Privacy-Preserving Alert Correlation: A Concept Hierarchy Based Approach [*]

Dingbang Xu and Peng Ning
Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
{dxu,pning}@ncsu.edu

Abstract

With the increasing security threats from infrastructure attacks such as worms and distributed denial of service attacks, it is clear that the cooperation among different organizations is necessary to defend against these attacks. However, organizations' privacy concerns for the incident and security alert data require that sensitive data be sanitized before they are shared with other organizations. Such sanitization process usually has negative impacts on intrusion analysis (such as alert correlation). To balance the privacy requirements and the need for intrusion analysis, we propose a privacy-preserving alert correlation approach based on concept hierarchies. Our approach consists of two phases. The first phase is entropy guided alert sanitization, *where sensitive alert attributes are generalized to high-level concepts to introduce uncertainty into the dataset with partial semantics. To balance the privacy and the usability of alert data, we propose to guide the alert sanitization process with the entropy or differential entropy of sanitized attributes. The second phase is* sanitized alert correlation. *We focus on defining similarity functions between sanitized attributes and building attack scenarios from sanitized alerts. Our preliminary experimental results demonstrate the effectiveness of the proposed techniques.*

1 Introduction

In recent years, the security threats from infrastructure attacks such as worms and distributed denial of service attacks are increasing. To defend against these attacks, the cooperation among different organizations is necessary. Several organizations such as CERT Coordination Center and DShield (`http://www.dshield.org/`) collect data (including security incident data) over the Internet, perform correlation analysis, and disseminate information to users and vendors. The security incident data are usually collected from different companies, organizations or individuals, and their privacy concerns have to be considered. To prevent the misuse of incident data, appropriate data sanitization through which the sensitive information is obfuscated is highly preferable. For example, DShield lets audit log submitters perform partial or complete obfuscation to destination IP addresses in the datasets, where partial obfuscation changes the first octet of an IP address to decimal 10, and complete obfuscation changes any IP address to a fixed value $10.0.0.1$.

To protect networks and hosts on the Internet, many security systems such as intrusion detection systems (IDSs) are widely deployed. However, current IDSs have some well-known limitations. They usually flag thousands of alerts per day, among which false alerts are mixed with true ones. To better understand the security threats, it is necessary to perform alert correlation. Current correlation approaches can be roughly divided into four categories: (1) similarity based approaches (e.g., [19, 17]), which perform clustering analysis through calculating the similarity between alert attributes, (2) approaches based on pre-defined attack scenarios (e.g., [5, 9]), which build attack scenarios through matching alerts to pre-defined scenario templates, (3) approaches based on prerequisites (pre-conditions) and consequences (post-conditions) of attacks (e.g., [4, 11]), which create attack scenarios through matching the consequence of one attack to the prerequisite of another, and (4) approaches based on multiple information sources (e.g., [13, 10, 20]), which correlate alerts from multiple security systems such as firewalls and IDSs.

Current alert correlation approaches generally assume all alert data (e.g., destination IP addresses) are available for analysis, which is true when there are no privacy concerns. However, when multiple organizations provide sanitized alert and incident data (because of privacy concerns) for intrusion analysis, alert correlation will be affected due to the lack of precise data. It is desirable to have techniques to perform privacy-preserving alert correlation such that the privacy of participating organizations is preserved, and at the same time, alert correlation can provide useful results.

[*]The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Science Foundation (NSF) under grants ITR-0219315 and CCR-0207297, and by the U.S. Army Research Office (ARO) under grant DAAD19-02-1-0219.

To our best knowledge, [7] is the only paper addressing privacy issues in alert correlation, which uses hash functions (e.g., MD5) and keyed hash functions (e.g., HMAC-MD5) to sanitize sensitive data. This approach is effective in detecting some high-volume events (e.g., worms). However, since hash functions destroy the semantics of alert attributes (e.g., the loss of topological information due to hashed IP addresses), the interpretation of correlation results is nontrivial. In addition, hash functions may be vulnerable to brute-force attacks due to limited possible values of alert attributes, and keyed hash functions may introduce difficulties in correlation analysis due to the different keys used by different organizations.

In this paper, we propose a privacy-preserving alert correlation approach based on concept hierarchies. This approach works in two phases: *entropy guided alert sanitization* and *sanitized alert correlation*. The first phase protects the privacy of sensitive alert data. We classify alert attributes into categorical (e.g., IP addresses) and continuous ones (e.g., the total time a process runs), and sanitize them through concept hierarchies. In a concept hierarchy, original attribute values are generalized to high-level concepts. For example, IP addresses are generalized to network addresses, and continuous attributes are generalized to intervals. We replace original attribute values with corresponding high-level concepts, thus introducing uncertainty while partially maintaining attribute semantics. To balance the privacy and usability requirements, alert sanitization is guided by *entropy* or *differential entropy* [3] of sanitized attributes, where the entropy or differential entropy for sanitization is determined according to the privacy policy.

To understand the security threats, the second phase of our approach is to correlate sanitized alerts. As we mentioned earlier, examining the similarity between alert attributes and building attack scenarios are two focuses in current correlation methods. We investigate both problems under the situation where alerts are sanitized. We first examine similarity functions based on original attribute values, and then show how to revise them to calculate similarity between sanitized attributes. To build attack scenarios from sanitized alerts, we propose an *optimistic approach*. As long as it is possible that two sanitized alerts have a *causal relation*, we link them together. Hence multiple alerts are connected through causal relations to form attack scenarios.

The remainder of this paper is organized as follows. Section 2 presents techniques on entropy guided alert sanitization. Section 3 discusses correlating sanitized alerts. Section 4 presents our experimental results. Section 5 discusses related work, and Section 6 concludes this paper.

2 Entropy Guided Alert Sanitization

Alert Types, Original Alerts and Sanitized Alerts. Intuitively, an alert type defines the possible attributes to describe a type of alerts. Formally, an *alert type* is a type name T and a set S of attribute names, where each attribute name $a_i \in S$ has an associated domain *Dom*(a_i). (For convenience, we may use type name T to represent either the type name or the corresponding alert type in this paper.) Original alerts are flagged directly by security systems. Formally, an *original alert* t_o of type T is a tuple on the attribute names S, where for each attribute name $a_i \in S$, the corresponding element v_i in the tuple is a value in a_i's domain *Dom*(a_i).

