Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 632–654
Computer Networks
www.elsevier.com/locate/comnet

# Improving the quality of alerts and predicting intruder's next goal with Hidden Colored Petri-Net ☆

Dong Yu *, Deborah Frincke

*Department of Computer Science, University of Idaho, Moscow, ID 83844-1010, United States*

Received 3 August 2005; received in revised form 27 January 2006; accepted 19 May 2006
Available online 22 June 2006

Responsible Editor: Christos Douligeris

**Abstract**

Intrusion detection systems (IDS) often provide poor quality alerts, which are insufficient to support rapid identification of ongoing attacks or predict an intruder's next likely goal. In this paper, we propose a novel approach to alert postprocessing and correlation, the Hidden Colored Petri-Net (HCPN). Different from most other alert correlation methods, our approach treats the alert correlation problem as an inference problem rather than a filter problem. Our approach assumes that the intruder's actions are unknown to the IDS and can be inferred only from the alerts generated by the IDS sensors. HCPN can describe the relationship between different steps carried out by intruders, model observations (alerts) and transitions (actions) separately, and associate each token element (system state) with a probability (or confidence). The model is an extension to Colored Petri-Net (CPN). It is so called "hidden" because the transitions (actions) are not directly observable but can be inferred by looking through the observations (alerts). These features make HCPN especially suitable for discovering intruders' actions from their partial observations (alerts) and predicting intruders' next goal. Our experiments on DARPA evaluation datasets and the attack scenarios from the Grand Challenge Problem (GCP) show that HCPN has promise as a way to reducing false positives and negatives, predicting intruder's next possible action, uncovering intruders' intrusion strategies after the attack scenario has happened, and providing confidence scores.


*Keywords:* Intrusion detection; Alert correlation; Hidden Colored Petri-Net

## 1. Introduction

One of the most important requirements of a good intrusion detection system (IDS) is the generation of high quality alerts. Unfortunately, IDS sensors usually generate massive amount of alerts [1], especially if those sensors have high sensitivity to potential misuse, as can be the case with tightly tuned anomaly based sensors. In the distributed

☆ This is an extended and enhanced version of the work published in the International Conference on Applied Cryptography and Network Security, Yellow Mountain, China, 2004 [39].

* Corresponding author. Tel.: +1 425 707 9282; fax: +1 425 706 7329.
*E-mail addresses:* dongyu@csds.uidaho.edu (D. Yu), frincke@cs.uidaho.edu (D. Frincke).

1389-1286/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2006.05.008

case, the situation is compounded because there are more sensors (and hence more data), and greater chance of time delay before sensor alerts are consolidated – with commensurate risk that some information will be inaccurate or stale. To make things worse, only 1% of the enormous amount of alerts generated by most IDS corresponds to unique attacks [1–3]. The remainders are false positives (i.e., alerts on non-intrusive actions), repeated warnings for the same attack, or alert notifications arising from erroneous activity or configuration artifacts.

Many ways are available to improve the quality of alerts. For example, we may achieve the goal by using better sensors, signatures, or analysis algorithms. In this paper, we focus on algorithms, and propose a novel alert correlation approach. In other words, our approach aims to reduce the false positives and false negatives by postprocessing (i.e., correlating) the alerts in a novel way: inferring an intruder's actions with a model named Hidden Colored Petri-Net (HCPN). The architecture of our system is depicted in Fig. 1. Raw audit data collected by sensors are first analyzed to generate alerts. These alerts, which contain large amounts of false positives, are then fed into our HCPN-based alert correlators to remove most of false positives and repetitions. Confidence scores of alerts from the HCPN components installed at different sites are then fused with our extended Dempster-Shafer theory of evidence to further improve the quality of alerts before they are sent to the active responser to make the reaction decision.

Alert correlation [1,4–20] is used to (a) reduce the number of alerts that an IDS would generate to more manageable levels while still retaining strong detection capacities, (b) improve IDS correctness by reducing the false positives and negatives in the alerts generated by the IDS sensors, and (c) unveil an intruder's intrusion strategy after the attack has happened.

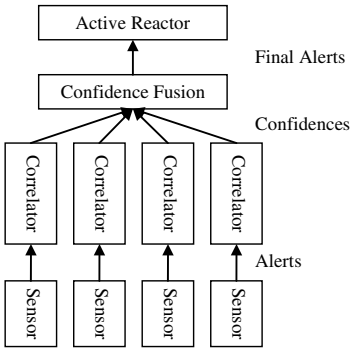


Fig. 1. Raw audit data collected by sensors are first analyzed to generate alerts. These alerts, which contain a large amount of false positives, are then fed into the alert correlators. Confidence scores provided by the alert correlators located at different sites are fused before being sent to the active responser to make the reaction decision.

One way to look at the alert correlation problem is to extract "true" alerts (or filter out the false alerts) from the raw alerts generated by the IDS sensors by utilizing relationships (e.g., similarities, sequential relationships, etc.) among alerts. This filter view of alert correlation, which will be discussed further in Section 5, has been taken by Cuppens et al. [13,14] and Ning et al. [17–19], to name a few. Approaches based on this filter view can remove (or filter out) large percentage of false positives. However, this formulation of the problem works directly upon the alerts. It does not distinguish between alerts and intruders' actions in the correlation process, and usually does not use information such as false negative rate and false positive rate to improve the correlation results.

In this paper, we take a different view and consider alert correlation as the problem of inferring an intruder's actions based on partial observations – alerts, progressively. Based on this perspective, we propose a novel methodology for analyzing alerts. Our approach is based on a theoretical model named Hidden Colored Petri-Net (HCPN). Based on this inference view, we assume an intruder's (either an actual intruder or an insider who is misusing the system) actions are unknown to the IDS and can be inferred only from the enormous amount of low quality alerts generated by the IDS sensors. We demonstrate that HCPN, as an extension to Colored Petri-Net [22,33,34], can describe the relationship between different steps carried out by intruders, model observations (alerts) and transitions (actions) separately, and associate each token element (system state) with a probability (or confidence). The model is called "hidden" because the transitions (actions) are modeled as hidden variables (i.e., not directly observable by the analyzer) in HCPN. However, it can look through the observations (alerts) to infer the transitions (actions). When no training data are available, HCPN behaves in a similar way as other alert correlation approaches that solely depend on the precondition–postcondition relationship. When training data are available, it can learn from the data and further improve the alert correlation result.

We report experimental results on the DARPA 2000 DDoS evaluation datasets and attack scenarios from the Grand Challenge Problem (GCP) provided by DARPA's Cyber Panel Program [21]. Our experiments[1] show that HCPN can significantly reduce false positives and negatives, effectively predict intruder's next possible action, uncover intruders' intrusion strategies after the attack scenario has happened, and provide confidence scores.

The remainder of the paper is organized as follows. In Section 2, we propose the HCPN framework to predicting attacks and understanding alerts. In Section 3, we discuss the basic computations underneath the operation of the HCPN model. We describe the model parameter estimation and action inference algorithms of HCPN in Section 4, describe the confidence fusion algorithm in Section 5, and report experimental results in Section 6. We discuss related work in Section 7, and conclude the paper in Section 8.

## 2. Hidden Colored Petri-Net: our framework

Our alert correlator is built upon a theoretical framework named Hidden Colored Petri-Net (HCPN). The model has been evolved through several stages. At the very beginning, we used CPN to model the preconditions and postconditions for misuse detection [25] since CPN allows unordered and/or concurrent event to be encoded in the network structure. Later on, we applied CPN to a higher level to improve the alert quality for scenario attacks by using the preconditions and postconditions built into the CPN. This is the time when we noticed that what encoded in the CPN are the preconditions and postconditions of ACTIONS instead of alerts. Although actions and alerts are highly correlated they are different (i.e., an alert is just an observation of an action). This suggests the separation of the alerts and actions in the model: the actions are hidden and the alerts are observable by the correlator. This separation of the actions and alerts evolved CPN into HCPN.

[1] DARPA datasets are the best labeled datasets we have access to, and have been used by many other researchers [17,18,37]. However, we would like to warn readers on the limitations of the DARPA datasets as indicated by McHugh [42] and others when interpreting our results.

### 2.1. Definition of HCPN

**Definition 1.** An HCPN is an 11-tuple $\mathrm{HCPN} = (\Sigma, Q, D, A, O, G, E, \Pi_0, Z, \Gamma, \Xi)$, where:

- $\Sigma = \{c_i | i = 1, \ldots, N_c\}$ (color set) is a non-empty finite set of agents (normal users and/or intruders);
- $Q = \{q_i | i = 1, \ldots, N_q\}$ (place set) is a finite set of states (e.g., a resource has been taken over);
- $D = \{d_i | i = 1, \ldots, N_d\}$ (transition set) is a finite set of actions agents might take;
- $A$ (arc set) is a finite set that $A = A_1 \cup A_2$, where $A_1 \subseteq (Q \times D)$ is the set of precondition links, and $A_2 \subseteq (D \times Q)$ is the set of postcondition links. We use $I(d)$ to denote the set of places of which the arc $\langle q, d\rangle \in A_1$, and use $O(d)$ to represent the set of places of which the arc $\langle d, q\rangle \in A_2$;
- $O = \{o_i | i = 1, \ldots, N_o\}$ (observation set) is a set of observations. It can be alerts or raw audit/traffic data. In this paper, observations are alerts;
- $G$ (precondition function set) is a set of precondition functions associated with arcs $A_1$, such that $G = \{g : A_1 \to S_{M(\Sigma)}\}$, where $S_{M(\Sigma)}$ is the superset of the multiset $M(\Sigma)$ [22]. Precondition functions represent the conditions to be met before an action can be conducted by the agents;
- $E$ (postcondition function set) is a set of postcondition functions associated with arcs $A_2$, such that $E = \{e : A_2 \to S_{M(\Sigma)}\}$. Postcondition functions represent the agent–resource ownership change due to an action;
- $\Pi_0$ (initial marking distribution) is the initial (or default) agent–resource ownership probability distribution $\Pi_0 = P_0(Q, S_{M(\Sigma)}) = \{\pi : (Q, S_{M(\Sigma)}) \to [0, 1]\}$;
- $Z$ (transition probability) is the probability that an action $d$ will be conducted given that the preconditions of the transition are satisfied (or, in other words, if the transition $d$ is enabled):

$$\begin{aligned} \Delta &= \{P(d \in D \text{ will fire next} | d \text{ is enabled})\} \\ &= \{z : D \to [0, 1]\}. \end{aligned}$$

  Note that, if the preconditions are not satisfied, the probability that an action will be taken is 0;
- $\Gamma$ (observation probability) is the probability that $O$ is observed given action $D$ and is defined as $\Gamma = P(O|D) = \{\gamma : (D, O) \to [0, 1]\}$;
- $\Xi$ (tolerance) is the tolerance function used to determine whether two states are indistinguishable.

HCPN is an extension to the Colored Petri-Net (CPN) first introduced by Jensen [23] and hence inherits many concepts and notations from CPN. Besides the key elements inherited from CPN (e.g., places, tokens, colors, and transitions), HCPN introduces an additional element called observations. The advantage of HCPN as compared to CPN is the ability to provide probabilities, and to model transitions and the observations separately.

Fig. 2 illustrates how HCPN can be used in the alert correlation task: colors represent agents; places represent resources; observations represent alerts; transitions represent actions; input arcs represent preconditions of the action; and output arcs represent postconditions of the action. Note that when an action is known to have been taken (from other information), we can deduce that the preconditions of the action are satisfied. In other words, there should be an output arc from the action to each of the input places in the HCPN to indicate this effect. For this reason, there is always a matched implicit output arc for each input arc in HCPN. With these implicit arcs, HCPN has the potential to recover the false negatives (i.e., those missing alerts).

