A Flexible Approach to Intrusion Alert Anonymization and Correlation

Dingbang Xu and Peng Ning
Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
Email: {dxu, pning}@ncsu.edu

Abstract

Intrusion alert data sets are critical for security research such as alert correlation. However, privacy concerns about the data sets from different data owners may prevent data sharing and investigation. It is always desirable and sometimes mandatory to anonymize sensitive data in alert sets before they are shared and analyzed. To address privacy concerns, in this paper we propose three schemes to flexibly perform alert anonymization. These schemes are closely related but can also be applied independently. In Scheme I, we generate artificial alerts and mix them with original alerts to help hide original attribute values. In Scheme II, we further map sensitive attributes to random values based on concept hierarchies. In Scheme III, we propose to partition an alert set into multiple subsets and apply Scheme II in each subset independently. To evaluate privacy protection and guide alert anonymization, we define local privacy *and* global privacy*, and use* entropy *to compute their values. Though we emphasize alert anonymization techniques in this paper, to examine data usability, we further perform correlation analysis for anonymized data sets. We focus on estimating similarity values between anonymized attributes and building attack scenarios from anonymized data sets. Our experimental results demonstrated the effectiveness of our techniques.*

1. Introduction

To defend against large-scale distributed attacks such as worms and distributed denial of service (DDoS) attacks, it is usually desirable to deploy security systems such as intrusion detection systems (IDSs) over the Internet, monitor different networks, collect security related data, and perform analysis to the collected data to extract useful information. In addition, different organizations, institutions and users may also have the willingness to share their data for security research as long as their privacy concerns about the data can be fully satisfied. For example, DShield [27] collects firewall logs from different users to learn global cyber threats, and Department of Homeland Security sponsors PREDICT [19] project to create a repository collecting network operational data for cyber security research.

Data generated by security systems may include sensitive information (e.g., IP addresses of compromised servers) that data owners do not want to disclose or share with other parties, where sensitive data are decided by data owners' privacy policy. It is always desirable and sometimes mandatory to anonymize sensitive data before they are shared and correlated. To address this problem, existing approaches [13, 29] usually perform transformation to sensitive data. Fox example, Lincoln et al. [13] propose to use hash functions and keyed hash functions to anonymize sensitive attributes such as IP addresses. Their approach is effective on detecting high-volume events, but may have limitations on alert correlation if different keys are introduced in keyed-hashing. Xu and Ning [29] propose to abstract original values to more general values (e.g., IP addresses are replaced by network addresses). Since general values may usually take different formats compared with original values (i.e., they have different attribute domains), observing this may let malicious users immediately realize that attributes are sanitized, which may infer organizations' privacy policy.

In this paper, we address this problem in another complementary direction. We start to hide original sensitive values through injecting more data into data sets. We also perform transformation to sensitive attributes, but this is carried over the same attribute domains. In this paper, we propose three schemes to anonymize sensitive attributes of intrusion alerts. These schemes are closely related and may also be applied independently. In Scheme I, we intentionally generate artificial alerts and mix them with original alerts, thus given any alert in the mixed set, it is not clear that this alert is original (i.e., IDS-reported) or artificial, which means its attributes may or may not be real. With both privacy and usability requirements in mind, during artificial alert genera-

tion, we preserve frequency distributions of attack types and non-sensitive attributes, while use concept hierarchies to facilitate the generation of sensitive attributes. A concept hierarchy can help us abstract attribute values, for example, IP addresses can be generalized to the corresponding network addresses. In Scheme II, we propose to map original sensitive attributes to random values based on concept hierarchies. And in Scheme III, we propose to partition an alert set into multiple subsets based on time constraints and perform Scheme II independently for each subset. To measure data privacy hence to guide the procedure of alert anonymization, we propose two measures: *local privacy* and *global privacy*, and use entropy to compute their values.

To understand security threats involved in alert data sets, various alert correlation methods have been proposed in recent years. These methods usually focus on original alerts and can be roughly divided into four groups. (1) Approaches based on similarity between alerts [24, 28, 5]. These approaches group alerts through computing similarity values between alert attributes. (2) Approaches based on prerequisites and consequences of attacks [6, 25, 17]. These methods model each attack through identifying its prerequisite (i.e., the necessary condition to launch an attack successfully) and consequence (i.e., the possible outcome if an attack succeeds), and matching consequences with prerequisites among different attacks to build attack scenarios. (3) Approaches based on known attack scenarios [15, 7]. These approaches build attack scenarios through matching alerts to known scenario templates. (4) Approaches based on multiple information sources [21, 16, 30]. Their goal is to analyze alerts from different security systems.

Though we emphasize alert anonymization techniques in this paper, we also perform alert correlation to anonymized alerts to examine data usability. We focus on two problems: estimating similarity values between anonymized attributes and building attack scenarios from anonymized alert sets. Our method on similarity measurement is a probability based approach, which estimates how possible two anonymized attributes may have the same original values. Our approach on building attack scenarios extends from an existing method [17], where the probabilities related to the matching of prerequisites and consequences among different attacks are estimated. Though it is closely related to the optimistic approach in [29], the probability estimation in our approach is based on our anonymization schemes. Based on these probabilities values, we can construct and further "polish" attack scenarios. Our experimental results demonstrated the effectiveness of our techniques in terms of various measures.

The remainder of this paper is organized as follows. In the next section, we present three schemes to anonymize alerts. In Section 3, we discuss techniques to correlate anonymized alert sets. In Section 4, we show our experimental results. In Section 5, we discuss related work. And in Section 6, we conclude this paper and point out further directions.

2. Three Schemes for Alert Anonymization

In this paper, we emphasize our alert anonymization techniques, which can flexibly protect data privacy. Before we go into the details of our techniques, we clarify some notions and definitions first.

An alert type is a type name T and a set $\mathcal{S}$ of attribute names, where each attribute name in $\mathcal{S}$ has a related domain denoting possible attribute values. As an example, an alert type *FTP_AIX_Overflow* has a set of six attribute names {*SrcIP, SrcPort, DestIP, DestPort, StartTime, EndTime*}, where the type name *FTP_AIX_Overflow* denotes that it is a buffer overflow attack targeting *AIX* ftp services, and all six attributes are used to describe this type of attacks. The domains of *SrcIP* and *DestIP* are all possible IP addresses, the domains of *SrcPort* and *DestPort* are possible port numbers (from port 0 to port 65535), and *StartTime* and *EndTime* denote the timestamps that the attack begins and finishes, respectively.

An original alert is an instance of alert types and is reported by security systems. Formally, a type T original alert t_o is a tuple on T's attribute set $\mathcal{S}$, where each attribute value in this tuple is a value in the related domain. For example, assume we have a type *FTP_AIX_Overflow* alert {SrcIP =$172.16.10.28$, SrcPort =1081, DestIP =$172.16.30.6$, DestPort =21, StartTime =01-16-2006 18:01:05, EndTime =01-16-2006 18:01:05}. This alert describes an *FTP_AIX_Overflow* attack from IP address $172.16.10.28$ to target $172.16.30.6$. A type T *artificial alert* has the same format as that of an original alert. The only difference between them is that original alerts are reported by security systems, while artificial alerts are synthetic, and may be generated by a human user, or some programs. Similarly, a type T *anonymized alert* has the same format as that of an original alert. However, sensitive attribute values in anonymized alerts are transformed (e.g., through randomization) to protect data privacy. As an example, if *DestIP* of the alert in the above example is sensitive, we may transform *DestIP*=$172.16.30.6$ to *DestIP*=$172.16.30.35$. In the rest of this paper, we call the alert set including both original and artificial alerts the mixed alert set. For convenience, we may also use attributes to represent either attribute names, attribute values, or both.

2.1. Scheme I: Artificial Alert Injection Based on Concept Hierarchies

Intuitively, artificial alert injection generates synthetic alerts and mixes them with original alerts. Given any alert in

a mixed alert set, identifying whether it is artificial or original is difficult, which means the information disclosed by any individual alert may not necessarily be true. The critical issue here is how to generate attribute values for artificial alerts, with both privacy and usability requirements in mind. We divide alert attributes into three classes: alert types, sensitive attributes, and non-sensitive attributes, and discuss them separately.

