Abstract—This paper proposes a measurement approach for estimating the privacy leakage from Intrusion Detection System (IDS) alarms. Quantitative information flow analysis is used to build a theoretical model of privacy leakage from IDS rules, based on information entropy. This theoretical model is subsequently verified empirically both based on simulations and in an experimental study. The analysis shows that the metric is able to distinguish between IDS rules that have no or low expected privacy leakage and IDS rules with a significant risk of leaking sensitive information, for example on user behaviour. The analysis is based on measurements of number of IDS alarms, data length and data entropy for relevant parts of IDS rules (for example payload). This is a promising approach that opens up for privacy benchmarking of Managed Security Service providers.

Keywords
Intrusion detection, privacy leakage, entropy, metrics

I. INTRODUCTION

The objective of this paper is to develop an entropy-based metric that can be used for privacy leakage detection in intrusion detection system (IDS) alarms. The approach should be able to identify IDS rules that according to stakeholders’ perception have a significant potential for leaking private or confidential information. It should also identify the worst IDS rules from a privacy or confidentiality perspective based on indicators that can be calculated automatically. For example IDS rules that:

- have a significant risk to leak information that is sensitive (privacy sensitive, security sensitive, business sensitive etc.);
- have an unclear or too simple definition of the attack detecting pattern, and therefore may trigger unnecessarily, in the worst case on person sensitive or confidential information.

Privacy policies can be used to define what information is sensitive. Examples of sensitive information may be certain IP ranges of classified systems or sampled payload that may reveal private or confidential information. Information can also be defined as person sensitive by law, for example the sampled payload from a health institution which may contain person sensitive information. Another example is critical infrastructures that may contain security sensitive or confidential information in the data traffic about the processes being controlled. Last, but not least, payment databases handling financial transactions may reveal sensitive information like credit card numbers.

In these cases, the information is per definition sensitive, which means that any leakage of information that can be identified may be problematic. For such use cases, an objective information leakage metric will be sufficient to identify problematic leakage of private or confidential information.

In other cases, the privacy sensitivity will be subjective, and can only be evaluated in a representative way by the owners of the data being sampled - the users themselves. It may even in this case be possible for the data controller to get realistic estimates of the perceived privacy sensitivity by asking a representative random set of users, for example using a random poll on the service being used, about how they would value privacy leakages. However this approach will be expensive and does not scale well. It is therefore only viable for smaller evaluations of privacy impact.

It is therefore assumed possible for an authority like the data controller, that is overseeing the privacy interests, to estimate the privacy impact, denoted by \( I \geq 0 \), that an identified information leakage \( L \geq 0 \) causes. The privacy impact could for example be the subjective value or expected liability from privacy or confidentiality breaches, as proposed by [19]. The privacy leakage, denoted by \( \pi_R \) for a given IDS rule \( R \) can then be defined as the product of the information leakage metric \( L \) and the privacy impact \( I \), i.e: \( \pi_R = I \cdot L \). However, if investigation shows that the information leakage is caused by activities from attack vectors that do not cause any risk of revealing private, business sensitive or confidential information, then the privacy impact for a given IDS rule may be set low or even to zero. The combined metric \( \pi_R \) can be regarded as a privacy leakage risk metric, that can be used to measure and perform incremental improvements of the Managed Security Service (MSS) operation from a privacy perspective.

Current IDSs typically provide an all or nothing solution for handling private or confidential information in the alarms. The payload of the alarms is either being sent in cleartext or may be pseudonymised, for example by only sending references to where more information can be found in a data forensics system. There does not exist a more fine-grained management nor any measurements of sensitive information flows in such systems. It is in particular common that Open Source based IDS’s like Snort, OSSEC or Prelude send payload in cleartext in the IDS alarms. Having a metric for how privacy invasive an MSS operation is will therefore be useful to benchmark.
the performance of different MSS providers from a privacy perspective. It will also be useful for tuning the IDS rulesets and for implementing anonymisation policies to reduce the privacy impact of the monitoring. Intuitively, such a privacy leakage model relates to the perceived preciseness of the IDS rule, i.e. how good it is at detecting only attack traffic without revealing non-attack traffic.

A promising candidate for a privacy leakage metric for IDS rules, is data entropy. This is a privacy leakage metric that is based on the variability of the underlying data. Examples of such metrics are Shannon-, Rnyi or Min-entropy, which previously have been proposed as anonymity metrics [32, 5]. Entropy can also be used to measure coding efficiency, for example whether sampled payload excerpts most likely are encrypted or compressed [32]. This paper investigates a model of privacy leakage from IDS rules that is based on the variation in entropy between IDS alarms. This is to the best of our knowledge the first comprehensive privacy leakage model for IDS rules based on quantitative measurements of information flow

The proposed privacy leakage metric has several practical applications. First, it can be used to identify imprecise IDS rules, since such rules typically will have more variation in the underlying data, and therefore also a larger variance in entropy than more precise IDS rules. Furthermore, an advantage with the proposed metric is that it can detect two common ways of preserving privacy or data confidentiality: anonymisation and pseudonymisation. Both encrypted and anonymised information can be expected to have zero entropy variance, given sufficiently long input. On the other hand, the entropy variance of plaintext data will be significantly larger than for encrypted data, as will be discussed in Section V-C.

This means that the entropy variance can be used as a metric to detect leakage of private or confidential information in message oriented data streams in general and IDS alarms in particular. It can also be used to verify whether an anonymisation/pseudonymisation or encryption scheme works as intended.

This paper is organised as follows: Section II discusses the motivation behind introducing an entropy variance based information leakage metric, based on existing knowledge of how common attack vectors work. Section III describes the threat model and scenario that is assumed when using the privacy leakage metric. Section IV develops the entropy-based privacy leakage model based on quantitative measurements of information flow analysis after introducing the necessary theoretical background information. The last part discusses how clustering based on the Expectation Maximisation algorithm can be used to identify the underlying attack vectors for IDS rules that detect more than one attack vector. Section V does a detailed analysis of the convergence speed as a function of amount of input data for the entropy algorithms and symbol definitions considered. This includes analysing the metrics’ abilities to distinguish between plaintext and encrypted data. Section VI analyses experimental results based on realistic measurements of IDS alarms. Section VII discusses related works; Section IX concludes the paper and Section X suggests future work and research opportunities.

II. Motivation

A precise IDS rule will in many cases report only one or a few different attack patterns corresponding to real attack vectors, as will be discussed below. One common type of attack vector that follows this behaviour, is stack or heap buffer overflow attacks [39]. These attack vectors frequently use large sequences of characters corresponding to the NOP operation or similar to increase the probability of successfully exploiting buffer overflow vulnerabilities. The attacker does then not need to know the exact memory location of injected shellcode, since returning to any address within the NOP sled will cause the shellcode to be executed. This makes it simpler for the adversary to exploit such vulnerabilities. The entropy of this NOP sled will be zero, and variance zero, as long as only NOP operations are being used in the sled and the attack vector does not mutate (e.g. by changing the length of the NOP sled). This is clearly distinguishable from ordinary traffic, and also easy to distinguish for rule-based IDSs.

Such naive attacks are however not so common nowadays, because the IDS and anti-virus technologies easily can detect such anomalies in the input. It is therefore increasingly common that the adversaries obfuscate the attack vector. Obfuscation of the NOP sled can for example be done using metamorphic coding, which means that instructions in the sled are substituted with other instructions that effectively perform the same function [21]. Furthermore, it is now common practice that also the shellcode of the attack is being obfuscated by using encryption techniques. This means that the attack after the NOP sled contains a small decryption program, with a decryption key that decrypts the obfuscated shellcode before it is being run [37]. Even the decryption program can be hidden by using metamorphic coding techniques [37], although this is still not very common [28].

This means that obfuscated attack vectors can be expected to have quite high entropy, in some cases indistinguishable from encrypted traffic [37] [18]. This means that the variation in entropy can be expected to go towards zero for a sufficiently large data sample from a polymorphic attack vector, given that it is indistinguishable from a perfect encryption scheme. Such an attack vector will behave like random uniform data. This means that the entropy variance of sufficiently large attack vector samples from both traditional NOP sled based attacks and modern obfuscated attacks also can be expected to have low entropy variance.

It can furthermore be observed that samples of encrypted user traffic, assuming that strong encryption is used, in itself does not leak any private or confidential information, hence can be expected to have low entropy variance. Ordinary non-encrypted user traffic, can however be expected to show a significant variance in entropy between different samples, as illustrated in Figure V.2. This indicates that entropy variance may be an interesting metric for measuring whether IDS alarms leak information, in particular for buffer overflow type of attacks. However this metric does obviously not understand the semantics of the data traffic, and can therefore not be used to evaluate whether the leaked information is private or confidential.
There also exist attack vectors that are indistinguishable from plaintext data. Examples of such attacks are nonobfuscated Javascript Trojans or SQL injection attacks. This means that the entropy standard deviation not necessarily can be assumed to be close to the extreme points: encrypted data (entropy close to 1) or NOP sleds (octet-entropy close to 0). However, there are still some other useful characteristics of such plaintext attacks in particular, and malware in general, that can be exploited by such a metric:

- Attacks are to a great extent automated and performed by large botnets of compromised hosts.
- Attack vectors do typically not yet mutate or change dynamically. This means that multiple attacks by a given host being controlled by an adversary typically has the same payload. Different hosts running the same version of a given malware can also be expected to typically have the same payload.
- Attack vectors are modular programs that are improved incrementally, which means that not all parts of a malware will change at the same time, and some parts of malware code are even shared between different malware families.
- Botherders, that manage large botnets of compromised hosts, will also have a self interest in a “well managed” botnet. This means that the malware of a botnet at regular intervals will be upgraded to include patches and new functionalities, amongst others to avoid being detected by Anti-Virus and IDS. It is therefore reasonable to believe that a large amount of the machines in a given botnet will run the same version of the malware and therefore also will use the same arsenal of attack vectors for attacking other hosts.

This means that if an IDS rule is able to detect a given attack, or attack variants, then there are several reasons to believe that the entropy variance between instances of the same attack vector may be small, even for nonobfuscated Javascript or SQL injection attacks. This furthermore means that if the underlying attack vectors detected by an IDS rule can be identified, then the entropy variance (or entropy standard deviation) around each attack vector can be considered a measure of the precision of that rule hence also an indicator of possible privacy leakages.

III. Threat Model

The paper assumes that intrusion detection services have been outsourced to a third party Managed Security Service (MSS) provider. Security monitoring is furthermore subdivided into two different security levels. An outsourced first-line service that is doing 24x7 monitoring of the computer networks, and a trusted second-line service that will have full knowledge of the IDS service, including capabilities to perform data forensic analysis. It is assumed that the MSS provider operates using a privacy-enhanced IDS, so that changes to the IDS ruleset must be agreed upon by both the data controller and the second line security analyst responsible for updating the IDS ruleset, to avoid that excessively privacy violating IDS rules are being deployed.

It is therefore assumed that the IDS services run in a controlled environment, where enforcement of a privacy policy supported by privacy leakage metrics makes sense. An example of such an environment is critical infrastructures or hospitals where security services have been outsourced to a third party, and privacy metrics are required to ensure compliance both to privacy and security policies. These policies must ensure that the first-line security analysts, that are not trusted to see sensitive information, do not get access to information considered private or confidential by the owner of the critical infrastructure. The objective is a stricter enforcement of the need-to-know principle than what IDSs typically have today. However, in order to enforce such privacy and security policies, suitable privacy metrics are needed, which will be developed here.

This paper mainly focuses on two adversaries: an external adversary that may want to manipulate the privacy metrics for example to reduce the chance of attacks being detected. The IDS ruleset is assumed public, so that an external adversary can investigate how the IDS rules work in order to perform targeted attacks on either privacy or security. However the external adversary will not know which IDS rules that are enabled.

Insiders are divided into two main groups. First-line security analysts are considered untrusted insiders, that only have limited authorisation to see information and no authorisation to modify information related to the IDS configuration. They do not have access to the data forensic tool to investigate attacks in detail. Second-line analysts are considered a trusted CERT team, that has authorisation to perform security investigations and reconfigure the IDS. A third actor is the data controller, who shares the responsibility for managing the IDS ruleset with the security officer, to ensure that both the privacy and security objectives are being considered. The paper furthermore assumes that suitable enforcement mechanisms exist, for example anonymisation or pseudonymisation schemes for sensitive information in IDS alarms, so that the privacy leakage metrics can be used for verification of the security or privacy policies.