In HCPN, a *token element* $(q, c)$ stands for the fact that the agent $c$ has access to resource $q$. An *action is enabled* if the preconditions of the corresponding action are satisfied. The *marking* $M$ represents the agent–resource ownership. The *marking distribution* $\Pi = P(Q, S_{M(\Sigma)}) = \{\pi : (Q, S_{M(\Sigma)}) \to [0,1]\}$ represents the agent–resource ownership probability. $\pi_t(q, c)$ represents the probability that at time $t$ resource $q$ has been taken over by agent $c$. The transition firing probability $\Delta$ represents the probability that each action will be taken next. $\delta_t(d, c)$ represents the probability that at time $t$ the action $d$ will be taken next by agent $c$. The *progress of intrusion* is represented by the change of marking distribution along time. The input of HCPN is the alerts, and the output of HCPN is the resources compromised by intruders. If the change of marking distribution between the current state and the initial state exceeds the tolerance, it is considered that a remarkable progress has been made by the agent.

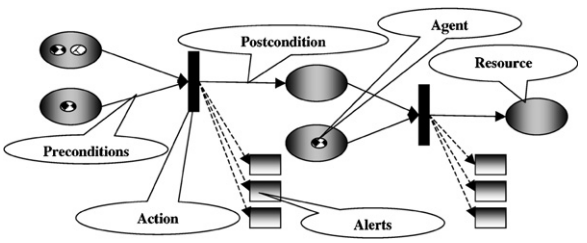


Fig. 2. Hidden Colored Petri-Net. In HCPN, colors represent agents; places represent resources; observations represent alerts; transitions represent actions; input arcs represent preconditions of the action; and output arcs represent postconditions of the action.

If the token element $(q, c)$ is in a marking $M$ (which is a multiset), agent $c$ has access to (or owns) resource $q$. Multiple token elements $(q, c)$ would not affect the agent–resource ownership. For this reason, we need to consider only the probability that the multiset $M$ is greater than (or contains) the multiset $\{(q, c)\}$, i.e., $\{(q, c)\} \leqslant M$ [22] and do not distinguish between one single token element and multiple ones. All precondition functions used in this paper thus have the same form: $\{(q, c)\} \leqslant M$, and the postconditions of any action $d$ have the same form: $M = M + \sum_{q \in O(d)} \{(q, c)\}$.

### 2.2. Examination of HCPN with examples

We use two examples to describe how the HCPN-based approach works. The first example is the classic local-to-root (L2R) attack scenario from [24,25], and the second example is the remote-to-local (R2L) and L2R attack scenario from [26]. These examples involve several steps and are described here just to show how HCPN works. A real world attack scenario may be far more complicated than the examples shown here.

The L2R attack scenario from [24,25] involves four actions: copy, chmod, touch, and mail. Each action would grant the intruder access to one resource. Fig. 3 depicts the HCPN model of this L2R attack. There are five named transitions in the graph: one for each action: copy, chmod, touch, and mail; and a special extra transition named "normal" whose preconditions are always satisfied and there is no postcondition to model the un-intrusive actions. (Note that our approach is to identify attacks by associating each alert with an action.) If an alert is inferred to be associated with the "normal" action, the alert is considered as a false positive. Six named places are used in the figure to represent resources (or attack states) involved. The place q1 is used to model the resource accessible to all agents. Arcs describe the preconditions and postconditions of actions. For example, an intruder needs to hold both q4 and q5 to be able to conduct the mail action. After the mail command is issued, the intruder would be able to hold q6. Each action might be identified by the IDS sensors as different

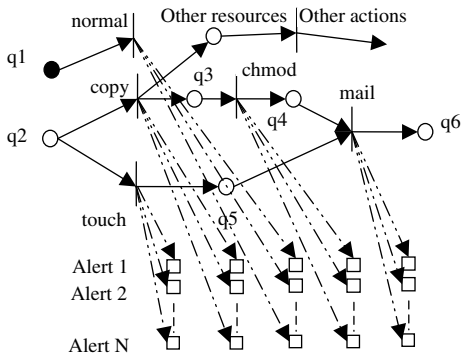


Fig. 3. An example HCPN model for an L2R attack. There are five transitions in the graph: transitions used to model the actions copy, chmod, touch, and mail, and the special "normal" transition used to model the un-intrusive actions. Six places are used in the figure to represent resources involved. The place q1 is a special place to model the resource that would be accessible to all users. Arcs in the figure describe the preconditions and postconditions of actions. Each action may be observed as different alerts with different probabilities. Note that resources and actions related to other attacks are also part of the whole HCPN.

alerts with different probabilities. These alerts (Alert 1 to Alert *N*), which might be reported by sensors at the host level or network level, are observations in the HCPN model. For example, the normal action might trigger different alerts in IDS, whose false positive rate is high. Note that, we only name the places, transitions, and arcs related to this L2R attack scenario in Fig. 3. The complete system would include additional elements (indicated as other resources and other actions in Fig. 3) associated with other attack scenarios. This specific L2R attack would then be only one path in the complete system.

The second example is the attack scenario from [26]. In this example, there are two hosts in a LAN. Host 2 has a database installed. Ftp service is running on both host 1 and 2, and ssh service is running on host 1. The intruder's goal is to compromise the information in the database stored on host 2.

Four actions (sometimes called atomic attacks) are available to the intruder in this scenario. The first action is a remote-to-root attack named Sshd Buffer Overflow (SBO). This attack immediately gives a remote user a root shell on the vulnerable machine (host 1 in our example). The precondition of this action is that the intruder has user-level privilege at the source host, and the target host is running a vulnerable version of sshd. The second action, named ftp-rhost (ftpR) attack, exploits an ftp vulnerability that allows an intruder to create a .rhosts file in the ftp home directory which establishes a remote login trust relationship between the source host and the target host. The precondition of this action is that the intruder has user-level privilege on the source host, and the target host is running a vulnerable ftp service. The third action is called remote login (rlogin) attack. In this attack, the intruder logs into the target host using an existing remote login trust relationship between two hosts. The fourth action is a local buffer overflow (LBO) attack. In this attack, the intruder already has the user-level privilege of the target host and gains the root privilege by exploiting the local setuid buffer overflow vulnerability.

An HCPN model for this attack scenario would include a normal action transition and a transition for each action described above. Note that since the precondition functions in the HCPN can take parameters, we do not need to have a separate transition and/or place for different hosts. The host (identified by the IP address) can be a parameter in the model conditions.

The intruder can have many strategies to compromise the database on the host 2. An example strategy is to use SBO attack to get the root privilege of host 1, use the ftpR attack to generate the trust relationship between host 1 and 2, use the login attack to gain the user-level privilege on host 2, and then launch the LBO attack to gain the root privilege.

It is obvious that the remote login is not conducted by an attacker most of the time. For this reason, you would see many warnings on non-intrusive remote login if an alarm is issued for each remote login. Using HCPN, most remote login alerts would be automatically associated with the special normal action transition, so no alarm would be issued for those remote login requests. However, HCPN would likely issue alarms if the previous steps have been conducted by the intruder. It may also issue an alarm if the next steps are perceived to have been conducted with high confidence.

HCPN behaves differently from the filter-based alert correlation approaches. Here we describe these differences with the L2R example we just described. To make discussion easier, we use alert_actionName to represent the alert related to the action named actionName.

First, instead of assuming a one-to-one mapping between alerts and actions in the filter-based alert correlation approaches, the HCPN models assume that the sensors may observe each action as different

alerts with different probabilities (named observation probabilities). These probabilities can be deduced from the false positive rate and false negative rate of each action, which will be discussed in Section 4. For example, the copy action might be correctly observed by IDS as alert_copy, missed by IDS (i.e., the action is observed as a normal behavior), or incorrectly identified as alert_touch with some low probability.

By using the observation probability, HCPN can determine with high accuracy whether the touch action really happened when an alert_touch was issued. Obviously, three possible conditions exist when the alert_touch was observed. The first possibility is that the touch action really happened. The second possibility is that alert_touch is just a false positive and is corresponding to a normal action. The third possibility is that a different action (such as the copy action) has happened since the sensors have small probability to mistakenly observe the copy action as touch and issue the alert_touch. HCPN would choose the best explanation based on all the information available at the current stage. Similarly, HCPN would not always think that the agent did not have access to the resource q3 just because alert_copy was not observed since the IDS sensors might miss the copy action (i.e., a false negative). The (implicit) output arcs from the action chmod to the resource q3 would allow HCPN to infer that q3 was accessible by the agent if the chmod action is believed to have been conducted from other information.

In brief, the correlation results in HCPN are determined by the alerts, the preconditions and postconditions of actions, the observation probabilities, the transition probabilities, and the number of each kind of alert. We can informally show that the information used in HCPN is a superset of features used by filter-based approaches and thus HCPN can potentially provide equivalent or better correlation results.

We now show that filter-based alert correlation approaches can be represented as special constructions of HCPN. For example, we can build an HCPN model for alert correlation approaches based ONLY on the preconditions and postconditions of malicious actions (a real filter-based system might be more complicated). To convert such a system to an HCPN-based system, we first construct a CPN (without the observation layer) that describes the same preconditions and postconditions of malicious actions with input and output arcs respectively. We then add the observation layer to make it an HCPN and set the observation probabilities to

$$P(o_i|d_j) = \begin{cases} 1 & i = j, \\ 0 & \text{otherwise}, \end{cases} \quad (1)$$

to indicate that alerts and actions are treated exactly the same (as in those systems). We further set the transition probabilities (i.e., the probability that an action will be taken given that all its preconditions are satisfied) to 1. With these settings, the HCPN behaves the same as the original filter-based model: For each new alert $o$, only the corresponding action $d$ (with which $P(o|d) = 1$) is likely to be taken since for any other transition $d'$, $P(o|d') = 0$. If the preconditions are not satisfied, the probability that the action has been taken is 0 (which is the same as to say that the alert is false if the preconditions are not satisfied). If the preconditions are satisfied, the probability that the action is taken is 1 (which is the same as to say that the alert is true if the preconditions are satisfied). If the alert is considered to be true, the action's postconditions are set to be true.

Second, HCPN presents the compromised resources instead of alerts of actions to administrators and/or active reactors. Because the number of compromised resources is usually smaller than the number of actions, this can effectively reduce the amount of data passed to the administrators and active reactors. Presenting the resources compromised by the intruders can also help administrators and active reactors to make better decisions on what to do to minimize further damage.

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

## 3. Basic algorithms

The initial state of HCPN is determined by the initial probabilities. After that, HCPN reevaluates its state (i.e., the marking distribution $\Pi_t$) after receiving each alert from the sensors. To reevaluate the state, HCPN needs to first determine the probabilities that each action is enabled (i.e., the preconditions of the action are satisfied). Given these probabilities, HCPN can determine which action is most likely to be taken next and thus which resources are in danger. It can also determine which action is most likely to have happened given the new alert, and thus which resources have been taken over by the agent. To make the equations easier to

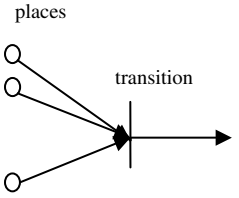


Fig. 4. An action is enabled if and only if all preconditions are satisfied.

read, agent $c$ is always assumed and is removed from all the equations from now on.