Alert types encode valuable information about the corresponding attacks. For example, RealSecure network sensor 6.5 [10] may report an *FTP_AIX_Overflow* attack. Based on the signature of this attack, we know that it is a buffer overflow attack targeting AIX FTP services. Alert types are crucial for security officers to learning security threats. So when we create artificial alerts, we propose to preserve the frequency of the original data set in terms of alert types, where the frequency of an alert type T is the ratio of the number of alerts with type T to the total number of alerts. In other words, if an original alert data set has n types of alerts with frequencies $p_1, p_2, \cdots, p_n$ where $p_1 + p_2 + \cdots + p_n = 1$, then in our artificial alert set, we will maintain this frequency distribution.

Sensitive attributes are decided by privacy policy, and their values are what we try to protect. Considering both privacy and usability concerns, we propose to use concept hierarchies to help us artificially generate sensitive attribute values. Creating attribute values based on concept hierarchies may preserve some useful information (e.g., prefixes of IP addresses), but also change attribute distributions to certain degree to protect alert privacy. Now let us first introduce concept hierarchies.

Concept hierarchies have been used in areas such as data mining [9] and also in privacy-preserving techniques (e.g., $k-$Anonymity approach [23] and privacy-preserving alert correlation [29]). A concept hierarchy is based on *specific-general* relations. Given two concepts c_1 and c_2 (e.g., attribute values), if c_1 is more specific than c_2 (or, c_2 is more general than c_1), we say there is a *specific-general* relation between c_1 and c_2, and denote it as $c_1 \preceq c_2$. As an example, given an IP address $172.16.10.3$ and a network address $172.16.10.0/24$, we have $172.16.10.3 \preceq 172.16.10.0/24$. Specific-general relations can be obtained through abstracting a set of low-level concepts to a high-level concept. For example, a set of individual IP addresses can be organized into a subnet. Based on specific- general relations, a *concept hierarchy* is a set of specific-general relations and is usually organized into a tree. Figure 1 shows a concept hierarchy for IP addresses $172.16.11.0, \cdots, 172.16.11.255$ and $172.16.12.0, \cdots, 172.16.12.255$, where each IP address is first generalized to its $/24$ network address, and then to its $/16$ network address.

For continuous attributes, we can group data into bins thus continuous values can be transformed into categorical. For example, given a set of timestamps within a one-hour time interval, we may partition the whole time interval into 60 equal-length bins, and put timestamps into the corresponding bins. Binning techniques have been extensively studied in the fields such as data mining and statistics, and we do not repeat them here. In this paper, our techniques focus on categorical data, though they can be extended to accommodate continuous data.

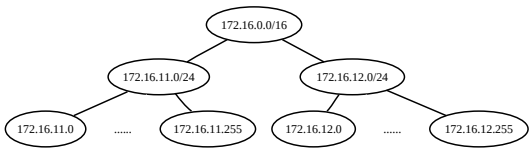


Figure 1. An example concept hierarchy

Given a concept hierarchy H, and two nodes v_s and v_g in H where $v_s \preceq v_g$ (v_s has a path to v_g in H), the *distance* between v_s and v_g is the number of edges over the path from v_s to v_g, denoted as *distance*(v_s, v_g). For example, in Figure 1, *distance*$(172.16.11.3,\ 172.16.11.0/24) = 1$. Given a concept hierarchy H and two nodes v_{s1} and v_{s2} in H, we call a node v_g in H the *least common parent* if (1) $v_{s1} \preceq v_g$, (2) $v_{s2} \preceq v_g$, and (3) $d_m =$*max*(*distance*(v_{s1}, v_g), *distance*(v_{s2}, v_g)) has a minimum value. In addition, if v_{s1} and v_{s2} are both leaf nodes in H and the least common parent v_g has total $\mathcal{L}$ leaf nodes including v_{s1} and v_{s2}, we call nodes v_{s1} and v_{s2} *$\mathcal{L}$-peer nodes*, or simply *$\mathcal{L}$-peers*. For example, in Figure 1, the least common parent of two leaf nodes $172.16.11.3$ and $172.16.11.5$ is node $172.16.11.0/24$. Since $172.16.11.0/24$ has totally 256 leaf nodes, $172.16.11.3$ and $172.16.11.5$ are 256-peers.

Now let us discuss how to generate sensitive attributes for artificial alerts. We assume each sensitive attribute value has a desirable general value in concept hierarchies, where these desirable general values can be derived through a given parameter $\mathcal{L}$ denoting the desirable number of peer nodes. For example, if $\mathcal{L} = 256$ for attribute *DestIP*, then the desirable general values for these IP addresses are the corresponding $/24$ network addresses. We first compute the frequency distribution of these generalized attribute values based on the original data set. Next, following the computed frequency distribution, we create generalized attribute values in the artificial alert set, finally we replace these generalized values using leaf nodes in the corresponding hierarchies (each leaf node value has equal probability to be chosen). Notice that during the above procedure, we preserve attribute frequency in terms of general values. This is because these general values partially maintain the usability of alert data (e.g., prefixes of IP address). We also replace these general values using their corresponding leaf nodes with uniform distribution. This may help us change the distribution of attributes in original sets. For example, if in

an original set, the values for attribute *DestIP* is only from $172.16.11.0$ to $172.16.11.31$. Further suppose we set parameter $\mathcal{L} = 256$. Then we will generate artificial attributes uniformly distributed from $172.16.11.0$ to $172.16.11.255$ to change attribute distribution and hence protect original values. Notice that desirable general values (or, parameter $\mathcal{L}$) for sensitive attributes are decided by privacy policy. For example, if we let *DestIP*'s desirable general values be the corresponding $/24$ network addresses, this means that ideally, we want each *DestIP* in its $/24$ network to be equally likely in alert sets, so malicious users may not be able to "guess" which values are more possible.

To generate non-sensitive attributes, we first compute the frequency distribution of their original values, then we generate artificial attribute values with the same frequency distribution in the artificial alert set. As a special case, for timestamp information, we first get the minimum and maximum timestamps for each alert type in the original alert set, then we uniformly choose timestamps between the minimum and maximum values for artificial alerts.

Another important issue on artificial alert generation is to decide how many alerts to generate. Notice that injecting artificial alert data usually may change the distribution of attribute values. Intuitively, if a large number of artificial alerts are generated, we may better protect alert privacy (attributes are more uniformly distributed), but alert usability may decrease. On the other hand, if only a small number of alerts are created, we may increase alert usability, but alert privacy may decrease. So the ideal case is to flexibly control the difference between attribute distributions based on the requirement of privacy protection.

We propose to use the distance between probability mass functions (PMFs) to measure the difference between attribute distributions. Given one attribute A_s in both original alert set S_o and mixed alert set S_m, assume the PMFs for A_s in S_o and S_m are $f_o(x)$ and $f_m(x)$, respectively, where x is possible values for A_s. Further assume the domain of attribute A_s is $Dom(A_s)$. To calculate the distance between two PMFs, we first need define distance function $D(f_o, f_m)$ between $f_o(x)$ and $f_m(x)$. As an example, we can set $D(f_o, f_m) = \sum_{x \in Dom(A_s)} |f_m(x) - f_o(x)|$. Through setting a desirable distance threshold, we may inject the number of alerts satisfying the threshold. (In case sensitive attributes in original sets are in uniform distribution, we may further specify a maximum number of artificial alerts to prevent infinite alert injection.)

Notice that in our above discussion, we do not consider the dependence between different attributes (e.g., the dependence between attributes *DestIP* and *DestPort*). When there are no such dependence, we can use the proposed method above to generate all alerts. However, when there does exist the dependence, we need to handle this situation with caution. As an example, if there are only one web server $172.16.10.5$ with port 80 open in a network, then usually all those web based attacks are targeting $172.16.10.5$ on port 80. This means IP address $172.16.10.5$ and port number 80 are dependent in web based attacks. Artificial alert generation, on the one hand, does not require to strictly satisfy this dependence, because the violation of this dependence may bring confusion to and require further investigation from malicious users, which in some sense may protect data privacy. However, on the other hand, if we try to make artificial and original alerts very difficult to distinguish, or data usability is our favorable concern, we can also maintain attribute dependence in artificial alert generation. We propose two ways to get dependence relationships.

