Evaluating the Utility of Anonymized Network Traces for Intrusion Detection

Kiran Lakkaraju
Department of Computer Science
University of Illinois, Urbana-Champaign
Urbana, Illinois
klakkara@illinois.edu

Adam Slagell
National Center for Supercomputing Applications
University of Illinois, Urbana-Champaign
Urbana, Illinois
slagell@ncsa.uiuc.edu

ABSTRACT

To intelligently create policies governing the anonymization of network logs, one must analyze the effects of anonymization on both the security and utility of sanitized data. In this paper, we focus on analyzing the utility of network traces post-anonymization. Any measure of utility is subjective to the type of analysis being performed. This work focuses on utility for the task of attack detection since attack detection is an important part of an incident responders daily responsibilities. We employ a methodology we developed that analyzes the effect of anonymization on Intrusion Detection Systems (IDS), and we provide the first rigorous analysis of single field anonymization on IDS effectiveness. Through this work we can begin to answer the questions of whether the field affects anonymization more than the algorithm; which fields have a larger impact on utility; and which anonymization algorithms have a larger impact on utility.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection

General Terms

Security

Keywords

Anonymization, Data Sanitization, FLAIM, Intrusion Detection, Metrics

1. INTRODUCTION

The ability to safely share log files and network traces has become increasingly important to several communities: networking research, computer security research, incident response, and education [23]. Synthetically generated data is abundant, but has been highly criticized for many uses [15]. Honeynets can be useful in generating exercises for students and to help meet the needs of educators, but very few honeynets have been setup on a scale to generate some of the large, cross-sectional data sets needed by computer security researchers [24]. Furthermore, using a honeynet does not necessarily release the owner from all legal responsibility when sharing data [20]. Lastly, nothing but real data will suit the needs of the incident responder who must share data about specific attacks under investigation on real machines. Therefore, there is a high demand for methods to share *real* log files and network traces within several communities.

At the same time as there is an increased need to share these data sets, there is also increased reluctance. First, many data owners recognize the inherent security risks of releasing detailed information that could be used to map out their own networks, services or sensor locations [2, 4, 10, 13, 18, 19]. Secondly, there are serious privacy concerns that companies have about releasing customer data, especially in light of recent incidents where some publicly released data—believed to be anonymous—leaked information about specific users [7, 17]. Lastly, recent research has questioned the legality of releasing data [20].

The burgeoning field of data sanitization has helped address this tension by providing organizations who wish to share their data with new tools to anonymize computer and network logs [18, 21, 22, 25, 26]. However, little has been done to help users negotiate the difficult trade-off between the data owner's need for security and privacy, and the data analyst's need for high quality data—what is called the *utility vs. security trade-off* [21]. As anonymization is an inherently lossy process, and the data analyst wants information as close to the original as possible, there is always this tension and a need to negotiate policies that meet the needs of both parties.

Necessary to solving this problem of negotiating policies for data sanitization is the ability to analyze the effects of anonymization on both the security of the sanitized data and the utility left after anonymization. In this paper, we focus upon the latter problem of evaluating the effects of anonymization on the utility of the data sets to be shared. Those wishing to use the data must understand how anonymization can affect utility, so that they can express constraints of what cannot be anonymized to the data owner. Then other tools based on a predicate logic we developed in [9] can be used to determine whether or not a policy exists that can meet the utility needs of the researcher and the security/privacy needs of the data owner.

Of course, utility is subjective since it depends upon who is using the data, or more specifically, for what purpose it is being used. Hence, what is important to a researcher in

SECURECOMM 2008 September 22–25, 2008, Istanbul, Turkey

the network measurements community may be completely irrelevant to the incident responder. As a first step we have focused our work on evaluating utility for attack detection.

The task of attack detection is an important part of the incident responder's daily job. When investigating broad attacks, of which their organization is only a part, they may have to settle for anonymized logs from the other sites involved. A similar case occures when using a distributed or collaborative intrusion detection system that crosses organizational boundaries. Output from the sensors may need to be anonymized. Not only is attack detection important in these "real world" applications, but it is important to the intrusion detection research community, as well. This community has often complained that their only good data sets to test new technologies against are synthetic. However, if they can still do their analysis with anonymized data, then they are more likely to obtain large, real data sets.

The utility of a data set is not only constrained by the type of analysis being done with it but also the type of data being shared. We have chosen to look at the effects of anonymization on one of the most commonly shared and general types of data, the pcap formatted network trace. From this type of data, many others can be derived (e.g., NetFlows [1]).