**Computation 1.** In HCPN, an action is enabled if and only if all preconditions are satisfied (Fig. 4). In other words, given a marking distribution $\Pi_t$, the probability that an action $d \in D$ is enabled with respect to agent $c$ can be estimated using Eq. (2). Note that by definition, the relationship between all preconditions is conjunctive. If a disjunctive relationship is needed, a duplicate action (possibly with a different ID and is treated as a different action) needs to be represented in the HCPN. Computation 1 is used to determine how likely each of the actions are ready (all preconditions are satisfied) to be taken by an agent at time $t$.

$$\begin{aligned} P(E(d)|\Pi_t) &= P(d \text{ is enabled}|\Pi_t) \\ &= P\left(\bigwedge_{q\in I(d)} (\Pi_t(q) \geqslant G(a=(q,d)))\right) \\ &\approx \prod_{q\in I(d)} P(\Pi_t(q) \geqslant G(a=(q,d))) \\ &= \prod_{q\in I(d)} \pi_t(q). \end{aligned} \tag{2}$$

**Computation 2.** Given a marking distribution $\Pi_t$, the probability that an action $d \in D$ will be taken next without knowing the alert can be determined by Eqs. (3) and (4). At any given time, multiple actions might be enabled (with different probabilities). This computation predicts the likelihood that each action will be taken next by the agent before receiving any new alert. As you can see, this computation depends on Computation 1 to calculate $P(E(d)|\Pi_t)$.

$$\begin{aligned} \delta'_t(d) &= P(D=d|\Pi_t) \\ &= P(d \text{ will fire next}|E(d))P(E(d)|\Pi_t) \\ &= z(d)\prod_{q\in I(d)} \pi_t(q), \end{aligned} \tag{3}$$

$$\delta_t(d) = \frac{\delta'_t(d)}{\sum_{d'} \delta'_t(d')}. \tag{4}$$

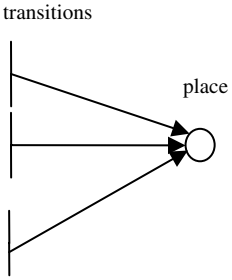


Fig. 5. A resource $q$ will be compromised by agent $c$ if and only if it is compromised by $c$ already or at least one of the action is to be taken next.

**Computation 3.** Given the state $S_t = (\Pi_t, \Delta_t)$ and without knowing the alert, a resource $q$ will be compromised by agent $c$ if and only if it is already compromised by $c$ or at least one of the actions (whose postcondition is to compromise resource $q$) is to be taken (Fig. 5). If we assume that the probability that each action is taken is independent with each other, the probability that resource $q$ will be compromised by agent $c$ can be calculated with Eq (5). This computation is used to predict which resources are likely to be compromised next without receiving the next alert. From (5), we can see that Computation 3 depends on Computation 2 to calculate $\delta_t(d)$.

$$\begin{aligned} &P(\{(q,c)\} \leqslant M_t|S_{t-1}) \\ &\qquad \approx 1-(1-\pi_{t-1}(q))\cdot \prod_{q\in O(d)} (1-\delta_t(d)). \end{aligned} \tag{5}$$

**Computation 4.** Given a state $S_{t-1} = (\Pi_{t-1}, \Delta_{t-1})$ and an alert $O_t$, the probability that the action $d$ is taken is denoted as $P(D_t = d|S_{t-1}, O_t)$ and can be determined as

$$\begin{aligned} P(D_t = d|S_{t-1}, O_t) &= \frac{P(D_t = d, O_t|S_{t-1})}{P(O_t|S_{t-1})} \\ &= \frac{P(D_t = d|S_{t-1})P(O_t|D_t = d, S_{t-1})}{P(O_t|S_{t-1})}. \end{aligned} \tag{6}$$

Since $P(D_t = d|S_{t-1}) = \delta_{t-1}(d)$, and $O_t$ only depends on $D_t$, it becomes:

$$\begin{aligned} = \frac{\delta_{t-1}(d)P(O_t|D_t=d)}{P(O_t|S_{t-1})} &= \frac{\delta_{t-1}(d)\gamma_t(O_t|d)}{\sum_{d'\in D}\delta_{t-1}(d')\gamma_t(O_t|d')} \\ &= \frac{\delta_{t-1}(d)\gamma(O_t|d)}{\sum_{d'\in D}\delta_{t-1}(d')\gamma(O_t|d')}. \end{aligned} \tag{7}$$

This computation is used to estimate the likelihood that an action has just been taken by an agent given the current state and the newly received alert.

**Computation 5.** Given a state $S_{t-1} = (\Pi_{t-1}, \Delta_{t-1})$, an alert $O_t$, and action $d$ is guessed to be taken, the probability of the next state is denoted as $P(\Pi_t|S_{t-1}, O_t, D_t = d)$ and can be determined as in (8) and (9). This computation is used to estimate the likelihood that each resource will be taken by an agent after receiving a new alert.

$$P(\{(q \in O(d), c)\} \leqslant M_t|S_{t-1}, O_t, D_t = d) \approx 1 - (1 - \pi_{t-1}(q)) \times \left(1 - \frac{\delta_{t-1}(d)\gamma(O_t|d)}{\sum_{d' \in D}\delta_{t-1}(d')\gamma(O_t|d')} \prod_{q' \in I(d)} \pi_{t-1}(q')\right) \quad (8)$$

$$P(\{(q \notin O(d), c)\} \leqslant M_t|S_{t-1}, O_t, D_t = d) = \pi_{t-1}(q \notin O(d)) \quad (9)$$

We have just described five basic computations used in HCPN. The HCPN starts with an initial state determined by the initial probabilities. At each state, the HCPN estimates the likelihood that each action's preconditions have been satisfied (i.e., the action is enabled) using Computation 1. It then evaluates the probability that each action will be taken by the agent next. In other words, it predicts the agent's next possible action using Computation 2. The HCPN can also estimate which resources are likely to be taken over by the agent using Computation 3. After receiving a new alert, the HCPN reevaluates the state by first calculating the probability that each action has just been taken using Computation 4, and then updating the state (i.e., probability that each resource has been taken over by an agent) using Computation 5.

Note that these computations are specific to our HCPN model to derive the information we need. The basic ideas behind these computations, however, have been applied to many areas.

## 4. Parameter estimation and action inference

To infer agents' actions with HCPN, we need to answer two questions: how to determine the model's parameters based on labeled datasets, and how to infer an agent's most probable action sequence given the model and the alerts. These questions are usually referred to as parameter estimation problem and action inference problem. In this section, we describe the algorithms to these problems. These algorithms serve as the basis of our approach to alert correlation.

### 4.1. Parameter estimation

The purpose of parameter estimation is to find the set of model parameters that best explains the known alerts $O_1, O_2, \cdots, O_t$ and associated actions given the model structure. Specific to our model, we need to estimate the observation probabilities and transition probabilities. These probabilities are estimated from all data available (not specific to a particular agent) so that probabilities estimated from past incidences can be used to detect attacks and intruders in the future. Note that, during the alert correlation process, the alerts are separated into different streams based on the agent identity.

The observation probabilities can be directly estimated with Maximum Likelihood (ML) principle [41]. The likelihood of alerts given the observation probability $\gamma(o|d)$ is defined as

$$\begin{aligned} L(\gamma, d) &= \sum_i \ln(P(O_i|\gamma, d)) \\ &= \sum_{O_i=o} \ln\gamma + \sum_{O_i \neq o} \ln(1-\gamma) \\ &= N\ln\gamma + L\ln(1-\gamma), \end{aligned} \quad (10)$$

where $N$ is the number of instances that $O$ is issued by IDS sensors when action $d$ is taken and $L$ is the number of instances that $O$ is NOT issued by IDS sensors when action $d$ is taken. The best estimation of the observation probability is the one that maximizes the above likelihood as shown in Eq. (11):

$$\begin{aligned} \frac{\partial L(\gamma, d)}{\partial\gamma} &= \frac{N}{\gamma} - \frac{L}{1-\gamma} = 0 \\ &\Rightarrow (N+L)\gamma = N \\ &\Rightarrow \gamma = \frac{N}{(N+L)}. \end{aligned} \quad (11)$$

This estimation is intuitive. It says that the observation probability equals to the relative frequency that a specific alert is issued by the IDS sensors when the action is taken. Note that the observation probability is a property of the IDS sensors. It indicates how likely an action would cause the IDS to raise a specific alert. For example, the observation probability $p(O_i|$ normal action) is the probability that a normal action causes the IDS to raise the alert $O_i$ (note that it is different from the probability $p(O_i|$ traffic data, normal action) in which case the traffic data value is also known), and $\sum_{i=1}^{N_o} p(O_i|$ normal action) is the false positive rate. Similarly, $p($no alert$|D_i)$ is the false negative rate

given the action $D_i$. Since false positive and false negative rates are encoded in the HCPN, they are used in the inference process to correlate alerts. The observation probability is hardly affected by the intruders' behavior given that it is an intrinsic property of the IDS sensors.

The estimation of the transition probabilities is a little bit trickier. In this paper, we strived to solve this problem using an Expectation Maximum (EM) based algorithm [27]. 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 (actions in HCPN) is estimated based on the current model parameters $\lambda_k$. In the $M$-Step, Maximum Likelihood (ML) estimation is used to estimate model parameters $\lambda_{k+1}$using estimated data. This process is iterated until the parameters are fixed.

The likelihood of action taking given the transition probability $\delta = \delta(d)$ can be defined as

$$\begin{aligned} L(\delta) &= \sum_i \ln(P(D_i|\delta, E_i(d))) \\ &= \sum_{D_i=d} \ln(\delta * E_i(d)) + \sum_{D_i\neq d} \ln(1-\delta * E_i(d)), \end{aligned} \tag{12}$$

where $E_i(d)$ is the probability that action $d$ is enabled at step $i$. The transition probability is chosen to maximize the above likelihood.

$$\begin{aligned} &\frac{\partial L(\delta)}{\partial \delta} = \sum_{D_i=d} \frac{1}{\delta} - \sum_{D_i\neq d} \frac{E_i(d)}{1-\delta * E_i(d)} = 0 \\ &\Rightarrow \delta = \frac{N}{\sum_{D_i\neq d} \frac{E_i(d)}{1-\delta * E_i(d)}}, \end{aligned} \tag{13}$$

where $N$ is the number of times that action $d$ is taken (or $|d|$). Since there is no close form solution for $\delta$ in (13). The transition probability is calculated iteratively with the initial $\delta = \delta(d)$ set to the value from the last EM step as indicated in the following steps:

1. Estimate $E_i(d)$ for each action $d$ at each step $i$ using Computation 1 in Section 3 based on the current model parameters.
2. Fix $E_i(d)$ and re-estimate $\delta$ iteratively using (14).

$$\delta_{k+1} = \frac{N}{\sum_{D_i\neq d} \frac{E_i(d)}{1-\delta_k * E_i(d)}}. \tag{14}$$

3. Exit if the change of transition probability is small enough, or go to step 1 otherwise.

Note that, the transition probability is the conditional probability that a transition will be taken if the preconditions of transition are satisfied, and is relatively consistent. It is different from the probability that the transition will be taken next given the current state, which changes case by case (and can be estimated using Computation 3 in Section 3). The transition probability has also served as the key element of other probabilistic alert correlation frameworks such as the one proposed by Zhai et al. [40].