1. Manually collect all dependence relationships through various means. For example, based on attack signatures, and host configurations, we can know the hosts and ports that some attacks are targeting.

2. Compute conditional probabilities between attribute values based on original alert sets, and follow these conditional probabilities to generate attribute values in artificial alert sets. This approach is similar to the data-swapping technique proposed by Reiss [22].

To see how well our anonymization schemes can protect alert data privacy, we classify alert data privacy into two levels: *local privacy* and *global privacy*. Local privacy is related to original attribute values in each individual alert. Intuitively, if the original value for a sensitive attribute in an alert has been known, the local privacy for the sensitive attribute in this alert is compromised. The second level of alert data privacy is global privacy, which is related to the distributions of sensitive attributes in alert set. Intuitively, if the distribution of a sensitive attribute in an original alert set is known, we can derive useful information about the original set (e.g., the most possible value in the data set). For both local and global privacy, we use *entropy* [4] to measure them. Suppose that we have a value v_s for a sensitive attribute A_s in a alert t. Based on v_s, if we can estimate the possible original values of A_s in t and the corresponding probability for each possible value, then we can compute alert t's local privacy $H_l(t) = -\sum p_i \log_2 p_i$, where p_i is the probability for a possible value. Given all attribute values for a sensitive attribute A_s in an alert set S, we can estimate S' global privacy $H_g(S) = -\sum P(v_i) \log_2 P(v_i)$, where v_i is a possible value for A_s in S, and $P(v_i)$ is the corresponding probability. To help us better understand global privacy, we may explain it from a random variable point of view. Assume that a sensitive attribute is a random variable, and all values for this attribute in an alert set is a realization of this random variable. Then the global privacy for this attribute in the alert set is a randomness (or uncertainty) measure about the realization of this random variable. Recall that we generate uniformly distributed data (based on concept hierarchies) during artificial alert injection. The reason

is that injecting uniformly distributed data generally may increase the randomness of data sets. Also notice that the distance between PMFs and the change in global privacy are closely related because we change attribute distributions through injecting uniformly distributed data. And it is also feasible to control the number of artificial alerts through adjusting the change in global privacy.

Back to artificial alert injection, if original alert set S_o has m alerts, we inject n artificial alerts in it. Thus in mixed set S_m including both original and artificial alerts, we have total $m+n$ alerts. In S_m, each individual alert has probability $\frac{m}{m+n}$ to be original, and probability $\frac{n}{m+n}$ to be artificial. So its local privacy is $-\sum p_i \log_2 p_i = \frac{m}{m+n}\log_2\frac{m+n}{m} + \frac{n}{m+n}\log_2\frac{m+n}{n}$. We can also calculate global privacy for both S_o and S_m, and compute their difference to see how well we can improve global privacy. One of our later experiments shows that through injecting 168 artificial alerts into an original set with 922 alerts, we achieve local privacy with value 0.62, and we improve global privacy from 4.696 to 5.692 (the distance between two PMFs is 0.3).

2.2. Scheme II: Attribute Randomization Based on Concept Hierarchies

In Scheme I, we inject artificial alerts into original data set. For any individual alert in the mixed alert set, it may be difficult to identify whether this alert is original or artificial, hence sensitive attribute values in any single alert have a chance to be faked.

Let us look at Scheme I in detail. Assume that we have m original alerts, and inject n artificial alerts into it. Hence in the mixed alert set, each alert has a probability $\frac{m}{m+n}$ to be original. Based on probability theory, randomly pick up k alerts in the mixed set, the probability that at least one alert is original is $1-(\frac{n}{m+n})^k$. As an example, let $m=1000$, $n=300$ and $k=2$, then the probability that at least one alert is original is 94.67%, while the probability that both alerts are artificial is only 5.33%. This tells us that when the ratio of the number of artificial alerts to the total number of alerts is small, it is very likely that some alerts in an alert subset are original even if this subset is small. Closely investigating these small subsets may disclose the privacy of alert data. This problem may be mitigated by injecting more artificial alerts. In the extreme case, if we inject a much larger number of artificial alerts (i.e., $m \ll n$), then in a randomly selected k-alert subset, the probability that at least one alert is original may be very small even if k is big. For example, let $m=1,000$ and $n=1,000,000$. When $k=100$, the probability that at least one alert is original is only 9.51%, and even when $k=1,000$, the probability value only increases to 63.19%. Notice that with the increase of the number of artificial alerts, the usability of mixed data sets may be decreasing, and more overhead is introduced to analyze mixed alert sets. Considering privacy, usability, and performance, we propose to apply randomization to sensitive attributes, which may allow us protect the privacy of original alerts without injecting a huge amount of artificial alerts. Notice that our randomization algorithm takes advantage of concept hierarchies, which can preserve some useful information in sensitive attributes (e.g., prefixes of IP addresses).

Though we motivate attribute randomization in term of mixed alert sets, it can be applied to original alert sets. Given a parameter $\mathcal{L}$ denoting the desirable number of peers in concept hierarchies, the basic idea of our algorithm is to randomize each different attribute value v_i uniformly to any of v_i's $\mathcal{L}$-peers (In other words, the mapping and mapped values has a common general value v_g where v_g has $\mathcal{L}$ leaf nodes). For example, based on the concept hierarchy in Figure 1, we may randomize a sensitive attribute *DestIP*=172.16.11.8 to *DestIP*=172.16.11.56 if $\mathcal{L}=256$. During randomization, we keep *consistency* in attribute mapping, which means if two alerts has the same original values for an attribute A_s, then the mapped values for A_s in both alerts are the same. For convenience, we call the mapped value the *image value*, or simply the *image*. This consistency is helpful to maintain data usability.

To see how attribute randomization may help protect alert privacy, let us take a look at both local and global privacy. Suppose that we randomize a sensitive attribute A_s in an original alert set. After performing randomization, given any image value v_r, we know the original value of v_r may be any of v_r's $\mathcal{L}$-peers with equal probability (i.e., $\frac{1}{\mathcal{L}}$). Thus the local privacy value is $-\sum_{i=1}^{\mathcal{L}}\frac{1}{\mathcal{L}}\log_2\frac{1}{\mathcal{L}} = \log_2\mathcal{L}$. If we randomize a mixed alert set, we can also derive the corresponding local privacy after considering the probability that a given alert may be original. On the other hand, based on concept hierarchies and requirements for local privacy values, we may choose desirable parameters (e.g., $\mathcal{L}$) satisfying the requirements. To consider global privacy, let us assume that there are k distinct values for sensitive attribute A_s in an alert set S. Since we keep consistency during randomization, there will be at most k distinct image values for A_s in randomized alert set S_r. If k distinct image values do exist, then S_r have the same global privacy as in S. When less than k distinct image values are generated (this is possible because two different original values may happen to be randomized to the same value), the global privacy value $H_g(S_r)$ in S_r may change compared with the value in S, where $H_g(S_r)$ may be slightly smaller. Our later experimental results confirm this conjecture. For example, in one data set, we set $\mathcal{L}=256$, and the global privacy slightly changes from 5.692 (before randomization) to 5.671 (after randomization). To summarize, our attribute randomization may result in slight change (or no change) in global privacy, and desirable change (through choosing appropriate param-

eters) in local privacy.

2.3. Scheme III: Alert Set Partitioning and Attribute Randomization

In Scheme II, we randomize sensitive attribute values to their $\mathcal{L}$-peers and maintain consistency during randomization. A limitation related to this scheme is that once attackers get to know some *(original value, image value)* pairs, then for all the image values in the known pairs, attackers know their corresponding original values in the whole alert set. Notice that (original value, image value) pairs can be obtained through various means, for example, deliberately attacking some special hosts and examining published alert data (e.g., counting frequencies of some attacks). To mitigate this problem, we propose to partition an alert set into multiple subsets and perform attribute randomization (Scheme II) in each subset independently. Thus one original value may be mapped to different image values in different subsets.