IV. A Privacy Leakage Model of IDS Rules

This section will first provide an information theoretic analysis of privacy leakage from IDS alarms, assuming a simple model of a perfect IDS rule $R_P$ that does not have any false alarms. This model is subsequently generalised to handle IDS rules that may leak potentially sensitive information, and we then show how this model corresponds to measuring the standard deviation of entropy from the IDS rule. It is finally shown how to measure the privacy leakage from IDS rules that detect more than one attack vector.

A. Basic Definitions

The definitions and notation in this section give a short introduction to quantitative information flow analysis, and is

\footnote{Although proof-of-concept polymorphic self-mutating worms has been demonstrated [23].}
based on \[34\]. It is throughout this paper assumed that the logarithm is taken to the base 2, i.e. \( \log(x) \) means \( \log_2(x) \). Shannon and Min-entropy can be considered instances of the more general Rnyi entropy \[30\], and we therefore use the Rnyi notation to describe the entropies. Any Rnyi entropy metric is denoted as \( H_\alpha(X) \), where \( \alpha \) is the entropy degree; \( \alpha = 1 \) represents Shannon entropy and \( \alpha = \infty \) represents Min-entropy. Given an IDS rule \( R \), which may leak sensitive information from a set of input data \( X \) and to a set of IDS alarms \( Y \), the objective is then to measure how much information \( R \) leaks.

Let \( X \) and \( Y \) be random variables whose set of possible values are \( X \) and \( Y \) respectively. The Shannon entropy is then defined by \[32\]:

\[
H_1(X) = \sum_{x \in X} P[X = x] \log \frac{1}{P[X = x]} \quad (IV.1)
\]

Shannon entropy indicates the number of bits that are required to transfer \( X \) in an optimal way. The conditional entropy denoted as \( H_1(X|Y) \) indicates the expected resulting entropy from input data \( X \) given a set of IDS alarms \( Y \) that pass through the IDS rule \( R \) \[34\]:

\[
H_1(X|Y) = \sum_{y \in Y} P[Y = y]H_1(X|Y = y) \quad (IV.2)
\]

where

\[
H_1(X|Y = y) = \sum_{x \in X} P[X = x|Y = y] \log \frac{1}{P[X = x|Y = y]} \quad (IV.3)
\]

Min-entropy is another entropy metric that is calculated based on the worst case (maximum) symbol occurrence probability, defined as the vulnerability \( V(X) \) that an adversary can guess the value of \( X \) correctly in one try \[34\]:

\[
V(X) = \max_{x \in X} P[X = x] \quad (IV.4)
\]

Min-entropy indicates the number of bits required to store \( V(X) \), and is defined as \[34\]:

\[
H_\infty(X) = \log \frac{1}{V(X)} \quad (IV.5)
\]

The conditional min-entropy can be defined as \[34\]:

\[
H_\infty(X|Y) = \log \frac{1}{V(X|Y)} \quad (IV.6)
\]

where

\[
V(X|Y) = \sum_{y \in Y} P[Y = y] \max_{x \in X} P[X = x|Y = y] \quad (IV.7)
\]

It is then possible to define the information leakage \( L_{XY} \) from \( X \) to \( Y \) using either Shannon or Min-entropy as proposed by \[34\]:

\[
L_{XY} = H_\alpha(X) - H_\alpha(X|Y). \quad (IV.8)
\]

\[\text{Figure IV.1: IDS rule 1:2003 SQL Worm Propagation attempt, behaving like } R_P.\]

\[\text{B. Perfect model IDS Rule}\]

Assume a perfect model IDS rule \( R_P \), that always detects the attack vector and does not have any false alarms or other entropy sources. Furthermore assume that the given attack vector does not change between different attack instances. The payload sample in the IDS alarm from \( R_P \) is also assumed to not contain any other entropy sources. The IDS will in this case always sample the same payload excerpt in every alarm according to the attack pattern definition.

This IDS rule is termed a perfect model IDS rule, since it is considered perfect according to the theoretical model of privacy leakage. \( R_P \) is in other words a perfect model of IDS rule behaviour from a privacy perspective. This is not a purely theoretical IDS rule behaviour. We observed three IDS rules that behaved like \( R_P \) in our experiments, for example the Snort IDS rule with SID 1:2003 SQL Worm Propagation attempt, as shown in Figure IV.1. This is obviously a simplistic model of an IDS rule, since it does not handle the fact that many IDS rules and also non-rule based technologies like anomaly-based IDS will be able to detect more than one attack vector, and also variants of attack vectors. The model is furthermore oblivious to whether the source of entropies is adversarial or ordinary user activities. An entropy-based metric can only measure whether information is leaking or not. Therefore the privacy impact \( I \) will need to be evaluated, as discussed earlier.

The perfect model IDS rule will under these assumptions provide a constant leakage denoted as \( c \) of information in each alarm, corresponding to the pattern matched by \( R_P \). The privacy impact \( I \) of this constant information leakage as a privacy leakage is however not known. The privacy impact of the information leakage from each IDS rule must therefore be evaluated by a data controller, to determine whether the expected information leakage from the IDS rule can be considered necessary and acceptable from a security perspective, and also that the effective privacy impact from the rule can be considered negligible if the rule is effective over time.
This manual quality assurance procedure makes it possible to detect and avoid IDS rules where \( I \cdot c \) in itself is judged to cause a significant privacy leakage, for example if the rule itself triggers on person sensitive information. The privacy leakage \( I \cdot c \) from each installed IDS rule is therefore in the rest of this paper considered as either necessary or negligible. If this constant privacy leakage is not considered tolerable, then it is assumed that this can be mitigated using anonymisation or pseudonymisation policies.

\( R_P \) will under these assumptions always triggers on the same attack pattern \( Y = \{y\} \), as illustrated in Figure IV.2. The inter-alarm entropy, assuming a set of input data \( X \), denoted as \( H^\text{int}_\alpha(X|Y) \), is defined as the entropy between different IDS alarms, calculated over the entire payload excerpt (i.e. each IDS alarm is considered as one “symbol”). The inter-alarm entropy will in this case be \( H^\text{int}_\alpha(X|Y) = 0 \), since \( P(Y = y) = 1 \). This means that a perfect model IDS rule according to this definition from an information theoretical perspective does not reveal any additional information apart from what can be infered from the limited and constant information leakage \( c \) in each alarm.

This does not mean that additional leakage of sensitive information cannot occur, since the resulting privacy leakage also will depend on the timing and context of the alarms. Additional information may for example be revealed by correlating the interdependencies between the IDS rules.

However, under the given assumptions, this means that when \( R_P \) triggers, then a known data pattern will have been sent in the input data stream. This information leakage is considered a tolerable privacy leakage under the assumptions in the previous subsection.

### C. A Non-perfect IDS rule \( R \)

Then consider a non-perfect IDS rule \( R \), which in addition to the assumed necessary and limited information leakage by the attack pattern, also may have false alarms or other entropy sources, as illustrated in Figure IV.3. However, it still only detects one attack vector, that does not change between attacks. This means that the entropy distribution function will be unimodal, perhaps with some outliers as illustrated in Figure IV.3. This is a simplistic model of how an IDS rule behaves. It does not assume any particular IDS rule implementation (e.g. whether string matching or regular expressions are being used) and does not take any position on the type of IDS technology being used. Experimental results have however shown that a significant amount of all IDS rules (35-53% in the experiments we have performed) actually behave in this way. However, this also means that many IDS rules actually do not behave this way. We will therefore later discuss how this restriction can be removed.

The model of a unimodal non-perfect IDS rule is illustrated in Figure IV.4. Assume that this IDS rule generates the ordered set of \( N \) IDS alarms denoted as \( Y = \{y_1, y_2, ..., y_N\} \), where \( P(Y = y_i) < P(Y = y_j) \) for \( i < j \), \( i,j \in 1,2, ..., N \). The inter-alarm entropy will in this case be greater than zero for both Shannon and Min-entropy, because \( \sum_{i=1}^{N} P(X|Y = y_i) = 1 \) and \( P(X|Y = y_1) < 1 \).

### D. Privacy Leakage Model

The next question is how to model the privacy leakage from the non-perfect IDS rule \( R \). One way to do this, is to measure the information leakage of the non-perfect IDS rule \( R \) relative to a perfect model IDS rule \( R_P \), as illustrated in Figure IV.5. The communication channel then consists of a cascade of two IDS rules (or two IDS rules connected in series), where the output of the first IDS rule serves as input to the second IDS rule.
information leakage is defined according to (IV.8) as:

\[ H \alpha(X) = \sum_{y \in \{0, 1\}} P[y \in \{0, 1\}] \log \frac{1}{P[y \in \{0, 1\}]} \]

\[ H_\alpha(X | Y = y) = \sum_{x \in \{0, 1\}} P[X = x | Y = y] \log \frac{1}{P[X = x | Y = y]} \]

\[ P[X = x | Y = y] = 0 \text{ for } x \neq y, \text{ which means that this can be expressed as:} \]

\[ H_1(X = y) = \sum_{y \in \{0, 1\}} P[X = y] \log \frac{1}{P[X = y]} \]

Figure IV.5: Channel model of privacy leakage from a non-perfect IDS rule R, measured relative to a perfect model IDS rule.

Both IDS rules have the objective to trigger on the same attack vector, however the first IDS rule \( R \) is non-perfect, and may have false alarms or other entropy sources, whereas the second IDS rule \( R_P \) is considered a perfect model IDS rule. The advantage of using a cascading model, is that this allows for comparing known values, and it is not dependent on the unknown Internet traffic \( X \). The set of alarms \( Y \) from \( R \) are known by the MSS provider and the set of expected alarms \( Z \) from \( R_P \) are also known given \( Y \).

Focusing on the inter-alarm entropies is not a fruitful approach here, since the difference in inter-alarm entropies is \( H_\alpha(\max(X | Y) - H_\alpha(\max(Y | Z) = H_\alpha(X | Y), \text{ because } H_\alpha(X | Y) = 0 \). What is needed, is therefore a measure of the limited information leakage that the perfect model IDS rule causes.

This initial information loss, denoted as the intra-alarm information loss \( H_\alpha(X) \), can be expressed by measuring the entropy of the IDS alarm in bits, instead of measuring the inter-alarm entropy \( H_\alpha(\max) \) (the entropy between IDS alarms, considering the entire IDS alarm as one symbol). The intra-alarm entropy for a perfect model IDS rule \( R_P \) can be calculated by assuming that the IDS alarm consists of a large sequence of bits. This can be expressed formally by considering a given IDS alarm as \( y \in \{0, 1\} \) where \( P[y = 1] = 1 - P[y = 0] \).

Considering the perfect model IDS rule first, then this IDS rule will always return the same IDS alarm \( Z = \{y\} \) where \( y \in \{0, 1\} \) with bit-probability \( \{P[y = 0], P[y = 1]\} \). The information leakage is defined according to (IV.8) as:

\[ L_{YZ} = H_\alpha(X | Y = y) - H_\alpha(Y = y | Z = y) - H_\alpha(X | Y = y) \]

(IV.9)

Since \( R_P \) is deterministic, then \( Z \) will be determined by \( Y \), which means that \( H_\alpha(Y | Z) = 0 \). Furthermore, for Shannon entropy:

\[ H_1(X | Y = y) = \sum_{x \in \{0, 1\}} P[X = x | Y = y] \log \frac{1}{P[X = x | Y = y]} \]

(IV.10)

\[ P[X = x | Y = y] = 0 \text{ for } x \neq y, \text{ which means that this can be expressed as:} \]

\[ H_1(X = y) = \sum_{y \in \{0, 1\}} P[X = y] \log \frac{1}{P[X = y]} \]

(IV.11)

This shows that the vulnerability \( V(X | Y = y) = 1 \) for \( P[y = 0] \in \{0, 1\} \). The lowest vulnerability is \( V(X | Y = y) = \frac{1}{2} \) for \( P[y = 0] = 0 \). This means that the Min-entropy for \( R_P \) can be expressed as:

\[ H_\infty(X | Y = y) = \log \frac{1}{1 - 2P[y = 0]}(1 - P[y = 0]) = c_\infty \]

(IV.15)

\[ V(X | Y = y) = P[y = 0]^2 + P[y = 1]^2 \]

(IV.16)

which can be expressed as:

\[ V(X | Y = y) = 1 - 2P[y = 0](1 - P[y = 0]) \]

(IV.17)

This means that \( R_P \) has a constant information leakage for both Shannon-entropy \( L_{YZ} = c_1 \) and Min-entropy \( L_{YZ} = c_\infty \). However these constants are different, except in the

Figure IV.6: Shannon vs. Min-entropy.
special cases where $P[y = 0] \in \{0, \frac{1}{2}, 1\}$, as can be expected (see Figure [IV.6]).