Unlike anomaly based sensors, HCPN does not adapt the transition probabilities completely unsupervised, and so it is not easy for an intruder to mislead the system. An extremely skilled intruder, however, may still behave differently from the transition probabilities learned from normal intruders and cheat the system to some extent.

### 4.2. Probability smoothing

The well-known law of large numbers assures us that the relative frequency is a good estimate of the probability (i.e. (11)) if the number of counts is very large. However, estimating the probability with (11) has two problems. First, the total number of occurrences of each action is usually not large enough in the training data, so the theory of large number does not hold. Second, some events may rarely happen, so it is seen only once or 0 times in the training data. In other words, not seeing an event in the training data does not mean the probability of it is 0. For the same reason, the probability of an event that happens only once is likely to be overestimated.

The question is what probability we should assign to events that occurs 0 or 1 time and whether the probability of an event that happens $n$ times should really be $n/m$ times as large as the probability of an event that happens $m$ times.

Our answer to the above question is based on the Good-Turing estimate [28], which can be derived by a variety of methods. The Good-Turing estimate states that the probability assigned to all the unobserved events is equal to the total probability that would have been assigned to singleton events by a relative frequency formula, i.e.,

$$\tilde{p}_0 n_0 = \frac{n_1}{N}, \tag{15}$$

where $\tilde{p}_0$ is the probability assigned to unobserved events, $n_0$ is the total number of events that do not happen, $n_1$ is the total number of singleton

events, and $N$ is the total number of event occurrences. The smoothed estimation of the probabilities is thus:

$$\tilde{p}_j = \begin{cases} \frac{n_{j+1}}{n_j}\frac{j+1}{N} & j = 0, 1, \ldots, M, \\ \alpha\frac{j}{N} & j > M, \end{cases} \tag{16}$$

where $\tilde{p}_j$ and $n_j$ are the estimated probability and the number of events observed $j$ times respectively. The normalization factor $\alpha$ is equal to

$$\alpha = \frac{\sum_{j>M+1} jn_j}{\sum_{j>M} jn_j} = \frac{N - \sum_{i=1}^{M+1} in_i}{N - \sum_{i=1}^{M} in_i}, \tag{17}$$

$M$ is usually a small to moderate integer. In our case, we choose $M$ to be 4.

Probability smoothing is very important to reliably estimating the probabilities, especially when the training data size is small. Without proper probability smoothing, some probabilities will be 0 and may lead to bad inference result.

### 4.3. Action inference

The purpose of action inference is to find the most probable action sequences taken by the agent given the alerts. In other words, the inference problem, or the correlation process in our model, can be stated as follows: given alerts $O = O_1, O_2, \cdots, O_t$, and the model parameter $\lambda$, which action sequence $D = D_1, D_2, \cdots, D_t$ is most likely to have led to the sequence of alert $O$ from $\lambda$? That is to say, we want to optimize the criteria indicated in (18):

$$\arg\max_D [P(D|O, \lambda)] = \arg\max_D \left[\frac{P(O|D, \lambda)P(D|\lambda)}{P(O|\lambda)}\right]. \tag{18}$$

Since the term $P(O|\lambda)$ is not related to $D$, we can discard it when selecting the best path, i.e., we need to only optimize:

$$\arg\max_D [P(O|D, \lambda)P(D|\lambda)] = \arg\max_D P(O, D|\lambda). \tag{19}$$

We solve this problem with dynamic programming by defining $\omega_t(j)$ as the maximum score of a length $t$ state sequence ending in action $j$ and producing the first $t$ alerts from $O$, as shown in (20):

$$\omega_t(j) = \max_{D_1 \cdots D_{t-1}} P(O_1, \ldots, O_t, D_1, \ldots, D_{t-1}, D_t = j|\lambda). \tag{20}$$

Thus, the overall maximum probability is

$$\begin{aligned} \max_j \omega_T(j) &= \max_j \max_{D_1 \cdots D_{T-1}} \\ &\quad (O_1, \ldots, O_T, D_1, \ldots, D_{T-1}, D_T = j|\lambda) \\ &= \max_D P(O, D|\lambda) \end{aligned} \tag{21}$$

and

$$\begin{aligned} \omega_t(j) &= \max_{D_1 \cdots D_{t-1}} P(O_1, \ldots, O_t, D_1, \ldots, D_{t-1}, D_t = j|\lambda) \\ &= \max_{D_1 \cdots D_{t-1}} P(O_1, \ldots O_{t-1}, D_1, \ldots, D_{t-1}|\lambda) \\ &\quad P(D_t = j|S_{t-1}, \lambda)P(O_t|D_t = j, \lambda) \\ &= \max_{D_{t-1}=i} [\max_{D_1 \cdots D_{t-2}} P(O_1, \ldots, O_{t-1}, D_1, \ldots, D_{t-1} = i|\lambda) \\ &\quad P(D_t = j|S_{t-1}, \lambda)P(O_t|D_t = j, \lambda)] \\ &\approx \max_{D_{t-1}=i} [\omega_{t-1}(i)\delta^i_{t-1}(j)\gamma(O_t|j)], \end{aligned}$$

where $\delta^i_{t-1}(j) \approx P(D_t = j|S'_{t-1}, \lambda)$ and $S'_{t-1}$ is the state corresponding to $\omega_{t-1}(i)$. From this inference process, we can see why HCPN has potential to reduce false positive and false negative rates: given the alert sequence, HCPN can infer the most possible action sequence using the precondition–postcondition relationship, the observation probability, and the transition probability. For example, if the best action sequence deduced by HCPN for the alert sequence:

alert_attack1, alert_attack1, alert_attack2, alert_attack5, alert_attack3, alert_attack5 is

attack1, attack1, attack2, normal, attack3, normal.

HCPN would indicate that attack1, attack2, and attack3 have all been taken by the agent. However alert_attack5s are considered false positive alerts since they are associated with the normal action.

In the inference algorithm, we calculate $\omega_t(j)$ for each alert $O_t$ and each action $j$. Note that we need to select the maximum value from all $N_d$ (the number of actions) values of $[\omega_{t-1}(i)\delta^i_{t-1}(j)\gamma(O_t|j)]$ when calculating $\omega_t(j)$. In other words, the time complexity of the action inference is $O(N_d^2 \cdot N_o)$, where $N_o$ is the total number of alerts issued by the IDS sensors. The efficiency of the algorithm can be further improved considering that most of the observation probabilities $\gamma(O_t|j)$ are close to 0, in which case we can simply assign a fixed floor probability for the related $\omega_t(j)$. Since each action is usually misrecognized as less than 10 (independent of $N_d$) other actions (i.e., the number of significant $\gamma(O_t|j)$ for each $j$ is less than 10), the time complexity of the

algorithm after the simplification can be reduced to approximately $O(N_d \cdot N_o)$, which is at least as good as any other approaches surveyed in Section 7.1. Note that the time complexity is not related to the number of attack steps.

## 5. Alert confidence fusion

The false positive and false negative rates can be further reduced by alert confidence fusion technologies (confidence fusion component in Fig. 1). The way to combining the confidence scores, however, is tricky for several reasons. First, alerts provided by different sensors should not be trusted equally. For example, information provided by remote sensors and analyzers is considered less trustworthy than that provided by local sensors and analyzers as noted earlier [29]. Second, sensors (and analyzers) installed at different locations may have different detection capabilities even if the sensors are of the same type since the raw events captured by these sensors are different. For example, a sensor installed in a LAN usually receives less external traffic but more internal traffic than a sensor installed at the DMZ zone. Third, different kinds of sensors and analyzers may detect the same type of attack with a different level of accuracy. For example, a sensor may detect a misuse activities such as Distributed Denial of Service (DDoS) attacks with 90% accuracy but detect escalation of privilege, such as (Local-to-Root) L2R attacks with only 50% accuracy. Another, perhaps host-based, sensor may not detect DDoS at all, but be quite accurate in detecting escalation of privilege.

To take into consideration all these factors, we have extended the Dempster–Shafer's (D–S) Theory of Confidence [30] into Exponentially Weighted Dempster–Shafer's (EWDS) Theory of Confidence [31] to fuse the confidence scores from different alert analyzers. In the basic D–S theory, the confidence combination rule is

$$m_{12}(H) = \frac{\sum_{B\cap C=H} m_1(B)m_2(C)}{\sum_{B\cap C\neq\phi} m_1(B)m_2(C)}, \tag{22}$$

where $B$, $C$, and $H$ are hypotheses. $m_i(H)$ is the confidence that observer $O_i$ believes that hypothesis $H$ is true and $m_{12}(H)$ is the combined confidence.

The EWDS theory incorporates a weighted view of evidence from different sources with the following modified combining rule:

$$m_{12}(H) = \frac{\sum_{B\cap C=H} [m_1(B)]^{w_1}[m_2(C)]^{w_2}}{\sum_{B\cap C\neq\phi} [m_1(B)]^{w_1}[m_2(C)]^{w_2}}, \tag{23}$$

where $w_i$ is the weight for observer $O_i$. When $w_1 = w_2 = 1$, (23) reduces to the basic D–S combining rule.

Since we are interested in whether an alert $A_k$ is true (a true positive) or false (a false positive), the *frame of discernment* [30] is $\Theta_k = \{A_k, \neg A_k\}$. The possible hypotheses are

$$2^{\Theta_k} = \{\phi, \{A_k\}, \{\neg A_k\}, \{A_k, \neg A_k\}\}. \tag{24}$$

It is clear that

$$m_i(\{A_k, \neg A_k\}) = 0, \tag{25}$$

$$m_i(\{\neg A_k\}) = 1 - m_i(\{A_k\}). \tag{26}$$

The combining rule then becomes:

$$m_{12}(\{A_k\}) = \frac{P(\{A_k\})}{P(\{A_k\}) + P(\{\neg A_k\})}, \tag{27}$$

$$m_{12}(\{\neg A_k\}) = \frac{P(\{\neg A_k\})}{P(\{A_k\}) + P(\{\neg A_k\})}, \tag{28}$$

where

$$P(X) = [m_1(X)]^{w_{1,k}}[m_2(X)]^{w_{2,k}} \tag{29}$$

In our system, we use the transition probability estimated for the action $A_k$ by the $i$th HCPN alert correlator as the confidence values $m_i(\{A_k\})$.

The weights can be estimated by minimizing the Mean Square Error (MSE) defined as

$$\mathrm{MSE} = \frac{\sum_{j=1}^{N} (m_{c,j}(\{A_k\}) - r_j)^2}{N}, \tag{30}$$

where $N$ is the total number of samples in the training set, $m_{c,j}$ is the $j$th combined observation score, and $r_j$ is the real (or true) value of the $j$th observation.

$$r_j = \begin{cases} 1 & \text{if } A_k \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

We used the standard gradient descending algorithm to search for the best weights, i.e.,

$$w_{i,k}^{t+1} = w_{i,k}^{t} - \lambda \frac{\partial \mathrm{MSE}}{\partial w_{i,k}}, \tag{32}$$

where $\lambda$ is the step size. Note that for each alert type $A_k$ and each observer $i$, there is a weight $w_{i,k}$. In other words, if observer one is good at detecting alert $A_k$ but is not good at detecting alert $A_j$, $w_{1,k}$ is large but $w_{1,j}$ is small.

There are two practical considerations in our implementation.

First, when $m_i(H)$ is close to 0, $\log(m_i(H))$ might be very small. This may result in computational underflow. To prevent underflow from happening, all the probabilities in the system have a floor value of 0.01. If the confidence of a hypothesis is less than 0.01, we set it to be 0.01.

