A Model for Partially Asynchronous Observation of Malicious Behavior

Mark M. Seeger
Center for Advanced Security Research Darmstadt (CASED)
Department Secure Services
Mornewegstraße 32, 64293 Darmstadt, Germany
Email: mark.seeger@cased.de

Stephen D. Wolthusen
Gjøvik University College
Department of Computer Science
N-2818 Gjøvik, Norway
and
Royal Holloway, University of London
Information Security Group
Egham, Surrey TW20 0EX, UK
Email: stephen.wolthusen@rhul.ac.uk

***Abstract*—For a non-trivial attack to be successful in compromising a target, multiple causally related operations must be performed. Detecting such potentially unknown sequences is the core problem in intrusion detection.**

In this paper, we focus on the problem of *observing* attacks over non-uniform, partially asynchronous event sets. We hence propose charactering attacks as partially ordered sets, and we show how these can be detected asynchronously, as will typically be the case in a modern computing architecture. By extending a naïve model incorporating subsets of known causal dependencies, enhanced observation strategies minimizing the number and cost of observations can be derived. This incorporation of knowledge regarding constraints on attack causality into observations allows for notable enhancements in the efficiency of detection. We also provide a simple example of an application of the model for the case of an intrusion detection system on a co-processor observing a host, although the model is intended for arbitrary non-uniform architectures and concurrent operations.

***Keywords*-Intrusion detection, asynchronous and partially asynchronous observation, causality models**

I. Introduction

The security of any host intrusion detection system (IDS) is limited by the ability of an adversary to compromise the IDS itself. The IDS must therefore be able to detect and mitigate attacks before itself becomes compromised. Recent advances in computer architecture with large numbers of (partially asynchronous) processing units have, moreover, made the conventional notion of an IDS questionable: While detection algorithms operate on a *snapshot* of observations, attacks may be ongoing in parallel to the detection and mitigation efforts.

Although it is possible to force synchronization of events (mainly memory access) in a host system, this is highly undesirable, raising the necessity of performing observations asynchronously or minimizing the need for them. Previous work has sought to characterize the effects of non-uniform memory architectures (NUMA) by such observations, but has not been concerned with characterizing the observations further. In this paper, we seek to describe a model of general attack characteristics which can be used to optimize observation strategy. To demonstrate this, we describe a naïve model of causal but equiprobable asynchronous events, observed by way of memory access forcing point-wise synchronization, and we expand this model into an explicit causality model where event sets are constituted from partially ordered sets. These sets are also assigned arbitrary probability distributions, demonstrating a significant gain in efficiency.

The remainder of this paper is structured as follows: Section II briefly reviews the related work, before we start with the actual contribution of this paper in Section III, where we detail our naïve model and our causality model. In order to present the practical relevance, especially of the latter model, we show the application of both models in Section IV before we end the paper with a conclusion and an outlook in terms of current and future work in Section V.

II. Related Work

Lee et al. propose an algebraic model for describing causality interfaces between actors, capable of, for instance, revealing causality loops, given that each actor is defined using a Turing-complete language. Schwarz [1] focuses on causality in distributed computations and presents a corresponding notion which is applicable in distributed, asynchronous systems. While not aimed at information security, his work covers some of the most important properties needed by an observer in order to reliably obtain the status of a distributed computation. That is, for instance, taking into account that "*every cause precedes its effect*".

Lamport's *happened before* relation[1] [2], written as $\rightarrow$, provides an intuitive definition for the case of partial orderings in distributed (as well as in asynchronous) systems; this can also, however, be extended to total orderings. Even though Tarafdar and Garg [3] describe the happened before relation as "*the best possible approximation of real time order that one can make in a message-passing distributed system without external channels*", they also claim that it is the wrong model for expressing and describing potential causality. To

[1] Also used by [1].

this extent, they propose to use the potentially caused relation, written as $\overset{p}{\hookrightarrow}$.

