\}$
10. Γ (observation probability) is the probability that O is observed given action D and is defined as $\Gamma = P\left(O|D\right) = \{\gamma : (D, O) \to [0,1]\}$.
11. Θ (*tolerance*) is the tolerance function used to determine whether two states are indistinguishable.

In HCPN-ACU, a *token element* (q, c) stands for the fact that the agent c has access to resource q. An *enabled transition* means that the prerequisites of the corresponding actions are met. The *marking distribution* Π represents the

agent-resource ownership probability. The *progress of intrusion* is represented by the change of marking distribution along time.

The HCPN-ACU can be further simplified with the following default settings due to the nature of the IDS:

1. Use a transition named *normal* to absorb the false positives.
2. The number of token elements (q, c) does not affect the agent-resource ownership. For this reason, we need to consider only the probability $\{(q, c)\} \leq M$ and don't distinguish between one single token element and multiple ones.
3. All guard functions need only to care about the probability $\{(q, c)\} \leq M$ with the same reason.
4. We may add in the model an arc from the transition to each input places to indicate that the carrying out of the action would also affect the input. With these additional arcs, the system will be able to automatically infer that the input places have been compromised if the action is determined to have been taken. Thus, the model has potential to infer missing alerts from other alerts to reduce false negatives.

Let us use the local-to-root (L2R) attack from [16,11] as an example. The attack involves four actions: copy, chmod, touch, and mail. Each action would grant the access of one resource to the attacker. Fig. 2 depicts the HCPN-ACU model of this L2R attack. There are five transitions in the graph. Transitions are used to model the actions copy, chmod, touch, and mail; a special transition named "normal" is used to model the unintrusive actions. Six places are used in the figure to represent resources involved. The place q1 is a special place to model the resource that would be accessible to all agents. Arcs in the figure describe the prerequisites and consequences of actions. For example, an attacker needs to hold both q4 and q5 to be able to conduct the mail action. After the mail command is issued, the attacker would be able to hold q6. Each attacker is assigned a color. For instance, user1 might be represented as red. If q3 is dyed with red, q3 is compromised by user1.

Although the HCPN-ACU also describes the prerequisites and consequences of actions, there are several differences between HCPN-ACU and the ACU approaches discussed earlier in this paper.

First, instead of assuming a one-to-one mapping between alerts and actions, we assume that the low level analyzers may observe each action as different alerts with different probabilities (named observation probabilities). These probabilities are induced from the false positive rate and false negative rate of each action. For example, the copy action might be observed as alert_copy, alert_touch, or normal (simply missed). For this reason, the correlation results in HCPN-ACU are determined by alerts, the observation probabilities, and the number of each kind of alerts.

Second, HCPN-ACU presents the compromised resources instead of alerts to the administrators and active reactors. Since the number of compromised resources is usually much smaller than the number of alerts, this can effectively reduce the amount of data passed to the administrators and active reactors.

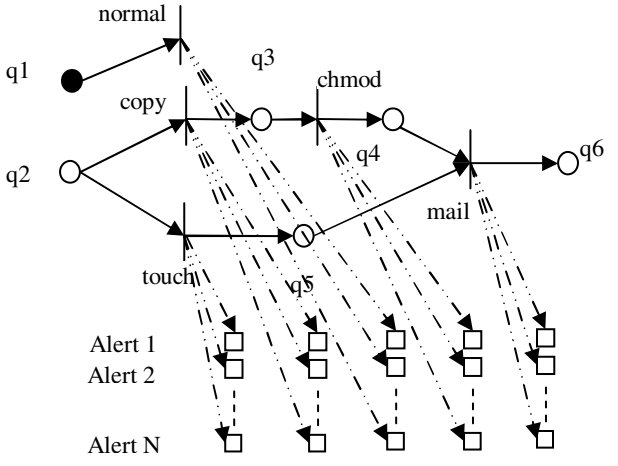

Fig. 2. An Example HCPN-ACU Model for the L2R Attack

Presenting compromised resources also helps administrators and active reactors to pick up a wise reaction.