Let the constant information leakage for either Shannon or Min-entropy be denoted as $c_\alpha$. The relative information leakage from the IDS rule $R$ can then be formally defined as follows:

**Definition 1.** Let $R$ be a non-perfect IDS rule, that in addition to the assumed necessary and limited information leakage by the attack pattern, also may have false alarms or other entropy sources. Let $R_P$ be a perfect model IDS rule with a limited privacy leakage $c_\alpha$, $\alpha \in \{1, \infty\}$. The relative information leakage $L_{YZ}$ for an IDS rule $R$ with input $X$, that generates a set of IDS alarms $Y = \{y_1, y_2, ..., y_N\}$, each with probability $P[Y = y_i]$, $i = 1, ..., N$ is then defined as the difference in intra-alarm entropy between $R$ and a perfect model IDS rule $R_P$ that both trigger on the same attack vector:

$$L_{YZ} = H_\alpha(X|Y) - c_\alpha \quad \text{(IV.19)}$$

If the probability distribution function (PDF) of the IDS alarm entropies for a given attack vector is symmetric, then the average entropy denoted as $\overline{H}_\alpha(X|Y)$ for input $X$ and a sufficiently large set of IDS alarms $Y$ can be considered as a good estimator of $c_\alpha$. For skewed distributions, the median may give a better estimate, given that the sample is sufficiently large. It can furthermore be observed that the precision of this estimator will improve with the precision of the IDS rule $R$. This means that the information leakage of $R$ for a given IDS alarm $y_i$ can be expressed as:

$$L_{YZ} = H_\alpha(X|Y = y_i) - \overline{H}_\alpha(X|Y) \quad \text{(IV.20)}$$

where the average entropy can be expressed as

$$\overline{H}_\alpha(X|Y) = \frac{1}{N} \sum_{i=1}^{N} P[Y = y_i]H_\alpha(y_i) \quad \text{(IV.21)}$$

for a set of input data $X$.

E. Information Leakage for a Sample of IDS Alarms

The average entropy per byte for a sample $y_1, y_2, ..., y_N$ of $N$ IDS alarms generated by an IDS rule $R$ that detects a single attack vector, can be expressed as

$$\overline{H}_\alpha = \frac{1}{N} \sum_{i=1}^{N} H_\alpha(y_i) \quad \text{(IV.22)}$$

The information leakage for any IDS alarm $y_j$, denoted as $L_R(y_j)$ can then be expressed as:

$$L_R(y_j) = H_\alpha(y_j) - \overline{H}_\alpha \quad \text{(IV.23)}$$

Further processing of the information leakage $L_R(y_i)$ for the IDS alarms $y_1, y_2, ..., y_n$ can now be calculated using traditional statistical analysis. The privacy leakage of the IDS rule can be expressed as the standard deviation $\sigma_\alpha$, error margin $2\sigma_\alpha$, or the 95% confidence interval $\pm 2\sigma_\alpha$ of the IDS rule. This gives an indication of the expected precision of the IDS rule. Another useful metric, is to consider the worst-case information leakage denoted as $L^\alpha_{max}$ where $L^\alpha_{max} = \max_i L_R$, or the minimum information leakage denoted as $L^\alpha_{min}$ where $L^\alpha_{min} = \min_i L_R$. Both of these can be useful in statistical analyses, in addition to the standard deviation. Furthermore, the privacy leakage can be calculated as $\pi^L_R = L_R \cdot I_R$, where $I_R$ is the privacy impact estimated by the data controller.

F. Sample Standard Deviation of Entropy $\sigma_\alpha$

1) Normal Distribution: Assuming that the probability distribution of alarms can be approximated using a Normal distribution, then the standard deviation can be calculated using the second norm.

Assume that the IDS generates a sample of $n$ IDS alarms $(y_1, y_2, ..., y_N)$. Each alarm $y_i$ contains payload or other potentially privacy leaking elements or attributes from the IDS alarms generated by an IDS rule $R$. The sample standard deviation of the entropy of the elements can then be expressed as:

$$\sigma_\alpha = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (L_R)^2} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (H_\alpha(y_i) - \overline{H}_\alpha)^2} \quad \text{(IV.24)}$$

The general properties of the variance of entropy measurements $\sigma_\alpha^2$ will fulfill the same requirements as the standard deviation of entropy measurements. However, the standard deviation is considered more appropriate, since it operates with the same unit of measure as the entropy.

2) Laplacian Distribution: An alternative distribution that during the experiment was shown to fit the data well, is the Laplacian (or double exponential) distribution. The Laplacian standard deviation, denoted as $\sigma_\alpha^L$ is based the $L^1$ norm (or Manhattan distance), and can be expressed as the sum of absolute deviations:

$$\sigma_\alpha^L = \sqrt{\frac{1}{N} \sum_{i=1}^{N} |H_\alpha(y_i) - \overline{H}_\alpha|} \quad \text{(IV.25)}$$

A well known advantage with $\sigma_\alpha^L$ is that it will be less influenced by outliers in the tail of the PDFs than the standard deviation of the Normal distribution.

The standard deviation of normalised entropy is a measure of the relative information leakage from an IDS rule, under the assumption that it detects only one nonmutating attack vector. If an IDS rule detects the attack vector perfectly without any false alarms, then the entropy of the IDS alarms will always be the same, and $\sigma_\alpha = 0$. If the IDS alarm is precise at detecting the attack, then only a few bits of information will vary between IDS alarms. This means that all alarms will have similar entropy with low standard deviation and therefore also
low information leakage. However if the IDS rule also has a significant amount of false alarms, or gets entropy from other sources then the entropy leakage from the IDS rule, will increase.

G. Aggregating $\sigma_\alpha$

This subsection shows how the standard deviation of entropy metric can be aggregated for a set of IDS rules. Assume that an IDS uses a rule set denoted as $R_{all}$ with $m$ IDS rules $R_{all} = \{R_1, R_2, ..., R_m\}$. Each IDS rule $R_i$ matches independently a set of $N_i$ IDS alarms:

$$Y_i = \{y_{i,1}, y_{i,2}, ..., y_{i,N_i}\}, \quad i = 1, 2, ..., m$$

where the number of IDS alarms $N_i$ typically will vary between IDS rules. Furthermore, assume that the IDS alarms are independent and non-overlapping, i.e. $Y_i \cap Y_j = \emptyset$ for $i \neq j$. This means that all IDS alarms, denoted $Y_{all}$, can be expressed as $Y_{all} = \bigcup_{i=1}^{m} Y_i$.

Assume that an IDS rule $R_i$ has entropy standard deviation denoted as $\sigma_i$ and resulting standard deviation denoted as $\sigma_{all}$. The aggregated metric should furthermore fulfill the following criteria in order to provide meaningful aggregation:

C1. If all IDS rules have the same standard deviation, say $\sigma_i$, then $\sigma_{all}$ should also be the same, i.e. $\sigma_{all} = \sigma_i$.

C2. The resulting entropy standard deviation should be weighted according to how many alarms that trigger on a given IDS rule $R_i$.

Each IDS rule should be assessed individually, in the same way as each underlying vulnerability should be assessed individually. This means that a weighted average, weighted by number of alarms from each IDS rule, can be used as aggregation function for $\sigma_{all}$, i.e.:

$$\sigma_{all} = \frac{\sum_{i=1}^{m} N_i \sigma_i}{\sum_{i=1}^{m} N_i} \quad \text{(IV.26)}$$

This function fulfills criterion C1, since the resulting average weighted sum is the same if $\sigma_i$ is the same for all IDS rules $R_i$ and it fulfills C2 by weighting the standard deviation against number of IDS alarms.

H. IDS Rules Detecting Several Attack Vectors

A significant part of the IDS rules will detect more than one attack vector, as illustrated in Figure [VI.1]. The data set used in this paper has 47% of the IDS rules with more than one attack vector. An earlier preliminary experiment at a commercial MSS provider shows even higher percentage (65%). An indication of an IDS rule that detects several attack vectors, is that the entropy probability distribution is multimodal. Figure [IV.7] shows an example IDS rule that matches three privacy leaking attack vectors. The Figure shows the payload entropy distribution of the Snort IDS rule with SID 1:11969 VOIP-SIP inbound 401 Unauthorized. A payload length correction causes the metric to be larger than one, and is required to make the metric incentive compatible[4]. The details of this can be ignored for now, since this will be discussed in Section [V-D]. Each attack vector cluster corresponds to a different SIP service provider.

A clustering algorithm is needed to identify each underlying attack vector for multi-modal distributions. Each individual cluster will in this case represent an attack vector, which behaves in a similar way as a non-perfect IDS rule described in Section [IV-C]. This means that the privacy leakage of each attack vector cluster can be calculated as the entropy standard deviation over all samples belonging to the cluster, and the resulting privacy leakage for the IDS rule can be calculated by aggregating the data over all IDS rules in the cluster using Equation [IV.26].

I. How to Perform the Clustering

There are two main types of clustering algorithms: hard clustering and soft clustering. Hard clustering algorithms assign each sample to a given cluster. Examples of a hard clustering algorithm is the popular k-means and k-mediann algorithms [25, 4]. Hard clustering is however not appropriate for clustering the IDS rules, since it cuts off the samples at the tail of the distribution where two distributions overlap. This will give a bias towards lower entropy standard deviation than can be expected.

Soft clustering is then a better approach, since it assigns the probability that each sample belongs to a given cluster, instead of assigning each sample to a given cluster. A commonly used soft clustering technique is the Expectation Maximisation (EM) algorithm [8]. This soft-clustering method provides a Maximum Likelihood estimate of the underlying data distribution as a mixture of assumed probability distributions. The EM-algorithm is basically a two-step hill-climbing technique where the first step (E-step) calculates the expectation of the

---

[4] Incentive compatibility – a characteristic of mechanisms whereby each agent knows that his best strategy is to follow the rules, no matter what the other agents will do [23].
log-likelihood using the current estimate of the parameters of the underlying probability distributions. The second step (M-step) computes the parameters that maximise the expected log-likelihood identified during the E-step.

There are however some drawbacks with the EM-algorithm. It is prone to get stuck in local minima, which means that it is sensitive to the initial cluster parameters. We use the cluster centers identified by k-means, since this is a generally recommended method of initialising the cluster centers.

Another issue is the selection of number of clusters. Too many clusters may give a poor representation of the distribution of the samples.

It is commonly assumed that the underlying probability distribution either is a mixture of Gaussian or Laplacian probability density functions. Both outliers and skewness have been found to be significant during the experimental analysis in Section[11]. We have therefore decided to model the probability distribution as a mixture of Laplacian probability density functions using the method proposed in [6]. This method is based on order statistics (uses a weighted median instead of the mean), and is therefore more robust against outliers and skewness than using a Gaussian mixture [6]. The remainder of this section highlights the necessary theory and notation to understand how we have implemented the Laplacian mixture based clustering.