Example 1 *An* FTP_Glob_Expansion *alert type has a set of attribute names* {SrcIP, SrcPort, DestIP, DestPort, StartTime, EndTime} , *where the domain of* SrcIP *and* DestIP *is all possible IP addresses, the domain of* SrcPort *and* DestPort *consists of all possible port numbers, and* StartTime *and* EndTime *are possible times an alert begins and ends.*

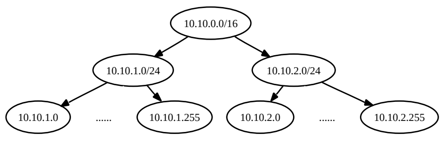
An original alert with type FTP_Glob_Expansion *is given as follows: {SrcIP=10.20.1.1, SrcPort=1042, DestIP=10.10.1.1, DestPort=21, StartTime =11-10-2004 15:45:10, EndTime =11-10-2004 15:45:10}.*

In this paper, the privacy of alerts is related to the original values of sensitive attributes in individual alerts. To ensure the privacy of individual alerts, these sensitive original values should be sanitized. A *sanitized alert* t_s with type T is a tuple on the attribute name set S, where for some attribute name $a_i \in S$, the corresponding element v_i in the tuple is a transformed value in domain $Dom_s(a_i)$ ($Dom_s(a_i)$ is *Dom*(a_i) or a different domain). To continue Example 1, assume *DestIP* of *FTP_Glob_Expansion* is sensitive. To sanitize the original alert, we let *DestIP*=$10.10.1.0/24$ (it is sanitized to its corresponding $/24$ network address). All the other attributes remain unchanged.

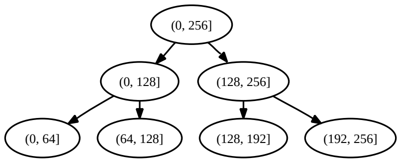
In the remainder of this paper, we may use attributes to represent either attribute names, attribute values or both if it is clear from the context. Likewise, we may use alerts to denote either original alerts, sanitized alerts, or both. In the following, we present concept hierarchy based sanitization for categorical and continuous attributes, respectively.

2.1 Categorical Attribute Sanitization

Categorical attributes have discrete values. Examples of categorical attributes are IP addresses and port numbers. Concept hierarchies abstract specific (low-level) concepts into general (high-level) ones, which are widely used in data mining. A concept hierarchy is based on *specific-general* relations. Given two concepts c_1 and c_2 (e.g., two attribute values), where c_2 is more general than c_1 (or equivalently, c_1 is more specific than c_2), we denote the specific-general relation between c_1 and c_2 as $c_1 \preceq c_2$. As a special case, we have $c \preceq c$ for any concept c. Given an attribute name with the corresponding domain, we can define specific-general relations through grouping a subset of attribute values and abstracting them into a more general concept. For example, a block of IP addresses can be organized as a subnet. Thus given an IP address $10.10.1.5$ and a subnet $10.10.1.0/24$, we have $10.10.1.5 \preceq 10.10.1.0/24$.



(a) A Concept Hierarchy for IP Addresses



(b) A Concept Hierarchy for *CPUProcessingTime*

Figure 1. Two Examples of Concept Hierarchies

A concept hierarchy consists of a set of specific-general relations, and usually is organized as a tree, where leaf nodes denote the most specific concepts (original attribute values), and the root node represents the most general concept in this hierarchy. For example, Figure 1(a) shows a concept hierarchy for IP addresses, where IP addresses from $10.10.1.0$ to $10.10.1.255$ and from $10.10.2.0$ to $10.10.2.255$ are organized into two subnets $10.10.1.0/24$ and $10.10.2.0/24$, respectively. For each attribute (e.g., source IP address), or a set of attributes having the same domain (e.g., both source and destination IP addresses), we can build a concept hierarchy based on the attribute domain. Then we perform alert sanitization by replacing original attribute values with more general values in the hierarchy.

Example 2 *To continue Example 1, assume* DestIP *of* FTP_Glob_Expansion *is sensitive. We use the concept hierarchy in Figure 1(a) to perform sanitization. We replace* DestIP=*10.10.1.1 with* DestIP=*10.10.1.0/24. The other attributes remain unchanged.*

To balance the privacy and usability of alert data, we need to design a satisfactory concept hierarchy to perform sanitization, or choose appropriate general values to replace original attribute values in a given concept hierarchy. We propose to guide these processes with *entropy* [3], an *uncertainty measure* for categorical attributes.

We start with calculating the entropy of a sanitized attribute. In a concept hierarchy for a categorical attribute, given an attribute value v, which is either an original or a generalized value, we use *Node*(v) to denote the node having value v. Given a general value v_g, we use *SubTree*(v_g) to denote the subtree rooted at *Node*(v_g), and *LeafCount*(v_g) to denote the number of leaf nodes in *SubTree*(v_g). When sanitizing a categorical attribute a, an original value v_o is replaced with a general value v_g in a concept hierarchy. Notice *Node*(v_o) should be a leaf node in *SubTree*(v_g). We denote the entropy of attribute a associated with v_g as $H_a(v_g)$, where $H_a(v_g) = -\sum_{i=1}^{LeafCount(v_g)} p(a = v_i) \log_2 p(a = v_i)$. Based on the frequencies of attribute values, we can compute attribute entropy using the above equation. For example, if all leaf nodes in *SubTree*(v_g) are equally likely to be generalized to v_g, then for any leaf node value v_i, the probability $p(a = v_i) = 1/LeafCount(v_g)$, thus we have $H_a(v_g) = \log_2 LeafCount(v_g)$. To continue Example 2 under the assumption of equal probabilities for leaf nodes, the entropy of *DestIP* associated with $10.10.1.0/24$ is $\log_2$LeafCount(10.10.1.0/24) = $\log_2 256 = 8$.

Attribute entropy can help us design a satisfactory concept hierarchy. For example, if we want to achieve an entropy value 8 when sanitizing *DestIP* distributed from $10.90.1.0$ to $10.90.1.255$ with equal probabilities, we can design a concept hierarchy with two levels, where the root node is a $/24$ network ($10.90.1.0/24$), and the leaf nodes are those individual IP addresses. Entropy can also help us choose an appropriate general value in a given concept hierarchy. For example, consider an original attribute *DestIP*=10.10.10.1 and a concept hierarchy in Figure 1(a), where leaf nodes in the hierarchy have equal probabilities. If we require an entropy value 8, we can choose the general value $10.10.1.0/24$ to sanitize the original attribute.

2.2 Continuous Attribute Sanitization

Some attributes in an alert take continuous values, for example, the CPU time a process uses. To sanitize a continuous attribute, we divide the domain of the attribute into mutually exclusive intervals, and replace the original values with the corresponding intervals. Formally, if the domain of an attribute a is $Dom(a)$, we partition $Dom(a)$ into n intervals $r_1, r_2, \cdots, r_n$ such that (1) $\cup_{k=1}^{n} r_k = Dom(a)$, and (2) for any i, j, where $1 \le i, j \le n$ and $i \ne j$, $r_i \cap r_j = \emptyset$.