To quantitatively measure the ability to identify attacks in anonymized data, we developed what we call the *IDS Utility Metric*. This measurement is based on how well attacks are detected by an IDS on an *anonymized* data set as compared to the same data set *unanonymized*. By doing this, we can automate an objective process to help us answer several questions: (1) How does the anonymization of fields affect the ability to detect an attack, (2) Which has greater impact on utility, the field being anonymized or the anonymization algorithms being used, and (3) Which anonymization algorithms have a larger impact on the utility of a log? In this paper, we present the results of anonymizing a portion of the 1999 MIT/Lincoln Labs DARPA data set—by using the FLAIM [21] anonymization framework—with over 150 separate anonymization policies to help us answer these questions. We use version 2.7.0.1 of the popular Snort Intrusion Detection System in these simulations, along with the publicly available signature set downloaded in December 2006 as well as the community ruleset version 2.4.

The rest of the paper is organized as follows. Section 2 describes the anonymization algorithms used by FLAIM in our experiments, while Section 3 describes the methodology and setup of our experiments. In Section 4, we present our results and analysis. We survey the related work in Section 5, and state our conclusions, as well as scope out future work to be done, in Sections 6 and 7.

2. FLAIM: FRAMEWORK FOR LOG ANONYMIZATION AND INFORMATION MANAGEMENT

We chose to use FLAIM [21] as our anonymization engine for several reasons. First, we can easily script its execution for a multitude of tests. Second, it has a very flexible XML policy language that makes it simple to generate hundreds of unique anonymization policies. Third, it can anonymize as many or more fields in PCAP traces as any other anonymization tool. Lastly, FLAIM has a very rich set of anonymization algorithms that can be applied to all these fields. With these properties, it was the ideal tool for our experiments.

2.1 Anonymization Algorithms

FLAIM implements a plethora of anonymization algorithms for several data types. The three basic data types available to most anonymization algorithms are *binary, string,* and *numeric*. Additionally, there are a few special data types like *timestamps*. Binary data is treated as a string of bits with no special structure. Algorithms anonymizing binary data output binary data of the same length. String data are variable length, terminated by a null character. Anonymization algorithms that take in strings will also output strings. However, the length may change. Numeric data is interpreted as a number of a given base, specified in the anonymization policy. This is useful, for example, when working with decimal numbers like a port number. There, one may want to act on individual digits, rather than bits (e.g., replacing the last 3 digits with 0's).

[12] lists the different anonymization algorithms in FLAIM along with the data types they operate upon. Further information on these algorithms can be found in [6].

2.2 Anonymization Policies in FLAIM

FLAIM provides an expressive and powerful method for specifying anonymization policies that can be modified at run time, thus enabling efficient automation. An anonymization policy is an XML file that specifies the anonymization algorithms that should be applied to the various fields in the log, along with any special parameters to be passed to those algorithms.

3. METHODOLOGY

For all of our experiments, we used a subset of the 1999 DARPA evaluation data set. The Defense Advanced Research Projects Office (DARPA) created an Intrusion Detection Evaluation testbed in 1998 and 1999. Data was captured from a simulated network that was subjected to various attacks. This data set has been frequently used in evaluating intrusion detection systems since its creation [14]. Thus, we found it appropriate to use in evaluating the effects anonymization of data can have on intrusion detection.

As we mentioned, we used but a portion of the 1999 DARPA data set. Specifically, we used the *inside* tcpdump data from Wednesday of the second week of the evaluation. Since FLAIM currently does not anonymize more esoteric network protocols, we kept only the TCP and UDP traffic—by far the majority—for our experiments. This data set may seem small, but it was still very dense, including 81 attacks. Furthermore, the experiments would not have been computationally feasible if we had used a trace vastly larger than the this one—which is already several hundred megabytes.

The *IDS Utility Metric* is constructed as follows. First, the unanonymized data set is processed by Snort to produce the "baseline" set of alerts, using the default Snort rule sets. We consider this the ideal set of alerts for the data set. Next, we take the same data set, and anonymize it—in our case, with FLAIM. We then take the anonymized data set, and we run Snort against it with the same rule set as before. The alerts generated from the unanonymized file are used as a baseline against which the alerts generated by the anonymized file are compared. The difference between the alerts in the anonymized file, versus the unanonymized file,

is used as a measure of the loss of utility in the log. The larger the difference, the more information was not available in the logs in order for the IDS to correctly identify an attack.

FLAIM schemas specify a set of anonymization algorithms that are appropriate for each field in a pcap log. These are summarized in [12]. In this evaluation, we consider anonymization policies that transform single fields. There are 152 single field policies that can be generated for pcap data. Each anonymization algorithm also has parameters that affect how it anonymizes the field. We will not go over the parameters in detail but instead refer readers to the FLAIM manual [6].