J. Laplacian Mixture Model

This section defines the general notation, which is based on the well-known theory of learning finite mixture models [6, 13]. Furthermore, the Laplacian Mixture Model used here, is based on [6]. Our implementation is simplified compared to the original solution, since only univariate clustering is needed. Let be a random variable representing the IDS alarm entropies of an IDS rule with component representing one particular outcome of . This random variable is expressed as:

\[ P(H_R = H_\alpha | \Theta) = \sum_{k=1}^{K} \beta_k P(H_R = H_\alpha | \Theta = \theta_k) \]  

where \( \beta_1, ..., \beta_K \) are the mixing probabilities, each \( \theta_k \) is the set of parameters defining the \( k \)-th component of the mixture and \( \Theta = \{ \theta_1, ..., \theta_K, \beta_1, ..., \beta_K \} \) is the complete set of parameters that define the mixture. Being probabilities, \( \beta_k \) must satisfy \( \beta_k \geq 0 \) and \( \sum_{k=1}^{K} \beta_k = 1 \). It is assumed that all the components of the mixture are Laplacian distributions:

\[ P(H_R = H_\alpha | \theta_k) = \mathcal{L}(H_\alpha | \theta_k = (\hat{\mu}_k, \lambda_k)) \]

where \( H_\alpha(y_i) \) is the entropy of the IDS alarm \( y_i \), \( \lambda_k > 0 \) is the scale parameter and \( \hat{\mu}_k \) is the median for mixture component \( \theta_k \). In the remainder, assume the shorthand notation that \( \mathcal{L}_{\alpha,i,k} = \mathcal{L}(H_\alpha(y_i) | \theta_k) \).

K. EM-Algorithm for Laplacian Mixture Model

The implementation of the EM-algorithm is based on [6, 13]. Assume that the EM-algorithm is performing cluster analysis on a sample of ordered entropy values \( H_\alpha(y_1), H_\alpha(y_2), ..., H_\alpha(y_N) \), where \( H_\alpha(y_i) < H_\alpha(y_j) \) for \( i < j \), \( i, j \in \{1, 2, ..., N\} \). These entropy values are calculated over the IDS alarms \( y_1, y_2, ..., y_N \) generated by an IDS rule \( R \). The Expectation Maximisation algorithm for the Laplacian Mixture Model then consists of two steps that are iterated until convergence is detected:

**E-step:** calculate the conditional expectation of the complete log-likelihood \( w_{i,k} = \log(P[H_R = H_\alpha(y_i) | \Theta = \theta_k]) \) that \( H_\alpha(y_i) \) comes from the \( k \)-th component of the mixture:

\[ w_{i,k} = \frac{\beta_k \mathcal{L}_{\alpha,i,k}}{\sum_{k=1}^{K} \beta_k \mathcal{L}_{\alpha,i,k}} \]  

**M-step:** estimate new model parameters \( \theta_k = (\hat{\mu}_k, \lambda_k) \) and weights \( \beta_k \) that maximise the log-likelihood \( \log(\mathcal{L}_{\alpha,i,k}) \) of the model:

\[ \hat{\mu}_k = \text{wmedian}(H_\alpha(k)) \]  
\[ \lambda_k = \frac{1}{\sum_{i=1}^{N} w_{i,k} \sum_{i=1}^{N} w_{i,k} |H_\alpha(y_i) - \hat{\mu}_k|} \]  
\[ \beta_k = \frac{1}{N \sum_{i=1}^{N} \sum_{k=1}^{K} w_{i,k}} \]

where the algorithm to calculate the weighted median for a given cluster \( k \), according to [6], is described in Algorithm 1.

**Algorithm 1** Weighted median.

1. function Wmedian(H_\alpha, k)
2. \( Q = (q_0 = 0, q_1 = 0, ..., q_N = 0) \)
3. \( \text{sum}=0 \)
4. for \( i \leftarrow 1, ..., N \) do
5. \( \text{sum} \leftarrow \text{sum} + w_{i,k} \)
6. \( q_i \leftarrow \text{sum} \)
7. end for
8. for \( i \leftarrow 1, ..., N \) do
9. if \( q_i > \frac{1}{2}q_N \) then
10. return \( H_\alpha(y_i) + H_\alpha(y_{i-1}) \)
11. else if \( q_i = \frac{1}{2}q_N \) then
12. return \( H_\alpha(y_i) \)
13. end if
14. end for
15. end function

The algorithm uses the Minimum Message Length (MML) as stop criterion [40], assuming one-dimensional data. We do
between two iterations is less than \( \epsilon \). The algorithm stops when the difference in MML length \(\Delta \text{MML} \) is less than \( \epsilon \). Determining the Optimal Number of Clusters

Cluster definitions. This is to avoid accidentally hitting a local and at least 20 iterations to converge after modifications of the addition to the MML criterion, the implementation of the EM-algorithm then easily got stuck in local modes. Overfitting no noisy or a mixture containing binomial distributions, since the distributions, however it did not work equally well for for skewed distributions, several components with the same median are used to represent data. A disadvantage by using \( \sigma_k \), is that this only will be correct if the model fits the data reasonably well. This may be true in some cases, however the sample distributions in the experiments do in several cases deviate significantly from the model due to outliers, heavy tails or noise. In these cases, it will be more correct to have a measure of \( \sigma_k \) that is based on the underlying samples \( H_{\alpha}(y_i) \) weighted according to the conditional expectation \( w_{i,k} \) of the model distributions defined by \( \Theta \), so that the weighted entropy is described by \( w_{i,k} H_{\alpha}(y_i) \). This means that the model distributions is used to specify how the samples are divided between the clusters, instead of defining the clusters directly. The mean value of the cluster entropies for cluster \( k \) can then be expressed as:

\[
\mu_k = \frac{\sum_{i=1}^{N} w_{i,k} H_{\alpha}(y_i)}{\sum_{i=1}^{N} w_{i,k}} \tag{IV.36}
\]

and the Normal standard deviation can be expressed in a similar way as:

\[
\sigma_k = \sum_{k=1}^{K} \beta_k \sigma_k \tag{IV.35}
\]

We implemented a simple user interface for managing the clusters. It supports configuration of the initial number of clusters \( k \) as well as managing the model definition \( \Theta \) after the initial configuration. The program also supports selecting type of entropy data and IDS rule to analyse from the datasets. The user interface for managing the clustering consists of the following functions:

- \text{setcl}(k, \tilde{\mu}_k)\) Assert that the cluster number \( k \) has a mode at \( \tilde{\mu}_k \).
- \text{delcl(clusterlist)}\) Delete clusters at index \( \text{clusterlist} \). Deleted clusters are marked with \( \theta_k = (\mu_k = 0, \rho_k = 0, \beta = 0) \).

The last term of Equation IV.33 is derived from the fact that the minimum of the MML criterion over \( \Theta \) can be obtained by using the negative maximum of the log-likelihood (the last term), since

\[
\max \left\{ \log \left( P[H_{\alpha}(\Theta)] \right) \right\} = \max_k \left\{ \sum_{i=1}^{N} \log(w_{i,k}) \right\} \tag{IV.34}
\]

The algorithm stops when the difference in MML length between two iterations is less than \( \epsilon \) MML = 1 × 10^{-4}. In addition to the MML criterion, the implementation of the EM-algorithm requires at least 40 iterations to converge initially, and at least 20 iterations to converge after modifications of the cluster definitions. This is to avoid accidentally hitting a local MML minimum before convergence has occurred.

M. Calculating the Privacy Leakage for Clusters

The privacy leakage for the identified clusters is calculated after the data controller has asserted that the relevant clusters have been identified and that the EM-algorithm subsequently has converged. All probability mass is then assigned to the clusters, which means that the privacy leakage can be calculated for the given IDS rule \( R \).

First, the model \( \Theta \) will in itself give an indication of the privacy leakage in the form of the entropy standard deviation of the Laplacian function \( L(H_{\alpha}(y_i) | \theta_k) \) for a given cluster \( k \). It is a well known fact that this can be calculated from the scale parameter \( \lambda_k \) for a Laplacian distribution as \( \sigma_k = \sqrt{2 \lambda_k} \). However to be able to aggregate the entropy standard deviation over all clusters, the relative proportion of the samples for a given cluster \( \theta_k \) must be estimated, which is exactly what \( \beta_k \) indicates. This means that the resulting entropy standard deviation for the IDS rule \( R \) can be calculated as the weighted average using Equation IV.26 substituting \( N_i \) with \( \beta_k \):

\[
\sigma_k^2 = \frac{1}{k} \sum_{k=1}^{N} \beta_k \sigma_k^2 \tag{IV.35}
\]

Pick the cluster to be asserted by clicking the mouse at the position to be asserted in the histogram showing the frequency distribution of the IDS alarm entropies. If there are no clusters that are marked as deleted, then the least significant cluster (with lowest \( \beta_k \)) will be chosen.
The Laplacian Mixture Model is implemented using the EM-algorithm. A semiautomatic process is used to identify the underlying clusters in the IDS alarms. The standard deviation of entropy metric is then calculated for each cluster and also the aggregated metric for the entire IDS rule. A possible attack on the clustering method, is an overfitting attack where a MSS provider decides to shirk by deliberately overfitting the attack vectors, by asserting too many clusters during the clustering process. It is therefore important that the role as data controller is separate from the role as security manager, and also that external quality assurance entities like certification organisations oversee the operation, to ensure that it is not overly privacy invasive. It must be emphasised that the objective not necessarily is to match the underlying probability distribution as closely as possible. The objective is rather to identify any likely attack vectors, and distribute the samples between these. The EM algorithm does this reasonably well.

The EM-based clustering generalises the privacy leakage metric to work for IDS rules that detect more than one attack vector. This generalisation is necessary, since our experiments have shown that a significant amount of all IDS rules trigger on more than one underlying attack vector. An advantage with this generalisation, is that it avoids the incentive incompatibility of the single cluster metric, which would encourage a shirking MSS provider to cheat by splitting up IDS rules into smaller IDS rules detecting a single attack vector.

V. Detailed Analysis of $\sigma_\alpha$

This section does a more thorough investigation of the standard deviation of entropy metric $\sigma_\alpha$. The objective of this discussion is to do an analysis of the convergence speed required to reliably detect random uniform input data as a function of the data length. It is expected that random uniform input data converges towards zero entropy standard deviation for a sufficiently long data series. This convergence speed is an important decision factor for the selection of entropy algorithm and symbol definition, since the IDS alarm entropies are calculated over a limited number of IDS alarms. Furthermore, it is discussed which metric and symbol definition that works best for distinguishing between plaintext and encrypted data. This analysis shows which entropy type (Min- or Shannon entropy) and symbol size (bit or octet) that is best for calculating privacy leakage in IDS rules.

A. Entropy Calculation

There are at least three obvious ways of selecting the symbol space that is used to calculate the entropies:

1) Define the payload of the IDS alarm as the symbol, i.e. calculate the inter-alarm entropy;
2) Use binary entropy, i.e. the intra-alarm entropy as described in Section V.A;
3) Use octets, i.e. 8-bit words, which commonly are used to define the character set in computer systems.

Other word sizes are possible, however these are considered the most common and interesting ones for our purpose. Each of these symbol definitions have their advantages and disadvantages, and it is important to note that the entropy values calculated from each of these definitions typically will be different. It has already been shown that the intra-alarm entropy calculated from bit-entropy is different from the inter-alarm entropy by a constant value. Furthermore, the inter-alarm entropy is not possible to use, since it can not be used to calculate the standard deviation of entropy.

Bit-entropy was used to develop the Equation [IV.20] since it is the easiest way to develop the theory for the privacy leakage metric. The entropy standard deviation formula is however not dependent on any particular symbol definition, as long as the

$$\sigma_k = \sqrt{\frac{\sum_{i=1}^{N} w_{i,k} (H_\alpha (y_i) - \mu_k)^2}{\sum_{i=1}^{N} w_{i,k}}}.$$  

Furthermore, the Laplacian standard deviation, based on the $L^1$ norm, can be expressed in terms of the conditional expectation $w_{i,k}$ and the median of the mixture component $\tilde{\mu}_k$ as:

$$\sigma_k^L = \sqrt{\frac{\sum_{i=1}^{N} w_{i,k} |H_\alpha (y_i) - \tilde{\mu}_k|}{\sum_{i=1}^{N} w_{i,k}}}.$$  

The resulting aggregated entropy standard deviation for the IDS rule $R$ can in both these cases be calculated from Equation [IV.35] by substituting the relevant standard deviation into the equation. The clustering analysis tool prints out both the individual standard deviations per cluster as well as the resulting standard deviation for the IDS rule based on both the standard deviation of the model $\sigma_k^R$, Normal standard deviation $\sigma_k$ and Laplacian standard deviation $\sigma_k^L$. It is useful to compare these, since a large deviation between $\sigma_k^R$ and the other standard deviations indicate a poor model fit, which may or may not be relevant depending on examination of the underlying data.