A related modeling approach to the one proposed here was described for security protocols by Backes et al. [4]. It relies on a finite graph (referred to as a *causal graph*) to express causality between events. Whilst [4] is only applicable to cryptographic protocols, it demonstrates the utility of relying on causality as an abstraction when reasoning over security properties. While many IDSs incorporate fixed causal models, such as rule sets, graphs, or automata, the more general case has not been fully explored for this application. However, King et al. [5] described a causality-based intrusion detection system (CIDS) incorporating network- and host-based alerts, reasoning about their severity by interpreting the causal dependency between network events (i.e., packets) and host events (i.e., systems calls). As a result of this, CIDS is able to reduce the number of false-positives, giving each alarm more significance. Subsequent work [6] extended this approach in conjunction with existing IDSs to provide causal relations for suspicious host- and network-based states and events.

Ning et al. [7] propose the correlation of IDS alerts in order to construct attack scenarios. They tested their formal framework using the DARPA dataset from 2000 and report a significant reduction of the false alert rate. In contrast to [7], we do not rely on alerts generated by third-party applications as we focus on low-level data elements being the subject of an subversion attempt rather than services. Also, our definition of an attack (scenario) is more finely-grained (i.e., [7] correlate several *service attacks* while we correlate several *data element attacks*).

Alert correlation as such is the subject of Cuppens and Miège [8], and Benferhat et al. [9]. Both works focus on the reduction of alerts generated by an IDS. In this context, our work could be seen as the mechanism which generates such alerts, not based on malicious network traffic but on a set of critical host based data elements.

The recognition of an intruder's intent is discussed by Cuppens et al. [10] and Wu et al. [11]. The former focus on the reduction of alerts presented to a network administrator, while we focus on the reduction of observation frequency as such, keeping the detection probability as high as possible. The latter apply the Dynamic Bayesian Network approach to model uncertainty, while we apply the noisy-or model to incorporate noise (i.e., uncertainty), and we focus on a manageable amount of host side elements rather than large amounts of intrusive data.

Very recently, Albanese et al. [12] published their work presenting a scalable analysis of attack scenarios. They analyze network infrastructure in order to reveal the dependency of available services. Based on the importance and dependency of network components and services, a dependency graph is constructed which is used later on for constructing (probabilistic) attack graphs. In contrast to [12], who rely on known vulnerabilities (i.e., known exploits) rendering it impossible to detect yet unknown attacks, we focus on loosely coupled data elements which may be attacked during even unknown subversion attempts. Furthermore, as our model does not focus on the analysis of vast amounts of data, we do not need algorithms that "can process between 25 and 30 thousands [sig!] alerts per second" [12].

Description of Assumption	Naïve Model	Causality Model
Uses the happened before relation $\rightarrow$	✓	✗
Uses the potentially caused relation $\overset{p}{\hookrightarrow}$	✗	✓
Causal dependency between attack sets	✗	✓
Uniform distribution of attack probabilities	✓	✗
Maximum detection probability	✓	✗
Maximum performance degradation on the host	✓	✗
High false-positives rate	✓	✗
Observation succeeds always	✓	✗
An observation attempt can be suppressed	✗	✓
Legitimate changes of value can happen	✗	✓

Table I
COMPARISON OF ASSUMPTIONS INCORPORATED INTO EACH MODEL.

III. OBSERVING PARTIALLY ORDERED ATTACK SETS

Section III-A shows a naïve model capturing the worst-case scenario in terms of observation frequency and performance degradation due to coprocessor-triggered host observations when studying modeling outcomes. In contrast to this, Section III-B presents an explicit causality model. Here, we apply partially tailored versions of well-known models in order to incorporate our assumptions. This results in only a moderate decrease of detection probability whilst at the same time reducing the performance impact.

We focus our attention on a finite $POASET$ (i.e., partially ordered attack sets) of shared host-side data elements. We assume an attacker needs to inevitably tamper with all $de_i \in POASET; i = 1, ..., n$ in order to complete his malicious intentions, which may either target the host system as such or the in previous publications (cf. [13] and [14]) proposed off-host host observation mechanism. Formally, we define $\alpha_j = \{de_i, ..., de_m\} \subseteq POASET; 1 \leq m \leq n$ to be attack sets, where each α_j consists of at least one de_i. No further assumptions other than partial ordering are made for the events constituting attacks.

In practice, the exact content of the $POASET$ depends on parameters such as observation goal (e.g. privilege escalation) and type of system. Despite this, in order to achieve his malicious goal, an attacker must tamper with the full $POASET$ and is thus confronted with the problem that some of the concerned data elements depend on others. Therefore, he must build attack sets which have to be tampered with in order. This fact highly limits an attacker's possibilities when choosing his course of action, and it leaves room for causality based observations (cf. Section III-B).