We iterate the process described above over each of the 152 anonymization policies, comparing the results pre- and post-anonymization. We describe how we compare these data sets in the next section, while the section after discuses the actual metric in more detail.

3.1 Comparing Snort Alerts

Alerts generated by Snort are defined by several properties (a full list of the properties is included in [12]). Each alert is associated with a specific packet. The relevant alert fields—for our purposes—are listed below:

timestamp : the timestamp from the offending packet.
sig_generator : the part of Snort generating alert.
sig_id : the Id. number of the signature that was fired.
msg : description of the alert.
proto : the protocol of the offending packet.
src : the source IP address of the offending packet.
srcport : The source port of the offending packet.
dst : the destination IP address of the offending packet.
dstport : the destination port of the offending packet.
id : Packet Id.

To determine whether two alert sets are equal we need a way of determining if two alerts are equal. Normally this can be done by comparing each field of the alerts. However, in this case the alerts generated from the anonymized log will cause alerts that are actually equal to appear unequal. To overcome this, we compare alerts on fields which will not change due to anonymization. This leads to two distinct field sets that must be used when comparing alerts. Field set 1 comprises the fields timestamp,sig_id, and id. Field set 2 is comprised of sig_id, src, srcport, dst, dstport, id, and tcpseq fields. Field Set 1 is used when the timestamp field has not been anonymized. Field Set 2 is used when the timestamp field has been anonymized.

3.2 Metrics for evaluating utility

The purpose of anonymization is to share logs while hiding sensitive information. Anonymization, while inherently an information reducing procedure, must be measured in terms of the amount of information that is lost in the file. However, by anonymizing we can introduce new false patterns into the data. The "best" anonymization policy should minimize information loss, while not adding any new false patterns to the log.

We can consider the IDS process as a pattern classification process. The data set is the input, and the IDS classifies each packet as malicious or benign. The alerts generated from the unanonymized data (which we call the baseline data) are considered to be the correct analysis of the data set. We then compare the alerts generated by the anonymized data to the baseline data alert set.

We will compare the anonymized alerts with the unanonymized alerts over three metrics. Let the set of alerts generated from the baseline file be called $\mathcal{A}_{baseline}$ and the set of alerts generated from the anonymized file be called $\mathcal{A}_{anony}$. We define three metrics:

True Positive $TP = \mathcal{A}_{anony} \cap \mathcal{A}_{baseline}|$ The number of alerts in $\mathcal{A}_{anony}$that are also in $\mathcal{A}_{baseline}$.

False Positive $FP = |\mathcal{A}_{anony} - (\mathcal{A}_{anony} \cap \mathcal{A}_{baseline})|$ The number of alerts that were generated by the anonymized file, but were not in the baseline file.

False Negative $FN = |\mathcal{A}_{baseline} - (\mathcal{A}_{anony} \cap \mathcal{A}_{baseline})|$ The number of alerts that were not caught by the anonymized file.

The True Positive rate indicates how much of the information was preserved in the anonymized log. The False Positive rate indicates how many additional patterns were added to the log through anonymization. The False Negative rate indicate the amount of information that was removed from the log. An optimal anonymization policy should make sure both False Positive and False Negative are low while maximizing the True Positive rate.

While the False Positive rate is an important factor, of primary importance is the False Negative rate, as it indicates the loss of information through anonymization. For the remainder of this paper, we use both False Positive and False Negative rates as a measure of the utility of a log post-anonymization, but focus more on the False Negative rate.

4. RESULTS AND ANALYSIS

In this section, we describe the results of our experiments with single field anonymization policies. These experiments provide a substantive start to answering these questions:

- What affects utility more, the fields that are anonymized or the anonymization algorithm?
- Which fields have a larger impact on the utility of a log?
- Which anonymization algorithms have a larger impact on the utility of a log?

To answer these questions, we evaluated all 152 pairs of fields and anonymization algorithms. For each pair, the number of alerts generated by Snort was calculated. The alerts generated for each pair were compared with the baseline alerts (See Table 1 and Table 2). False Positives/False Negatives were calculated based on the definitions above, whose detailed results we discuss later.

The unanonymized file produced 81 alerts. The number and types of alerts produced are summarized in Table 3. There are a total of 81 alerts generated in the baseline file, but only 19 unique types of alerts.

4.1 Fields or Anonymization Algorithms?

It is important to understand which causes a greater impact on utility; the field that is being anonymized or the anonymization algorithm that is being applied.

To evaluate the effects of anonymizing a field we calculate the *marginal* of a field. The marginal of a field is the average

Table 1: Alerts generated for Anonymization-Field pairs. AnonyAlg is the anonymization algorithm used; Field is the field which it was applied on; NumAlerts is the number of alerts that were generated; NumTypesOfAlerts is the number of different *types* of alerts generated. Table 1 of 2.