Third, HCPN-ACU not only presents the compromised resources but also indicates the probability that a specified resource has been compromised by a specific intruder.

Three assumptions are made in HCPN-ACU:

First, the action prerequisites and effects are known as domain knowledge. This assumption is reasonable since the prerequisites and consequences of the alerts are usually known when the alerts are defined in an IDS,. All intention (or plan) recognition approaches are based on this assumption and they usually include this knowledge as rules in a database [7].

Second, the initial probability of resources owned by the agent can be determined by the system through such ways as policy and logon credentials. To deal with the situation where information is incomplete, we can assign a small probability to all agent/resource pairs using smoothing technology [31] to indicate that each resource may be accessible by an agent through unknown approaches.

Third, agents do not cooperate with each other. With this assumption, we can represent agents (identified as different source IPs and user IDs) with different colors and consider them separately. This assumption is valid for many intrusion cases because many attacks happened today are launched by isolated, script-based intrusion such as worms[1]. However, this assumption is not valid for sophisticated intrusions where a skilled attacker controls several agents and attacks the same system at the same time. To handle attacks launched by coop-

[1] We perceive that future worms may act as cooperative agents and would thus be more dangerous.

erating agents, an improved model is needed to correlate cooperating agents as "one" agent. We consider this as our future work.

4 Inference and Learning Algorithms

In this section, we briefly introduce the algorithm to infer an attacker's most probable action sequence given the model and the observations, as well as the algorithm to learn the model's parameters based on intrusion logs.

4.1 Basic Operations

In HCPN-ACU, a transition (action) is enabled iff all input places (prerequisites) satisfy the guards. In other words, given a marking distribution Π_t, the probability that a transition $d \in D$ is enabled can be determined by the following calculation:

$$P\left(E\left(d\right)|\Pi_t\right) = P\left(d \text{ is enabled}|\Pi_t\right) = P\left(\underset{q\in I(d)}{\wedge}\left(\Pi_t\left(q\right) \geq G\left(a=(q,d)\right)\right)\right)$$
$$= \prod_{q\in I(d)} P\left(\Pi_t\left(q\right) \geq G\left(a=(q,d)\right)\right) = \prod_{q\in I(d)} \pi_t\left(q\right) \ . \tag{1}$$

Similarly, a place q will be compromised by the color c iff it's compromised by c or at least one of the transitions (of which q is an output place) is enabled. Given a state $S_t = (\Pi_t, \Delta_t)$, the probability that a transition $d \in D$ will be fired next without knowing the observation can be determined by this calculation:

$$\delta_t^{'}\left(d\right) = P\left(D=d|\Pi_t\right) = \delta\left(d\right) P\left(E\left(d\right)|\Pi_t\right) = \delta\left(d\right) \prod_{q\in I(d)} \pi_t\left(q\right) \ . \tag{2}$$

$$\delta_t\left(d\right) = \frac{\delta_t^{'}\left(d\right)}{\sum\limits_{d'} \delta_t^{'}\left(d'\right)} \ . \tag{3}$$

Given the state S_{t-1}=(Π_{t-1}, Δ_{t-1}) and an observation O_t, the probability that the action d is taken is denoted as $P\left(D_t=d|S_{t-1},O_t\right)$ and can be determined by the following calculation

$$P\left(D_t=d|S_{t-1},O_t\right) = \frac{P\left(D_t=d,O_t|S_{t-1}\right)}{P\left(O_t|S_{t-1}\right)}$$
$$= \frac{P\left(D_t=d|S_{t-1}\right) P\left(O_t|D_t=d,S_{t-1}\right)}{P\left(O_t|S_{t-1}\right)} \ . \tag{4}$$