Table I details the different assumptions made for each model, with respect to the performance degradation caused on the host when applying it.

A. The Naïve Model

The naïve model makes no explicit assumptions regarding certain data elements and their likelihood of becoming the

subject of an attack, nor does it assume any causal dependency between attack sets. It presents the worst-case scenario in terms of observation frequency and performance degradation and validates the assumption that modeling causal dependency between data elements is likely to reduce the loss of performance.

While the set of all attack sets $\mathcal{A} = \bigcup_{j=1}^{k} \alpha_j$ forms a partial order fulfilling the *happened before* relation [2], each α_j itself does not represent an order of any kind. The happened before relation defines – among other – that if α_j is attacked before α_{j+1}: $\alpha_j \rightarrow \alpha_{j+1}$ and if $\alpha_j \rightarrow \alpha_{j+1}$ and $\alpha_{j+1} \rightarrow \alpha_{j+2}$: $\alpha_j \rightarrow \alpha_{j+2}$. Obviously, $\rightarrow$ is transitive.

In other words, an attacker may freely decide in which order he wants to alter the data elements *within* each attack set α_j. This gives him $\sum_{j=1}^{k} |\alpha_j|!$ possibilities to do so, presuming that he alters each data element only once. He must, however, finish the attack against α_j before he can attack α_{j+1}.

This confronts the observer with two extreme cases where $\alpha_1 \equiv POASET$ and where $|\alpha_j| = 1$. We deduce from the former that an attacker has $|POASET|!$ possibilities to complete his attack while the latter extreme represents full serialization. Since any attack will be composed of loosely coupled attack sets, and because of the constraint that these sets have to be executed in order, any detection mechanism faces the problem of the inability to be one step ahead of the attacker. That is, the full $POASET$ has to be observed according to the busy-wait approach and at maximum observation speed, which is not efficient due to the performance degradation this causes on the host-side [15]. This fact already implies that any reasonable assumption made with respect to a causal dependency between attack sets is likely to reduce the impact on the host-side, as target-oriented observations can be performed (cf. Section III-B).

As shown before, an attacker can freely choose the order in which he alters the data elements within attack sets, while at the same time being restricted by the happened before relation with regard to the execution of data elements belonging to different attack sets. Once the attack sequence is determined, the probability that a specific de_i within the targeted α_j will be attacked (assuming uniform distribution) is:

$$P(de_i|\alpha_j) = \frac{1}{|\alpha_j|} \tag{1}$$

The existence of attack sets clearly limits an attacker's alternatives for achieving any malicious goal. If the number of attack sets and their composition would be visible to an observer, targeted and ordered observations would be possible. Unfortunately, the existence of these sets is not visible to anybody but the attacker himself. Without any further assumptions being made, an observer is therefore always confronted with the problem of observing *all* data elements of the $POASET$ with equal frequency. As no assumptions about causal dependency between attack sequences have been made, each data element is equally likely to be subject of an attack at point in time t_0; i.e., the starting point of an attack:

$$P(de_i|t_0) = \frac{1}{|POASET|} \tag{2}$$

We note that this is of particular interest for multiple concurrent scheduling entities (as is the case in multi-core, multi-processor architectures) and denote $AP = \{ap_1, ..., ap_N\}$ to be the set of entities (processors) used by the attacker. In the worst case, from a security point of view, an attacker can utilize all processors to capacity by assigning each processor a new data element to work with as soon as the work with the former data element has been finished. Following this schema, rather than having several processors alter one data element in parallel, limits the need of inter processor communication, reduces forced synchronizations, and is therefore faster.

For simplicity, we assume that each alteration targeted at each data element requires an equivalent amount of processor cycles and that attacking a data element costs as much as observing it. Hence, the abstract[2] *time* the attacker needs to accomplish his goal is given by the maximum number of data elements to be processed by a single processor:

$$t_{attack} = \left\lceil \frac{|POASET|}{|AP|} \right\rceil \tag{3}$$

As one data element cannot be split over several processors, the result is mapped to the smallest subsequent integer (i.e., ceiling).