AnonyAlg	Field	Num Alerts	Types Of Alerts
BinaryBlackMarker	SRC_MAC	81	19
BytesTruncation	SRC_MAC	81	19
Annihilation	SRC_MAC	81	19
MacRandomPermutation	SRC_MAC	81	19
BinaryBlackMarker	DST_MAC	81	19
BytesTruncation	DST_MAC	81	19
Annihilation	DST_MAC	81	19
MacRandomPermutation	DST_MAC	81	19
IPv4PrefixPreserving	IPV4_SRC_IP	1014	5
BinaryBlackMarker	IPV4_SRC_IP	9	4
Annihilation	IPV4_SRC_IP	9	4
RandomPermutation	IPV4_SRC_IP	9	4
NumericTruncation	IPV4_SRC_IP	9	4
IPv4PrefixPreserving	IPV4_DST_IP	759	5
BinaryBlackMarker	IPV4_DST_IP	9	4
Annihilation	IPV4_DST_IP	9	4
RandomPermutation	IPV4_DST_IP	14	5
NumericTruncation	IPV4_DST_IP	9	4
BinaryBlackMarker	IPV4_ID	81	19
Annihilation	IPV4_ID	81	19
NumericTruncation	IPV4_ID	81	19
RandomPermutation	IPV4_ID	81	19
Classify	IPV4_ID	81	19
Annihilation	IPV4_OFFSET	81	19
BinaryBlackMarker	IPV4_TTL	81	19
Annihilation	IPV4_TTL	81	19
NumericTruncation	IPV4_TTL	81	19
RandomPermutation	IPV4_TTL	81	19
Classify	IPV4_TTL	81	19
Annihilation	IPV4_CHECKSUM	81	19
BinaryBlackMarker	TCP_DST_PORT	520290	2
NumericTruncation	TCP_DST_PORT	457410	6
Substitution	TCP_DST_PORT	923033	2
Annihilation	TCP_DST_PORT	923033	2
RandomPermutation	TCP_DST_PORT	18	2
Classify	TCP_DST_PORT	521670	2
BinaryBlackMarker	TCP_SRC_PORT	380527	4
NumericTruncation	TCP_SRC_PORT	294519	6

Table 2: Alerts generated for Anonymization-Field pairs. AnonyAlg is the anonymization algorithm used; Field is the field which it was applied on; NumAlerts is the number of alerts that were generated; NumTypesOfAlerts is the number of different *types* of alerts generated. Table 2 of 2

AnonyAlg	Field	Num Alerts	Types Of Alerts
Substitution	TCP_SRC_PORT	922171	6
Annihilation	TCP_SRC_PORT	922171	6
RandomPermutation	TCP_SRC_PORT	20	4
Classify	TCP_SRC_PORT	381954	4
BinaryBlackMarker	TCP_SEQUENCE	81	19
NumericTruncation	TCP_SEQUENCE	65	15
Annihilation	TCP_SEQUENCE	65	15
Classify	TCP_SEQUENCE	81	19
BinaryBlackMarker	TCP_ACK_NO	62	16
NumericTruncation	TCP_ACK_NO	56	14
Annihilation	TCP_ACK_NO	56	14
Classify	TCP_ACK_NO	81	19
BinaryBlackMarker	TCP_FLAGS	5	3
NumericTruncation	TCP_FLAGS	5	3
Annihilation	TCP_FLAGS	5	3
BinaryBlackMarker	TCP_WINDOW	81	19
NumericTruncation	TCP_WINDOW	81	19
Annihilation	TCP_WINDOW	81	19
Classify	TCP_WINDOW	81	19
Annihilation	TCP_CHECKSUM	81	19
BinaryBlackMarker	TCP_URGENT	81	19
NumericTruncation	TCP_URGENT	81	19
Annihilation	TCP_URGENT	81	19
Classify	TCP_URGENT	81	19
Annihilation	TCP_OPTIONS	81	19
BinaryBlackMarker	UDP_DST_PORT	81	19
NumericTruncation	UDP_DST_PORT	81	19
Substitution	UDP_DST_PORT	81	19
Annihilation	UDP_DST_PORT	81	19
RandomPermutation	UDP_DST_PORT	81	19
Classify	UDP_DST_PORT	81	19
BinaryBlackMarker	UDP_SRC_PORT	81	19
NumericTruncation	UDP_SRC_PORT	81	19
Substitution	UDP_SRC_PORT	81	19
Annihilation	UDP_SRC_PORT	81	19
RandomPermutation	UDP_SRC_PORT	81	19
Classify	UDP_SRC_PORT	81	19
Annihilation	UDP_CHECKSUM	81	19
RandomTimeShift	TS_SEC	81	19
TimeUnitAnnihilation	TS_SEC	81	19
Annihilation	TS_SEC	81	19
BinaryBlackMarker	TS_SEC	83	20
TimeEnumeration	TS_SEC	399	19
Annihilation	TS_USEC	81	19

number of false positive/false negatives over all anonymization algorithms. Similarly, the marginal of an anonymization algorithm is the false positive/false negative rate averaged over all fields. The marginal provides a concise summary of the effect of anonymizing a particular field or using a particular anonymization algorithm.