Because $P\left(D_t=d|S_{t-1}\right)$ is equal to $\delta_{t-1}\left(d\right)$ and O_t is independent of S_{t-1} given D_t, the above equation becomes:

$$= \frac{\delta_{t-1}\left(d\right) P\left(O_t|D_t=d\right)}{P\left(O_t|S_{t-1}\right)} = \frac{\delta_{t-1}\left(d\right)\gamma\left(O_t|d\right)}{\sum\limits_{d'\in D} \delta_{t-1}\left(d'\right)\gamma\left(O_t|d'\right)} \ . \tag{5}$$

4.2 Inference Problem

The inference problem (the correlation process) can be stated as follows: Given observations O_1,$O_2, \cdots, O_t$, and the model parameter λ, which action sequence, represented by D_1,$D_2, \cdots, D_t$, is most likely to have produced O from λ? That's to say, which sequence of state transitions is most likely to have led to this sequence of observations? In other words, we want to optimize the following criteria:

$$\arg\max_{D} \left[P\left(D|O,\lambda\right)\right] = \arg\max_{D} \left[\frac{P\left(O|D,\lambda\right)P\left(D|\lambda\right)}{P\left(O|\lambda\right)}\right] \ . \tag{6}$$

Since the term $P\left(O|\lambda\right)$ is not related to D, we can discard it when selecting paths. So we need to only optimize:

$$\arg\max_{D} \left[P\left(O|D,\lambda\right)P\left(D|\lambda\right)\right] = \arg\max_{D} P\left(O,D|\lambda\right) \ . \tag{7}$$

This problem can be solved with dynamic programming (DP) by defining $\omega_t\left(j\right)$ as the maximum score of a length t state sequence ending in action j and producing the first t observations from O, as shown in the following equation:

$$\omega_t\left(j\right) = \max_{D_1 \cdots, D_{t-1}} P\left(O_1, \cdots, O_t, D_1, \cdots, D_{t-1}, D_t = j|\lambda\right) \ . \tag{8}$$

where $\delta^i_{t-1}\left(j\right) \approx P\left(D_t = j|S^{'}_{t-1},\lambda\right)$ and $S^{'}_{t-1}$ is the state corresponding to $\omega_{t-1}\left(i\right)$.

4.3 Model Parameter Estimation Problem

The model parameter estimation problem can be stated as: Given observations $O_1, O_2, \cdots, O_t$, the model structure, and associated attacks, how can we estimate the model parameters so that the model best explains the known data.

We solve this problem with Expectation Maximum (EM) algorithm [25]. The EM algorithm consists of two major steps: an expectation step (E-Step), followed by a maximization step (M-Step). In the E-Step, the unobserved data (transitions in HCPN) is estimated based on the current model parameters λ_k. In the M-Step, Maximum Likelihood (ML) estimation is used to estimate model parameters λ_{k+1} using estimated data. This process is iterated until the segmentation is fixed. In our current system, we assume that the initial probabilities are determined based on the security policies. We need to estimate the observation probabilities and transition probabilities. Likelihood of observations given the observation probability $\theta = \gamma\left(o|d\right)$ is defined as:

$$\begin{aligned} L\left(\gamma, d\right) &= \sum_{i} \ln\left(P\left(O_i|\gamma, d\right)\right) = \sum_{O_i = o} \ln\gamma + \sum_{O_i \neq o} \ln\left(1-\gamma\right) \\ &= N\ln\gamma + L\ln\left(1-\gamma\right) \ . \end{aligned} \tag{9}$$

where N is the number of instances that O is observed when transition d is taken and L is the number of instances that O is NOT observed when transition d is taken. Observation probability is chosen to maximize the above likelihood as shown in the following equation:

$$\frac{\partial L(\theta, d)}{\partial \theta} = \frac{N}{\theta} - \frac{L}{1-\theta} = 0 \Rightarrow (N+L)\,\theta = N \Rightarrow \theta = \frac{N}{(N+L)} \quad . \tag{10}$$

Transition probabilities can be estimated similarly.