Alterations of data elements can be detected if the observation mechanism is able to observe de_i before and after it has been attacked, and the data read during both observations differ[3]. While, due to our assumptions, an analogous equation accounts for the time needed to observe all data elements (i.e., $t_{observation}$), we are interested in gaining the maximum probability of detecting an ongoing attack. The probability to do so is 1 if we are able to observe the set of all data elements within the $POASET$ in less than the time the attacker needs to alter one data element $de_i \in POASET$. This correlation can be expressed by writing:

$$t_{observation} < \left\lceil \frac{t_{attack}}{|POASET|} \right\rceil \tag{4}$$

Clearly, Equation 4 is a simple model leaving aside any causal dependency between data elements and assuming uniform distribution of attack probabilities. Also, employing this model would cause a high false positive rate, as even legitimate changes to any $de_i \in POASET$ would be treated as an attack. Furthermore, since data elements cannot be spread over processors, altering one data element exclusively and altering up to $|AP|$ independent elements in parallel consumes the same amount of time.

With no further assumptions made, the likelihood of being attacked is the same for all data elements within the $POASET$. We have therefore shown that, for maximum detection probability, an observer mechanism must be much faster than the processors used for attack. This is neither a surprise nor a problem in reality: Coprocessors such as modern GPUs are able to operate much faster than modern CPUs. Hence, the problem is not the speed as such but the performance degradation high-speed bulk observations cause

[2]The concrete time depends on the actual processor speed.

[3]For simplicity, we assume that the observation results are not stored.

on the host-side.

We define the performance degradation D to be maximum (i.e., 1) if Equation 4 is fulfilled:

$$P(D = 1|t_{observation} < \left\lceil \frac{t_{attack}}{|POASET|} \right\rceil) = 1 \quad (5)$$

Below, we present the causality model, incorporating resilient assumptions regarding the success probability of an observation as well as possible causal dependencies between attack sets. This reduces performance degradation whilst maintaining acceptable detection probability.

B. The Causality Model

In the previous section, we have associated each data element $de_i \in POASET$ to have the same probability of being the *first* attack target (cf. Equation 2). Also, we have assumed that an observer has no knowledge about the sequence according to which an attacker will aim at a specific de_i, nor that he knows anything about the correlation between different data elements.

For the causality model, we allow further assumptions to be made but limit our attention to causality chains, despite the fact that more complex relations between sets of data elements may be possible. As we do not know the exact course of any attack, our assumptions are probabilistic in type.

The discussion up to this point has tacitly assumed that costs are equal for each event observation. However, this is not desirable for a NUMA architecture with different memory pathways, where each of them possibly has a different priority. As a result, not every attempt to observe a data element is assumed to be successful. This is intrinsic to our coprocessor/host combination (i.e., a NUMA architecture) which imposes different priorities on concurrent memory accesses depending on their source.

We distinguish between two sets of memory pathways, being H, the ones triggered by the host itself, and C, the ones triggered by a coprocessor. Each memory pathway $h_s \in H$ and $c_t \in C$, respectively, is associated with a priority (i.e., p_s and p_t). With respect to an operation on de_i, H and C form a partial order over the happened before relation, such that $p_s \geq p_t \Rightarrow h_s \rightarrow c_t$. Whenever this relation is true for any two competing memory pathways h_s and c_t, we regard the observation as failed. For simplicity, we postulate that the set of memory pathways follows a binomial distribution. Therefore, assuming host-triggered events to have a higher priority, the probability of a successful observation can be expressed as follows: Let $p = \frac{|C|}{|C \bigcup H|}$. Then:

$$\hat{x}_i = B(x|p, n) = \binom{n}{x} p^x (1-p)^{n-x} \quad (6)$$

where x denotes the number of possibly conflicting memory pathways, p, the ratio between coprocessor and host-triggered events targeting a certain data element, and n, the number of successive observation tries.