Figure 1 shows the false positive and false negative marginals of the fields. The full data for these graphs is in Table 5. Figure 2 shows the false positive and false negative marginals of the anonymization algorithms. The full data for these graphs is in Table 4.

Figure 1 indicates that the majority of fields generate no false positives or false negatives. The fields for which this is true (such as DST_MAC) did not affect the utility of the log under any anonymization algorithm. We can conclude that anonymizing these fields has no impact on the utility of a log, with respect to the IDS metric.

Consider the false positive rate first. We can see that very few fields generated any false positives. This indicates that anonymization, usually, does not add new patterns.

The exception to this is anonymizing the fields TCP_DST_PORT and TCP_SRC_PORT. These fields produce a large

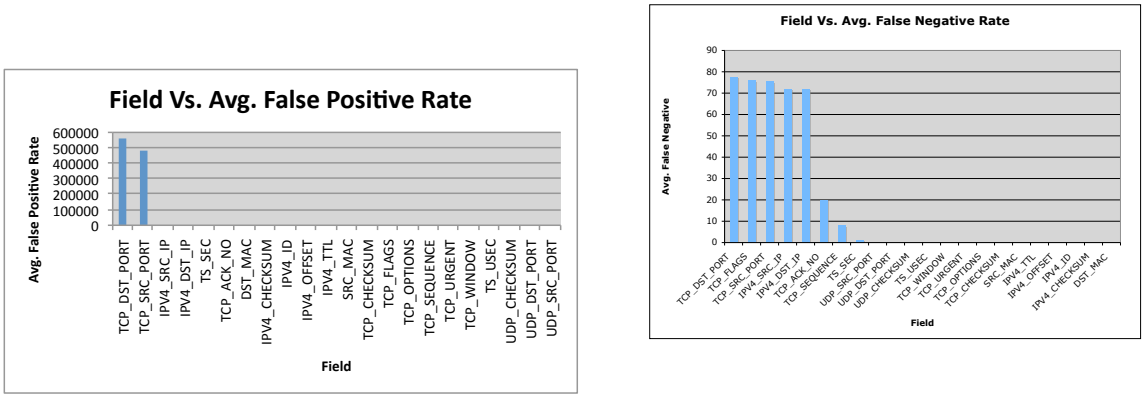

Figure 1: The left hand chart shows the marginal of each field with respect to false positives (i.e, the average number of false positives for a field, averaged over all anonymization algorithms). The right hand chart shows the marginal of each field with respect to false negatives.

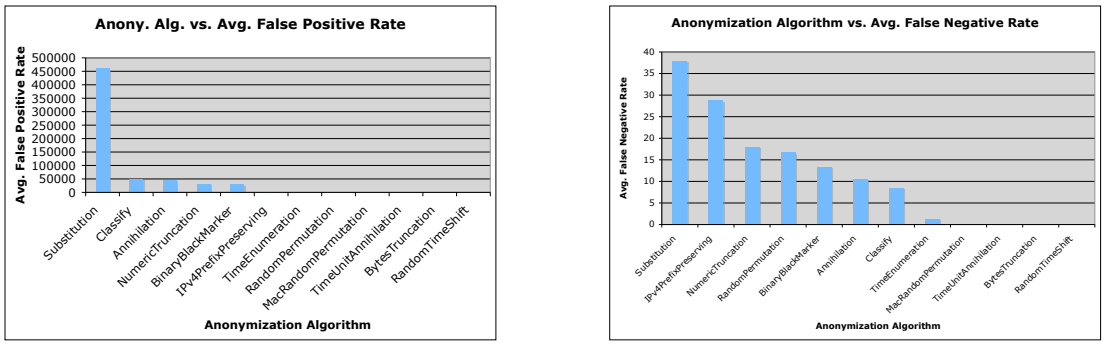

Figure 2: The left hand chart shows the marginal of all the anonymization algorithms, with respect to false positives. The right hand chart shows the marginal of all anonymization algorithms with respect to false negatives.

number of false positives. In the next section we describe why this happens.