5 Experiments on DARPA Dataset

We have developed an off-line alert correlation system based on our HCPN-ACU framework and performed several experiments using the two DARPA 2000 intrusion detection evaluation datasets [24]. Each dataset includes the network traffic data collected from both the DMZ and the inside part of the evaluation network. In the datasets, attackers probe, break-in, install the DDoS daemon, and launch DDoS attacks.

Instead of running our low level analyzers to generate alerts, we used alerts generated by RealSecure Network Sensor 6.0 as what Ning et. al. [28] did: "In all the experiments, the Network Sensor was configured to use the *Maximum Coverage* policy with a slight change, which forced the Network Sensor to save all the reported alerts." We choose to use RealSecure Network Sensors because attack signatures used in RealSecure Network Sensor 6.0 are well documented, and Ning et. al. already have a set of rules to describe action's prerequisites and consequences.

In the experiments, we used the second dataset and one set of data (associated with one host) from the first dataset as the training set. We do this because the second dataset is lack of representative data. We used the first dataset as the testing set. We performed two sets of tests, one on the DMZ traffic and one on the inside network traffic.

The HCPN-ACU model used in the experiments consists of 20 places (resources), 29 transitions (actions), and 28 alerts. The actions used in the experiments have the same names as the alerts. However, each action might be observed by the sensors as different alerts. We used 0.02 as the initial probability for all resources other than the resource known to all users - SystemExisits. The training takes less than 20 seconds and the inference takes less than 5 seconds for about 900 alerts on a Celeron 1.0GHz PC.

As mentioned in section 3, our HCPN-ACU system outputs resources compromised instead of alerts themselves. Table 3 lists the correlation result for the inside network traffic. From the table we can see that the attacker has installed daemons on hosts 172.016.112.010, 172.016.112.050, and 172.016.115.020, and ready to launch the DDoS attack. Note, however, our system does not report that it's ready to launch DDoS attack on host 172.016.115.020 due to false negatives.

Table 3. Correlation results for the inside traffic

Host	Place Name	Probability
172.016.112.010	SystemCompromised	1.00
	VulnerableSadmind	0.66
	DaemonInstalled	0.55
	ReadyToLaunchDDOSAttack	0.95
172.016.112.050	SystemCompromised	1.00
	VulnerableSadmind	0.66
	DaemonInstalled	0.55
	ReadyToLaunchDDOSAttack	0.95
172.016.115.020	SystemCompromised	1.00
	VulnerableSadmind	0.66
	DaemonInstalled	0.80
131.084.001.031	DDoSHappened	0.90

Table 4 shows the detection and false alert rates for RealSecure Network Sensor 6.0. Table 5 shows the experiment results of our approach. We separated them into two tables because our approach presents different information.

Table 4. Detection Rate (DR) and False Alert Rates (FAR) for RealSecure Network Sensor 6.0: AD = Attacks Detected; RA = Real Attacks

Dataset	# of Attacks	# of Alerts	# of AD	DR	# of RA	FAR
DMZ	89	891	51	57.30%	57	93.60%
Inside	60	922	37	61.67%	44	95.23%

Table 5. Detection and False Positive Rates (FPR) for HCPN-ACU: CR = Compromised Resources; T = True; D = Detected

Dataset	# of CR	# of CR Shown	# of D CR	Detect Rate	# of T CR	FPR
DMZ	12	15	12	100.00%	12	20%
Inside	13	12	12	92.31%	12	0%

We counted the numbers in Table 4 the same way as what Ning et. al. did [28]. When counting the compromised resources in Table 5, we noticed that the attacker tried Sadmind_Amslverify_Overflow towards the targets 172.016.114.010, 172.016.114.020, and 172.016.114.030. No additional attacks were carried out against these hosts. This suggests that the Sadmind_Amslverify_Overflow attacks were failed. However, since our rules indicate that the consequence of the Sadmind_Amslverify_Overflow attack is SystemCompromised, our HCPN-ACU would report that these hosts are compromised. In our calculation, we consid-

ered these reports as false positives. These false positives might be eliminated by using the system configuration information in the prerequisites.