The partitions of an attribute domain can be organized into a concept hierarchy. For example, Figure 1(b) shows a concept hierarchy for attribute *CPUProcessingTime* (assuming its domain is interval $(0, 256]$). Several approaches have been proposed to create concept hierarchies for continuous attributes in data mining. For example, one simple approach organizes a hierarchy into multiple levels, where each level has different number of equal-length intervals.

Example 3 *Consider a* JVM_Malfunction *alert with a sensitive attribute* CPUProcessingTime = *82.6 milliseconds. Using the concept hierarchy in Figure 1(b), we let* CPUProcessingTime = $(64, 128]$.

To design a satisfactory concept hierarchy for sanitization, or choose an appropriate interval to replace an original value in a concept hierarchy, we use *differential entropy* [3], an *uncertainty measure* for continuous attributes.

We first discuss how to compute differential entropy. When sanitizing a continuous attribute a, an original value v_o is replaced with an interval v_g that includes value v_o. The length of v_g is critical to the calculation of the attribute uncertainty. We let $Length(v_g)$ denote the difference between the upper and lower bounds of interval v_g. We denote the differential entropy of a associated with v_g as $H_a(v_g)$.

$$H_a(v_g) = -\int_{v_g} f(a) \log_2 f(a) da, \qquad (1)$$

where $f(a)$ is the probability density function for attribute a over interval v_g.

Equation 1 is derived and simplified from the standard form of differential entropy [3]. In the standard form, $H_a(Dom(a)) = -\int_{Dom(a)} f_o(a) \log_2 f_o(a) da$, where $f_o(a)$ is the probability density function over attribute domain $Dom(a)$. Under our sanitization technique, although we cannot know the exact value of attribute a, we are certain that it is in interval v_g, where v_g may be a part of $Dom(a)$. Then we know that the probability density function $f(a)$ is 0 outside interval v_g. Thus the integration in Equation 1 only needs to be performed over v_g [1].

Based on Equation 1, we can compute differential entropy for sanitized attributes where their original values are in different distributions. As an example, we derive a formula for uniformly distributed attributes. The original attributes in other distributions can be computed in a similar way. Assume an attribute a is in uniform distribution and is sanitized to interval $[\alpha, \beta]$. Thus its probability density function $f(a)$ is $1/(\beta - \alpha)$ when $\alpha \leq a \leq \beta$; otherwise $f(a) = 0$. Based on Equation 1, we have $H_a(v_g) = -\int_{\alpha}^{\beta} f(a) \log_2 f(a) da = \log_2(\beta - \alpha) = \log_2 Length(v_g)$.

This equation tells us that differential entropy can be greater than, equal to, or less than 0. Consider a random variable X uniformly distributed over an interval with length 1. For a sanitized continuous attribute, if its differential entropy is greater than 0, then its uncertainty is greater than variable X; if its differential entropy is equal to 0, its uncertainty is equal to X; otherwise its uncertainty is less than X. As noted by Shannon [16], an important difference between the differential entropy and the entropy for categorical attributes is that differential entropy is "relative to the coordinate system". In other words, if the measurement units are changed (e.g., from milliseconds to seconds), differential entropy values may also change. To continue Example 3, further assume attribute *CPUProcessingTime* is uniformly distributed in interval $(64, 128]$. The differential entropy of *CPUProcessingTime* associated with $(64, 128]$ is $\log_2(128 - 64) = 6$.

The differential entropy can help design a satisfactory concept hierarchy. For example, assume the domain of an attribute is $[0, 64]$ with uniform distribution. If we require a differential entropy value 5, we can build a concept hierarchy with two levels, where the root node is $[0, 64]$, and there are two leaf nodes $[0, 32]$ and $(32, 64]$. The differential entropy can also help us choose an appropriate interval to replace an original value. For example, consider an original attribute *CPUProcessingTime*=82.6 milliseconds and a concept hierarchy in Figure 1(b). Assume attributes are in uniform distribution. If we require a differential entropy value 6 for sanitization, we can choose $(64, 128]$ to replace the original value.

[1] To let the probability density function $f(a)$ satisfy $\int_{v_g} f(a)da = 1$, $f(a)$ can be derived from $f_o(a)$. Assume $\int_{v_g} f_o(a)da = q \leq 1$. We can let $f(a) = f_o(a)/q$ in interval v_g; otherwise $f(a) = 0$. Another method to get $f(a)$ is to compute the distribution parameters, which is straightforward for uniformly distributed attributes.

3 Correlation Analysis of Sanitized Alerts

The second phase of our approach is sanitized alert correlation. As we stated in the Introduction, examining the similarity between alert attributes and building attack scenarios are two focuses in current correlation approaches. In Subsections 3.1 and 3.2, we discuss how to compute the similarity between sanitized attributes and building attack scenarios for sanitized alerts, respectively.

3.1 Similarity between Sanitized Attributes

Sanitized Categorical Attributes. Several functions or heuristics (e.g., techniques in [19, 17]) have been proposed to calculate the similarity between (original) attribute values. Here we first give a simple heuristic, and then discuss how to revise this heuristic to calculate the similarity between sanitized categorical attributes. Other simple heuristics can be revised using a similar approach.

If two original attributes x_o and y_o are known, we give a similarity function between them as follows.

$$Sim(x_o, y_o) = \begin{cases} 1, & \text{if } x_o = y_o, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

After sanitization, x_o and y_o become generalized values x_g and y_g, respectively. There are several ways to compute the similarity between x_g and y_g. For example, we can treat the sanitized attributes as the original ones, and use Equation 2 to compute their similarity. This is a coarse-grained similarity measurement because even if the sanitized values are the same, their corresponding original values may be different. We propose to compute their similarity by estimating the probability that x_g and y_g have the same original value. Intuitively, in a concept hierarchy, two nodes *Node*(x_g) and *Node*(y_g) are possible to have the same original value only if they are in the same path from the root to a leaf node (*Node*(x_g) and *Node*(y_g) may be the same). In other words, there is a specific-general relation between x_g and y_g. If the probability that x_g and y_g have the same original value is large, we interpret it as a high similarity between them; otherwise their similarity is low.

Now we show how to compute the probability that x_g and y_g have the same original value. To simplify our discussion, we assume leaf node values in the concept hierarchy have equal probabilities. Using probability theory, the revised similarity function based on Equation 2 is as follows.

$$Sim(x_g, y_g) = \begin{cases} \frac{1}{LeafCount(x_g)}, & \text{if } y_g \preceq x_g, \\ \frac{1}{LeafCount(y_g)}, & \text{if } x_g \preceq y_g, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where "$\preceq$" denotes specific-general relations.