Second, an HCPN alert correlator reports to the alert confidence fusion component only if the perceived state of a resource has been changed, so alert correlators send the state change information asynchronously. To deal with this problem, we store every correlator's last reported evidence. When a new report regarding alert $A_k$ comes in, we update the corresponding evidence and re-estimate the combined confidence for the alert $A_k$. We thus gauge confidence based on evidence "in hand," and increase/reduce confidence when new information arrives. Note that, the fusion process produces derived information from multiple sources. However, it does not damage information stored at each source locations.

## 6. Experiments and evaluation

We have developed an off-line alert correlation system based on our HCPN framework and performed two sets of experiments: one on the DARPA 2000 DDoS evaluation datasets [32], and one on the attack scenarios from the Grand Challenge Problem (GCP) provided by DARPA's Cyber Panel Program [21]. The HCPN model used in our experiments consists of 23 places (resources), 34 transitions (33 malicious actions + 1 normal action), and 33 alerts (corresponding to malicious actions) for both experiments.

### 6.1. Quality of alerts

Quality of alerts is usually measured with the total number of alerts (TNA), the detection rate (DR), the false positive rate (FPR), the false negative rate (FNR), and the repeated true alert rate (RTR):

$$\mathrm{DR} = \frac{\text{Number of True Attacks Reported}}{\text{Number of Total Observable Attacks}}, \tag{33}$$

$$\mathrm{FPR} = 1 - \frac{\text{Number of True Alerts Reported}}{\text{Number of Alerts Reported}}, \tag{34}$$

$$\mathrm{FNR} = 1 - DR, \tag{35}$$

$$\mathrm{RTR} = \frac{\text{Number of True Alerts} - \text{Number of True Attacks}}{\text{Number of True Attacks}}. \tag{36}$$

The reason above measures are reasonable is that for each true intrusive action taken by an agent, we want to produce only one alert. However, though one cannot reduce both the false positives and the false negatives at the same time, these qualities are not necessarily functions of one another either. It is thus difficult to compare different approaches based on one measure alone. In this paper, we define quality of alerts as

**Definition 2.** The Quality of Alerts (QoA) is defined as

$$\mathrm{QoA} = 100\% - \frac{\mathrm{fp} + \mathrm{fn} + \mathrm{rt}}{\mathrm{ta}}, \tag{37}$$

where fp is the number of false positives, fn is the number of false negatives, rt is the number of repeated true alerts, and ta is the number of intrusive actions (or expected alerts). Note that the QoA value can be negative (which means the number of incorrect and/or redundant alerts exceeds the number of true attacks) and the best QoA value is 100%. QoA is a combined score to measure the alert quality. It is borrowed from the speech recognition community where a speech recognizer may make insertion errors (similar to false positives and repeated true alerts in IDS), and deletion errors (similar to false negatives in IDS).

Sometimes, the cost incurred by different types of alerts is different. For example, the cost incurred by the repeated true alerts is often less than that brought by the false positives, which, in turn, is usually less than that from the false negatives. To incorporate the cost information, we introduce the concept of Weighted Quality of Alerts (WQoA):

**Definition 3.** The Weighted Quality of Alerts (WQoA) is defined as

$$\mathrm{WQoA} = 100\% - \frac{w_{\mathrm{fp}} \cdot \mathrm{fp} + w_{\mathrm{fn}} \cdot \mathrm{fn} + w_{\mathrm{rt}} \cdot \mathrm{rt}}{\mathrm{ta}}, \tag{38}$$

where $w_{\mathrm{fp}} \geqslant 0$ is the weight for the false positives, $w_{\mathrm{fn}} \geqslant 0$ is the weight for the false negatives, and $w_{\mathrm{rt}} \geqslant 0$ is the weight for the repeated true alerts. WQoA reduces to QoA when $w_{\mathrm{fp}} = w_{\mathrm{fn}} = w_{\mathrm{rt}} = 1$.

Note that the weights in the WQoA are usually different in different systems. For this reason, we use QoA instead of WQoA in our experiments.

### 6.2. Experiments on the DARPA DDoS evaluation dataset

Our first set of experiments was conducted on the DARPA 2000 intrusion detection evaluation data-

sets [32]. The dataset includes the network traffic data collected from both the DMZ and the inside part of the evaluation network. In the dataset, an intruder probes, breaks-in, installs the DDoS daemon, and launches a DDoS attack against an offsite server (Table 1). In this set of experiments, alerts are generated using RealSecure Network Sensor 6.0 with the maximum coverage policy which forced the Network Sensor to save all the reported alerts. We choose data produced by RealSecure Network Sensors because attack signatures used in RealSecure Network Sensor 6.0 are well documented, and we could draw upon Ning et al. [19] for the precondition–postcondition rule set.

The alerts are first separated into two parts: those alerts generated at the DMZ and those generated inside the network. Each part is then separated into the training set and the testing set. The training set was used to estimate the HCPN model parameters, and the testing set was used to evaluate the effectiveness of the model. Each set has about 900 alerts. The training takes less than 5 s and the inference takes about 1 s (without speed up approximation) on a 2.8 GHz PC.

As mentioned in Section 2, our HCPN system outputs resources compromised. For example, it reports whether a daemon has been installed instead of the intruder's action that leads to the installation of the daemon, although the action itself is available to the administrators and/or reactors if needed. Tables 2 and 3 list the correlation results for the inside network traffic and DMZ network traffic, respectively.

Table 4 shows the actions detected by the HCPN-based alert analyzer along the time. After the action is inferred to have happened, the alert analyzer automatically estimates the new perceived state of the system, and infers the next goal of the intruder. In this experiment, our HCPN-based alert analyzer predicted intruder's next goals with 100% accuracy. This indicates that our HCPN-based alert analyzer can not only understand an intruder's progress but also predict an intruder's next possible action and goal. If the whole attack scenario has been completed, HCPN potentially uncovers the intruder's intrusion strategy. Note that the accuracy of the action and goal prediction is directly related to the perplexity of the HCPN model and the knowledge about the actions and their preconditions and postconditions. When many actions might be enabled at the same

Table 1
Steps of the DDoS attack

| Step # | Attack |
|---|---|
| 1 | IPsweep for live host IPs from a remote site |
| 2 | Probe with SadmindPing the identified live IPs for sadmind daemon running on Solaris hosts |
| 3 | Compromise target hosts via the sadmind vulnerability (SadmindBOF) |
| 4 | Install the Trojan mstream DDoS software on compromised hosts |
| 5 | Register DDoS Trojan to the master computer |
| 6 | Launch the DDoS from compromised hosts against target |

Table 2
Correlation result for the inside network traffic

| Host | State | 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 3
Correlation result for the DMZ traffic

| Host | State | Probability |
|---|---|---|
| 172.016.112.010 | SystemCompromised | 1.00 |
| | VulnerableSadmind | 0.66 |
| | DaemonInstalled | 0.55 |
| 172.016.112.050 | SystemCompromised | 1.00 |
| | VulnerableSadmind | 0.66 |
| | DaemonInstalled | 0.55 |
| 172.016.114.010 | SystemCompromised | 1.00 |
| | VulnerableSadmind | 0.66 |
| 172.016.114.020 | SystemCompromised | 1.00 |
| | VulnerableSadmind | 0.66 |
| 172.016.114.030 | SystemCompromised | 1.00 |
| | VulnerableSadmind | 0.66 |
| 172.016.115.020 | SystemCompromised | 1.00 |
| | VulnerableSadmind | 0.66 |
| | DaemonInstalled | 0.80 |

Table 4
Actions detected along the time from the inside network traffic

| Step | Host | Action | State | Next goal |
|---|---|---|---|---|
| 1 | 172.016.115.020 | Sadmind_Ping | VulnerableSadmind | SystemCompromised |
| 2 | 172.016.112.010 | Sadmind_Ping | VulnerableSadmind | SystemCompromised |
| 3 | 172.016.112.050 | Sadmind_Ping | VulnerableSadmind | SystemCompromised |
| 4 | 172.016.115.020 | Sadmind_BOF | SystemCompromised | DaemonInstalled |
| 5 | 172.016.112.010 | Sadmind_BOF | SystemCompromised | DaemonInstalled |
| 6 | 172.016.112.050 | Sadmind_BOF | SystemCompromised | DaemonInstalled |
| 7 | 172.016.115.020 | Rsh | DaemonInstalled | ReadyForDDoS |
| 8 | 172.016.112.050 | Rsh | DaemonInstalled | ReadyForDDoS |
| 9 | 172.016.112.010 | Rsh | DaemonInstalled | ReadyForDDoS |
| 10 | 172.016.112.050 | Mstream_Zombie | ReadyForDDOS | LaunchDDoS |
| 11 | 172.016.112.010 | Mstream_Zombie | ReadyForDDOS | LaunchDDoS |

The step number indicates the order in which the malicious action is detected by our HCPN alert analyzer. The state column indicates the state change after the action is detected. The next goal is the predicted intruder's next goal after the detected action.

time for a given agent (as in a very big IDS system), the action prediction accuracy will likely decline.

Table 5 illustrates the detection rate, false positive rate, and QoA for RealSecure Network Sensor 6.0 and our HCPN-based system. In the table, Raw and Raw2 are results from RealSecure sensors. The difference between Raw and Raw2 is the way the number of attacks is calculated. Each attempt of the same attack from the same agent is considered as a different attack in Raw (which is used by Ning et al. [19]), while all attempts of the same attack is considered as one attack in Raw2. In Raw2, an attack is considered detected as long as one of the alerts corresponding to the attack has been reported. Note that Raw2 provides a higher detection rate but lower QoA comparing to Raw due to the decrease of the number of true attacks. DMZ-uniform and Inside-uniform are results of the HCPN model without training (i.e., the transition probabilities are set to 1, and the observation probabilities are set according to Eq. (1)). This is corresponding to the simple precondition–postcondition based correlation approaches. DMZ-HCPN and inside-HCPN are results of HCPN after the parameter estimation.

From the table, we can see that HCPN-based alert analyzer can greatly reduce the number of alerts presented to the administrators and active reactors. Without using our alert analyzer, about 900 alerts are reported at both the DMZ and the inside network sites. With our HCPN-based alert analyzer, the total number of alerts reported is reduced to less than 20 at each site. Note that, our system can not only reduce the total number of alerts but also improve the quality of the alerts. With our HCPN-based approach, the false positive rate is decreased from 94% to 20% at the DMZ site and from 95% to 0% at the inside network site. From the table, we can also see that comparing with the default parameters, both the detection rate and the false positive rate are improved after the training. In other words, HCPN can improve the QoA even without training data. When training data are available, however, HCPN can utilize the

Table 5
Detection and false alert rates for RealSecure Network Sensor 6.0 and the HCPN-based system

| Setting | NOA | NA | NTAD | DR (%) | NRA | FPR (%) | RT | RAR (%) | QoA (%) |
|---|---|---|---|---|---|---|---|---|---|
| DMZ-Raw | 89 | 891 | 51 | 57.30 | 57 | 93.60 | ***6*** | ***11.8*** | ***−887*** |
| DMZ-Raw2 | 12 | 891 | 12 | 100 | 57 | 93.60 | ***45*** | ***375*** | ***−7225*** |
| DMZ-uniform | 12 | 42 | 11 | 91.7 | 11 | 73.8 | ***0*** | ***0*** | ***−167*** |
| DMZ-HCPN | 12 | 15 | 12 | 100.00 | 12 | 20 | ***0*** | ***0*** | ***75.0*** |
| Inside-Raw | 60 | 922 | 37 | 61.67 | 44 | 95.23 | ***7*** | ***18.9*** | ***−1413*** |
| Inside-Raw2 | 13 | 922 | 12 | 92.31 | 44 | 95.23 | ***32*** | ***267*** | ***−7483*** |
| Inside-uniform | 13 | 30 | 11 | 84.6 | 11 | 63.3 | ***0*** | ***0*** | ***−61.5*** |
| Inside-HCPN | 13 | 12 | 12 | 92.31 | 12 | 0 | ***0*** | ***0*** | ***92.3*** |

NOA = Number of observable attacks, NA = number of alerts, NTAD = number of true attacks detected, DR = detect rate, NRA = number of real alerts, FPR = false positive rate, RT = repeated true alerts, RTT = repeated true alert rate.

information to further improve the correlation results. Note that while HCPN with the default parameters (as when no training data is available) can reduce the false positives, it may decrease the detection rate at the same time as shown in Table 5.