Our algorithm on partitioning an alert set is based on a *time constraint*. Given a time interval $\mathcal{I}$ (e.g., 2 hours) and a set $\mathcal{S}$ of alerts, we say $\mathcal{S}$ satisfy time constraint $\mathcal{I}$ if $|max(EndTime) - min(StartTime)| \leq \mathcal{I}$ (i.e., the difference between the maximum *EndTime* and the minimum *StartTime* in $\mathcal{S}$ is less than or equal to $\mathcal{I}$). Based on a time constraint, we partition an alert set into multiple subsets such that each subset satisfies the time constraint, then we randomize each subset through Scheme II independently.

Now let us look at local and global privacy under Scheme III. For local privacy, since image values are chosen uniformly from $\mathcal{L}$-peers, we have a similar analysis as in Scheme II. However, for global privacy, since it is possible that one original value in different subsets may be randomized to different image values, global privacy usually may increase compared with applying Scheme II. Our later experiments confirm our conjecture. For example, in one experiment, we partitioned one data set into four subsets, and global privacy increased from 5.692 to 6.955.

3. Anonymized Alert Correlation

Though our focus in this paper is alert anonymization, we are also interested in the usability of anonymized alert sets. Notice that current alert correlation approaches usually concentrate on computing similarity values between alert attributes, or building attack scenarios to reflect attackers' activities. So in this section, we will focus on these two problems under the situation that alerts are anonymized. Notice that our approach is closely related to a previous method [29], however, the alerts that we correlate are anonymized by the schemes proposed in this paper.

3.1. Similarity Estimation

Similarity measurement computes how similar two attributes are, usually with a value between 0 and 1. Existing approaches [28, 24, 11] focus on measuring similarity between original attributes. Our anonymization techniques (Schemes II and III) randomize original attributes to their peers. Thus it is crucial and helpful to estimate similarity between original attributes only using anonymized values. In the following, we first give an example function on computing similarity between original values, then discuss how to estimate similarity based on anonymized attributes.

Assume that we have a sensitive attribute A_s and two original attributes values x_1 and x_2 for A_s. As an example similarity function, we let $Sim(x_1, x_2) = 1$ if $x_1 = x_2$, and $Sim(x_1, x_2) = 0$ if $x_1 \neq x_2$. We further assume x_1 and x_2's images are y_1 and y_2, respectively, after randomization. We consider two cases regarding whether x_1 and x_2 are in the same subset (an set is partitioned in Scheme III).

(1) When x_1 and x_2 are in the same subset, we have the following observation.

Observation 1 *For a sensitive attribute A_s, given two original attribute values x_1 and x_2 where they are in the same subset, and two image values y_1 and y_2 randomized from x_1 and x_2 using Scheme II (with same parameters such as $\mathcal{L}$), we know that (i) if $x_1 = x_2$, then $y_1 = y_2$; (ii) if $y_1 = y_2$, x_1 and x_2 may or may not be the same.*

We explain our observation through examples. Assume x_1 and x_2 are both destination IP addresses where $x_1 = 172.16.11.5$. Using the concept hierarchy in Figure 1, suppose we randomize x to one of its 256-peer nodes, and x_1 is mapped to y_1 with value $172.16.11.98$. Since we keep consistency in Scheme II, if $x_2 = 172.16.11.5$, then we know $y_2 = y_1 = 172.16.11.98$. On the other hand, if $x_2 \neq x_1$, for example, let $x_2 = 172.16.11.9$, then y_2 can be any IP address from $172.16.11.0$ to $172.16.11.255$, which has a chance to be $172.16.11.98$. To better characterize this observation, we compute the following probabilities.

For simplicity, assume the domain of x is $\mathcal{L}$ specific values where each value has equal probability to be chosen, and two original values x_1 and x_2 are randomized using their $\mathcal{L}$-peer nodes. Based on conditional probability and total probability theorems, we can derive $P(x_1 = x_2|y_1 = y_2) = \frac{\mathcal{L}}{2\mathcal{L}-1}$. (How we derive this probability is shown in Appendix A.) Similarly, we can get $P(x_1 \neq x_2|y_1 = y_2) = \frac{\mathcal{L}-1}{2\mathcal{L}-1}$, $P(x_1 = x_2|y_1 \neq y_2) = 0$, and $P(x_1 \neq x_2|y_1 \neq y_2) = 1$. Notice that though we use the assumption of uniform distribution about original values to derive $P(x_1 = x_2|y_1 = y_2) = \frac{\mathcal{L}}{2\mathcal{L}-1}$, we can prove that for other distributions, we have $P(x_1 = x_2|y_1 = y_2) \geq \frac{\mathcal{L}}{2\mathcal{L}-1}$ as long as the attribute domain is the same. We prove it through Lemma 1 in Appendix A.

(2) When x_1 and x_2 are in different subset, x_1 and x_2 are randomized independently. Assume their randomization has the same input parameters (e.g., $\mathcal{L}$). With the similar reasoning as in (1), we know that as long as x_1 and x_2 are $\mathcal{L}$-peers, y_1 and y_2 have a chance to be the same. Under the same assumption as in (1), we derive $P(x_1 = x_2|y_1 = y_2) = \frac{1}{\mathcal{L}}$, $P(x_1 \neq x_2|y_1 = y_2) = \frac{\mathcal{L}-1}{\mathcal{L}}$, $P(x_1 = x_2|y_1 \neq y_2) = \frac{1}{\mathcal{L}}$, and $P(x_1 \neq x_2|y_1 \neq y_2) = \frac{\mathcal{L}-1}{\mathcal{L}}$.

As an example to estimate similarity between y_1 and y_2, we estimate how possible their corresponding x_1 and x_2 are the same, and use this probability as their similarity value. Based on our above assumption and reasoning, we can get an example similarity function for anonymized attributes as follows. $Sim(y_1, y_2) =$

- $\frac{\mathcal{L}}{2\mathcal{L}-1}$, if ($y_1$ and y_2 are $\mathcal{L}$-peers) $\wedge$ ($y_1 = y_2$) $\wedge$ (y_1 and y_2 in the same subset),
- $\frac{1}{\mathcal{L}}$, if (y_1 and y_2 are $\mathcal{L}$-peers) $\wedge$ (y_1 and y_2 in different subsets),
- 0, otherwise.

Notice that to derive the new similarity function above, we only consider a simple case regarding how possible original attributes may be the same. For more complicated case, we may apply a similar, probability-based approach to derive new functions for anonymized attributes.

3.2. Building Attack Scenarios

Attack scenarios help us understand what steps adversaries take to attack victim machines. For example, an attacker may first run IP sweep to detect live IP addresses, followed by port scanning attacks to look for open ports, and finally launch buffer overflow attacks to compromise live hosts. To build attack scenarios, it is crucial to identify causal relations between individual attacks. For example, in the scenario above, there is a causal relation between the earlier IP sweep attack and the later port scanning attack because the first one may detect live IP addresses for the second one to further probe.

There are several approaches being proposed to build attack scenarios. For example, known attack scenario based approaches [7, 15], and prerequisite and consequence based methods [25, 6, 17]. These approaches can build attack scenarios when original attributes are known. To build attack scenarios from anonymized alerts, we use a probability based approach, which is extended from previous correlation methods [17, 29]. In the following, we start with an overview of a previous correlation method [17], with slight modification to facilitate our discussion.

3.2.1. An Overview of a Correlation Method [17]. In correlation method [17], each alert type is associated with its *prerequisite* and *consequence*, where the prerequisite of an attack is the necessary condition for the attack to be successful, and the consequence of an attack is the possible outcome if the attack succeeds. We use predicates to represent prerequisites and consequences, where the variables in predicates are the attribute names in alert types.

Example 1 Given an alert type *FTP_AIX_Overflow*, its prerequisite is *ExistService(DestIP,DestPort)*, and its consequence is {*GainAccess(DestIP)*}, which means the necessary condition of an *FTP_AIX_Overflow* attack is that *FTP* service is running on *DestIP* at *DestPort*, and the consequence is that attackers may gain unauthorized access to *DestIP*.