When the leaf node values in a concept hierarchy are not evenly distributed, to compute the similarity value for x_g and y_g, we can first compute the probability for each original value based on attribute frequencies, then calculate the similarity value based on x_g, y_g, the concept hierarchy, and the probability for each leaf node.

Sanitized Continuous Attributes. The similarity function between continuous attributes is different from that of categorical attributes due to various reasons. For example, due to the clock drift, two *CPUProcessingTime* may not be reported the same even if their actual time is the same. Considering these situations, here we first give a simple similarity function as follows. (Other similarity functions are possible and may be revised in a similar way to our approach.)

$$Sim(x_o, y_o) = \begin{cases} 1, & \text{if } |x_o - y_o| \leq \lambda, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where x_o and y_o are original attribute values, and λ is a predefined threshold. For example, if the difference between two *CPUProcessingTime* is less than 5 milliseconds, we say their similarity is 1.

When x_o and y_o are generalized to intervals x_g and y_g, respectively, there are several ways to compute the similarity between x_g and y_g. For example, assuming $Length(x_g) = Length(y_g) > \lambda$, their similarity is 1 if $x_g = y_g$, and 0 otherwise. This certainly is a rough, imprecise estimation, because even if x_g and y_g are not the same interval, it is possible that the difference between their original values is less than λ. Similar to the categorical case, we propose to compute their similarity by estimating the probability that the difference between their original values is within threshold λ.

To simplify our discussion, suppose that original values of x_g and y_g are independent and uniformly distributed over intervals x_g and y_g, respectively, and we also assume *Length*(x_g) = *Length*(y_g) $> \lambda$. More sophisticated cases such as *Length*(x_g) $\neq$ *Length*(y_g) can be covered by an approach similar to the following calculation. We notice the difference between two original values may be within λ only if x_g and y_g fall into any of the following two cases. (1) x_g and y_g are the same interval, or (2) for x_g and y_g, the difference between the lower bound of the higher interval and the upper bound of the lower interval is within λ. Intuitively, this second case means x_g and y_g either are adjacent intervals (e.g., $[0, 5]$ and $(5, 10]$), or there is a small "gap" between them (e.g., $[0, 5]$ and $[6, 11]$).

Using probability theory, the revised similarity function based on Equation 4 is as follows.

$$Sim(x_g, y_g) = \begin{cases} \frac{2\lambda[Length(x_g)] - \lambda^2}{[Length(x_g)]^2}, & \text{if } x_g = y_g, \\ \frac{(\lambda - d)^2}{2[Length(x_g)]^2}, & \text{if } 0 \leq d \leq \lambda, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where d is the difference between the lower bound of the higher interval and the upper bound of the lower interval. Note that similarity computation based on Equation 5 is symmetric ($Sim(x_g, y_g) = Sim(y_g, x_g)$).

We notice that in the probability computation, we have taken several assumptions such as *Length*(x_g)= *Length*(y_g)$> \lambda$ to simplify our calculation. However, the essential steps involved in the probability computation have been demonstrated. More sophisticated cases can be covered by a similar approach.

3.2 Building Attack Scenarios

An attack scenario is a sequence of steps adversaries performed to attack victim machines. The essence of creating attack scenarios from security alerts is to discover causal relations between individual attacks. For example, there is a causal relation between an earlier *SCAN_NMAP_TCP* attack and a later *FTP_Glob_Expansion* attack if the earlier one is used to probe a vulnerable ftp service for the later one.

We extend a previous correlation method [11], which targets at building attack scenarios from original alerts, to build attack scenarios from sanitized alerts. In the following, we first give an overview of that correlation method with a slight modification, which simplifies our discussion without losing the essence of the previous method.

A Previous Method for Building Attack Scenarios [11]. The correlation method in [11] models each attack type through specifying its prerequisite and consequence, where the prerequisite is the necessary condition to launch the attack successfully, and the consequence is the possible outcome if the attack succeeds. Prerequisites and consequences are modeled by predicates. For example, the consequence of a port scanning attack may be *ExistService(DestIP, DestPort)*, denoting that an open port *DestPort* is found on host *DestIP*. Formally, given an alert type T, the prerequisite of T is a logical combination of predicates, and the consequence of T is a set of predicates, where the variables in the predicates are attribute names in type T.

Example 4 *Consider alert types* T_1=SCAN_NMAP_TCP *and* T_2=FTP_Glob_Expansion. T_1*'s prerequisite is* ExistHost(DestIP), *and* $\{$ExistService(DestIP,DestPort)$\}$ *is its consequence.* T_2*'s prerequisite is* ExistService(DestIP, DestPort) $\wedge$ VulnerableFtpRequest(DestIP), *and its consequence is* $\{$GainAdminAccess(DestIP)$\}$.

Given a type T alert t, the prerequisite and consequence of t can be obtained through instantiating T's prerequisite and consequence using t's attribute values and timestamps. We model causal relations between alerts (i.e., detected attacks) as *prepare-for* relations. Intuitively, an earlier alert t_1 *prepares for* a later alert t_2 if the consequence of t_1 can contribute to the prerequisite of t_2. Formally, t_1 *prepares for* t_2 if and only if (1) one of the instantiated predicates in t_1's consequence implies one of the instantiated predicates in t_2's prerequisite, and (2) t_1.EndTime $<$ t_2.StartTime.

Example 5 *To continue Example 4, consider a type* T_1 *alert* t_1 *and a type* T_2 *alert* t_2. *Assume that* t_1

and t_2 *both have* DestIP=*10.10.1.1 and* DestPort=*21,* t_1*'s* EndTime *is 11-15-2004 20:15:10, and* t_2*'s* StartTime *is 11-15-2004 20:15:15. Through predicate instantiation,* t_1*'s consequence is* {ExistService*(*10.10.1.1, 21*)*}*,* t_2*'s prerequisite is* ExistService*(*10.10.1.1, 21*)* $\wedge$ VulnerableFtpRequest*(*10.10.1.1*). Notice* t_1*.EndTime* $<$ t_2*.StartTime. Then we know* t_1 *prepares for* t_2*.*

Alert correlation graphs are used to represent the attack scenarios discovered through alert correlation. Formally, an alert correlation graph is a directed graph (N, E), where each node $n \in N$ is an alert, and each directed edge $(n_1, n_2) \in E$ represents that n_1 *prepares for* n_2. For convenience, we may use causal relations and *prepare-for* relations interchangeably in this paper. Given two alerts t_1 and t_2, where t_1 *prepares for* t_2, we call t_1 the preparing alert, and t_2 the prepared alert.