One can for example expect good model fit for IDS rules with some Gaussian or Laplacian noise, since this is close to the expected model of privacy leakage. However very noisy rules that match random traffic will get a poor model fit. An example of this is the IDS rule 1:1394000 in our experiments that detects random traffic. It has a standard deviation over all data of 6.7 for both Normal and Laplacian standard deviation, but only a model standard deviation of $\sigma_k^R = 1.44$ . In such cases the standard deviation of the model $\sigma_k^R$ will not be usable. Another example is if $\sigma_k$ is significantly larger than $\sigma_k^L$, then $\sigma_k$ may be unduly influenced by outliers, which means that $\sigma_k^L$ would be the more robust estimate. In general, the Laplacian standard deviation can be expected to give the most conservative estimate, which is least influenced by skewedness and outliers.

N. Summary of EM-based Clustering

The Laplacian Mixture Model is implemented using the EM-algorithm. A semiautomatic process is used to identify the underlying clusters in the IDS alarms. The standard deviation of entropy metric is then calculated for each cluster and also the aggregated metric for the entire IDS rule. A possible attack on the clustering method, is an overfitting attack where a MSS provider decides to shirk by deliberately overfitting the attack vectors, by asserting too many clusters during the clustering process. It is therefore important that the role as data controller is separate from the role as security manager, and also that
symbol definition ensures that the entropy standard deviation in the worst case, i.e. for random, uniform data, can be measured to be sufficiently close to zero for encrypted traffic. It is assumed that $\sigma$, converges towards zero for random, uniform data as a function of input data length, however the convergence speed is unknown and must be investigated. It can furthermore be observed that for a perfect encryption scheme that is approximated by random uniform data, the symbol definition does not matter, since random uniform data does not leak any information. This means that if the objective is to purely detect whether the information conveyed is encrypted or not, then the entropy scheme with fastest convergence speed may make sense to use.

This means that the minimum length of data required to reliably detect that random uniform data has zero variance (i.e. speed of convergence) is an important design factor that this metric relies on. It can be expected that different entropy metrics will have different convergence speed. In particular, can Min-entropy be expected to converge more slowly, since it only considers the maximum symbol occurrence probability, and not a weighted sum of all symbol occurrence probabilities, as Shannon entropy does.

B. Entropy Bias of Finite Length Encrypted Data

A question that needs to be investigated is therefore how different entropy standard deviation metrics $\sigma$ (Shannon- or Min-entropy) respond to random uniform data strings of varying length, and also how it is influenced by the symbol width, i.e. whether bit-entropy or octet-based entropy is used. The reason for this, as discussed in Subsection [IV-F2], is that the metric shall be able to measure privacy leakage sufficiently close to zero in the following three cases:

1) For a perfect model IDS rule $R_F$ which detects and displays one or more non-changing attack vectors perfectly;
2) For anonymised IDS alarms from the IDS rule;
3) As a limit case for encrypted (e.g. pseudonymised) IDS alarms from the IDS rule, as the number of bits $n$ in the IDS alarm goes towards infinity.

The entropy standard deviation bias for finite length encrypted data, denoted as $\sigma_{bias}$, can be analysed by simulating the response function of $\sigma_{bias}$ as a function of number of bits of data. The simulation is based on a set of Monte-Carlo experiments, one for each octet of data. Each standard deviation is the average of an ensemble of 10000 experiments. Bit-length is calculated for each octet as eight times the octet length, in order to have comparable x-axis values for bit- and octet-based data. The experiments are based on simulations using random uniform data selection, which means that a Normal distribution can be assumed.

Figure V.1 shows a log-log plot of the entropy standard deviation. The log-entropies both appear to be log-linear, which means that the bias for detecting a perfectly encrypted IDS alarm with length $n$ bits can be expressed as $\log_2(\sigma_{bias}) = \log_2(\gamma' + \psi_0n)$, where $\gamma'$ is the offset and $\psi_0$ is the slope of the log-log scale. This gives $\sigma_{bias} = 2^\gamma' n^{\psi_0}$, where $2^\gamma'$ is constant. The slope can be calculated from the experimental data, which shows that $\psi_1 = -1.005 \approx -1$ for Shannon bit-entropy and $\psi_\infty = -0.479 \approx -\frac{1}{2}$ for Min-entropy. This means that $\sigma_{bias} \approx 2^{\frac{\gamma'}{n}}$, whereas $\sigma_{bias} \approx \sqrt[n]{n}$, which means that Shannon bit-entropy converges by an order of $O(n^{-\frac{1}{2}})$ faster towards zero than Min-entropy. Shannon bit-entropy has initially 2.7 times less bias than Min-entropy for perfectly encrypted (i.e. random uniform) data.

The octet-based entropies perform very poorly during the initial transient phase, but are then stabilised on a slope similar to the respective bit-entropy slopes, as shown in Figure V.1. This means that there is a significant, but approximately constant, difference between the bit- and octet-based metrics after the initial transient phase. Shannon bit-entropy entropy ends up with a precision 143 times better than Shannon octet-entropy after 80 kbit. The difference in precision between bit- and octet-based Min-entropy is smaller, only 25 times.

A nice property is that the bias is systematic, which means that the entropy standard deviation calculations may be able to compensate for it by subtracting the expected bias from the entropy standard deviation, given that the number of samples (IDS alarms) is sufficiently large. However, this only makes sense if it is known that the data are encrypted. Since this in general is not known for the payload from IDS rules, and it will be wrong to correct for this bias for nonencrypted data, this means that the metric with fastest convergence speed is preferable.

It must also be noted that bit-based entropies (both Shannon Min-entropy) are computationally less complex than octet-based Shannon entropy, which needs to calculate the weighted logarithm expression for each symbol in an octet. Counting the number of bits set to one in an octet or word (list of octets) can be done by calculating the Hamming weight, which is implemented in hardware on most modern Intel or AMD processors using the `popcnt` (population count) operator.

---

Figure V.1: Log-log plot of entropy standard deviation as a function of number of bits input data for Min and Shannon entropy and bit and octet symbol definition.
This opens up for efficient implementations of bit-entropy calculations for up to 64 bits word chunks [7], which is more efficient than iterating to calculate the octet frequencies, as required by octet-based entropies.

C. Entropy Standard Deviation Difference between Encrypted and Plaintext data

Another foundational scenario that must be investigated, is how well the proposed entropy algorithms and symbol definitions distinguish between encrypted and plaintext information. The entire theory behind $\sigma_\alpha$ hinges on the assumption that there is a difference in entropy standard deviation between plaintext and as a limit case encrypted information. To determine whether this assumption is true or not, and which entropy configuration that works best, we set up another Monte-Carlo simulation, this time comparing the entropy standard deviation of plaintext data with the entropy standard deviation of random uniform data for both Min- and Shannon-entropy, using both bit and octet-based symbol definition.

The experiment configuration calculates the average and the 95% confidence band ($\pm 2\sigma$) from an ensemble of 10000 experiments. Each experiment calculates the standard deviation over 50 samples for varying input data length in bits, assuming that this is the smallest number of samples that in practice will be used to reliably distinguish between encrypted and plaintext data. If less samples are used per experiment, then the confidence band will widen out, meaning that longer payload will be needed to reliably distinguish between encrypted and plaintext data. There is in other words a tradeoff between the payload length and the number of samples required to reliably detect encrypted content.

Random uniform data was measured in a similar way as the previous experiment. The plaintext data was extracted using randomly selected contiguous quotes from the Brown corpus [16], with varying data length in bits along the x-axis.

Figure V.2 shows the difference between $\sigma_\alpha$ for plaintext and random data using bit-entropy for varying input data length in bits. Figure V.3 shows that Shannon octet entropy is able to distinguish reliably between cleartext and encrypted data over a sample of 50 IDS alarms within a 95% confidence interval from 5 octets (40 bits) and onwards, despite the poor convergence properties for random traffic in the range [5, 131] octets.

However, due to the slightly hourglassed shape of the entropy difference, it is not possible to achieve any larger precision between 40 and 3000 bits (375 bytes), unless the sample size is increased to narrow the confidence band sufficiently. Plaintext data is 11 times larger than encrypted data.
D. Payload Length Correction for Bit-entropy

A deficiency with the entropy standard deviation metrics, is that they decrease as the data length increases. This is the desired behaviour for random uniform data, however it is not necessarily desirable for plaintext data, since this means that the metric can not be considered incentive compatible: it will then pay off for an adversary to match as large plaintext data packets as possible, since this in effect reduces the measured information leakage. An obvious way to mitigate this problem might be to multiply the entropy values with the length \( n_i = |y_i| \) of the IDS alarm, i.e. \( n_i H_{\alpha}(y_i) \), and then take the standard deviation of the length corrected entropy values. This correction will however be too strong, since the expected bias for random uniform data of length \( n_i \) then would be constant: \( \sigma_1^{bias} \approx \frac{2 \mu_{\alpha}}{n_i} = \gamma_1 \). This means that the metric would not converge to zero for encrypted traffic.

This problem can be mitigated by multiplying the entropy values with the square root of the payload length \( n_i \). This means that the length corrected entropy values for bit-entropy can be described as \( H_{\alpha}^{\prime}(y_i) = \sqrt{n_i} H_{\alpha}(y_i) \).

The length-corrected privacy leakage metric \( \pi_{R_L}^{\prime} \), can be expressed as:

\[
\pi_{R_L}^{\prime} = I \cdot \sigma_k^{\prime} = I \cdot \sqrt{2 \sum_{i=1}^{N} w_{i,k} \left[H_{\alpha}^{\prime}(y_i) - \tilde{\mu}_k\right]}
\]

\( (\text{V.1}) \)

where \( \tilde{\mu}_k \) is the median from the LMM.

The payload length corrected Shannon bit-entropy standard deviation function is shown in Figure [V.5]. It can be observed that the term \( \sqrt{n_i} H_{\alpha}(y_i) \) essentially reduces the convergence speed to detect random uniform traffic for Shannon entropy by a factor of \( O(n^{-\frac{3}{2}}) \) to \( \sigma_1^{bias} \approx \gamma_2 \), similar to Min-entropy originally. However random uniform traffic will still converge towards 0, as required, although somewhat more slowly. Furthermore, the measured privacy leakage for plaintext data will now increase exponentially as a function of payload length, instead of decreasing, as long as the payload length is larger than the required 100 bytes (800 bits). These modifications avoids the incentive incompatibility for Shannon bit-entropy, since the metric now increases with increasing payload length.

E. Payload Length Correction for Shannon Octet-based Entropy

Shannon octet-based entropy has the same convergence speed as Min-entropy after an initial transient phase, as shown

Figure V.5: Payload length corrected Shannon bit-entropy with 95% confidence band as a function of input data length in bits for plaintext and random data.

at 5 octets (40 bits), whereas at around 128 octets (1024 bits), is down to 1.8 times larger than the encrypted data, before the random data reaches its knee point where the octet-based metric again improves.

Shannon bit-entropy is more well-behaved than Shannon octet entropy, in that the difference in entropy seems to be a strictly convex function, as opposed to the octet-based entropies. Min-bit-entropy also seems to be well behaved, and has the advantage that the 95% confidence band for Min-bit-entropy is narrower than for Shannon bit-entropy. However it is still overall a much poorer measure of entropy difference than Shannon bit-entropy, since it requires at least 6000 bits (750 octets) to reliably distinguish between plaintext and encrypted data. Octet-based Min-entropy, \( \gamma \), as shown in Figure [V.3], behaves extremely poorly, and is not usable for distinguishing between plaintext and encrypted text.

Overall, this strengthens the conclusion that Shannon entropy is the best metric, regardless of symbol definition since it converges faster than the other alternatives and it distinguishes better between cleartext and encrypted data as long as the payload is longer than the minimum threshold of 5 octets for octet-based entropy or 50 octets for bit-entropy for minimum 50 samples.
in Figure [V.1] This means that $\sqrt{n_i}$ can be used as a length correction factor also for Shannon octet-entropy for large entropy values ($\gtrsim 200$ octets or 1600 bits), to ensure that the measured privacy leakage increases with the payload length for plaintext data, and decreases with the payload length for random data.