To further characterize prerequisites and consequences of attack instances (i.e., alerts), we can instantiate prerequisites and consequences using alert attribute values. To continue Example 1, assume that we have a type *FTP_AIX_Overflow* alert {*SrcIP=172.16.10.28, SrcPort=1081, DestIP=172.16.30.6, DestPort=21, StartTime =01-16-2006 18:01:05, EndTime =01-16-2006 18:01:05* }. The alert's instantiated prerequisite is *ExistService(172.16.30.6,21)*, and its instantiated consequence is {*GainAccess(172.16.30.6)*}.

We model *causal relations* between alerts as *prepare-for* relations. Formally, given two alerts t_1 and t_2, t_1 *prepares for* t_2 if (1) one of t_1's 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 2 Assume that we have two alert types *Port_Scan* and *FTP_AIX_Overflow* where the consequence of type *Port_Scan* is {*EixistService(DestIP, DestPort)*}. Further assume that we have two alerts t_1 and t_2 where t_1 is a type *Port_Scan* alert {*SrcIP=172.16.10.28, SrcPort=1073, DestIP=172.16.30.6, DestPort=21, StartTime=01-16-2006 18:00:02, EndTime=01-16-2006 18:00:02* }, and t_2 is a type *FTP_AIX_Overflow* alert {*SrcIP =172.16.10.28, SrcPort =1081, DestIP =172.16.30.6, DestPort =21, StartTime =01-16-2006 18:01:05, EndTime =01-16-2006 18:01:05* }. Thus t_1's instantiated consequence is {*ExistService (172.16.30.6,21)*}, and t_2's instantiated prerequisite is *ExistService (172.16.30.6,21)*. Further due to t_1.EndTime $<$ t_2.StartTime, we know t_1 prepares for t_2.

We model attack scenarios as *alert correlation graphs*. *An alert correlation graph* (or simply *a correlation graph*) is a directed graph (N, E), where (1) any vertex $n \in N$ is an alert, and (2) for any directed edge $(n_1, n_2) \in E$, we have n_1 prepares for n_2.

3.2.2. A Probability Based Approach to Building Attack Scenarios. To build attack scenarios, it is critical to identify prepare-for relations. When all original values are known, this identification is straightforward. However, when alerts are anonymized through randomization, identifying prepare-for relations requires more efforts.

Example 3 Let us re-consider Example 2. Assume that *DestIP* is sensitive, and we do not know their original values in both alerts t_1 and t_2. Based on the prerequisites, the consequences, and the available non-sensitive values, we know that t_1 prepares for t_2 only if t_1 and t_2 have the same original destination IP addresses. For simplicity, assume that the original values of *DestIP* are uniformly distributed from $172.16.30.0$ till $172.16.30.255$, and attribute randomization uniformly chooses IP addresses from $172.16.30.0$ to $172.16.30.255$ to replace original values ($\mathcal{L} = 256$). We consider two cases. (1) Suppose that t_1 and t_2 are in the same subset (regarding alert set partitioning in Scheme III). If t_1 and t_2's anonymized destination IP addresses are the same (e.g., both are $172.16.30.52$), then based on our reasoning in similarity estimation (Subsection 3.1), we know that with probability $\frac{\mathcal{L}}{2\mathcal{L}-1} = \frac{256}{511} = 0.501$, t_1 and t_2 may have the same original *DestIP*. Equivalently, t_1 and t_2 may have different original destination IP addresses with probability 0.499. In addition, if after randomization, t_1 and t_2 have different anonymized destination IP addresses, we know that their original destination IP addresses are different. (2) Suppose that t_1 and t_2 are in different subsets. we know that t_1 and t_2 may be possible (with probability $\frac{1}{\mathcal{L}} = \frac{1}{256}$) to have the same original *DestIP* as long as their anonymized values are $\mathcal{L}$-peers.

Based on the above observation, we realize that we can only identify possible prepare-for relations after attribute randomization. To characterize this observation, we propose to associate a probability value to each possible prepare-for relations when building attack scenarios from anonymized alerts. Notice that sometimes precisely computing the probability that one alert prepares for another is difficult because analysts do not know probability distributions of original attribute values. However, as we mentioned in Subsection 3.1 and also proved in Appendix A, we can get lower bound probability values under the assumption of uniform distributions. We take advantage of this observation and define *possibly-prepare-for* relation. Formally, given two anonymized alerts t_1 and t_2, t_1 *possibly prepares for* t_2 with at least probability p if (1) the probability that t_1 prepares for t_2 is no less than p, and (2) $p > 0$. Our probability based approach is closely related to the optimistic approach in [29]. However, here probabilities related to possibly-prepare-for relations are lower bound values and are estimated based on the anonymization schemes in this paper. To continue Example 3, (1) when t_1 and t_2 are in the same subset, if t_1 and t_2 have the same anonymized *DestIP*, we know t_1 possibly prepares for t_2 with at least probability 0.501; (2) when t_1 and t_2 are in different subsets, if their anonymized destination IP addresses are $\mathcal{L}$-peers, t_1 possibly prepares for t_2 with at least probability $\frac{1}{256}$.

In the above example, there are only one implication relationship between instantiated predicates for two alerts. When there are multiple implication relationships, similar as in [29], we assume each implication relationship is independent, and then estimate the probability that at least one implication relationship is true. In particular, if the probability that each implication relationship is true is at least p_1, p_2, $\cdots$, p_n, respectively, then the probability that at least one implication relationship is true has a lower bound value $1 - (1 - p_1)(1 - p_2) \cdots (1 - p_n)$.

To build attack scenarios from anonymized alerts, we identify all possibly-prepare-for relations and connect them as a graph. For convenience, we may use prepare-for relations to represent either prepare-for relations, possibly-prepare-for relations, or both in the rest of this paper.

Lower-bound probabilities related to prepare-for relations may also be used to "polish" alert correlation graphs. Actually if we take a close look at how we compute these probability values, the basic problem involved is to decide how possible the related attributes have the same original values. In some cases, we can estimate precise probability values. For example, suppose the desirable number of peers is $\mathcal{L}$ when applying Scheme III. If two anonymized attributes are in different subsets, and they are $\mathcal{L}$ peers, then the probability that they have the same original value is $\frac{1}{\mathcal{L}}$. However, in some other cases, for example, two anonymized attributes are in the same subset and have the same anonymized values, we usually may assume uniform distribution to get lower-bound probability values. Since prepare-for relations are identified through this probability based approach, it is natural that some prepare-for relations may not be true. A common way to filter out false prepare-for relations is to examine their related (lower-bound) probabilities. If the probability is greater than a given probability threshold, we keep it in the correlation graph, otherwise we remove it. Notice that in [29], a probability based refinement to alert correlation graphs also has been proposed. However, the approach in [29] is to calculate the probability for a set of prepare-for relations, and use this probability to perform aggregation, while the approach in this paper is to filter out individual prepare-for relations. In addition, we agree that the approach in [29] is complementary to the approach in this paper, and may be applied here. Though our approach on polishing alert correlation graphs may help us remove false prepare-for relations, we also notice that it has a chance to prune true prepare-for relations. It is necessary to examine both alert correlation graphs with and without probability-polishing to understand attackers' activities.

4. Experimental Results

To evaluate the effectiveness of our techniques, we did a set of experiments to evaluate our anonymization schemes, similarity estimation and building correlation graphs. The data sets we used are 2000 DARPA intrusion detection sce-

nario specific data sets [14]. These data sets were collected in simulation networks and include two scenarios: LLDOS 1.0 and LLDOS 2.0.2. Both scenarios have two data sets collected over different parts of the networks: the inside part and the DMZ part. In LLDOS 1.0, the attackers first probed the network, then launched buffer overflow attacks against vulnerable *Sadmind* services, next installed DDoS software on victim hosts, and finally ran DDoS attacks. LLDOS 2.0.2 has a similar scenario as in LLDOS 1.0. We used RealSecure network sensor 6.0 to generate alerts from the data sets. Similar as done by DShield, we set attribute *DestIP* (destination IP addresses) in all alerts as sensitive attribute, and anonymized them using the schemes in this paper.

4.1. Alert Anonymization via Schemes I, II and III

In our first set of experiments, our goal is to evaluate the effectiveness of artificial alert injection. We injected artificial alerts into all four data sets based on Scheme I. We set the desirable general value for each *DestIP* to its corresponding $/24$ network address ($\mathcal{L} = 256$). We used distance function $D(f_o, f_m) = \sum_{x \in Dom(A_s)} |f_m(x) - f_o(x)|$, and let distance threshold (between PMFs) be 0.3.