In terms of false negatives, a minority of the fields generate a substantial amount. Most of the fields generate zero false negative alerts.

Looking at the marginals over anonymization algorithms, we get a slightly different picture. As with the marginals of fields, we get very few anonymization algorithms that generate a large amount of false positives. In fact, the substitution algorithm is the only one that generates a substantial number of false positives. The reason for this specific case will be explained in the next section.

The false negative marginals for anonymization algorithms are quite different from the false negative marginals for fields. We can see in Figure 2 that the false negatives are distributed over nearly all the anonymization algorithms. This is unlike the false negative marginals for fields, which are concentrated on a few fields.

The more uniform distribution of false negatives over fields indicates that for many of the fields the exact anonymization algorithm does not matter, i.e. no anonymization algorithm applied to the field will impact utility. For only a few fields will utility be impacted.

While we do not have enough evidence yet to state it conclusively, we can certainly say that the evidence is leaning towards the field being anonymized having a greater impact than the anonymization algorithm being used. If it was the other way around, we would expect false positives and false negatives to be more evenly distributed over all the fields.

4.2 Which fields have higher impact on utility?

As Figure 1 and Table 5 clearly show, the majority of fields do not generate false positives. The fields that do are TCP_DST_PORT, TCP_SRC_PORT, IPV4_SRC_IP, IPV4_DST_IP, TS_SEC, and TCP_ACK_NO.

We can see that several of the fields resulted in an average of 81 alerts. These fields (shown in italics in the table) had 0 error. Judging from these results, we can see that this set of fields did not affect the utility of the log as measured by

Table 3: Alerts generated in the baseline unanonymized file. AlertID is the id of the alert; Num is the number of that type of alert generated; Desc is a description of the alert

AlertID	Num	Desc
1	1	(portscan) TCP Portscan [1]
323	1	FINGER root query
330	1	FINGER redirection attempt
332	1	FINGER 0 query
356	1	FTP passwd retrieval attempt
359	1	FTP satan scan
503	4	MISC Source Port 20 to <1024
1200	9	ATTACK-RESPONSES Invalid URL
1201	12	ATTACK-RESPONSES 403 Forbidden
1288	4	WEB-FRONTPAGE /_vti_bin/ access
1292	30	ATTACK-RESPONSES directory listing
1418	2	SNMP request tcp
1420	2	SNMP trap tcp
1421	1	SNMP AgentX/tcp request
2467	1	NETBIOS SMB D$ unicode share access
2470	1	NETBIOS SMB C$ unicode share access
2473	1	NETBIOS SMB ADMIN$ unicode share access
3151	6	FINGER / execution attempt
3441	2	FTP PORT bounce attempt

the IDS metric.

TCP_DST_PORT and TCP_SRC_PORT generated the most false alerts on average. Upon inspection of the generated alert files, we can see that most of the pairs generated only 2 types of alerts. In the case of BinaryBlackMarker there were 520286 alerts of type 524. Alert 524 is "BAD-TRAFFIC tcp port 0 traffic". The other anonymization algorithms produced the same pattern—the majority of alerts were of type 524.

The reason for this is the value we substitute for the port field. The BinaryBlackMarker, Substitution, NumericTruncation, and Annihilation algorithms all replaced the port field with 0. This resulted in the alert being triggered for nearly all the packets in the log (see [12] for the parameter settings of the anonymization algorithms). The same problem occurs for the TCP_SRC_PORT field.

The RandomPermutation algorithm replaced the port number with another, random port number, thus infrequently causing the generation of the BAD-TRAFFIC alert. It is clear from this that the false positive rate was greatly affected by the choice of the substitution port. However, the effect would have been less pronounced had we counted the types of new alerts rather than the raw alerts, themselves.

Arguably, it is the false negative count that is most important in determining the utility of a log. A low false negative count indicates that little information was lost in the process of anonymization. In terms of false negatives, we find that there are 8 fields that have an impact on the average false negative rate: TCP_DST_PORT,TCP_FLAGS,TCP_SRC_PORT, IPV4_DST_IP, IPV4_SRC_IP,TCP_ACK_NO, TCP_SEQUENCE, and TS_SEC.

4.3 What anonymization algorithms have a higher impact on utility?

Figure 2 summarizes the false positive and false negative rates with respect to anonymization algorithm. For each anonymization algorithm, the average false positive/false negative rate is calculated over all the fields. Table 4 contains the data for the graphs.

We can see from these that most anonymization algorithms have an impact on the utility of a log. In contrast, the field data that we saw before showed strong structure in what fields affected utility. When viewed from the perspective of anonymization algorithms, there is no single anonymization algorithm that stands out.