From Table 4 and 5, we can clearly observe that HCPN-ACU can reduce the number of "alerts" presented to the administrators and active reactors, improve the detect rate, and reduce the false positive rate.

6 Conclusions and Future Work

In this paper, we described a novel framework named HCPN-ACU for the alert correlation and understanding task. We showed that HCPN-ACU has the following features:

- It combines alert fusion and intention recognition in one system.
- It presents resources compromised to show the progress of an attack instead of alerts themselves. Since the number of resources compromised are much smaller than the number of alerts generated, HCPN-ACU can reduce the number of alerts shown to the administrators and active reactors.
- It can reduce false positives with a special transition named "normal action". In the inference process, false positives are automatically associated with this transition.
- It can reduce the false negatives because later alerts would increase the probability that a missing action has happened.
- It provides confidence scores to the detection result by assigning probabilities to each mark indicating how likely an attacker has compromised a resource.
- The inference process is very efficient and the HCPN can be organized in layers to scale up. This makes it applicable in real world systems.

We perceive three weaknesses of HCPN-ACU. First, it requires the knowledge of the alerts and the system to be protected. For large networks, complete system information may not be easily available. Second, it requires training data to learn the system parameters. Training data might be difficult to get in real system. Third, a careful intruder may fool the system by carrying out specially designed steps.

Our system can be improved in the following two areas:

- **Experiments on alerts from multiple sources**: Our current experiments are carried out on DMZ and inside network traffics separately. It would be interesting to see the results using the information from both sources.
- **Detection of coordinated attacks**: One of the major assumptions in the current framework (and all other intention recognition based approaches) is that no attacks are cooperative. This may not be true when sophisticated attacks happen. To eliminate this assumption, we plan to integrate attacker correlation into the HCPN.

References

1. D. Armstrong, S. Carter, G. Frazier, T. Frazier: A Controller-Based Autonomic Defense System. Proc. of DARPA Information Survivability Conference and Exposition (DISCEX), 2003
2. J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner: State of the Practice of Intrusion Detection Technologies. Technical Report CMU/SEI-99-TR-028, 1999
3. J.P Anderson: Computer Security Threat Monitoring and Surveillance. Technical report, James P Anderson Co., Fort Washington, Pennsylvania, April 1980
4. S. Axelsson: The base-rate fallacy and its implications for the difficulty of intrusion detection. In 6th ACM Conference on computer and communications security, pp 1-7, November 1999
5. D. Barbara, S. Jajodia: Applications of Data Mining in Computer Security. Kluwer Academic Pub, June 2002
6. R. C. de Boer: A Generic Architecture for Fusion-Based Intrusion Detection Systems. Master Thesis, Erasmus University Rotterdam, October 2002
7. F. Cuppens, F. Autrel, A. Miège and S. Benferhat: Correlation in an intrusion detection process. Internet Security Communication Workshop (SECI'02), Septembre 2002
8. F. Cuppens, A. Miège: Alert Correlation in a Cooperative Intrusion Detection Framework. IEEE Symposium on Security and Privacy, May 2002
9. F. Cuppens: Managing Alerts in a Multi-Intrusion Detection Environment. In 17th Annual Computer Security Applications Conference, New-Orleans, USA, December 2001
10. H. Debar and A. Wespi: Aggregration and Correlation of Intrusion-Detection Alerts. In Proceedings of the 4th International Symposium on Recent Advances in Intrusion detection (RAID), 2001.
11. D. Frincke, D. Tobin, and Y. Ho: Planning, Petri Nets, and Intrusion Detection. In Proceedings of the 21st National Information Systems Security Conference (NISSC'98), 1998
12. C. Geib and R. Goldman: Plan Recognition in Intrusion Detection Systems. In DARPA Information Survivability Conference and Exposition (DISCEX), June 2001
13. R. P. Goldman, W. Heimerdinger, S. Harp, C. W. Geib, V. Thomas, and R. Carter: Information Modeling for Intrusion Report Aggregation. In Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX), June 2001
14. J. Haines, D. K. Ryder, L. Tinnel, S. Taylor: Validation of Sensor Alert Correlators. IEEE Security and Privacy, January-February 2003 (Vol. 1, No. 1) pp. 46-56
15. M.-Y. Huang, and T. M. Wicks: A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis. Web proceedings of the First International Workshop on Recent Advances in Intrusion Detection (RAID'98), 1998
16. K. Ilgun, R. Kemmerer, and P. Porras: State Transition Analysis: A Rule-Based Intrusion Detection System. IEEE Transactions on Software Engineering, 21(3), Mar. 1995
17. K. Julisch and M. Dacier: Mining intrusion detection alarms for actionable knowledge. In Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining, pp 366-375, July 2002