Thus, $\hat{x}_i$ gives us the probability that exactly x memory pathways can successfully observe data element de_i without being overridden by a host-triggered memory pathway. While parameter x serves as a lower bound, we are in fact interested in the probability that *at least* x memory pathways can be used unrestrictedly. This is obtained by applying Equation 6 to all values $w = t, ..., n$:

$$\hat{X}_i = \sum\nolimits_{w=t}^{n} \binom{n}{w} p^w (1-p)^{n-w} \quad (7)$$

The smaller $\hat{X}_i$ is, the lower the probability that we can detect a change in value at de_i while an attack is ongoing, as this is the probability that we can observe a given data element without being suppressed by memory pathways triggered by the host. We refer to $\hat{X}_i$ as noise (i.e., λ_i) and make use of the noisy-or model [16] in order to deal with the possibility that an observation attempt is not successful. In addition to another memory pathway suppressing our observation, we have to incorporate a second uncertainty: The probability that we observe a benign change in value at de_i which we cannot distinguish from a malicious one. With respect to the noisy-or model, we speak of this factor as a global confidence probability, denoted as λ_0 [4].

Since we are interested in the causal dependency between attack sets, we make use of the *potentially causes* relation $\overset{p}{\hookrightarrow}$, proposed by Tarafdar and Garg [3] who describe it as a relation capable of expressing the connection of two events not sharing a local clock while at the same time having the possibility of one causing the other. In our case, we say that O_j^m and O_{j+1}^m are two observations of malicious changes that fulfill the happened before relation (i.e., $O_j^m \rightarrow O_{j+1}^m$). Furthermore, if the changes observed have the potential to cause other data elements to change in value, then $\{O_j^m \rightarrow O_{j+1}^m\} \overset{p}{\hookrightarrow} O_{j+2}^m$ and if $\{O_j^m \rightarrow O_{j+1}^m\} \overset{p}{\hookrightarrow} O_{j+2}^m$ and $\{O_{j+2}^m \rightarrow O_{j+3}^m\} \overset{p}{\hookrightarrow} O_{j+4}^m$, then $\{O_j^m \rightarrow O_{j+1}^m\} \overset{p}{\hookrightarrow} O_{j+4}^m$. Just as in its original form, our version of the potentially caused relation is transitive.

The probability that a change in value of a subset of the $POASET$ is malicious can be written as:

$$P(O_j^m|de_i, ..., de_n) = 1 - ((1-\lambda_0) \prod\nolimits_{i=1}^{n} (1-\lambda_i)) \quad (8)$$

Here, O_j^m is a random variable with the probability that an observation of a change in $de_i, i = p, ..., n$ is malicious. λ_0 is the confidence probability, i.e., $1 - \lambda_0$ is the likelihood that a benign event causes a malicious change in value. λ_i is a noise parameter which is associated with the data being subject to observation. In cases where we have causal dependency between data elements, we simply take the result from Equation 8 as a noise-parameter, define a global confidence probability, and apply the noise model as before for the purposes of this model.

$$\widehat{O_k^m} = P(O_k^m|O_j^m \rightarrow O_l^m) = 1 - ((1-\lambda_0) \prod\nolimits_{j=1}^{l} (1-\lambda_j)) \quad (9)$$

Equation 9 implies that $\{O_j^m \rightarrow O_l^m\} \overset{p}{\hookrightarrow} \widehat{O_k^m}$ and gives us a probabilistic answer to the question of how likely it is to see $\widehat{O_k^m}$ after we have seen the attack sequence $O_j^m \rightarrow O_l^m$. With this probability, we can start a targeted observation of the data elements included in observation O_k^m and thus, have the ability to adjust the observation frequency for each set of data

[4] $\lambda_i \in [0; 1];\ 0 \leq i \leq n$

elements being observed according to the causalities assumed.

Clearly, prior to applying the causality model proposed above, a thorough analysis of concerned data elements, their relation to each other, and possible attack types they could be used for, is mandatory. Once this has been accomplished, the model is capable of giving the probability with which certain observation results cause another one. Thus, this model not only allows an observation mechanism to be tuned and therefore, to gain in efficiency, but also provides limited predictive ability.

IV. Practical Application of Proposed Models

The following provides a highly simplified instantiation of the model proposed in the preceding section, again divided into the naïve and causality models. We assume our set of data elements comprises 12 elements: $POASET = \{de_1, ..., de_{12}\}$. We also assume without loss of generality that an attacker has decided to perform an attack $\mathcal{A}$ having four attack sets $\alpha_{1,...,4}$ containing the following data elements: $\alpha_1 = \{de_2, de_5, de_6, de_{10}, de_{12}\}$; $\alpha_2 = \{de_1, de_8\}$; $\alpha_3 = \{de_4, de_7, de_{11}\}$, and $\alpha_4 = \{de_3, de_9\}$.