It might seem like the substitution algorithm has the largest effect with a huge number of alerts. However, this is because of the parameter setting used. By looking at the false negative rate we can see that while substitution still has a large effect on utility, most of the other algorithms have an effect as well.

Table 4: False negative/positive counts for an anonymization algorithm, aggregated over all log fields.

Anonymization Alg.	False Negatives	False +
Annihilation	10.62	47312.69
BinaryBlackMarker	13.43	30027.37
BytesTruncation	0	0
Classify	8.5	50200.83
IPv4PrefixPreserving	28.8	351
MacRandomPermutation	0	0
NumericTruncation	17.96	32692.13
RandomPermutation	16.72	2.11
RandomTimeShift	0	0
Substitution	38	461298.5
TimeEnumeration	1.25	80.75
TimeUnitAnnihilation	0	0

5. RELATED WORK

After performing our initial experiments, a new piece of work closely related to ours appeared [26]. Though this is mostly a paper presenting another anonymization tool for pcap logs, at the end the authors introduce a similar method of analyzing the effects of anonymization of pcap traces by use of an IDS. Their analysis is cursory, only considering anonymization for a few different fields. Also, their analysis only considered the number of alerts. They neglected distinctions such as false positive and negative rates. We were unable to reproduce their results (which may be because they used an unspecified subset of the LBNL data set[2]), but more troubling is the use of this data set in the first place. To evaluate the effects of anonymization, one must start with unanonymized data. However, this LBNL data set is already anonymized, and therefore, they have no clean baseline with which to compare the re-anonymized data.

In [11], the authors do an even simpler metric of utility that just counts the number of IDS alerts. However, they

[2]http://www.icir.org/enterprise-tracing/

Table 5: False negative/positive counts for a field, aggregated over all anonymization algorithms

Field	False Negatives	False Positive
DST_MAC	0	0
IPV4_CHECKSUM	0	0
IPV4_DST_IP	72	151
IPV4_ID	0	0
IPV4_OFFSET	0	0
IPV4_SRC_IP	72	201
IPV4_TTL	0	0
SRC_MAC	0	0
TCP_ACK_NO	20	2.75
TCP_CHECKSUM	0	0
TCP_DST_PORT	77.67	557572.33
TCP_FLAGS	76	0
TCP_OPTIONS	0	0
TCP_SEQUENCE	8	0
TCP_SRC_PORT	75.5	483554.83
TCP_URGENT	0	0
TCP_WINDOW	0	0
TS_SEC	1.2	65.2
TS_USEC	0	0
UDP_CHECKSUM	0	0
UDP_DST_PORT	0	0
UDP_SRC_PORT	0	0

only use a single policy (per tool) because their goal is not to evaluate how different policies affect an IDS, but different tools. So they compare how Snort performs on an unanonymized data set, this same data set anonymized with *tcpdpriv* [16] and the data anonymized with the tool they are presenting in their paper. By showing that they lose fewer alerts, they claim to have a better anonymization tool than *tcpdpriv*.

A few people have created general entropy-based metrics of anonymity [3, 5]. The goal here is very complimentary, that is, to measure the risk of re-identification posed by an adversary. As such, they are working on the flip-side of the same problem, to balance the utility needs of the data analyst with the security/privacy needs of the data owner. Creating a good anonymization policy requires a way to measure both utility and security.

Lastly, a reviewer pointed out a recent paper [8] that employees an analogous approach to a different type of utility measure. In this paper, the authors are investigating the affects of data sanitization on tools which detect insider threats from specialized syslog data. While the type of data and the measure of utility is different, it approaches the problem in the same spirit by investigating the effects of anonymization empirically.

6. CONCLUSIONS

Anonymization can be a powerful tool to allow greater cooperation between organizations. The need for cooperation is strikingly clear. However, a clear understanding of the needs of the data provider and the client is necessary before flexible, effective sharing between organizations can occur.

The objective of this paper has been to begin to formally evaluate the utility vs. security trade off. The *IDS metric* is simple yet effective in evaluating the difference in utility when anonymizing different fields in different ways.

In this paper, we have focused on answering three questions: whether the field affects anonymization more than the algorithm; which fields have a larger impact on utility; and which anonymization algorithms have a larger impact on utility.

We have provided a thorough evaluation of single field anonymization polices upon pcap formatted network traces. We found that the primary impact on the utility of a log is not the particular anonymization algorithm, but rather the field that was anonymized.

In addition, we were able to empirically show a range of utilities for a log based on the field that was anonymized. The loss of utility was largest for ports and IP addresses. There was some loss of utility for the fields of ID, sequence number, flags, timestamp, and ACK number. However, for many of the fields there was no change in utility when anonymized.