18. K. Jensen: An Introduction to the Theoretical Aspects of Coloured Petri Nets. In J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science vol. 803, Springer-Verlag 1994, pp230-272
19. K. Jensen: Colored Petri-Nets–Basic Concepts, Analysis Methods, and Practical Use, 2nd ed. New York: Springer-Verlag, 1996, vol. 1.
20. K. Julisch: Mining Alarm Clusters to Improve Alarm Handling Efficiency. In Proceedings of the 17th ACSAC, New Orleans, December 2001
21. L. M. Kristensen, S. Christensen, K. Jensen: The practitioner's guide to coloured Petri nets. Int. Journal on Software Tools for Technology Transfer, 2 (1998), Springer-Verlag, pp98-132
22. S. Kumar and E.H. Spafford: A Pattern-Matching Model for Intrusion Detection. Proceedings of the National Computer Security Conference, 1994
23. S. Kumar and E. Spafford: A Pattern Matching Model for Misuse Intrusion Detection. In 17th National Computer Security Conference, 1994
24. Lincoln Lab, MIT. DARPA 2000 intrusion detection evaluation datasets. http://ideval.ll.mit.edu/2000 index.html, 2000.
25. T. Moon: The Expectation-Maximization algorithm. IEEE Signal Processing Magazine, pp. 47–60, Nov. 1996
26. P. Ning, Y. Cui, D. S. Reeves: Analyzing Intensive Intrusion Alerts Via Correlation. In Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002), LNCS 2516, pp 74-94, October 2002
27. P. Ning, Y. Cui, D. S. Reeves: Constructing Attack Scenarios through Correlation of Intrusion Alerts. In Proceedings of the 9th ACM Conference on Computer & Communications Security, pp 245-254, November 2002
28. P. Ning, D. S. Reeves, Y. Cui: Correlating Alerts Using Prerequisites of Intrusions. Technical Report, TR-2001-13, North Carolina State University, Department of Computer Science, December 2001
29. P. A. Porras,M. W. Fong, and A. Valdes: A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. Proceedings Recent Advances in Intrusion Detection. October, 2002. Pp 95-114
30. P. A. Porras and P. G. Neumann: EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. 1997 National Information Systems Security Conference, October, 1997
31. J. S. Simonoff: Smoothing Methods in Statistics. Springer-Verlag, 1998
32. S. J. Templeton, K. Levitt: A requires/provides model for computer attacks. Proceedings of the 2000 workshop on New security paradigms, Pp 31-38, 2001
33. A. Valdes and K. Skinner: Probabilistic Alert Correlation. In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID) 2001
34. N. Ye, X. Li, Q. Chen, S. M. Emran, and M. Xu: Probabilistic techniques for intrusion detection based on computer audit data. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 31, No. 4, 2001, pp. 266-274
35. N. Ye, J. Giordano, J. Feldman, and Q. Zhong: Information Fusion Techniques for Network Intrusion Detection. 1998 IEEE InformationTechnology Conference, Information Environment for the Future, 1998.