In the second set of experiments, our goal is to see the effectiveness of attribute randomization using Scheme II. We applied Scheme II to all four mixed data sets generated in the first set of experiments. Specifically, we randomized each *DestIP* in mixed data sets to its 256-peer node (Any IP address in its $/24$ network).

In the third set of experiments, our goal is to evaluate the effectiveness of Scheme III. we applied Scheme III to all four mixed data sets. Specifically, we randomized each *DestIP* in mixed data sets to its 256-peers. In addition, we set time interval $\mathcal{I} = 1$ hour to partition data sets. Consequently, we got 4 subsets for LLDOS 1.0 Inside alert set, 4 subsets for LLDOS 1.0 DMZ alert set, 2 subsets for LLDOS 2.0.2 Inside alert set, and 2 subsets for LLDOS 2.0.2 alert set.

To see how our anonymization schemes may change the distributions of sensitive attributes and protect data privacy, we plotted the PMFs of attribute *DestIP*, and computed local and global privacy values for all three sets of experiments. Due to space constraint, we listed all PMFs of *DestIP* in Appendix B (in Figure 3). For local and global privacy values, we listed them in Table 1.

Based on the PMFs in Figure 3, we observed that (1) Scheme I can change the distributions of *DestIP* compared with original data sets, (2) Scheme II may greatly change the distribution of *DestIP* in mixed alert sets, and (3) Scheme III may further change *DestIP* distributes compared with the results in Scheme II.

To precisely evaluate alert privacy, now let us look at local and global privacy values in Table 1. Based on this table, we noticed that (1) through Scheme I, we can better protect alert privacy because both local and global privacy values increase. For example, in LLDOS 1.0 inside part, we injected around 15% of artificial alerts among all alerts, and local privacy increases from 0 to 0.620, and global privacy increases from 4.696 to 5.692. (2) Through Scheme II, local privacy has significantly increased, which is highly desirable, and global privacy stays almost the same, which results from consistency keeping during randomization. And (3) after applying Scheme III, local privacy stays the same, and global privacy further increases, which results from independent randomization in each subset.

	LLDOS 1.0		LLDOS 2.0.2	
	Inside	DMZ	Inside	DMZ
Scheme I: R_{cc}	100%	100%	100%	100%
Scheme I: R_{mc}	0.0305%	0.0649%	0.0418%	0.0736%
Scheme II: R_{cc}	100%	100%	100%	100%
Scheme II: R_{mc}	0%	0%	0%	0%
Scheme III: R_{cc}	100%	100%	100%	100%
Scheme III: R_{mc}	10.01%	8.27%	6.58%	4.32%

Table 2. Similarity estimation

4.2. Similarity Measurement

In our experiments, similarity computation is based on our discussion on Subsection 3.1. For original data set, if two attribute values are the same, we set their similarity to 1; otherwise their similarity is 0. Next we applied three anonymization schemes independently, where we chose similar settings as in Subsection 4.1 (the only difference is that here we applied Schemes II and III to original alert sets instead of mixed alert sets). For the data sets after applying Scheme I, we measured attribute similarity using the function for original sets. And for the data sets after applying Schemes II and III, we used the function in Subsection 3.1 to estimate their (lower-bound) similarity values. Next, similar as done in [29], we also used *correct classification rate* R_{cc} and *misclassification rate* R_{mc} to measure the effectiveness of similarity estimation. Given two attribute values, if their similarity value is greater than 0, we call them *"similar" pair*. Assume the alert sets before and after anonymization are S and S_r, respectively, then $R_{cc} = \frac{\text{\# common similar pairs in } S \text{ and } S_r}{\text{\# similar pairs in } S}$, and $R_{mc} = \frac{\text{\# similar pairs in } S_r - \text{\# common similar pairs in } S \text{ and } S_r}{\text{\# total pairs} - \text{\# similar pairs in } S}$. The results are shown in Table 2.

From Table 2, we observed that the correct classification rate is 100%, and misclassification rate is low (the maximum is around 10%). We also notice that the results from Scheme II are very desirable. These results tell us that the data usability is still significantly preserved after performing our anonymization techniques.

	LLDOS1.0 Inside	LLDOS1.0 DMZ	LLDOS2.0.2 Inside	LLDOS2.0.2 DMZ
# original alerts	922	886	489	425
# artificial alerts injected	168	164	89	78
local privacy in S_o	0	0	0	0
local privacy in S_m (Scheme I)	0.620	0.625	0.620	0.623
local privacy in S_m (Scheme II)	7.387	7.376	7.388	7.382
local privacy in S_m (Scheme III)	7.387	7.376	7.388	7.382
global privacy for S_o	4.696	4.845	4.461	4.519
global privacy for S_m (Scheme I)	5.692	5.806	5.372	5.383
global privacy for S_m (Scheme II)	5.672	5.792	5.360	5.363
global privacy for S_m (Scheme III)	6.955	7.041	6.097	6.033

Table 1. Local and global privacy (S_o: original set, S_m: mixed set, attribute: *DestIP*).

4.3. Building Alert Correlation Graphs

In this subsection, we evaluate the effectiveness of building attack scenarios. We did experiments using all four data sets. In the first set of experiments, we identified all possibly-prepare-for relations and built correlation graphs. Due to space constraint, here we do not list prerequisites and consequences for alert types and implication relationships, but they can be found in [18] (Tables III and IV). Figure 2 shows a correlation graph from LLDOS 1.0 Inside data set (after applying Scheme I and then Scheme II).

In Figure 2, the string inside each node is the alert type followed by an alert ID. For comparison purpose, we also built the correlation graph from the original data set, where the nodes in this scenario are those without gray-color marking in Figure 2. For all those gray nodes, the ellipse nodes are in original data sets, while the rectangle nodes are artificial alerts. The attack scenario in Figure 2 tells us that attackers probed the vulnerable service using *Sadmind_Ping*, compromised *Sadmind* services using *Sadmind_Amslverify_Overflow* attacks, used *Rsh* to start *mstream* master and zombies, let *mstream* master and zombies communicate with each other (*Mstream_Zombie*), and finally launched *Stream_DoS* attacks, which is consistent with the real attack scenario involved in this data set.

In Figure 2, we observed that artificial alerts (e.g., *Sadmind_Amslverify_Overflow100091*) and false alerts (e.g., *Email_Almail_Overflow67302*) may be involved in alert correlation graphs. To further measure the quality of alert correlation graphs, we computed two measures *recall* M_r and *precision* M_p for all four data sets, where we let $M_r = \frac{\text{\# detected attacks}}{\text{\# total attacks in the real scenario}}$, and $M_p = \frac{\text{\# true alerts}}{\text{\# alerts}}$. The results are shown in Table 3.

In Table 3, the number of alerts in the correlation methods are the number of alerts in the correlation graphs. From Table 3, we observed that artificial alerts may or may not be included in correlation graphs. This tells us that attackers cannot use alert correlation graphs to distinguish between original and artificial alerts, which is desirable for privacy protection. In addition, we also noticed that, for example, precision measures from anonymized data sets are higher than those from RealSecure network sensors, but lower than those from original data sets. This is reasonable because original data sets are more precise than anonymized ones.

We also did experiments to "polish" alert correlation graphs through examining the (lower-bound) probability of each edge. In our experiments, we set probability threshold to $\frac{1}{256}$. Due to space constraint, here we only discuss the result on polishing Figure 2. After probability polishing, the number of nodes in the resulting graphs reduced from 57 to 45. We noticed that some false prepare-for relations have been removed, which may further filter out false alerts (e.g., *Email_Almail_Overflow67292*) and artificial alerts (e.g., *Sadmind_Amslverify_Overflow100009*). However, true prepare-for relations also have a chance to be ruled out (e.g., the prepare-for relation between *Rsh67542* and *Mstream_Zombie67777*), which is not desirable. So it is helpful to examine both correlation graphs with or without probability based pruning to learn attackers' activities.