A. Application of the Naïve Model

If the observation mechanism were to be aware of the above defined attack sets, the probability that, e.g., de_5 will be attacked is $P(de_5|\alpha_1) = \frac{1}{5}$ (cf. Equation 1). Without further assumptions or information, we assume events to be equiprobable:

$P(de_i|t_0) = \frac{1}{12}, i \in |POASET|$ (cf. Equation 2)

Assuming no relation between all data elements within the $POASET$ and being confronted with an attacker employing 4 processors $AP = \{ap_1, ..., ap_4\}$, at least one processor needs to process $t_{attack} = \frac{12}{3} = 4$ data elements, which consumes the corresponding abstract time (cf. Equation 3). In order to achieve the maximum detection probability, an observation of the full $POASET$ has to be finished in less than the time the attacker needs to attack one single data element: $t_{observation} < \frac{4}{12} = \frac{1}{3}$ (cf. Equation 4). While achieving this would result in a detection probability of 1, presupposing that the $POASET$ is complete, it would also cause the maximum performance degradation on the host-side due to forced synchronisations: $P(D = 1|t_{observation} < \frac{1}{3}) = 1$ (Equation 5)

B. Application of the Causality Model

We will now apply the probability model introduced in Section III-B in order to show that observing all $de_i \in POASET$ with equal frequency is not necessary.

First, we define one host-triggered and one coprocessor-triggered set of memory pathways[5] for each of the 12 data elements used in the example. Next, we need the probability that at least two coprocessor triggered memory pathways (parameter x in Equation 6) can be employed subsequently. Ideally, the first two memory pathways selected (parameter n) are of this type, as this implies a failure rate f of only 1 (i.e., $\frac{n}{x}$). The more tries we need in order to select two coprocessor triggered memory pathways, the higher the failure rate. Table II shows the results for $\hat{x}_i, \hat{X}_i$ and f_i, $i = 1, ..., 12$, where the success factor n is set to 2, 4 and 8, respectively. Table II reveals the connection between failure rate and success probability. That is, the more errors we allow, the higher the probability that we get two coprocessor triggered memory pathways. Correspondingly, minimizing the failure rate results in a lower success probability. The correlation between success probability, the coprocessor/host memory pathway ratio (i.e., $\frac{|C_i|}{|H_i|}$) and the failure rate is depicted in Figure 1. As we ordered the results according to the memory pathway ratio, it is clear to see that we can overcome the *failure rate problem* by assigning more coprocessor triggered memory pathways to each data element (i.e., success probability of up to 80% with a failure rate of 1 and a coprocessor/host memory pathway ratio of $8.57 : 1$). Given the success probabilities, we can now apply Equations 8 and 9.

We then have to define our observation sets O_j, $j = 1, ..., g$. Ideally, we are not only able to define as many observation sets as the attacker has defined attack sets, but we also have them including the same data elements. While this may not be possible in general, we assume the following observation sets, gained from analyzing selected data elements: $O_1 = \{de_1, de_2, de_5, de_6, de_8, de_{10}, de_{12}\}, O_2 = \{de_4, de_7\}$, and $O_3 = \{de_3, de_9, de_{11}\}$. From our analysis we know that data elements de_3, de_9 and de_{11} (i.e., O_3) show a causal dependency to all other data elements. Therefore, we are interested in the strength of this connection. We start by calculating the probability that a change in value within O_1 and O_2 is malicious (cf. Equ 8). We express the confidence in the preceding analysis by setting the global parameter λ_0 to be 70% for O_1^m, 60% for O_2^m and 65% for O_3^m. Furthermore, we use the success probabilities from Table II (success factor $n = 2$) as λ_i (i.e., $\lambda_i \equiv \hat{X}_i$). This associates the probability of successfully performing an observation with the probability that the observation result is correct:

$$P(O_1^m|O_1) = 1 - ((1 - 0.70) \prod_{i \in O_1} (1 - \lambda_i)) = 0.99$$
$$P(O_2^m|O_2) = 1 - ((1 - 0.60) \prod_{i \in O_2} (1 - \lambda_i)) = 0.68$$