Optimistic Approach to Building Attack Scenarios from Sanitized Alerts. We notice that identifying *prepare-for* relations between alerts is essential to building attack scenarios. However, after alert sanitization, we may not be certain whether *prepare-for* relations are satisfied if sanitized attributes are involved. Without loss of generality, we assume alert type data is not sanitized. We propose an *optimistic* approach to identifying *prepare-for* relations between sanitized alerts. This approach identifies a *prepare-for* relation between two alerts t_1 and t_2 as long as it is possible that (1) one of the instantiated predicates in t_1's consequence *may* imply one of the instantiated predicates in t_2's prerequisite, and (2) t_1 and t_2's timestamps *may* satisfy t_1.EndTime $<$ t_2.StartTime. In other words, based on sanitized attributes, we "guess" what possible original values are, and if these original values have a chance to satisfy the implication relationship between instantiated predicates, and also satisfy the timestamp requirement, we identify a *prepare-for* relation. Example 6 illustrates this idea.

Example 6 *To continue Examples 4 and 5, assume* DestIP *of alerts* t_1 *and* t_2 *are sanitized based on the concept hierarchy in Figure 1(a), where* DestIP=10.10.1.1 *is replaced with* DestIP=10.10.1.0/24*. So* t_1*'s consequence becomes* {ExistService*(*10.10.1.0/24, 21*)*}*, and* t_2*'s prerequisite is* ExistService*(*10.10.1.0/24, 21*)* $\wedge$ VulnerableFtpRequest *(*10.10.1.0/24*). It is possible that the instantiated predicate* ExistService*(*10.10.1.0/24, 21*) in* t_1*'s consequence implies the instantiated predicate* ExistService*(*10.10.1.0/24, 21*) in* t_2*'s prerequisite if both sanitized* DestIP *attributes have the same original IP address in network* 10.10.1.0/24*. Further due to* t_1*.EndTime* $<$ t_2*.StartTime, we identify a prepare-for relation between* t_1 *and* t_2*.*

Attack Scenario Refinement Based on Probabilities of Prepare-for Relations. Our optimistic approach certainly may introduce false *prepare-for* relations between alerts. Without knowledge of original values, we cannot guarantee that one instantiated predicate implies another if sanitized attributes are involved. To improve this approach, it is desirable to estimate how possible each pair of sanitized alerts has a *prepare-for* relation. To do so, we can first compute the probability that one instantiated predicate implies another, and then consider timestamp requirement.

Example 7 *To continue Example 6, consider* ExistService(DestIP,DestPort) *in* T_1*'s consequence and* T_2*'s prerequisite. After predicate instantiation using sanitized alerts, we compute probabilities* $P(t_1$*.DestIP=*t_2*.DestIP)=*$\frac{1}{256}$*, and* $P(t_1$*.DestPort=*t_2*.DestPort)* =1*. Hence the probability that the instantiated predicate* ExistService*(*10.10.1.0/24, 21*) in* t_1*'s consequence implies the instantiated predicate* ExistService*(*10.10.1.0/24, 21*) in* t_2*'s prerequisite is* $\frac{1}{256}$*. Further note* $P(t_1$*.EndTime*$<$ t_2*.StartTime)=1. Then we know the probability of this prepare-for relation to be true is* $\frac{1}{256}$*.*

Notice that between two alerts, sometimes there may exist several pairs of instantiated predicates such that in each pair, one instantiated predicate may imply the other. It is difficult to estimate the probability that at least one implication relationship is true because we do not know the dependency among them. To simplify the probability estimation, we assume n pairs of instantiated predicates that may have implication relationships are independent with probabilities p_1, p_2, $\cdots$, p_n, respectively. Then the probability that at least one implication relationship is satisfied is $1 - (1 - p_1)(1 - p_2) \cdots (1 - p_n)$. Next we consider timestamp requirement to further compute the probability for the *prepare-for* relation.

After the probabilities of *prepare-for* relations are computed, it is desirable to use these probability values to prune false *prepare-for* relations in an alert correlation graph (e.g., remove *prepare-for* relations with lower probabilities). However, we observe that this ideal case may not help much. As shown in Example 7, after sanitizing IP addresses to $/24$ network addresses, the probability that two alerts have a *prepare-for* relation is only $\frac{1}{256}$, which may imply that this *prepare-for* relation is *false*. However, considering that when the IP addresses in a $/24$ network are sanitized, the probabilities of all *prepare-for* relations involving these IP addresses are small. If we remove all the low-probability *prepare-for* relations, it is very likely that some *true prepare-for* relations are pruned.

We further observe that if we calculate the probability for a set of *prepare-for* relations instead of only one, we can gain more interesting hints. Assume n pairs of *prepare-for* relations have probabilities $p_1, p_2, \cdots, p_n$, respectively. Further suppose they are independent. Thus the probability that at least one *prepare-for* relation is *true* is $1-(1-p_1)(1-p_2)\cdots(1-p_n)$. This result may help us refine an alert correlation graph.

To further refine an alert correlation graph constructed from the optimistic approach, we propose to aggregate alert correlation graphs. This is performed according to *temporal constraints* and probability thresholds. Consider a set S of alerts and a time interval with length δ (e.g., 6 seconds), where alerts in S are sorted in increasing order

```
Algorithm 1. Aggregation to an alert correlation graph.
Input: An alert correlation graph $CG = (N, E)$, a temporal
       constraint $\delta$, and a probability threshold $\theta$.
Output: An aggregated correlation graph $ACG$.
Method:
  1. Partition edge set $E$ into subsets $E_1$,$E_2$, $\cdots$, $E_l$ such
  that in any $E_i$ ($1 \leq i \leq l$), all edges have the same preparing
  alert type, and the same prepared alert type.
  2. For each subset $E_i$ in $E$
  3.     Further partition $E_i$ into groups $E_{i1}$, $E_{i2}$, $\cdots$, $E_{ij}$ such
         that the preparing alerts and prepared alerts in $E_{ik}$
         ($1 \leq k \leq j$) satisfy temporal constraint $\delta$, respectively.
  4.     For each group $E_{ik}$ in subset $E_i$
  5.         Compute the probability $\mathcal{P}$ that at least one
             prepare-for relation in $E_{ik}$ is true.
  6.         If $\mathcal{P} \geq \theta$ Then
  7.             Aggregate edges in $E_{ik}$ into one; merge preparing
                 and prepared alerts, respectively.
  8.         Else Remove all edges in $E_{ik}$.
                 Remove preparing and prepared alerts in $E_{ik}$ if
                 they are not linked by other edges.
  9. Let $CG$ after the above operations be $ACG$. Output $ACG$.
```

Figure 2. Aggregating alert correlation graphs

based on *StartTime*. We call two alerts *consecutive alerts* if their *StartTime* timestamps are neighboring to each other in S. S satisfies temporal constraint δ if and only if for any two consecutive alerts t_i and t_j in S where t_i.StartTime $\leq$ t_j.StartTime, t_j.StartTime $-t_i$.EndTime $\leq \delta$. Intuitively, this means the time intervals (in the form of [*StartTime*, *EndTime*]) of any two consecutive alerts overlap, or the "gap" between them is within δ.