When we use QoA as the performance measurement, the improvement is also very significant. In the DMZ zone, the QoA is −887% for the Raw setting, −7225% for the Raw2 setting, −167% for the uniform setting, and 75% for the HCPN setting. In the inside network, the QoA is −1413% for the Raw setting, −7483% for the Raw2 setting, −62% for the uniform setting, and 92% for the HCPN setting. Under both conditions, HCPN performs much better than the system without alert correlation and the system without separating alerts and actions. Please note, however, the DARPA evaluation dataset has some limitations as indicated by McHugh [42]. The improvements in QoA for other datasets may not be as significant as what we have got for the DARPA dataset.

Our HCPN model after training is not perfect. As indicated in Table 5, there are still false negatives and false positives after using HCPN.

An example of false negative can be observed in Table 2, where we can see that the intruder has installed daemons on hosts 172.016.112.010, 172.016.112.050, and 172.016.115.020, and is ready to launch the DDoS attack. Note, however, our system does not detect that the intruder is ready to launch DDoS attack on host 172.016.115.020.

An example of false positive is shown in Table 3. According to the description of the dataset, the intruder tried Sadmind_BOF 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_BOF attacks had failed. However, since our rules indicate that the postcondition of the Sadmind_BOF attack is SystemCompromised, our HCPN would report that these hosts are compromised as indicated in Table 3.

In these experiments, we used different default initial probabilities from 0.01 to 0.05 and get the same results.

### 6.3. Test on attack scenarios from the GCP

Our second set of experiments was conducted on attack scenarios from the GCP. Note that the access to the original GCP dataset is very restrictive. However, we have managed to generate the scenarios using our own simulation tool. To generate the alerts, we used a program to simulate the behavior of an intruder and another program to simulate the noise. The simulated intruder picks the next step based on the precondition–postcondition relationship and predefined probabilities. The noise generator randomly generates noise traffic at random time intervals. The traffic generated from both programs is merged together based on the time stamp. Using the simulation tool, we have generated both the training set and the testing set. The attack steps of the scenario used in this experiment are summarized in Table 6.

Table 7 shows the quality of alerts with and without HCPN. In the DMZ zone, the QoA is −954% for the Raw setting, −6829% for the Raw2 setting, −71% for the uniform setting, and 93% for the HCPN setting. In the inside network, the QoA is −1224% for the Raw setting, −9686% for the Raw2 setting, −29% for the uniform setting, and 93% for the HCPN setting. It is obvious that our HCPN model can greatly improve the QoA by reducing the false positive rate and increase the detection rate for this task.

A phenomena not observed in the DARPA DDoS experiment but happened in this experiment is the potential of HCPN to recover from the missing alerts. The attacks of IllegalFileAccess are missed in the raw alerts at DMZ. However, with the implicit output arcs in the HCPN model, HCPN correctly deduced that the files have been illegally accessed.

The inference time for the DMZ traffic is about 2 s and the inside network traffic is about 3 s on a 2.8 GHz PC if the speed up approximation is not used. Note that the inference time is $O(N_d^2 \cdot N_o)$ if speed up approximation is not used as indicated in Section 4.3. In other words, the inference time is proportional to the total number of alerts reported by the sensors, and the square of the total number of possible actions encoded into the HCPN. It is not related to the number of atomic attacks taken by an agent. We have generated variable length of alerts with variable length of (artificial) attack steps as test data, and our experiments shown in Table 8 confirmed our estimation.

### 6.4. Alert confidence fusion

Ideally, the confidence score should be equal to the probability, i.e., reflect the likelihood that the alert is true. For this reason, the effectiveness of the confidence fusion can be measured with the cross entropy between the confidence score and

Table 6
Attack steps in the attack scenario from GCP

| Step | Host | Action | Description |
|---|---|---|---|
| 1 | 129.0.1.8 | Port_Scan | Scan the ports to detect FTP service |
| 2 | 129.0.1.7 | Port_Scan | Scan on a different host |
| 3 | 129.0.1.13 | Port_Scan | Scan on a different host |
| 4 | 129.0.1.11 | Port_Scan | Scan on a different host |
| 5 | 129.0.1.12 | Port_Scan | Scan on a different host |
| 6 | 129.0.1.9 | Port_Scan | Scan on a different host |
| 7 | 129.0.1.13 | FTP_Globbing_Attack | Get root access to the host using the FTP Globbing buffer overflow attack |
| 8 | 129.0.1.13 | AgentInstall | Download malicious code to the host |
| 9 | 129.0.1.13 | IllegalFileAccess | Illegally collects information from the host |
| 10 | 129.0.1.13 | Loki | Upload sensitive information to the external server |
| 11 | 129.0.1.11 | FTP_Globbing_Attack | Attack a different host |
| 12 | 129.0.1.13 | AgentInstall | Attack a different host |
| 13 | 129.0.1.11 | IllegalFileAccess | Attack a different host |
| 14 | 129.0.1.11 | Loki | Attack a different host |

Table 7
Quality of alerts with and without HCPN

| Setting | NOA | NA | NTAD | DR (%) | NRA | FPR (%) | RT | RTR (%) | QoA (%) |
|---|---|---|---|---|---|---|---|---|---|
| DMZ-Raw | 87 | 980 | 58 | 66.7 | 63 | 93.6 | ***5*** | ***8.6*** | ***−954*** |
| DMZ-Raw2 | 14 | 980 | 12 | 85.7 | 63 | 93.6 | ***51*** | ***425*** | ***−6829*** |
| DMZ-uniform | 14 | 34 | 12 | 85.7 | 12 | 64.7 | ***0*** | ***0*** | ***−71.4*** |
| DMZ-HCPN | 14 | 15 | 14 | 100 | 14 | 6.7 | ***0*** | ***0*** | ***93.3*** |
| Inside-Raw | 98 | 1384 | 77 | 78.6 | 86 | 93.8 | ***9*** | ***11.7*** | ***−1224*** |
| Inside-Raw2 | 14 | 32 | 14 | 100 | 86 | 56.3 | ***72*** | ***514*** | ***−9686*** |
| Inside-uniform | 14 | 32 | 14 | 100 | 14 | 56.3 | ***0*** | ***0*** | ***−28.6*** |
| Inside-HCPN | 14 | 15 | 14 | 100 | 14 | 6.7 | ***0*** | ***0*** | ***93.3*** |

NOA = number of observable attacks, NA = number of alerts, NTAD = number of true attacks detected, DR = detect rate, NRA = number of real alerts, FPR = false positive rate, RT = repeated true alerts, RTT = repeated true alert rate.

Table 8
Inference time in HCPN

| # of Alerts | 1054 | 1384 | 1868 | 2568 | 2568 | 2568 | 2568 | 2568 |
|---|---|---|---|---|---|---|---|---|
| # of Attack steps | 14 | 14 | 14 | 14 | 20 | 26 | 30 | 35 |
| Time (s) | 1.8 | 3.1 | 4.0 | 7.4 | 8.3 | 7.1 | 7.3 | 8.1 |

the true distribution of the attacks. However, since our ultimate goal is to reduce the false positive and false negative rates, we evaluate the effectiveness of the confidence fusion with QoA in this paper.

Table 9 compares the result with and without the alert confidence fusion for the DARPA DDoS evaluation dataset. Note that the detection and false alert rate reported in Table 9 is different from that in Table 5 because in Table 5 we focus on the performance of individual analyzers, so the detectable alerts in Table 5 are specific to the information available to that analyzer. In Table 9, however, we focus on the performance of the whole fused system,

Table 9
Detection and false positive rates of our HCPN-based system with and without alert confidence fusion

| Setting | NOA | NA | NTAD | DR (%) | NRA | FPR (%) | QoA (%) |
|---|---|---|---|---|---|---|---|
| DMZ | 16 | 15 | 12 | 75.0 | 12 | 20 | 56.3 |
| Inside | 16 | 12 | 12 | 75.0 | 12 | 0 | 75.0 |
| Confidence fusion | 16 | 15 | 15 | 93.8 | 15 | 0 | 93.8 |

NOA = number of observable attacks, NA = number of alerts, NTAD = number of true attacks detected, DR = detect rate, NRA = number of real alerts, FPR = false positive rate.

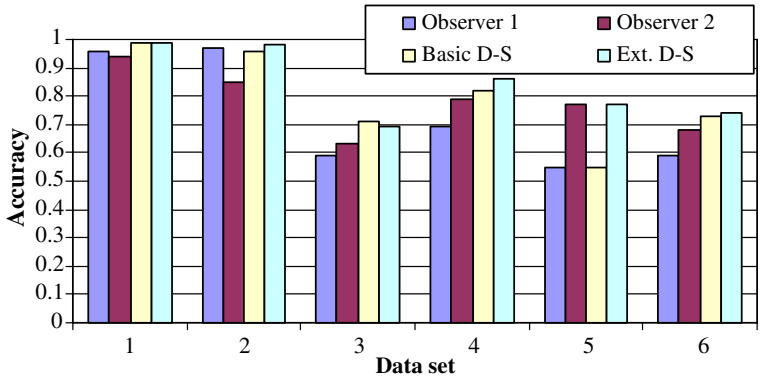


Fig. 6. Comparison of classification accuracy based on confidence scores from observer 1, observer 2, the combined confidence with the basic D–S theory, and the combined confidence with our extended D–S theory.

so the number of detectable alerts is subject to information available to the whole system. Table 9 shows that with the alert confidence fusion technologies, the overall detection rate can be increased from 75% at each site to 94%, while keeping the false positive rate low on the DARPA DDoS evaluation dataset. The QoA is increased from 56% using the DMZ zone alerts alone and 75% using the inside network alerts alone, to 94% using the fused confidence scores.

We have run the confidence fusion experiments on other simulation data. In these experiments, confidence scores from alert correlation components are simulated and separated into the training and the test sets. Fig. 6 compares the classification accuracy (whether an alert should be reported) on confidence scores from observer 1, observer 2, the combined confidence with the basic D–S theory, and the combined confidence with our extended D–S theory. It is not difficult to see that in most cases, confidence fusion with our extended D–S theory outperforms the basic D–S theory as well as observations from individual observations.

## 7. Related work

### 7.1. Alert correlation as a filter and an inference problem

Alert correlation in IDS has been an active research area. Most existing alert correlation approaches consider the alert correlation problem as the problem of extracting the true alerts (i.e., filtering out the false positive alerts) directly from the alerts generated by the IDS sensors based on the relationships (e.g., similarities, sequential relationships, etc.) among alerts. These approaches are often implemented in two steps: alert aggregation (also known as fusion or clustering) and intention recognition (also known as plan recognition). Fig. 7 shows the full correlation process based on alert aggregation and plan recognition.