This length correction does however not work well below 200 octets, since Shannon octet entropy initially rises quickly until a knee point at 50 bits for plaintext data and 150 bits for random data, and then starts falling, as shown in Figure [V.4]. It is desirable to reduce the effect of this knee point, in order to have an easier functional relationship between plaintext and random data, so that a fixed threshold can be used to distinguish between cleartext and random traffic. Introducing an additional length correction factor of $\frac{1}{\log_2(n_i)}$ where $n_i$ is the length of the payload $y_i$ can be used to reduce the effect of this knee point, as shown in Figure [V.6]. This means that the payload length correction function for Shannon octet-based entropy is $H'_\alpha(y_i) = \frac{\sqrt{n_i}}{\log_2(n_i)}$.

Payload length corrected Shannon octet-entropy standard deviation as a function of payload length is shown in Figure [V.6]. The initial slightly hour-glassed shape of the standard deviation functions means that the octet-based function despite the payload correction still is reduced slightly for plaintext data between 48 and 800 bits (5 and 100 bytes) payload length. This means that the metric is not entirely incentive compatible in this range, since it is slightly decreasing for plaintext instead of increasing, however the deviation is not very large. The octet-based metric is however incentive compatible beyond 100 bytes, since the metric then increases with increasing payload length for plaintext data. An advantage with Shannon octet-entropy, is that it is able to detect whether short strings of data is encrypted or cleartext, for example from pseudonymisation schemes, assuming that the data is at least 5 octets and encrypted using a perfect encryption scheme.

Another advantage with the payload length corrected entropy metrics, is that a fixed threshold can be used to distinguish between plaintext and random data, regardless of payload length for a sufficiently large sample (minimum 50 samples). For Shannon bit-entropy this threshold is 0.028, whereas Shannon octet-entropy has a threshold of 0.14 (five times larger).

It must be noted that it is possible to construct data that falls between the two example entropies used here. The first example that comes to mind, is partially encrypted IDS alarms, where for example a header part is nonencrypted and a payload part is encrypted or coded (e.g. compressed). In these cases, some IDS alarms would be interpreted as encrypted, whereas others may be interpreted as nonencrypted. However, an advantage with the octet-based metric, is that relatively few octets are required to calculate it, which means that the header and remaining payload in such cases can be calculated separately.

**F. Standard Deviation of Entropy for Base64-encoded Data**

Another interesting case is how $\sigma_\alpha$ copes with quoting techniques used to transfer binary data on transport protocols that are not 8-bit clean. A common encoding technique is Base64-encoding, which can be used to transfer binary information in SMTP and XML-based formats like HTML or SOAP. Figure [V.7] shows the standard deviation of Shannon octet-entropy for plaintext and Base64 encoded random data. The Base64-encoding adds redundancy, which means that the encoded data is closer to plaintext data. This can be seen from the Figure, since the confidence bands now overlap for less than 800 bit (100 bytes). However, for longer input data, the Base64-encoded random data behaves in a similar way like plain random data, since the standard deviation goes towards 0.

This means that at least 100 bytes are required to reliably distinguish between Base64-encoded random data and plaintext data. If it is known that the information is Base64-encoded, then it will be possible to decode the information before the entropy is calculated. This may be useful if the information leakage of shorter Base64-encoded strings are being measured. However this decoding will add additional parsing overhead, which may not be desirable from a performance perspective. This is however avoidable, as long as the payload is larger than 100 bytes as shown above.

**G. Semantic Information of Symbols**

The symbol definition for the entropy algorithms will also need to take into account the semantic information that symbols convey. The definition of bytes (or more precisely octets) is in particular important for computer systems, since this is used to define the basic character set used for communicating both text and binary codes. Octet-based symbol definition is also important for many of the attack vectors discussed in the introduction. Buffer overflow attacks for example frequently use the single octet NOP instruction (0x90 on Intel machines) for the NOP sled. There also exist multi-octet NOP variants and other techniques for generating an obfuscated sled [11]. However for now consider single byte based NOP sleds, which
are common, not the least because they are easier to exploit. Using this strategy means that the shellcode does not need to be placed on an exact 32- or 64 bits word boundary, as compilers typically enforce for normal programs. The single-byte NOP sled (0x90) is a unique symbol for octet-based entropy, however for bit entropy, this represents the binary string 10010000, which has two out of eight bit set. The problem is that this value is not unique. There will in general be 8!/(2!·6!) = 28 different octets, where any combination of these can produce the same two-bit based entropy value as this NOP opcode. In fact, bit-entropy means that 256 different octet values are mapped down to 9 different bit-entropies. Furthermore, whereas the octet entropy of a list of NOP opcodes will be zero, the bit-entropy will be greater than zero, except if all bits are ‘1’ or ‘0’. The Shannon bit-entropy of the NOP sled is 0.81, which is very different from the octet-entropy (0). Furthermore, if one octet of information is changed, this means that somewhere between one and eight bits will change. There is in other words a less clear correlation between the change in information and change in entropy for bit-entropy than for octet-based entropy.

This means that octet-based entropy is closer to representing the meaning of the information being exchanged, and therefore should be the preferred symbol definition for the privacy metrics. The discussion above has also identified that the standard deviation of Shannon octet-entropy is the metric that overall has the best properties for distinguishing between cleartext and encrypted data, despite its poor convergence properties over part of the usable range. Octet-based entropy is furthermore able to uniquely identify that a sequence of the same octet has zero entropy, something bit-entropy does not identify. This means that Shannon octet-entropy will provide the largest possible difference in entropy between plaintext and strings consisting of sequences of the same character. Shannon octet-entropy is in other words a better privacy leakage metric than Shannon bit-entropy with better distinguishing capability according to our requirements and needs within the operating range. Min-entropy is not usable for our purpose.

VI. EXPERIMENTAL RESULTS

The experimental results are based on IDS alarms from my own home network between 2009 and 2011. Some of the IDS alarms are also from the KDD-Cup’99 data set. We included the 32 most noisy IDS rules with at least 50 IDS alarms per cluster in the measurements. The threshold of 50 IDS alarms per cluster is chosen to stay within the 95% confidence bands discussed in the simulations in Section V.C. This is a limited data set that will not reflect the privacy leakage measured at a professional MSS provider doing large-scale measurements. The main difference that can be expected from a larger MSS provider, is that there would be a greater selection of IDS alarms with more than 50 alarms per cluster, and that the number of attack clusters would be greater. Furthermore, a larger set of IDS alarms may be enabled by commercial MSS providers to counter for emerging threats that are not yet in the Snort VRT ruleset, which we used. Furthermore, traffic from a commercial MSS provider would not be influenced by the synthetic KDD-Cup’99 data set.

However, despite these deficiencies, there are also some advantages by using our own data. One of the main advantages, is that this allows for discussing the IDS rules that may be leaking private or confidential information in detail, something that it according to our experience would be difficult or impossible to do for a commercial MSS provider due to business confidentiality and repudiation concerns. We have attempted to get agreement for such measurements for commercial MSS providers, however this is only possible if the IDS ruleset is not revealed, which makes it difficult to discuss in a convincing way that the proposed privacy leakage metrics work as intended. More elaborate tests at a commercial MSS provider is therefore left as future work. We decided to use a privacy impact factor I = 1 to only show the information leakage part of the privacy leakage metrics.

The experiment includes an IDS rule that we created (sid:1:1394000) which tests the worst-case scenario from a privacy perspective. This is a threshold-based IDS rule that essentially samples every 10th packet from the network. This is intended to show the maximum value that the privacy metric typically is able to detect, which is useful to see how far away the IDS rules in the measurements are from a worst-case scenario.

A. Number of Attack Vectors

The number of attack vectors per IDS rule for the given experiment is summarised in Figure VI.1. For this experiment, 53% (17 rules) have one attack vector, 31% (10 rules) have two clusters identifying attack vectors, 13% (4 rules) have three clusters and 3% (1 rule) have 4 clusters identifying attack vectors. Please note that these numbers are specific to the given experiment. A preliminary experiment at a commercial MSS provider indicates that large-scale operations can expect the distribution to be shifted somewhat towards more attack vectors.

![Figure VI.1: Number of attack vectors estimated per IDS rule for Shannon octet-entropy.](image-url)
vectors. It is in other words common that IDS rules may trigger on more than one attack vector, which means that clustering must be used to calculate the entropy of each underlying attack vector.

VII. INFLUENCE BY OUTLIERS

Figure VII.1 shows the Normal standard deviation $\sigma_1$ and Laplacian standard deviation $\sigma_2^L$ based on the $L^1$ norm for length corrected normalised Shannon octet-entropy. The Figure shows that the Normal standard deviation $\sigma_1$ for some IDS rules indicate a significantly larger privacy leakage than the Laplacian standard deviation $\sigma_2^L$. The most extreme cases are SID 109:14 which detects non-standard characters in web requests and SID 1:399 ICMP Host unreachable. The reason for the deviation is in both these cases outliers far out from the main cluster. The Normal standard deviation will give too high weight to the outliers in these cases, since it measures the root of the squared distances. Other IDS rules where the Normal standard deviation of entropy is somewhat influenced by outliers are amongst others SIDs 119:4, 119:15 and 1:1201.

In all these cases, the Laplacian standard deviation will give a more realistic estimate of the privacy leakage than the Normal standard deviation. The Laplacian standard deviation is only significantly larger than the Normal standard deviation for SID 1:402 ICMP Destination Port unreachable. This IDS rule has a left skewed noisy distribution, with several peaks reflecting the servers that were attempted contacted, but did not respond. We interpreted this as one cluster, since the failed services strictly speaking cannot be considered attack vectors. The median for this IDS rule (at 7.5) deviates somewhat from the mean (at 7.2), which gives more weight to the leftmost peaks for the Laplacian standard deviation than the Normal standard deviation does in this case, causing the Laplacian standard deviation to be larger than the Normal standard deviation. This is a pathological case where the normal standard deviation may give a better estimate than the Laplacian standard deviation. However, overall the Laplacian standard deviation $\sigma_2^L$ should be used to calculate the privacy leakage metric, since this in most cases is the more robust statistic.

A. Measured Information Leakage

Figure VII.1 shows the measured privacy leakage for the experiment using length corrected standard deviation (Normal $\sigma_1$ and Laplacian $\sigma_2^L$) of normalised Shannon octet-entropy as a function of Snort IDS rule. Further details can be found in Table II. This discussion is based on the Laplacian standard deviation, since the previous section shows that the Normal standard deviation has problems with outliers in the dataset. First, it can be observed that the metric works as expected for the extreme cases. The IDS rule that performs random sampling of payload (SID 1:1394000) has the highest privacy leakage. On the other hand, there also exist 5 IDS rules that are very precise at matching the attack vector, and behaves like the perfect model IDS rule $R_P$ with zero privacy leakage. IDS rules that fall into this category are attack vectors like SID 1:2050 SQL Version Overflow attempt, SID 1:2003 SQL Worm Propagation attempt, SID 105:2 BO traffic and SID 106:4 spp_rpc_decode preprocessor which detect amongst others incomplete RPC segments. All these IDS rules indicate possibly malicious activities, and are precise at detecting the attack. SID 1:382 which detects ICMP Echo requests (Ping) for Windows also behaves like a perfect IDS rule. It typically sends the alphabet in the payload.

There are furthermore 9 additional IDS rules with privacy leakage lower than the threshold of 0.14 for distinguishing between plaintext and encrypted traffic that was identified in Section VI.D. Rules in this category can be considered to have insignificant privacy leakage, since it is not distinguishable from encrypted traffic. These include ICMP rules matching ICMP Echo Request and Reply for various platforms (SIDs 1:384, 1:408 and 1:368) and ICMP traceroute (SID 1:385). These ICMP protocols are part of the TCP/IP protocol suite and are benign in themselves, however the Ping protocol is also frequently used for malicious activities like Denial of Service attacks or Ping scans. Furthermore pre-attack activities like portscanning (SIDs 122:1 and 122:3), and unauthorised inbound SIP calls (SID 1:11969) are potentially malicious activities that fall into this category. Last, SID 1:1437 detects download of Windows media files. This would normally be considered a benign activity, and it may also be concerning from a privacy perspective if this IDS rule is activated, since it could be used to monitor user activities. This rule detects download of Windows media files as two narrow clusters, where the upper cluster at an entropy close to 1 probably indicates download of the compressed media file. This is an example of a pathological case where the entropy standard deviation in itself, as an indirect measure of privacy leakage, does not match the perceived privacy leakage. The data controller may in this case consider whether the privacy impact $I$ of this IDS rule should be increased.