Given an alert correlation graph $CG = (N, E)$ constructed from the optimistic approach, a temporal constraint δ, and a probability threshold θ, we perform aggregation to CG through the algorithm shown in Figure 2. The basic idea is that we aggregate the edges with the same preparing and the same prepared alert types into one such that the probability that at least one *prepare-for* relation (represented by these edges) is *true* is greater than or equal to threshold θ. (The related nodes are merged accordingly.)

As we stated earlier, the alert correlation graphs constructed from our optimistic approach may include both *false* and *true prepare-for* relations. They may also have large numbers of nodes and edges such that understanding these scenarios can be time-consuming. Algorithm 1 helps us improve the quality of alert correlation graphs in that it reduces the numbers of nodes and edges, and may improve the certainty about *prepare-for* relations (in the aggregated sense). Note that after aggregation, a node in the aggregated correlation graph is actually a place holder which may represents multiple alerts. Our aggregation also has some limitations because we may remove some *true prepare-for* relations from alert correlation graphs when the probability for them is less than the threshold. In our future work, we will investigate additional techniques to refine alert correlation graphs to reduce both false alerts and false *prepare-for* relations in the graphs.

Table 1. Evaluating similarity functions

	Categorical	Continuous
R_{cc} for "similar" pairs	100%	100%
R_{mc} for "similar" pairs	5.88%	9.95%
R_{cc} for "distinct" pairs	94.12%	90.05%
R_{mc} for "distinct" pairs	0%	0%

4 Experimental Results

4.1 Evaluating Similarity Functions

We first evaluate the revised similarity functions (Equations 3 and 5). We are interested in how possible sanitized datasets can provide similarity classification as that from original datasets. In our experiments, we randomly generated a set S_o of alerts with only one categorical (or continuous) attribute, and then sanitized it to get a new set S_s. For each pair of alerts in S_o, we used Equation 2 (or Equation 4, resp.) to calculate attribute similarity. While for each pair of alerts in S_s, we used Equation 3 (or Equation 5, resp.) to compute their similarity. Then we applied an optimistic classification. If the similarity value is greater than 0, we classify this pair of alerts as "similar" pair; otherwise we classify them as "distinct" pair. We compared the results from S_s with those from S_o. We used two quantitative measures: *correct classification rate* R_{cc} for S_s based on S_o and *misclassification rate* R_{mc} for S_s based on S_o. We define R_{cc} and R_{mc} for "similar" pairs as follows. $R_{cc} = \frac{\#\text{common "similar" pairs in both } S_o \text{ and } S_s}{\#\text{"similar" pairs in } S_o}$, and $R_{mc} = \frac{\#\text{"similar" pairs in } S_s - \#\text{common "similar" pairs in } S_o \text{ and } S_s}{\#\text{total alert pairs} - \#\text{"similar" pairs in } S_o}$. Note that R_{cc} and R_{mc} are only for sanitized datasets, and both measures can be computed for "similar" or "distinct" pairs. Likewise, we define correct classification rate and misclassification rate for "distinct" pairs by replacing "similar" with "distinct" in the above equations.

Our first experiment is for categorical attributes. We generated a set S_o of $2,560$ alerts with *DestIP* attributes uniformly distributed over 256 IP addresses in network $10.60.1.0/24$ (from $10.60.1.0$ to $10.60.1.255$). Next we partitioned this network into 16 subnets. Each subnet ($/28$ subnet) has 16 addresses. We sanitized S_o to S_s such that *DestIP* of each alert is generalized to the corresponding $/28$ subnet ID. We applied Equation 2 to S_o and Equation 3 to S_s. The results are shown in the left part of Table 1.

Our second experiment is for continuous attributes. We generated a set S_o of $1,000$ alerts with *CPUProcessingTime* attributes uniformly distributed over interval $[0, 100]$. Then we divided $[0, 100]$ into 20 small equal-length intervals (the length of each small interval is 5). Next we sanitized S_o to S_s by replacing original values with the corresponding small intervals (a boundary value between two adjacent intervals is put into the lower interval). Let $\lambda = 2.5$. We applied Equation 4 to S_o and Equation 5 to S_s. The results are shown in the right part of Table 1.

In these two experiments, the entropy and differential entropy for attributes *DestIP* and *CPUProcessingTime* are $log_2 16 = 4$ and $log_2 5 = 2.3219$, respectively. Our correct classification rates for both "similar" and "distinct" pairs are high (greater than 90%), while the misclassification rates for both pairs are low (less than 10%). This demonstrates that the privacy of alert attributes can be protected with sacrificing the data functionality (similarity classification) slightly.

4.2 Building Attack Scenarios

To evaluate the techniques on building attack scenarios, we performed experiments on 2000 DARPA intrusion detection scenario specific data sets [8]. The datasets include two scenarios: LLDOS 1.0 and LLDOS 2.0.2, where each scenario includes two parts (inside and DMZ).

In the first set of experiments, our goal is to evaluate the effectiveness of our optimistic approach to building attack scenarios. We first used RealSecure network sensor 6.0 to generate alerts from four datasets: LLDOS 1.0 inside, LLDOS 1.0 DMZ, LLDOS 2.0.2 inside, and LLDOS 2.0.2 DMZ. The prerequisites and consequences for all alert types can be found in [12] (Tables III and IV). Due to space constraint, we do not list them here. We first constructed alert correlation graphs for the original alert datasets using the previous method [11]. Then we sanitized the destination IP address of each alert (a sanitization policy used by DShield) by replacing it with its corresponding $/24$ network ID. We applied our optimistic approach to building alert correlation graphs for the four datasets. To save space, here we only list one alert correlation graph in Figure 3.