This empirical evaluation provides the basis for further work on studying the impact of more complex anonymization schemes on the utility of a log.

7. FUTURE WORK

It is clear that evaluating utility via Snort generated IDS alerts will cause the utility to depend upon the rule set. Complementary to the current approach would be to analyze the ruleset and see what fields are often part of a rule. From this we can estimate the loss in utility by anonymizing those fields. The more often a field is seen in the rule set, the larger the loss of utility.

The strength of an anonymization algorithm is a measure of how difficult it is to "break" or "de-anonymize" a log that has been anonymized via the algorithm. It is beneficial to have strong anonymization algorithms so that attackers will have a difficult time breaking the algorithm. As [21] points out, there is a trade off between the security of an anonymization algorithm and the utility of the log. While we have discussed the loss in utility, we have not discussed the strength of an anonymization algorithm.

Our work is currently limited to anonymization policies for just one field. Our next step would be to extend this work to multiple field anonymization policies and to work with more realistic data. The DARPA evaluation data set is useful because it is supervised, however it is still synthetic. In the future, we will be gaining access to other large *unanonymized* data sets that we can use instead of the DARPA data.

Finally, we have focused on utility for just one task, attack detection. An incident responder does more than just detect attacks, and in the future, we could look at how anonymizing logs affects other important security related tasks—such as alert correlation.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation (NSF) under Award No. CNS 0524643. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF. We would like to thank Jim Basney for his thoughtful critique of this paper and the rest of the LAIM Working Group at the NCSA. Lastly, we would like to thank the anonymous reviewers for their constructive comments that have helped reshape this final version.

9. REFERENCES

[1] Argus: Qosient software. http://www.qosient.com, March 2008.

[2] J. Bethencourt, J. Franklin, and M. Vernon. Mapping internet sensors with probe response attacks. In *Proceedings of the 14th USENIX Security Symposium*, August 2005.

[3] M. Bezzi and A. Kounine. Assessing Disclosure Risk in Anonymized Datasets. In *FloCon*, January 2008.

[4] S. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. Reiter. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium*, February 2007.

[5] S. E. Coull, C. V. Wright, A. D. Keromytis, F. Monrose, and M. K. Reiter. Taming the Devil: Techniques for Evaluating Anonymized Network Data. In *NDSS '08: 15th Annual Network and Distributed System Security Symposium*, February 2008.

[6] L. W. Group. Flaim core user's guide, March 2008.

[7] K. Hafner. Researchers yearn to use aol logs, but they hesitate. New York Times, August 23 2006.

[8] K. Killourney and R. Maxion. Toward realistic and artifact-free insider-threat data. In *ACSAC '07:* 23^{rd} *Annual Computer Security Applications Conference*, pages 87–96, Dec. 2007.

[9] J. King. *A Taxonomy, Model, and Method for Secure Network Log Anonymization*. PhD thesis, University of Illinois, Urbana-Champaign, 2008.

[10] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2005.

[11] D. Koukis, S. Antonatos, D. Antoniades, E. Markatos, and P. Trimintzios. A Generic Anonymization Framework for Network Traffic. In *ICC '06: IEEE International Conference on Communications*, volume 5, pages 2302–2309, June 2006.

[12] K. Lakkarau and A. Slagell. Evaluating the utility of anonymized network traces for intrustion detection. ArXiv, Dec 2008.

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

[14] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 2000.

[15] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratories. *ACM Transactions on Information and System Security*, 3(4):262–294, 2000.

[16] G. Minshall. Tcpdpriv: Program for eliminating confidential information from traces.

[17] A. Narayanan and V. Shmatikov. How to break anonymity of the netflix prize dataset. In *ACM Computing Research Repository*, October 2006.

[18] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *Computer Communication Review*, January 2006.

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

[20] D. Sicker, P. Ohm, and D. Grunwald. Legal issues surrounding monitoring during network research. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, August 2007.

[21] A. Slagell, K. Lakkaraju, and X. Luo. Flaim: A multi-level anonymization framework for computer and network logs. In *Proceedings of the 20th USENIX Large Installation System Administration Conference*, December 2006.

[22] A. Slagell, Y. Li, and K. Luo. Sharing network logs for computer forensics: A new tool for the anonymization of netflow records. In *Proceedings of the Computer Network Forensics Research Workshop*, August 2005.

[23] A. Slagell and W. Yurcik. Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization. In *Proceedings of SECOVAL: The Workshop on the Value of Security through Collaboration*, August 2005.

[24] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In *Proceedings of the 20th Symposium on Operating System Principles*, October 2005.

[25] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, November 2002.

[26] W. Yurcik and etc. Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs. In *Proceedings of the 3rd SECOVAL Workshop*, September 2007.