The privacy leaking IDS rules can broadly be subdivided into two groups: IDS rules with large privacy leakage ($\sigma_2^L > 1$) and IDS rules with medium privacy leakage ($\sigma_2^L \in [0.14, 1]$). There are 13 IDS rules with medium privacy leakage. The most privacy leaking of these IDS rules, is SID 1:1394 “SHELLCODE x86 inc ecx NOP” which triggers on any packet that contains a sequence of 31 ’A’ characters ($\sigma_2^L = 0.97$). The problem is that this sometimes occurs in hex-encoded URLs or hex-encoded data in web pages. It may also occur in non-compressed images, as well as for other protocols. This means that the rule most likely will trigger on a lot of random traffic, which is problematic from a privacy perspective.

Many of the rules with medium privacy leakage may be triggered by normal user behaviour, for example SID 1:486 ICMP Destination Unreachable, SID 1:402 ICMP Destination Port Unreachable and SID 1:399 ICMP Host unreachable. These can be problematic from a privacy perspective, since the ICMP error message often contains the payload of the original request, and these error messages can for example be triggered by high traffic volume (or DoS attacks) towards a server. This means that these ICMP messages essentially sample random user traffic. There are also other IDS rules in this category that will sample random traffic from users,
Figure VII.1: Privacy leakage measured using length corrected standard deviation (Normal $\sigma_1$ and Laplacian $\sigma_L$) of normalised Shannon octet-entropy as a function of Snort IDS rule.

| Snort SID | Alarms | Clusters | $\sigma_1$ | $\sigma_L$ | Description |
|-----------|--------|----------|------------|------------|-------------|
| 1:1394000 | 95096  | 1        | 6.71       | 6.70       | Samples random traffic |
| 119:14    | 3104   | 1        | 4.10       | 3.49       | http_inspect non-standard characters in web request |
| 1:402     | 36224  | 1        | 2.74       | 2.73       | ICMP Destination Port unreachable |
| 1:1201    | 680    | 1        | 1.96       | 1.77       | HTTP 403 Forbidden |
| 1:1394    | 720    | 1        | 1.40       | 1.02       | http_inspect over-long URL |
| 1:1394    | 1384   | 2        | 0.90       | 0.97       | Shellcode x86 NOP AAAAAA |
| 119:4     | 576    | 1        | 1.24       | 0.91       | http_inspect preprocessor (IIS decoding attacks) |
| 1:1852    | 10392  | 1        | 0.96       | 0.75       | robots.txt access |
| 1:1463    | 288    | 1        | 0.80       | 0.72       | IRC Chat |
| 119:2     | 21744  | 2        | 0.38       | 0.61       | http_inspect double encoded characters |
| 1:399     | 631840 | 1        | 1.02       | 0.58       | ICMP Host unreachable |
| 119:7     | 1520   | 2        | 0.48       | 0.43       | http_inspect unicode encoded web request |
| 1:12592   | 312    | 1        | 0.33       | 0.40       | SMTP command injection attempt |
| 1:2925    | 12960  | 2        | 0.42       | 0.35       | 1x1 GIF attempt (web bug) |
| 1:1560    | 360    | 2        | 0.27       | 0.30       | WEB-MISC /doc access |
| 1:486     | 368    | 1        | 0.37       | 0.27       | ICMP Destination Unreachable |
| 128:4     | 306616 | 3        | 0.25       | 0.27       | spp_ssh |
| 119:18    | 22700  | 2        | 0.32       | 0.18       | http_inspect directory traversal outside web server root. |
| 122:1     | 576    | 2        | 0.08       | 0.10       | stFortiscan preprocessor (tcp portsweep) |
| 122:3     | 2088   | 1        | 0.09       | 0.09       | stFortiscan preprocessor (tcp portsweep) |
| 1:384     | 566016 | 4        | 0.04       | 0.08       | ICMP Ping (general) |
| 1:1437    | 1056   | 2        | 0.08       | 0.08       | MULTIMEDIA Windows Media download |
| 1:408     | 205903 | 3        | 0.04       | 0.04       | ICMP Echo Reply |
| 1:366     | 202552 | 1        | 0.04       | 0.04       | ICMP Ping *NIX |
| 1:368     | 202552 | 1        | 0.04       | 0.04       | ICMP Ping BSD |
| 1:11969   | 2896   | 3        | 0.03       | 0.03       | VOIP-SIP inbound 401 Unauthorized |
| 1:385     | 462    | 2        | 0.04       | 0.03       | ICMP traceroute |
| 1:382     | 2192   | 1        | 0.00       | 0.00       | ICMP Ping Windows (alphabet) |
| 1:2050    | 32024  | 1        | 0.00       | 0.00       | SQL Version Overflow attempt. |
| 1:2003    | 1777264| 1        | 0.00       | 0.00       | SQL Worm Propagation attempt. |
| 105:2     | 192    | 2        | 0.00       | 0.00       | BO traffic (spp_bo) |
| 106:4     | 464    | 3        | 0.00       | 0.00       | spp_rpc_decode preprocessor - e.g. incomplete RPC segment. |

Table I: Privacy leakage measured using length corrected standard deviation based on Shannon octet-entropy for the IDS rules in the experiment.
which for example may be used in user profiling. Examples of such rules are SID 1:2925 1x1 GIF attempt that detects web bugs, SID 1:1560 that triggers on access to /doc on the web server root and SIDs 119:2, 119:4, and 119:7 that aim at detecting anomalies in HTTP requests like double encoded requests, IIS decoding attacks and unicode encoded requests. These may indicate attacks, but will in most cases probably be false alarms that essentially sample random user traffic, something that may be problematic from a privacy perspective.

1:1852 robots.txt access, which normally indicates indexing of a web server by a web crawler, also falls into this category. There are also some other attack rules with medium privacy leakage that do not target ICMP or web traffic. SID 128:4 detects non-SSH traffic on an SSH port, or a protocol mismatch (e.g. SSH1 traffic on an SSH2 port). This rule triggers on the initial key negotiation phase, where some information in the SSH protocol goes in cleartext. This can probably not be considered a significant primary source of privacy leakage, since no sensitive information is transferred in the packets. The data controller may consider reducing the privacy impact for this IDS rule. SID 1:12592 detects SMTP command injection attempts, that aims at exploiting a bug in the ClamAV antivirus system. The rule definition is a very simple regular expression which is likely to have false alarms. This rule may therefore be concerning from a privacy perspective, although it mostly triggered on spam. Rule 1:1463 triggers on IRC chat traffic, which also may be concerning from a privacy perspective. The reason for implementing this rule, is that IRC bots also often have been used to control botnets of compromised hosts. However, the rule does not check whether the traffic is benign or not.

There are four IDS rules with high privacy leakage, not including the test rule that samples random traffic. Three of these trigger on web traffic: SIDs 119:14, 119:15 and 1:1201. The most privacy leaking ordinary IDS rule (SID 119:14, $\sigma^L_1 = 3.49$) triggers on non-standard character encodings in HTTP requests, which are getting increasingly common, especially after IANA allowed non-ASCII domain names. The second most privacy leaking IDS rule is SID 1:402 ICMP Destination Port Unreachable with $\sigma^L_1 = 2.73$. This protocol typically copies the failed request in the ICMP message, and therefore samples random traffic requests. On third place is SID 1:1201 HTTP 403 Forbidden ($\sigma^L_1 = 1.77$), which also is quite common also for benign traffic, for example on web sites referring to internal material that require subscription. On fourth place is SID 119:15 that tests for over-long URL's ($\sigma^L_1 = 1.02$), something that frequently happens for blogs or search engines that use URL referencing. All of these rules may be problematic from a privacy perspective, since they in many cases will trigger on normal user behaviour. It is especially problematic if the IDS rules monitoring web services are set up in an uncritical way, so that these rules trigger for any web server accesses and not only for relevant web servers (e.g. the company's own web servers).

This discussion shows that the privacy leakage metric is able to distinguish between IDS rules that most likely may trigger on ordinary user activities, and therefore may be problematic from a privacy perspective, from the IDS rules that are precise at detecting the underlying attack vector, or that perform a very specific task without leaking any significant amount of data about user behaviour. However there were also two pathological cases where it may make sense to adjust the privacy impact, since using entropy as an indirect measure of privacy leakage not always will gives a true picture of whether the underlying information is sensitive from a privacy/confidentiality perspective or not. Overall, this demonstrates that the privacy leakage metric works as intended. However larger studies involving commercial MSS providers will be needed in the future to confirm these results.

**B. The Effect of Anonymisation**

The resulting privacy leakage over all IDS alarms in the experiment, weighted according to number of alarms, is 0.31. However, if the test IDS rule with SID 1:1394000 that samples random data is removed, then the resulting privacy leakage is reduced to 0.16. If all the IDS rules with high privacy leakage are removed, then the resulting leakage is reduced by 0.02 to 0.14.

Surprisingly, it is then more efficient to anonymise all ICMP Destination Host unreachable alarms, since there are many of them (631840) in the data set, and each of them has a significant measured privacy leakage ($\sigma^L_1 = 0.58$). Anonymising ICMP Destination Host unreachable alarms would reduce the overall privacy leakage by 0.07 to 0.07. This can probably be done without reducing the usability for the security analyst significantly, since it still would be known which host that was attempted contacted from the IP-address element of the IDS alarm. SID 1:402 ICMP Destination Port unreachable also triggers quite often (32360 times) and has the second highest measured privacy leakage ($\sigma^L_1 = 2.73$). Anonymising this rule reduces the privacy leakage by 0.02 to 0.05, and can probably also be done without reducing the possibility to do root cause analysis significantly, since the number of services running on a server normally is limited. Classification based on the EM-clustering can if necessary be used to indicate which server that failed without revealing the original user request. These examples show that the total privacy leakage, calculated as the product of number of IDS alarms $N_R$ for the given rule $R$ and the entropy standard deviation $\sigma^L_{1,R}$, must be used as the optimisation criterion to reduce the overall privacy leakage. The total privacy leakage is calculated as

$$L_{tot} = N_R \sigma^L_{1,R}.$$

Another IDS rule, that either benefits from anonymisation, alternatively by setting the privacy impact to zero, is SID 128:4 which detects ssh anomalies. This rule triggers quite often (306616 times) with $\sigma^L_1 = 0.27$, which means that the overall privacy leakage can be reduced by 0.02 to 0.03 if this rule is anonymised. If the IDS rules with low privacy leakage, that are not relevant from a privacy perspective (all with privacy leakage less than 0.14, except SID 1:1437), are either anonymised or removed by setting the privacy impact to zero, then the resulting privacy leakage index is reduced from 0.03 to 0.011.

If the two IDS rules from the http.inspect preprocessor with largest total leakage (SID 119:14 and 119:2) also are anonymised, then the measured privacy leakage is reduced to 0.005.
This illustrates how a structured method can be used to reduce the privacy leakage of the IDS ruleset based on measured privacy leakage and number of IDS alarms. It is furthermore also clear that many of the IDS rules can be anonymised without significantly reducing the usability for the security analysts. Especially since the clustering model used to identify attack vectors in many cases can be used to help the security analysts in identifying the necessary properties of the underlying data without having to reveal the user payload.

VIII. RELATED WORKS

The research area of quantitative information flow based on information theory adds a comprehensive theoretical framework for analysing privacy leakage based on entropies [34, 33]. Our research is based on this, and extends the theory to cover privacy leakage in IDS alarms. There is as far as we are aware of no other research that proposes a comprehensive model of privacy leakage in IDS alarms based on quantitative information flow analysis.