Conceptually, alert aggregation is simple: by aggregating alerts into meta-alerts (or hyper-alerts) based on feature similarities, researchers have found that they can reduce the total number of alerts presenting to the administrators and/or AR [1,4–

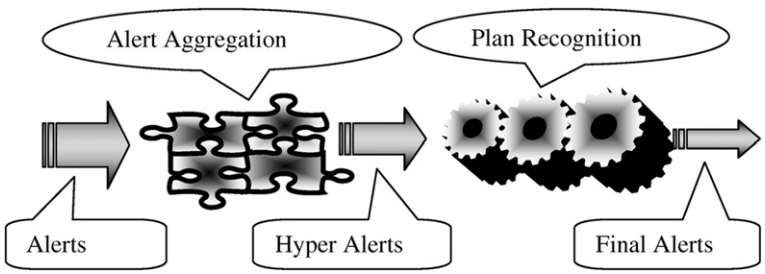


Fig. 7. A typical alert correlation process. Alerts are first grouped into hyper-alerts through alert aggregation component. Hyper-alerts are then processed through plan recognition component which utilizes the precondition–postcondition information of the attack actions. The number of alerts decreases after each step.

7,9,11,12]. The aggregation process usually involves merging features of the two alerts. For example, alerts from the same sensor and belonging to the same attack (identified by the same source and target IP address) are considered similar alerts [11]. In alert aggregation, 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 [6,9]. The main purpose of the alert aggregation is to reduce the number of alerts to be provided to the administrators. In other words, the alert aggregation approach can improve the quality of the alerts (defined in Section 5) by reducing the number of repeated alerts. Alert aggregation itself, does not directly address false negatives or false positives. False negatives occur when an attack is "missed", so aggregating alerts alone will not be of assistance. False positives may be reduced somewhat – for instance, if a set of alerts tied to acceptable behaviors is flagged as misuse, aggregation will produce only one meta-alert (which will be a false positive) rather than the full collection of raw false positive alerts. However, this does not truly eliminate the fact that something is raising a false positive, only reducing the number of times it is reported.

To filter out false positives, researchers have proposed using plan recognition [8,13–15,17,18,20,24] after the alert aggregation step. Plan recognition utilizes the precondition–postcondition relationships between activities. The concept is that in a given attack scenario, system misusers perform a sequence of steps to violate system security policies, with earlier steps preparing for the later ones. An alert is likely to be true if the precondition steps and/or postcondition steps are also reported in the alerts. By keeping only those alerts, we can potentially remove part of the false positives, and hence improve the quality of the alerts. Plan recognition seeks to recognize an intruder'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, multistage scenarios. Detecting an intruder's plan at an early stage would make it easier to prevent the intruder 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.

In this paper, we have taken a different view and considered the problem of alert correlation as the problem of INFERRING an agent's actions (which is not directly observable) from the alerts, instead of filtering out false alerts from all alerts reported by sensors. Note that this is a completely different view comparing to the filter view of the problem. Here, we consider and treat alert and action as two different entities. The agent's actions are unknown. However, their actions may be partially observed by IDS and reported as alerts that contain incomplete and false information. The task of alert analysis is not to find those "good" alerts but to infer the most possible actions taken by the agent given the observations. During the inference process, information such as the false positive rate, false negative rate, and the number of same alert issued can all be used and integrated into the process as discussed in the previous sections. When no training data are available, HCPN behaves the same as the filter-based approaches that rely on only the precondition–postcondition relationship. When training data are available, the model parameters can be tuned and better correlation results can be achieved.

### 7.2. Probabilistic framework for IDS

Zhai et al. [40] proposed a Bayesian network based probabilistic framework to integrate and reason about complementary intrusion evidence such as alerts generated by IDS and reports by system monitoring or vulnerability scanning tools. In their framework, the basic conditional probability table is the same as the transition probability used in HCPN. The difference between their approach and HCPN is that HCPN also uses the observation probabilities in the inference process, while their approach integrates additional information from system monitoring or vulnerability scanning tools to improve the alert quality.

### 7.3. Colored Petri-Net

A CPN [22,33,34] model of a system provides an explicit description of both the states and actions of the system. CPN has strong modeling ability to model systems with concurrent and distributed events and has been widely used in modeling the Discrete Event Dynamic System.

Petri-Net has also been introduced to model the intruder's misuse behaviors [25,35,36]. Kumar and Spafford [35,36] proposed to use CPN as the model to match misuse patterns. In their model, "Guards define the context in which signatures are matched. The notion of start and final states, and paths between them define the set of event sequences matched by the net. Partial order matching can also be specified in this model. The main benefits of the model are its generality, portability and flexibility." Ho et al. [25] suggested combining partial order planning and executable Petri-Net to detect misuse intrusions. The main benefit of their approach is to allow unordered event to be encoded in the intrusion signature. They argue that traditional state transition analysis which is based on strictly ordered and non-overlapping sequence of events is not flexible to model the real misuse scenario. For this reason, they "use a partial ordering of events. A partial order of events specifies that some events are ordered with respect to each other while others are unordered. A partial order state transition analysis allows more than one sequence of events in the state transition diagram." Thus, using Petri-Net to model the partial order relationship between steps, they can represent the set of all intrusion attempts with one diagram.

The advantage of HCPN as compared to CPN is the ability to provide probabilities, and to model transitions and the observations separately. More specifically, HCPN differs from CPN in the following two ways.

First, in CPN, each place either contains exactly $N$ tokens or not. For example, if the statement that "*place A contains one token*" is true, the statement that "*place A contains two tokens*" is false, and vice versa. In HCPN, however, each token element is associated with a probability (or confidence). For example, a place may have a probability of 0.4 to hold one token, a probability of 0.2 to hold two tokens, and a probability of 0.4 to contain no token. Thus, HCPN is a statistical model that represents the probability of the system being in different states as opposed to a representational model that maps precisely to a particular state of affairs.

Second, no concept of observation exists in CPN. In other words, observations and transitions are considered the same in CPN. In HCPN, however, observations are explicitly separated from transitions. Each transition can be observed only as a specific observation with some probability. The transitions (actions) are not directly observable and can be inferred only through the observations (alerts). This property makes HCPN especially suitable for alert correlation task since the true action might be observed as different alerts by IDS as we have indicated in Section 2.2.

## 8. Discussions and conclusion

In this paper, we compared two different ways of looking at the alert correlation problem. We argued that approaches based on the filter view of the problem carry with intrinsic limitations. The main limitation comes from the ignorance of the difference between alerts and the actions during the correlation process. A better view of the problem is to consider it as an inference problem, where we seek to infer an intruder's actions based on the alerts.

We described our novel framework named HCPN to predict attacks and understand alerts based on our new take of the problem. We showed that HCPN has promise as a way to reducing the false positive and false negatives in IDS.

### 8.1. Effectiveness of HCPN

The HCPN-based approach has several advantages to make it a promising approach to improving the quality of alerts in IDS. First, it uses the knowledge on actions' preconditions and postconditions in the correlation process. This knowledge provides additional information for the system to more reliably determine whether an alert issued by the IDS sensors is true or not. The false alerts are removed when they are associated to the special transition named "normal action" in the inference process. Second, it uses information such as the false positive rate, and the false negative rate naturally in the correlation process (since this information is encoded in the observation probabilities). Note that false positive rate and false negative rate of a specific alert type indicate the reliability of alerts in that type. The usage of this information can thus help to further reduce the false alerts. Third, it presents resources compromised (instead of actions) to show the progress of an attack. Since the number of resources compromised is much smaller than the number of raw alerts, HCPN can further reduce the total number of alerts shown to the administrators and active reactors. Fourth, the HCPN-based approach has potential to reduce the false negatives. This is due to the fact that there is an arc from a resource back to an action (whose postconditions

include compromising that resource) in the HCPN model. If later alerts (whose preconditions include the compromise of that resource) are confirmed to be true with high probability, the probability that a missing action has happened will be increased. Fifth, it provides confidence scores to the detection result by assigning probabilities to each mark indicating how likely an intruder has compromised a resource. This confidence score is very useful for the confidence fusion components to further improve the quality of the alerts. Sixth, HCPN works with and without training data. If training data are not available, HCPN behaves the same as the filter-based approaches that rely on only the precondition–postcondition relationship. If training data are available, the model parameters can be tuned and better correlation results can be achieved.

The HCPN model is a generic model not tied to a specific attack. It uses an attack action's precondition and postcondition information to remove false alerts and recover missing attacks. In other words, HCPN targets on the scenario attacks in which previous steps prepare for the later ones (a non-scenario attack can be considered as a one-step scenario attack). Note that, HCPN does not need to see all possible attack scenarios from the training data. Instead, possible attack scenarios are automatically derived from the preconditions and postconditions. The HCPN is especially effective to remove the false alerts whose preconditions are not satisfied. As long as the knowledge about the attacks is available, the gain we have seen in the experiments is very likely to be extended to other scenario attacks and other datasets.

HCPN can be deployed to different locations and at different levels. For example, there can be one HCPN for the DMZ zone, one for the inside network, and one for each host. Alternatively, there can be only one HCPN that analyzes alerts from all different sources. It is easy to see that recovering missing alerts is much harder than removing false alarms. This is endorsed by our experiments. For this reason, it is a good idea to optimize the detection rate in the sensors and rely on HCPN to remove false alerts. Under this setting, we expect 90% false positive rate reduction with HCPN.

### 8.2. Assumptions and limitations in HCPN

In HCPN, we have made several assumptions. The applicability of the model is limited under the conditions that the assumptions do not hold.

First, we assume that malicious actions' preconditions and postconditions are known as domain knowledge. This assumption is not new to HCPN. Almost all filter-based alert correlation approaches rely on this assumption and include this knowledge as rules in a database (e.g. [13]). This knowledge is usually available when the attacks are analyzed and alerts are designed into the IDS sensors. Alternatively, we can use statistical approaches to get this knowledge [37]. HCPN heavily depends on this knowledge to build the links between resources and actions. If this knowledge is not available or is incorrect, the effectiveness of the model will be greatly affected.

Second, we assume that the initial probability of resources owned by the users and/or intruders can be estimated by the system. For example, the privilege of a user can be determined by the logon credential (e.g., anonymous user) he/she originally uses when entering the system. However, information available to the system is incomplete and so probabilities determined this way may be incorrect. For example, an intruder may already gain access to some resources with other approaches (e.g., social engineering) before entering the system and the IDS would not know this. To deal with this situation, we can assign a small floor probability to all agent/resource pairs to indicate that each resource may be accessible by an intruder through unknown approaches. From our experiments, HCPN is not sensitive to the floor probability chosen as long as it is within a reasonable range such as 0.01–0.05. Note that other precondition–postcondition based alert correlation approaches also require information about the initial state.

Third, the inference process deduces agent's actions using observation probabilities and transition probabilities. Although these probabilities are hard to be affected by agent's behaviors as discussed in Section 4, it is still possible that an extremely skillful intruder may mislead the system and cause incorrect results.