Since we are interested in the probability that O_1^m and O_2^m cause O_3^m (i.e., $\{O_1^m \rightarrow O_2^m\} \overset{p}{\hookrightarrow} O_3^m$) we now apply Equation 9 as follows:

$$\widehat{O_3^m} = P(O_3^m|O_1^m \rightarrow O_2^m) =$$
$$1 - ((1 - 0.65) \prod_{j \in O_1^m \bigcup O_2^m} (1 - \lambda_j)) = 0.99$$

This result is to be interpreted as follows: We have made resilient assumptions based on thorough analysis and thus, were able to form observation sets. We paid respect to the existing chance that our observation triggered by a coprocessor will not succeed due to host-triggered memory pathways of higher priority and subsequently used the resulting confident values as noise parameters of the noisy-or model.

[5]For a better understanding, one can think of these sets as sets of threads.

x	2	2	2
n	2	4	8
C_1	10		
H_1	5		
$\hat{x}_1$	0.44	0.30	0.02
$\hat{X}_1$	0.44	0.88	0.99
f_1	1	2	4

x	2	2	2
n	2	4	8
C_2	8		
H_2	4		
$\hat{x}_2$	0.33	0.30	0.02
$\hat{X}_2$	0.33	0.88	0.99
f_2	1	2	4

x	2	2	2
n	2	4	8
C_3	10		
H_3	10		
$\hat{x}_3$	0.25	0.38	0.11
$\hat{X}_3$	0.25	0.69	0.96
f_3	1	2	4

x	2	2	2
n	2	4	8
C_4	4		
H_4	8		
$\hat{x}_4$	0.11	0.30	0.27
$\hat{X}_4$	0.11	0.41	0.80
f_4	1	2	4

Table II
EXTRACT OF RESULTS WHEN APPLYING EQUATIONS 8 AND 9 FROM SECTION III-B TO THE 12 SETS OF MEMORY PATHWAYS FROM SECTION IV; ONE HOST AND ONE COPROCESSOR SET FOR EACH OF THE 12 DATA ELEMENTS.

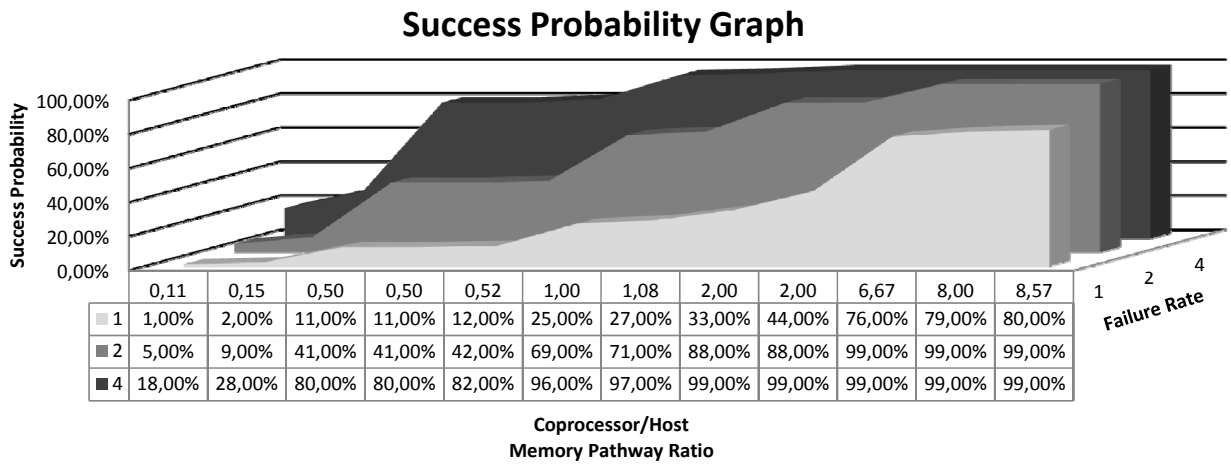

Figure 1. The results for $\hat{x}_i, \hat{X}_i$ and fr_i, $i = 1, ..., 12$

.