5. Related Work

Our approach is related to intrusion alert correlation techniques (discussed in Introduction) and privacy-preserving techniques. Various privacy-preserving techniques have been proposed in the fields of statistical databases and data mining. In statistical databases, data swapping technique [22] is applicable to categorical attributes. It first computes the frequency counts of the combinations of attribute values for the original data set, then re-generates a new data set that preserving the computed frequency counts. There are also other approaches such as data perturbation techniques [26] and data distortion techniques [12]. A good survey about privacy-preserving techniques in statistical databases is [1].

Privacy-preserving data mining techniques [3, 8, 2] focus on constructing data mining models under the assumption that the original values of individual records are protected. Our work is also related to k-Anonymity approach

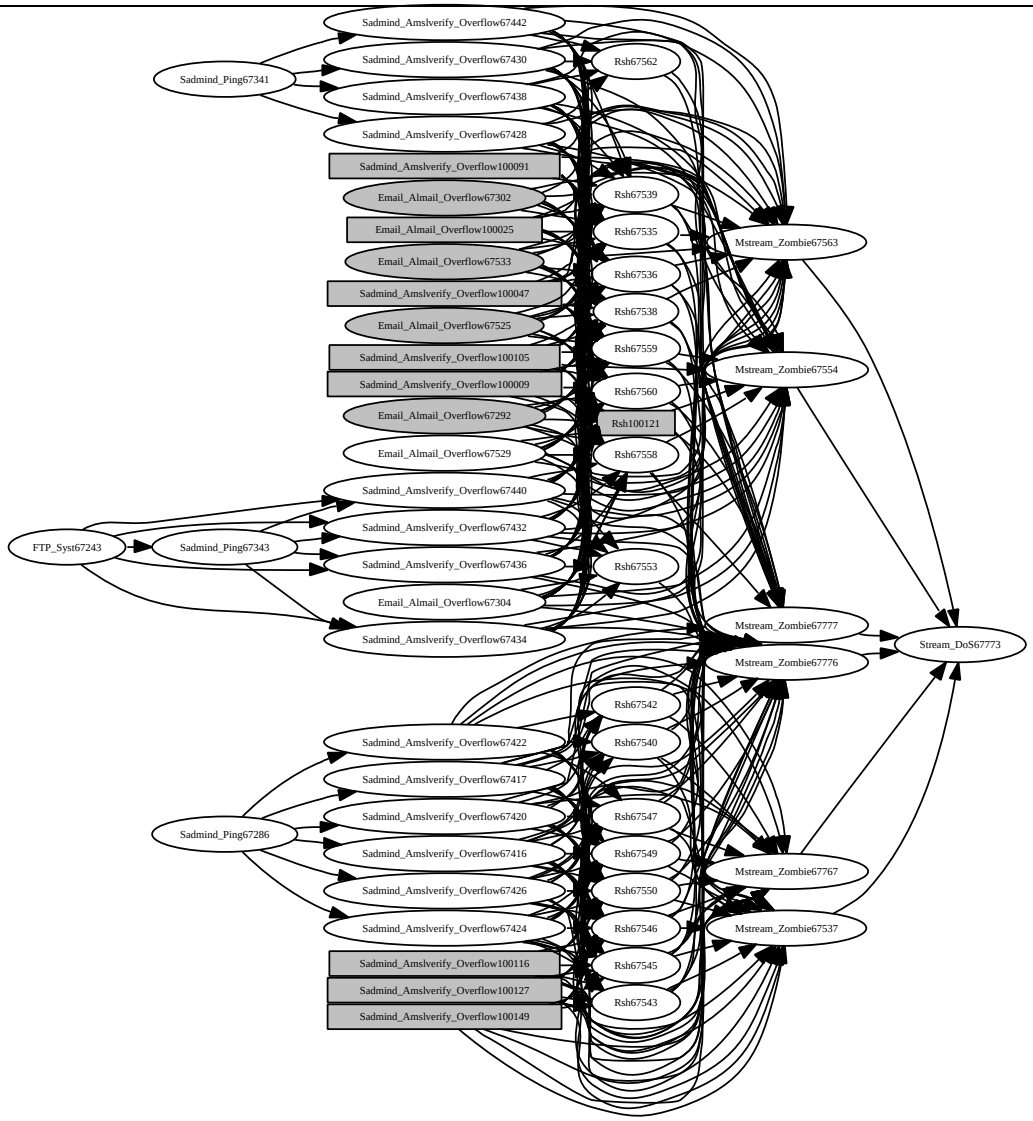


Figure 2. An alert correlation graph in LLDOS 1.0 Inside data set

		LLDOS 1.0		LLDOS 2.0.2	
		Inside	DMZ	Inside	DMZ
RealSecure	# alerts	922	886	489	425
Correlation for original sets	# alerts	44	57	13	5
Correlation for anonymized sets	# original alerts	48	61	20	5
	# artificial alerts	9	3	2	0
RealSecure	Recall M_r	61.67%	57.30%	80.00%	57.14%
Correlation for original sets	Recall M_r	60.00%	56.18%	66.67%	42.86%
Correlation for anonymized sets	RecallM_r	60.00%	56.18%	66.67%	42.86%
RealSecure	Precision M_p	4.77%	6.43%	3.27%	1.41%
Correlation for original sets	Precision M_p	93.18%	94.74%	76.92%	60.00%
Correlation for anonymized sets	Precision M_p	77.19%	84.38%	45.45%	60.00%

Table 3. Recall and precision measures in our experiments

[23] and concept hierarchy based privacy-preserving alert correlation approach [29], where concept hierarchies are used to generalize original values, while in this paper we use concept hierarchies to facilitate artificial alert generation and attribute randomization. Other approaches such as high-level language based packet anonymization [20] are also complementary to ours.

6. Conclusions and Future Work

To protect the privacy of intrusion alert data sets, in this paper we propose three schemes to anonymize sensitive attributes of alerts. Our techniques include injecting artificial alerts into original data sets, applying attribute randomization, and partitioning data sets and then performing randomization. To examine the usability of anonymized alerts, we use probability based method to estimate attribute similarity and build attack scenarios. Our experimental results demonstrated the usefulness of our techniques.

There are several directions worth further investigation. One is additional techniques to perform alert anonymization. Our techniques on injecting artificial alerts may introduce additional overhead on alert correlation analysis. Other directions include additional correlation analysis approaches that can help us understand security threats from anonymized alerts.

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] D. Agrawal and C. Aggarwal. On the design and quantification of privacy-preserving data mining algorithms. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, May 2001.

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

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

[5] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference*, December 2001.

[6] 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.

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

[8] A. Evfimievski, J. E. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2003)*, June 2003.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[10] Internet Security Systems. RealSecure intrusion detection system. `http://www.iss.net`.

[11] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security*, 6(4):443–471, Nov 2003.

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

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] The Department of Homeland Security Science and Technology directorate. PREDICT - protected repository for the defense of infrastructure against cyber threats. `https://www.predict.org`.

[20] R. Pang and V. Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of ACM SIGCOMM 2003*, August 2003.

[21] P.A. Porras, M.W. 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.

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

[23] 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.

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

[25] S. Templeton and K. Levitt. A requires/provides model for computer attacks. In *Proceedings of New Security Paradigms Workshop*, pages 31 – 38. ACM Press, September 2000.

[26] F. Traub, Y. Yemini, and H. Woźniakowski. The statistical security of a statistical database. *ACM Transactions on Database Systems*, 9(4):672–679, December 1984.

[27] J. Ullrich. DShield - distributed intrusion detection system. `http://www.dshield.org`.

[28] 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.

[29] D. Xu and P. Ning. Privacy-preserving alert correlation: A concept hierarchy based approach. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC '05)*, December 2005.

[30] 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.