Quantitative information flow analysis that in a similar way uses information entropy has however been proposed used to derive an intrusion detection capability metric in [20]. This metric aims at modelling the uncertainty about the input given the IDS output. The uncertainty as it is termed in this paper is the same as the information leakage defined here based on [34], which in turn is based on the notion of mutual information from [32]. The IDS capability metric is defined as the mutual information between the IDS input and output to the entropy of the input:

$$C_{ID} = \frac{H(X) - H(X|Y)}{H(X)} \quad \text{(VIII.1)}$$

The numerator is the same as the information leakage defined in [34], however these data are normalised with respect to the entropy of the input data, something our model does not do. This model assumes that the input data $H(X)$ is the labels (attack or not) from a labeled IDS test set, and the output data $H(X|Y)$ is the classification by the IDS, which also is different from our conceptual model of an IDS rule. It is from this reason that the proposed metric is different from the privacy leakage metric proposed here, since it assumes different input data, a different information model and normalises the indicator to the input data. However an interesting similarity is that the effect of false positives in Figure 3b) in this paper follows a similar falling exponential curve as Figure VII.1 as can be expected, since the false alarms here will increase the entropy up to the point where the classifier is not better than random decisions. However this paper does not make the connection to privacy leakage metrics for IDS rules.

There are also some similarities between the proposed approach and the concept of Differential Privacy in statistical databases [10, 11, 12]. Both methods use a Maximum Likelihood (ML) estimate, however the estimate is interpreted differently. Differential Privacy uses the ML estimate to indicate the aggregate value of underlying perturbed data, whereas we use the ML estimate as a measure of underlying attack vectors. Both methods use robust statistics (first norm) for calculating aggregated values. However, Differential Privacy typically adds Laplacian noise to hide individual elements of privacy sensitive information, whereas our privacy leakage metric works in the opposite way - assumed Laplacian noise from an IDS rule is used as an indication of IDS privacy leakage. So although there are similarities, our proposed metric is clearly different to Differential Privacy.

Entropy has previously been proposed as a measure of privacy [5, 3]. Claude Shannon’s seminal paper on information theory was the first publication where entropy was proposed to measure the level of ambiguity or equivocation in transferred information [32]. Min-entropy has been proposed as a metric of anonymity that in particular considers local aspects, i.e. the worst case scenario for the user [38]. The more general Rnyi entropy has been proposed as a metric of anonymity in [31, 3]. Neither of these have used entropy to measure privacy leakage in IDS alarms.

The paper is also related to field of privacy-preserving intrusion detection systems [36, 14, 35, 29, 26, 15], however neither of these solutions focus on privacy metrics.

IX. CONCLUSIONS

In this paper we propose an entropy-based privacy leakage metric founded on quantitative information flow analysis. An advantage is that this metric can be calculated based on already existing information in the IDS alarm database. From a privacy perspective, it provides a structured approach to identify which IDS rules that may be leaking sensitive information and also for handling such privacy leakages.

An advantage with the metric, is that it also is a measure of IDS rule precision. This is clearly desirable, since the objective is to tune the IDS ruleset to reduce the leakage of private or confidential information over time, for example through improving the precision of the IDS rule or by applying anonymisation techniques. This is also an advantage from a security perspective, since more precise IDS rules mean less effort spent on false alarm handling.

We have demonstrated that the proposed approach is feasible based on a set of real IDS alarms. It is furthermore shown that different entropy algorithms and ways to calculate the standard deviation have different strengths and weaknesses. Not surprisingly, the Laplacian standard deviation based on the $L^1$ norm provides the most robust statistic to avoid problems with outliers, a problem that has been shown to occur in the experimental data. The experiments have shown that Shannon octet-entropy is the best entropy metric with fastest convergence speed for reliably detecting encrypted traffic, and it is also the entropy metric that is is best at distinguishing between plaintext and encrypted traffic. It is also shown how the metric can avoid being incentive incompatible by taking into account the length of the input data.

The Laplacian Mixture Model of the underlying data will in itself be useful for classification purposes. If a given model of the data has been identified, then this can be used for subsequent classification of the underlying samples, for
example to anonymise IDS alarms from data clusters, or to further classify the attack vectors of the IDS alarms, for example to detect Denial of Service attacks. The clustering can therefore be used as a post-processing step to modify the IDS alarms according to cluster, which means acting as a higher order IDS solution.

A possible attack on the clustering method, is an overfitting attack where a MSS provider decides to shirk by deliberately overfitting the attack vectors. The proposed method to avoid this, is to ensure separation of duties between privacy and security interests and also that third party certification organisations oversee the operation.

The proposed privacy leakage metric only measures the primary privacy leakage sources in IDS alarms. It does not consider secondary sources of information leakage, like correlation of different information sources. However, being able to measure the primary sources of privacy leakage in IDS alarms is at least an initial approach that can and should be considered before more elaborate analyses of the anonymity set are performed. Furthermore, the ability to verify that the anonymisation policies reduce measured information leakages means that policy verification, in the form of a privacy leakage gap analysis, will be possible in order to provide incremental reductions of privacy leakage in IDS alarms over time.

X. Future Work

Future work includes doing comparative studies of the performance of different MSS providers from a privacy perspective. Adapting the privacy leakage metric to support anomaly-based IDS is also left as future work. This will amongst others require subdivision of the alarms, for example based on service etc., to avoid that the entropy space becomes too crowded by attack vectors.

Investigating possible secondary privacy leakages that may occur due to inference or cross correlation between different information sources both within the IDS alarm and outside is also left as future research. This would require taking the privacy leakage metrics and evaluation even further in order to evaluate the anonymity set that can be expected for private or sensitive information, using metrics like differential privacy \[ \text{[10][11][12]} \], k-anonymity \[ \text{[9]} \], or l-diversity \[ \text{[24]} \].

References

[1] P. Akritidis, E. P. Markatos, M. Polychronakis, and K. Anagnostakis. STRIDE: polymorphic sled detection through instruction sequence analysis. In In 20th IFIP International Information Security Conference, 2005.
[2] T L Bailey and C Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. Proceedings of the International Conference on Intelligent Systems for Molecular Biology : ISMB. International Conference on Intelligent Systems for Molecular Biology, 2:28–36, 1994. ISSN 1553-0833. PMID: 7584402.
[3] M. Bezzi. An entropy based method for measuring anonymity. In Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on, pages 28–32, 2008.
[4] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In Advances in Neural Information Processing Systems -9, page 368–374. MIT Press, 1997.
[5] Sebastian Clauß and Stefan Schiffer. Structuring anonymity metrics. In Proceedings of the second ACM workshop on Digital identity management, pages 55–62, Alexandria, Virginia, USA, 2006. ACM.
[6] Aurélien Cord, Christophe Ambroise, and Jean-Pierre Cocquerez. Feature selection in robust clustering based on laplace mixture. Pattern Recognition Letters, 27(6): 627–635, April 2006. doi: 10.1016/j.patrec.2005.09.028.
[7] Intel Corporation. Intel SSE4 Programming Reference. Intel, 2007.
[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological), page 1–38, 1977.
[9] Sabrina De Capitani di Vimercati and Pierangela Samarati. Privacy in the electronic society. In Information Systems Security, pages 1–21. Springer, 2006.
[10] C. Dwork. Differential privacy. Automata, languages and programming, page 1–12, 2006.
[11] C. Dwork. Differential privacy: A survey of results. Theory and Applications of Models of Computation, page 1–19, 2008.
[12] C. Dwork and J. Lei. Differential privacy and robust statistics. In Proceedings of the 41st annual ACM symposium on Theory of computing, page 371–380, 2009.
[13] Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. IEEE Trans. Pattern Anal. Mach. Intell., 24(3):381–396, March 2002. ISSN 0162-8828. doi: 10.1109/34.990138.
[14] Simone Fischer-Hübner. IDA - An Intrusion Detection and Avoidance System (in German). Aachen, Shaker, 1997.
[15] Ulrich Flegel. Privacy-Respecting Intrusion Detection. Springer, 1 edition, October 2007.
[16] W. N. Francis and H. Kučera. A standard corpus of present-day edited American English, for use with digital computers (Brown). Brown university. providence, Rhode Island, 1964, 1971, 1979.
[17] Felix C. Freiling, Thorsten Holz, and Georg Wicherski. Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks. In Proceedings of the 10th European conference on Research in Computer Security, ESORICS’05, pages 319–335, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-28963-1, 978-3-540-28963-0.
[18] Jean Goubault-larrecq and Julien Olivain. Detecting Subverted Cryptographic Protocols by Entropy Checking. Laboratoire Spécification et Vérification, ENS Cachan, France, 2006.
[19] S. Gritzalis, A. Yannacopoulos, C. Lambrinoudakis, P. Hatzopoulos, and S. Katsikas. A probabilistic model for optimal insurance contracts against security risks
and privacy violation in IT outsourcing environments. *International Journal of Information Security*, 6(4):197–211, July 2007. doi: 10.1007/s10207-006-0010-x.

[20] Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skoric. Measuring intrusion detection capability: an information-theoretic approach. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS ’06, page 90–101, New York, NY, USA, 2006. ACM. ISBN 1-59593-272-0. doi: 10.1145/1128817.1128834.

[21] C. Jordan, J. ROYES, and J. WHYTE. Writing detection signatures. *USENIX; login*, 30(6):55–61, 2005.

[22] O. Kolesnikov and W. Lee. Advanced polymorphic worms: Evading IDS by blending in with normal traffic. Technical report, Georgia Institute of Technology, 2005.

[23] John O. Ledyard. Incentive compatibility. In Steven N. Durlauf and Lawrence E. Blume, editors, *The New Palgrave Dictionary of Economics*, pages 158–164. Nature Publishing Group, Basingstoke, 2 edition, 2008. ISBN 978-0-333-78676-5.

[24] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *Cornell University*, page 52, March 2007.

[25] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14, 1967.

[26] Ruoming Pang and Vern Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 339–351, Karlsruhe, Germany, 2003. ACM.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Michalis Polychronakis, Kostas G. Anagnostakis, and Evangelos P. Markatos. An empirical study of real-world polymorphic code injection attacks. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, LEET’09, pages 9–9, Berkeley, CA, USA, 2009. USENIX Association.

[29] R. Büschkes, D. Kesdogan. Privacy enhanced intrusion detection. In G. Müller and K. Rannenberg, editors, *Multilateral Security in Communications, Information Security*, pages 187–204. Addison Wesley, 1999.

[30] A. Rényi. On measures of information and entropy. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability* 1960, pages 547–561, 1961.

[31] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, page 259–263, 2003.

[32] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[33] G. Smith. Quantifying information flow using min-entropy. In *Quantitative Evaluation of Systems (QEST)*, 2011 Eighth International Conference on, page 159–167, September 2011. doi: 10.1109/QEST.2011.31.

[34] Geoffrey Smith. On the foundations of quantitative information flow. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures*, number 5504 in Lecture Notes in Computer Science, pages 288–302. Springer Berlin Heidelberg, January 2009. ISBN 978-3-642-00595-4, 978-3-642-00596-1.

[35] Michael Sobirey, Birk Richter, and Hartmut König. The intrusion detection system AID - architecture and experiences in automated audit trail analysis. In *Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security*, pages 278–290, 1996.

[36] Michael Sobirey, Simone Fischer-Hübner, and Kai Rannenberg. Pseudonymous audit for privacy enhanced intrusion detection. In *Proceedings of the IFIP TC11 13th International Conference on Information Security (SEC’97)*, pages 151–163, May 1997.

[37] Y. Song, M. E Locasto, A. Stavrou, A. D Keromytis, and S. J Stolfo. On the infeasibility of modeling polymorphic shellcode. In *Proceedings of the 14th ACM conference on Computer and communications security*, page 541–551, 2007.

[38] G. Tóth, Z. Hornák, and F. Vajda. Measuring anonymity revisited. In *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*, pages 85–90, 2004.

[39] M. Vallentin. On the evolution of buffer overflows. [http://matthias.vallentin.net/course-work/buffer_overflows.pdf](http://matthias.vallentin.net/course-work/buffer_overflows.pdf) 2007.

[40] C. S. Wallace and D. M. Boulton. An information measure for classification. *The Computer Journal*, 11(2): 185–194, January 1968. doi: 10.1093/comjnl/11.2.185.