Supposing the reasonability of our assumptions, we are 99% confident that a malicious change in value within the data elements covered by O_3 is potentially caused by malicious changes in value of the data elements observed by O_1 and O_2. In other words, once we have observed O_1^m and O_2^m, we can instantly lock O_3 and thus, will counter the subversion attack while it is ongoing with a probability of 99%.
With this result given, it is no longer necessary to observe all $de_i \in POASET$ with equal frequency which automatically influences the performance degradation positively. At the same time, there is a negligible loss in terms of detection rate (i.e., $\widehat{O_3^m} = 0.99$), compared to the busy-wait approach proposed in Section III-A.

V. CONCLUSION AND FUTURE WORK

We no longer have to be one step behind the attacker. Therefore, we are not interested in observing a full set of data elements without reasoning about what an attacker may do next. To this extent, we presented a model that focuses its observation on subsets of a partial order over host-side data elements. By associating each subset with a potential to cause the next action and applying two slightly tailored well-known models, we gain the following advantages: first, we are able to reduce the performance degradation on the host-side due to less interference. Second, we are able to jump at least one step ahead of the attacker, locking data elements likely to be attacked soon.

Our future work will be concentrated on implementing a proof of concept of the proposed models. We will focus on the access times of file system parts such as `/etc/passwd` and `/etc/shadow` and correlate the data with the frequency of, for instance, root logins. This first example is based on the assumption that after user specific data has been modified, a rising number of root logins will be the result.

REFERENCES

[1] R. Schwarz, "Causality in Distributed Systems," in *5^{th} ACM SIGOPS European Workshop*. ACM, 1992.
[2] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Commun. ACM*, vol. 21, pp. 558–565, Jul. 1978.
[3] A. Tarafdar and V. K. Garg, "Happened Before is the Wrong Model for Potential Causality," University of Texas at Austin: Parallel & Distributed Systems Group, Tech. Rep., Jul. 1998.
[4] M. Backes, A. Cortesi, and M. Maffei, "Causality-based abstraction of multiplicity in security protocols," in *CSF 2007*. IEEE Computer Society, 2007.
[5] S. T. King, Z. M. Mao, , and P. M. Chen, "CIDS: Causality Based Intrusion Detection System," University of Michigan, Tech. Rep., 2004.
[6] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen, "Enriching intrusion alerts through multi-host causality," in *NDSS 2005*, 2005.
[7] P. Ning, Y. Cui, and D. S. Reeves, "Constructing Attack Scenarios through Correlation of Intrusion Alerts," in *CSS 2002*. ACM, Nov. 2002, pp. 245–254.
[8] F. Cuppens and A. Miege, "Alert Correlation in a Cooperative Intrusion Detection Framework," in *SP 2002*. IEEE Computer Society, May 2002, pp. 202–215.
[9] F. A. u. F. C. Salem Benferhat, "Enhanced Correlation in an Intrusion Detection Process," in *MMM-ACNS 2003*. Springer-Verlag, Sep. 2003, pp. 157–170.

[10] A. M. S. B. F. Cuppens, F. Autrel, "Recognizing Malicious Intention in an Intrusion Detection Process," in *HIS 2002*. IOS Press, Dec. 2002, pp. 806–817.

[11] P. Wu, Y. Shuping, C. Junhua, and W. Zhigang, "Recognizing Intrusive Intention Based on Dynamic Bayesian Networks," in *IEEC 2009*. IEEE Computer Society, May 2009, pp. 241–244.

[12] M. Albanese, S. Jajodia, A. Pugliese, and V. S. Subrahmanian, "Scalable Analysis of Attack Scenarios," in *ESORICS 2011*. Springer-Verlag, Sep. 2011, pp. 416–433.

[13] T. R. McEvoy and S. D. Wolthusen, "Using Observations of Invariant Behavior to Detect Malicious Agency in Distributed Environments," in *IMF 2008*. GI, Sep. 2008, pp. 55–72.

[14] M. M. Seeger and S. D. Wolthusen, "Observation Mechanism and Cost Model for Tightly Coupled Asymmetric Concurrency," in *ICONS 2010*. IEEE Computer Society, Apr. 2010, pp. 158–163.

[15] M. M. Seeger, S. D. Wolthusen, C. Busch, and H. Baier, "The Cost of Observation for Intrusion Detection: Performance Impact of Concurrent Host Observation," in *ISSA 2010*. IEEE Computer Society, Aug. 2010, pp. 1–8.

[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.