In Figure 3, the string inside each node is an alert type followed by an alert ID. Notice that to show the difference between the alert correlation graphs created from the original dataset and the sanitized one, we marked the additional nodes obtained only from the sanitized dataset in gray. From Figure 3, it is clear that the alert correlation graph from the sanitized dataset is a supergraph of the one from the original dataset. This is because our optimistic approach identifies *prepare-for* relations even if the related probabilities are low. Figure 3 represents a multi-stage attack scenario, which is consistent with the major steps adversaries performed.

We notice that false alerts may be involved in an alert correlation graph (e.g., alert *Email_Debug67705* in Figure 3). To further evaluate the effectiveness of our approach, similar to [11], we used two quantitative measures: *soundness* M_s and *completeness* M_c, where $M_s = \frac{\#\text{correctly correlated alerts}}{\#\text{correlated alerts}}$, and $M_c = \frac{\#\text{correctly correlated alerts}}{\#\text{related alerts}}$. We computed both measures for the correlation approaches based on original datasets and the sanitized ones. The results are in Table 2. Table 2 shows the correlation approach based on original datasets is slightly better than our optimistic approach, which is reasonable because original datasets are more precise than sanitized datasets. Nevertheless, our optimistic approach is relatively good: the majority of soundness measures are greater than 70%, and all completeness measures are greater than 60%.

Table 2. Soundness and completeness

	LLDOS 1.0		LLDOS 2.0.2	
	Inside	DMZ	Inside	DMZ
M_s (original)	93.18%	94.74%	92.31%	100%
M_s (sanitized)	70.69%	85.71%	48.00%	83.33%
M_c (original)	93.18%	94.74%	66.67%	62.50%
M_c (sanitized)	93.18%	94.74%	66.67%	62.50%

In the second set of experiments, our goal is to verify whether correlation methods can help us differentiate between true and false alerts. We conjecture that correlated alerts are more likely to be true alerts, and false alerts have less chance to be correlated. This conjecture has been experimentally verified in [11] when original alerts are available. We try to see the results when alerts are sanitized. Similar to [11], we compute detection rate as $\frac{\#\text{detected attacks}}{\#\text{observable attacks}}$, and false alert rate as $1 - \frac{\#\text{true alerts}}{\#\text{alerts}}$. We calculated detection rates and false alert rates for RealSecure network sensor, the correlation approach based on original datasets, and the correlation approach (our optimistic approach) based on sanitized datasets. The results are shown in Table 3. In Table 3, the numbers of alerts for correlation approaches are the numbers of correlated alerts. We observe that our optimistic approach still has the ability to greatly reduce false alert rates, while slightly sacrificing detection rates. In addition, comparing the detection rates and false alert rates, the approach based on original datasets is slightly better than our optimistic approach since original datasets have more precise information than sanitized ones.

In the third set of experiments, our goal is to evaluate the effectiveness of the aggregation to alert correlation graphs. Due to space constraint, we only show one case for LLDOS 1.0 inside dataset. We aggregated the alert correlation graph in Figure 3, where we set temporal constraint $\delta = \infty$ and probability threshold $\theta = 0.1$. The result is shown in Figure 4. In Figure 4, we notice that some false alerts are ruled out (e.g., *Email_Debug67705*), which is highly preferable. However, we also observe that some true alerts are pruned (e.g., three *Sadmind_Ping* alerts), which is undesirable. Though it is possible to mitigate this undesirable case through setting a lower probability threshold, we can never guarantee that only false alerts will be ruled out. Thus the aggregation should be applied with caution. The alert correlation graphs created from the optimistic approach and the aggregated correlation graphs are complementary to each other, and they should be referred to each other to better understand the security threats.