A. A Lower Bound on Similarity Estimation

Lemma 1 *Given a sensitive attribute A_s with domain $\{v_1, v_2, \cdots, v_n\}$ (v_i is a possible attribute value for A_s), suppose x_1 and x_2 are two original values for A_s, and y_1 and y_2 are two image values for x_1 and x_2, respectively, after applying Scheme II. Further assume the number of desirable peer nodes in Scheme II is $\mathcal{L}$. $P(x_1 = x_2|y_1 = y_2)$ has a lower bound when $v_1, v_2, \cdots, v_n$ are in uniform distribution.*

PROOF. First, let us assume that in the original alert set, $P(x_1 = x_2) = \alpha$. Then we have

$$\begin{aligned} & P(x_1 = x_2|y_1 = y_2) \\ = & \frac{P(x_1=x_2 \wedge y_1=y_2)}{P(y_1=y_2)} \\ = & \frac{P(x_1=x_2 \wedge y_1=y_2)}{P(y_1=y_2|x_1=x_2)P(x_1=x_2)+P(y_1=y_2|x_1\neq x_2)P(x_1\neq x_2)} \\ = & \frac{\alpha}{\alpha+\frac{1-\alpha}{\mathcal{L}}} \\ = & \frac{\mathcal{L}}{\mathcal{L}-1+\frac{1}{\alpha}} \end{aligned}$$

Now let us discuss how to compute α. Suppose the probabilities for possible attribute values $v_1, v_2, \cdots, v_n$ in A_s's domain are $p_1, p_2, \cdots, p_n$, respective, where $p_1+p_2+\cdots+p_n = 1$. Then $\alpha = p_1^2 + p_2^2 + \cdots + p_n^2$.

Based on Cauchy's inequality $(\sum_{i=1}^n a_i b_i)^2 \leq (\sum_{i=1}^n a_i^2)(\sum_{i=1}^n b_i^2)$, we can derive $\alpha = \sum_{i=1}^n p_i^2 \geq (\sum_{i=1}^n p_i)^2/n = \frac{1}{n}$.

Then we know that the minimum value of α is $\frac{1}{n}$, where $p_1 = p_2 = \cdots = p_n = \frac{1}{n}$ (uniform distribution). Next we prove $P(x_1 = x_2|y_1 = y_2) = \frac{\mathcal{L}}{\mathcal{L}-1+\frac{1}{\alpha}}$ is monotonically increasing when $0 < \alpha < 1$.

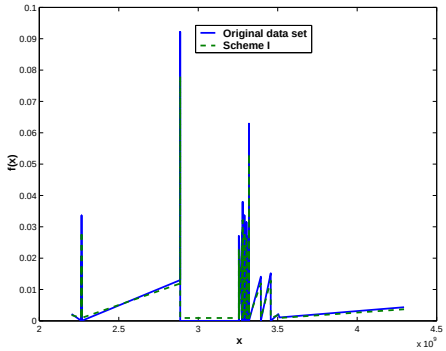
Assume $0 < \alpha_1 < \alpha_2 < 1$. Then $\frac{1}{\alpha_1} > \frac{1}{\alpha_2}$. Next we have $\mathcal{L} - 1 + \frac{1}{\alpha_1} > \mathcal{L} - 1 + \frac{1}{\alpha_2}$. Finally we get $\frac{\mathcal{L}}{\mathcal{L}-1+\frac{1}{\alpha_1}} < \frac{\mathcal{L}}{\mathcal{L}-1+\frac{1}{\alpha_2}}$.

To summarize, we know that $P(x_1 = x_2|y_1 = y_2)$ have a minimum value $\frac{\mathcal{L}}{\mathcal{L}-1+\frac{1}{\alpha}}$ when $v_1, v_2, \cdots, v_n$ are in uniform distribution. □

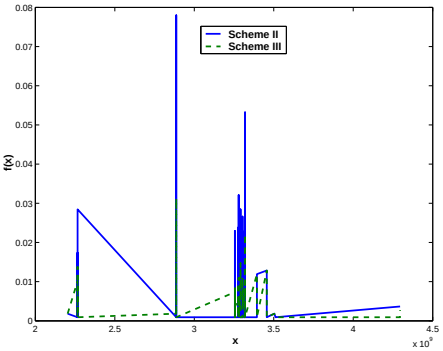
B. Attribute Distributions in Our Experiments

In this section, we listed attribute *DestIP* distributions (i.e., PMFs) in each original data set, in each mixed alert set after applying Scheme I, in each mixed alert set after applying Scheme II, and in each mixed alert set after applying Scheme III. The results are shown in Figure 3.

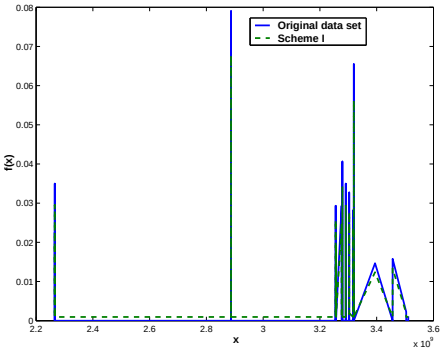
Note that in Figure 3, each destination IP address is transformed into an integer. Specifically, if the format of an IP address is $A.B.C.D$ where A, B, C and D are integers between 0 to 255, then $A.B.C.D$ is transformed to an integer $x = A \times 2^{24} + B \times 2^{16} + C \times 2^8 + D$.



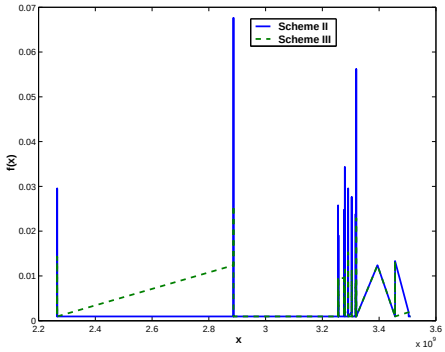
(a) LLDOS1.0 Inside (Original and Scheme I)



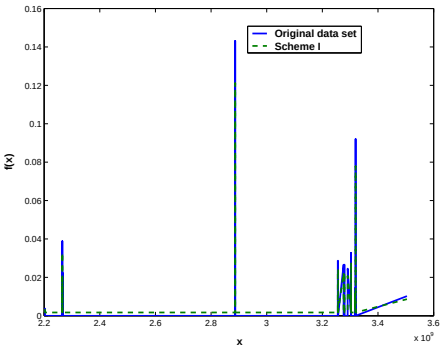
(b) LLDOS1.0 Inside (Schemes II and III)



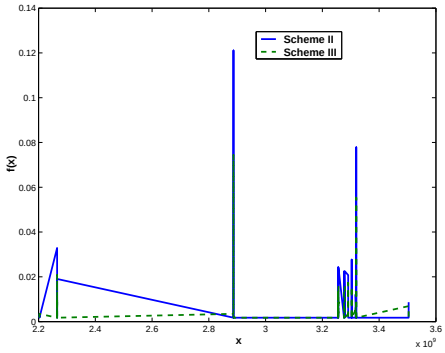
(c) LLDOS1.0 DMZ (Original and Scheme I)



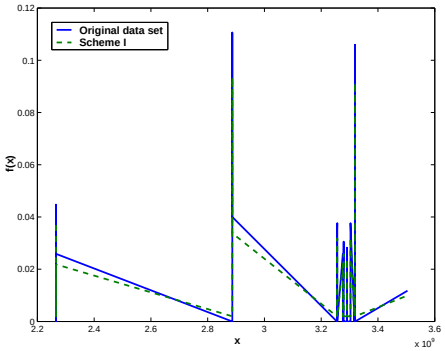
(d) LLDOS1.0 DMZ (Schemes II and III)



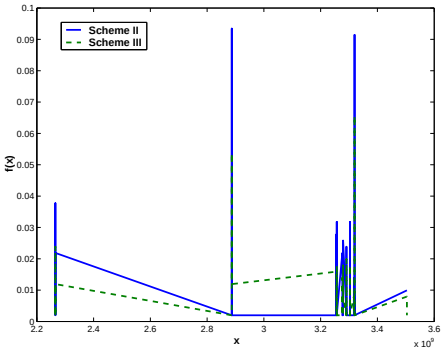
(e) LLDOS2.0.2 Inside (Original and Scheme I)



(f) LLDOS2.0.2 Inside (Schemes II and III)



(g) LLDOS2.0.2 DMZ (Original and Scheme I)



(h) LLDOS2.0.2 DMZ (Schemes II and III)

Figure 3. PMFs in original alert sets and after applying Schemes I, II and III