Fourth, we assume that malicious users do not cooperate with each other. With this assumption, we can treat each agent (indicated by a different source IP and/or user ID) separately. This assumption is valid for many intrusion cases since most attacks are isolated and script-based (e.g., most worms). However, this assumption is not valid for sophisticated intrusions, where a skilled intruder controls several agents and attacks the same system at the same time, as for instance the case of a bot

network [38]. To handle attacks launched by cooperating agents, we need to enhance the model to automatically correlate cooperating agents as "one" agent. In other words, the augmented model needs to test whether two agents are cooperating with each other, and combine the agents if the answer is yes. A possible approach to testing whether two agents are actually one is to compare the likelihood quotient $q$ with a threshold.

$$q = \frac{p(\text{one agent, raw alerts})}{p(\text{two agents, raw alerts})}. \quad (39)$$

We consider this as one of the future work items.

## Acknowledgements

Authors would like to thank members in Center for Secure and Dependable Software in University of Idaho for the valuable discussion. Thanks are also given to anonymous reviewers for their great comments and suggestions in improving this paper.

## Appendix

Actions, alerts, and resources used in the experiments

| Actions | Alerts | Resources |
|---|---|---|
| Admind | Alert_Admind | SystemExists |
| DNS_HInfo | Alert_DNS_HInfo | CiscoCatalyst3500XL |
| Email_Almail_Overflow | Alert_Email_Almail_Overflow | ActiveXEnabledBrowser |
| Email_Debug | Alert_Email_Debug | SystemCompromised |
| Email_Ehlo | Alert_Email_Ehlo | GainInformation |
| Email_Turn | Alert_Email_Turn | JavaEnabledBrowser |
| FTP_Pass | Alert_FTP_Pass | MailLeakage |
| FTP_Put | Alert_FTP_Put | SMTPSupportEhlo |
| FTP_Syst | Alert_FTP_Syst | SMTPSupportTurn |
| FTP_User | Alert_FTP_User | VulnerableCGIBin |
| HTTP_ActiveX | Alert_HTTP_ActiveX | VulnerableAlMailPOP3Server |
| HTTP_Cisco_Catalyst_Exec | Alert_HTTP_Cisco_Catalyst_Exec | VulnerableSadmind |
| HTTP_Java | Alert_HTTP_Java | DNS_HInfo |
| HTTP_Shells | Alert_HTTP_Shells | DDOSHappened |
| Mstream_Zombie | Alert_Mstream_Zombie | ExistService |
| Port_Scan | Alert_Port_Scan | SendMailInDebugMode |
| RIPAdd | Alert_RIPAdd | ExistFTPService |
| RIPExpire | Alert_RIPExpire | ReadyToLaunchDDOSAttack |
| Rsh | Alert_Rsh | GainTerminalType |
| Sadmind_Amslverify_Overflow | Alert_Sadmind_Amslverify_Overflow | DaemonInstalled |
| Sadmind_Ping | Alert_Sadmind_Ping | SensitiveInfoAccessed |
| SSH_Detected | Alert_SSH_Detected | SensitiveInfoLeaked |
| Stream_DoS | Alert_Stream_DoS | AgentInstalled |
| TCP_Urgent_Data | Alert_TCP_Urgent_Data | |
| TelnetEnvAll | Alert_TelnetEnvAll | |
| TelnetTerminaltype | Alert_TelnetTerminaltype | |
| TelnetXdisplay | Alert_TelnetXdisplay | |
| UDP_Port_Scan | Alert_UDP_Port_Scan | |
| EventCollector_Info | Alert_EventCollector_Info | |
| IllegalFileAccess | Alert_IllegalFileAccess | |
| Loki | Alert_Loki | |
| AgentInstall | Alert_AgentInstall | |
| FTP_Globbing_Attack | Alert_FTP_Globbing_Attack | |
| Normal_Action | | |

## References


[1] K. Julisch, M. Dacier, Mining intrusion detection alarms for actionable knowledge, in: Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining, July 2002, pp. 366–375.

[2] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, E. Stoner, State of the practice of intrusion detection technologies, Technical Report CMU/SEI-99-TR-028, 1999.

[3] S. Axelsson, The base-rate fallacy and its implications for the difficulty of intrusion detection, in: 6th ACM Conference on Computer and Communications Security, November 1999, pp. 1–7.

[4] K. Julisch, Mining alarm clusters to improve alarm handling efficiency, in: Proceedings of the 17th ACSAC, New Orleans, December 2001, pp. 12–21.

[5] R.C. de Boer, A Generic architecture for fusion-based intrusion detection systems, Master Thesis, Erasmus University Rotterdam, October 2002.

[6] F. Cuppens, Managing alerts in a multi-intrusion detection environment, in: 17th Annual Computer Security Applications Conference, New Orleans, USA, December 2001, pp. 22–31.

[7] H. Debar, A. Wespi, Aggregation and correlation of intrusion–detection alerts, in: Proceedings of the 4th International Symposium on Recent Advances in Intrusion detection (RAID), 2001, pp. 87–105.

[8] R.P. Goldman, W. Heimerdinger, S. Harp, C.W. Geib, V. Thomas, R. Carter, Information modeling for intrusion report aggregation, in: Proceedings of the DARPA Information Survivability Conference and Exposition II (DISCEX-II), June 2001, pp. 329–342.

[9] P.A. Porras, M.W. Fong, A. Valdes, A mission-impact-based approach to INFOSEC Alarm correlation, in: Proceedings Recent Advances in Intrusion Detection, October 2000, pp. 95–114.

[10] P.A. Porras, P.G. Neumann, EMERALD: event monitoring enabling responses to anomalous live disturbances, in: 1997 National Information Systems Security Conference, October 1997, pp. 353–365.

[11] A. Valdes, K. Skinner, Probabilistic alert correlation, in: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID), 2001, pp. 54–68.

[12] N. Ye, J. Giordano, J. Feldman, Q. Zhong, Information fusion techniques for network intrusion detection, in: 1998 IEEE Information Technology Conference, Information Environment for the Future, 1998, pp. 117–120.

[13] F. Cuppens, F. Autrel, A. Miège, S. Benferhat, Correlation in an intrusion detection process, Internet Security Communication Workshop (SECI'02), September 2002, pp. 153–172.

[14] F. Cuppens, A. Miège, Alert correlation in a cooperative intrusion detection framework, in: 2002 IEEE Symposium on Security and Privacy, May 2002, pp. 202–215.

[15] C. Geib, R. Goldman, Plan recognition in intrusion detection systems, in: DARPA Information Survivability Conference and Exposition (DISCEX), June 2001, pp. 46–55.

[16] M.-Y. Huang, T.M. Wicks, A Large-scale distributed intrusion detection framework based on attack strategy analysis, in: Web Proceedings of the First International Workshop on Recent Advances in Intrusion Detection (RAID'98), 1998, pp. 2465–2475.

[17] 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, vol. 2516, October 2002, pp. 74–94.

[18] 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, November 2002, pp. 245–254.

[19] P. Ning, D.S. Reeves, Y. Cui, Correlating alerts using preconditions of intrusions, Technical Report, TR-2001-13, North Carolina State University, Department of Computer Science, December 2001.

[20] S.J. Templeton, K. Levitt, A requires/provides model for computer attacks, in: Proceedings of the 2000 workshop on New security paradigms, 2001, pp. 31–38.

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

[22] 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, LNCS, vol. 803, Springer-Verlag, 1994, pp. 230–272.

[23] K. Jensen, Coloured Petri Nets and the invariant methodTheoretical Computer Science, vol. 14, North-Holland, 1981, pp. 317–336.

[24] K. Ilgun, R. Kemmerer, P. Porras, State transition analysis: a rule-based intrusion detection system, IEEE Transactions on Software Engineering 21 (3) (1995) 181–199.

[25] D. Frincke, D. Tobin, Y. Ho, Planning, Petri Nets, and intrusion detection, in: Proceedings of the 21st National Information Systems Security Conference (NISSC'98), 1998, pp. 346–361.

[26] Somesh Jha, Oleg Sheyner, Jeannette M. Wing, Minimization and reliability analyses of attack graphs, Technical Report, CMU-CS-02-109, February 2002.

[27] T. Moon, The expectation–maximization algorithm, IEEE Signal Processing Magazine (1996) 47–60.

[28] I.J. Good, The population frequencies of species and the estimation of population parameters, Biometrika 40 (Parts 3 and 4) (1953) 237–264.

[29] D. Frincke, Balancing cooperation and risk in intrusion detection, ACM Transactions on Information and System Security (TISSEC) 3 (1) (2000) 1–29.

[30] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, 1976.

[31] D. Yu, D. Frincke, Alert Confidence fusion in intrusion detection systems with extended Dempster–Shafer theory, in: Proceedings of ACMSE 2005.

[32] Lincoln Lab, MIT. DARPA 2000 intrusion detection evaluation datasets, 2000. Available from: <http://ideval.ll.mit.edu/2000index.html>.

[33] K. Jensen, Colored Petri-Nets—Basic Concepts, Analysis Methods, and Practical Use, second ed., vol. 1, Springer-Verlag, New York, 1996.

[34] L.M. Kristensen, S. Christensen, K. Jensen, The practitioner's guide to coloured Petri nets, International Journal on Software Tools for Technology Transfer 2 (1998) 98–132.

[35] S. Kumar, E.H. Spafford, A pattern-matching model for intrusion detection, in: Proceedings of the National Computer Security Conference, 1994, pp. 11–21.

[36] S. Kumar, E. Spafford, A Pattern matching model for misuse intrusion detection, in: 17th National Computer Security Conference, 1994, pp. 11–21.
[37] X. Qin, W. Lee, Statistical causality analysis of INFOSEC alert data, in: Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID), 2003, pp. 73–93.
[38] D. Dittrich, Dissecting distributed malware networks. Availabel from: <http://security.isu.edu/ppt/pdfppt/Core02.pdf>.
[39] D. Yu, D. Frincke, A Novel framework for alert correlation and understanding, in: International Conference on Applied Cryptography and Network Security (ACNS), Springer's LNCS series, vol. 3089, 2004, pp. 452–466.
[40] Y. Zhai, P. Ning, P. Iyer, D.S. Reeves, Reasoning about complementary intrusion evidence, in: Proceedings of 20th Annual Computer Security Applications Conference, December 2004, pp. 39–48.
[41] J.W. Harris, H. Stocker, Maximum likelihood method §21.10.4, in: Handbook of Mathematics and Computational Science, Springer-Verlag, New York, 1998, pp. 824.
[42] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA off-line intrusion detection system evaluation as performed by Lincoln laboratory, ACM Transactions on Information and System Security 3 (4) (2000).

**Dong Yu** holds a Ph.D. degree in computer science from University of Idaho, an MS degree in computer science from Indiana University/Bloomington, an MS degree in electrical engineering from Chinese Academy of Sciences, and a BS degree (with honor) in electrical engineering from Zhejiang University (China). He has been working in Microsoft since 1998. His research interests are in speech processing, computer and network security, and machine learning. He has published dozens of refereed journal and conference papers in the above areas. He is a senior member of IEEE and a member of ACM.

**Deborah Frincke** is Chief Scientist of the CyberSecurity groups at Pacific Northwest National Laboratory. Prior to joining PNNL, she was a member of the University of Idaho faculty, departing as a Full Professor. She co-founded one of the first NSA Centers of Academic Excellence in Information Assurance, through University of Idaho's Center for Secure and Dependable Systems. Her mid-90s research served as the basis for TriGeo Network Systems' commercial product. Her research spans a broad cross section of computer security, including very large system defense, security risk/benefit analysis, alert consolidation, and computer security education. She has published over eighty articles and technical reports. She received her Ph.D. in computer science with a focus on computer security from University of California, Davis in '92.