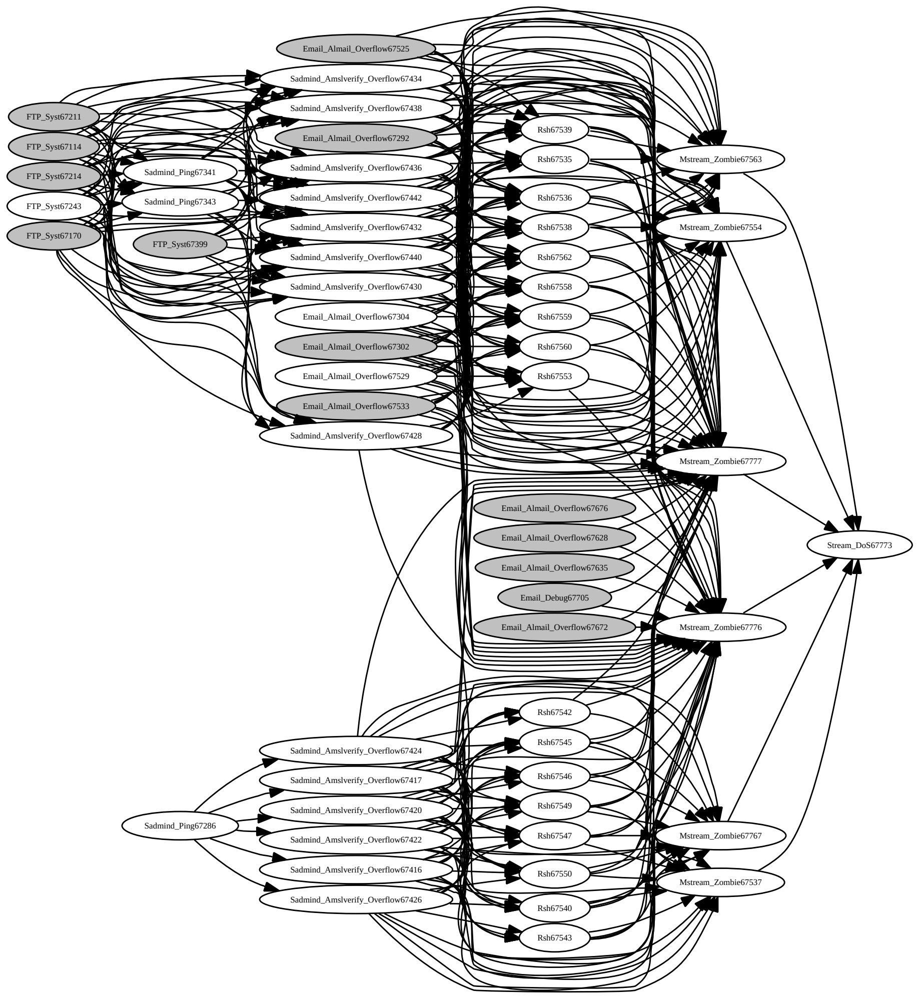

Email_Almail_Overflow67525
Sadmind_Amslverify_Overflow67434
Sadmind_Amslverify_Overflow67438
Email_Almail_Overflow67292
FTP_Syst67211
FTP_Syst67114
FTP_Syst67214
FTP_Syst67243
FTP_Syst67170
Sadmind_Ping67341
Sadmind_Ping67343
FTP_Syst67399
Sadmind_Amslverify_Overflow67436
Sadmind_Amslverify_Overflow67442
Sadmind_Amslverify_Overflow67432
Sadmind_Amslverify_Overflow67440
Sadmind_Amslverify_Overflow67430
Email_Almail_Overflow67304
Email_Almail_Overflow67302
Email_Almail_Overflow67529
Email_Almail_Overflow67533
Sadmind_Amslverify_Overflow67428
Rsh67539
Rsh67535
Rsh67536
Rsh67538
Rsh67562
Rsh67558
Rsh67559
Rsh67560
Rsh67553
Mstream_Zombie67563
Mstream_Zombie67554
Mstream_Zombie67777
Stream_DoS67773
Email_Almail_Overflow67676
Email_Almail_Overflow67628
Email_Almail_Overflow67635
Email_Debug67705
Email_Almail_Overflow67672
Mstream_Zombie67776
Rsh67542
Rsh67545
Rsh67546
Rsh67549
Rsh67547
Rsh67550
Rsh67540
Rsh67543
Sadmind_Amslverify_Overflow67424
Sadmind_Amslverify_Overflow67417
Sadmind_Amslverify_Overflow67420
Sadmind_Amslverify_Overflow67422
Sadmind_Amslverify_Overflow67416
Sadmind_Amslverify_Overflow67426
Sadmind_Ping67286
Mstream_Zombie67767
Mstream_Zombie67537

Figure 3. An alert correlation graph in LLDOS 1.0 inside dataset

Table 3. Detection rates and false alert rates in our experiments

	Detection approach	LLDOS 1.0		LLDOS 2.0.2	
		Inside	DMZ	Inside	DMZ
# alerts	RealSecure	922	891	489	425
	Correlation for original datasets	44	57	13	5
	Correlation for sanitized datasets	58	63	25	6
Detection rate	RealSecure	61.67%	57.30%	80.00%	57.14%
	Correlation for original datasets	60.00%	56.18%	66.67%	42.86%
	Correlation for sanitized datasets	60.00%	56.18%	66.67%	42.86%
False alert rate	RealSecure	95.23%	93.60%	96.73%	98.59%
	Correlation for original datasets	6.82%	5.26%	23.08%	40.00%
	Correlation for sanitized datasets	29.31%	14.29%	60.00%	50.00%

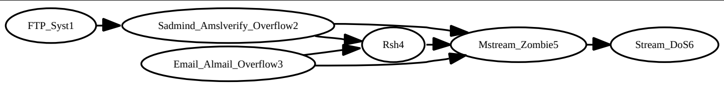


Figure 4. Aggregation to the alert correlation graph in Figure 3

5 Related Work

To our best knowledge, [7] is the only paper that explicitly addresses privacy issues in alert correlation. This approach is complementary to ours. DShield lets audit log submitters perform partial or complete obfuscation to destination IP addresses to sanitize sensitive information. Our approach can be considered an extension to the DShield approach; the sanitization process in our approach is guided by the desirable entropy, which can be determined by the privacy policy, and thus leaves maximum allowable information for further analysis.

Our work is also closely related to the k-*Anonymity* approaches [15, 18] where an entity's information may be released only if there exist at least $k - 1$ other entities in the released data that are indistinguishable from this entity. These approaches also apply generalization hierarchies to help obfuscate attributes, where k is the pre-defined parameter to control the generalization process. Our approach differs in that we use entropy to control the attribute sanitization as well as to help design satisfactory concept hierarchies. Moreover, we also study methods to correlate sanitized alerts.

We notice that several other techniques may also be used to protect the privacy of alerts, such as data perturbation techniques [14, 6] used in statistical databases [1], and privacy-preserving data mining techniques [2].

6 Conclusion and Future Work

In this paper, we proposed a concept hierarchy based approach for privacy-preserving alert correlation. It works in two phases. The first phase is entropy guided alert sanitization. We sanitize sensitive attributes through concept hierarchies, where original attribute values are generalized to high-level concepts to introduce uncertainty into the datasets, and also partially maintain attribute semantics. We further proposed to use entropy and differential entropy to measure the uncertainty of sanitized attributes, and also guide the generalization of original attributes. The second phase is sanitized alert correlation, where we focus on defining similarity functions between sanitized attributes and building attack scenarios from sanitized alerts.

There are several future research directions. One of the focuses in this paper is to define similarity functions for sanitized attributes. Our results are still preliminary. We are not clear how to get new similarity functions if the heuristics between original attributes are very complex. We notice that alert correlation graphs constructed from our optimistic approach may include false prepare-for relations, we will investigate how to further refine them in our future work.

References

[1] N. Adam and J. Wortmann. Security-control methods for statistical databases: A comparison study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, May 2000.

[3] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[4] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.

[5] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Recent Advances in Intrusion Detection*, LNCS 2212, pages 85 – 103, 2001.

[6] C. Liew, U. Choi, and C. Liew. A data distortion by probability distribution. *ACM Transactions on Database Systems*, 10(3):395–411, September 1985.

[7] P. Lincoln, P. Porras, and V. Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *Proceedings of 13th USENIX Security Symposium*, August 2004.

[8] MIT Lincoln Lab. 2000 DARPA intrusion detection scenario specific datasets. `http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html`, 2000.

[9] B. Morin and H. Debar. Correlation of intrusion symptoms: an application of chronicles. In *Proceedings of the 6th International Conference on Recent Advances in Intrusion Detection (RAID'03)*, September 2003.

[10] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2D2: A formal data model for IDS alert correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, pages 115–137, 2002.

[11] P. Ning, Y. Cui, and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 245–254, Washington, D.C., November 2002.

[12] P. Ning and D. Xu. Hypothesizing and reasoning about attacks missed by intrusion detection systems. *ACM Transactions on Information and System Security*, 7(4):591–627, November 2004.

[13] P. Porras, M. Fong, and A. Valdes. A mission-impact-based approach to INFOSEC alarm correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, pages 95–114, 2002.

[14] S. Reiss. Practical data-swapping: The first steps. *ACM Transactions on Database Systems*, 9(1):20–37, March 1984.

[15] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, Computer Science Laboratory, SRI International, 1998.

[16] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July 1948.

[17] S. Staniford, J. Hoagland, and J. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

[18] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, October 2002.

[19] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, pages 54–68, 2001.

[20] V. Yegneswaran, P. Barford, and S. Jha. Global intrusion detection in the domino overlay system. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS'04)*, February 2004.