

# 13 Security Measurements and Metrics for Networks

Thorsten Holz

University of Mannheim, Germany

This chapter surveys research in two areas of security metrics: The first area is the field of malicious network traffic. The measurements made there are used to estimate the “health” of a network. The second area is the field of intrusion detection systems. These measurements can be taken as indicators on the quality of the system, i.e., its assurance to reliably detect intrusions.

## 13.1 Measuring Malicious Network Traffic

In today’s Internet, we observe more and more security threats. Examples of these malicious attacks include network attacks against vulnerable services, host based attacks such as privilege escalation, unauthorised logins and access to sensitive files, data driven attacks on applications, or many other attack vectors. Up to now, there is no real metric to classify this malicious network traffic. In this section, we try to come up with at least some possibilities to *measure* malicious traffic. This data can then hopefully be used in the future to establish metrics for this area.

Currently, there are several different attempts to measure security-related network activity. Measurement is the first step towards a real metric, so we present in this section the basics. On the one hand, there are tools to measure actual malicious network traffic. These tools include amongst others *honeypots*, *network telescopes / Internet Motion Sensor*, and *flow-based techniques*. On the other hand, there are several attributes that help us to describe malicious network traffic, e.g., *backscatter*. In the following, we will describe each of these tools and attributes in detail and elaborate how the results can be used as a metric. We start with *honeypots* and *honeynets*.

### Honeypots and Honeynets

A honeypot can be defined as an information system resource whose value lies in unauthorised or illicit use of that resource. Honeypots are used to learn more about attack patterns and attacker behaviour in general. The concept is rather simple: electronic decoys, i.e., network resources deployed to be probed, attacked, and compromised, are used to lure in attackers. A honeypot is usually a computer system with no conventional task in the network. This assumption aids in attack detection: every interaction with the system is suspicious and could point to a possibly malicious action.

In honeypot-based research, there is a distinction between two fundamental methodologies: *high-interaction* and *low-interaction* honeypots. A high-interaction honeypot is a conventional computer system, e.g., a commercial off-the-shelf (COTS) computer, a router, or a switch. This system does not offer any production-related services in the network and no regularly active users access it. Thus it should neither have any unusual

processes nor generate any network traffic, besides regular daemons or services running on the system. These assumptions aid in attack detection: every interaction with the high-interaction honeypot is suspicious and could point to a possibly malicious action. Hence, all network traffic to and from the honeypot is logged. In addition, system activity is recorded for later analysis. Several honeypots can also be combined to a network of honeypots, a *honeynet*. Usually, a honeynet consists of several honeypots of different type (different platforms and/or operating systems). This allows the operator to simultaneously collect data about different types of attacks. Usually high-interaction honeypots help to learn in-depth information about attacks and therefore qualitative results of attacker behaviour [400, 494]. An example of this type of honeypot is the so-called *GenIII honeynet* [36].

A honeynet creates a fishbowl environment that allows attackers to come, while giving the operator the ability to capture all of their activity. This fishbowl also controls the attacker's actions, mitigating the risk of them doing harm to any non-honeynet systems. It is within this fishbowl environment that the operators place any honeypots they want, real systems with real services. The key element to a honeynet deployment is called the *Honeywall*, a layer-two bridging device that separates the honeynet from the rest of the network. This device mitigates risk through data control and captures data for analysis, known as data capture. Tools on the Honeywall allow for analysis of attacker's activities. Any inbound or outbound traffic to the honeypots must pass the Honeywall. Information is captured using a variety of methods, including passive network sniffers, Intrusion Detection System alerts, firewall logs, and the kernel module known as "Sebek". The attacker's activities are controlled at the network level, with all outbound connections filtered through both an Intrusion Prevention System and connection limiter, to mitigate outgoing attacks.

A similar approach in the area of high-interaction honeypots is more lightweight: instead of deploying a physical computer system which acts as a honeypot, it is also possible to deploy one physical computer which hosts several virtual machines which act as honeypots. This leads to easier maintenance and lower physical requirements. Usually VMware [488] or User-mode Linux (UML) [477] are used in order to set up such *virtual honeypots* [403]. These tools allow the operator to run multiple operating systems and their applications concurrently on a single physical machine, thus enabling an easy way of data collection which can eventually be used as a basis for a metric.

If the operator of a honeynet is primarily interested in quantitative results, it is possible to even go one step further and pursue the deployment of a whole computer system. This approach is called *low-interaction honeypot* in contrast to the high-interaction honeypots described above. This type of honeypot does not simulate all aspects of a system, but only simulates specific services or some parts of an operating system, e.g., the network stack. Low-interaction honeypots can primarily be used to gather statistical data about attacks and to collect high-level information about attack patterns. Furthermore, they can be used as a kind of intrusion detection system where they provide an early warning, i.e., a kind of burglar alarm, about new attacks (see Sect. 13.2 for more information about intrusion detection systems). Moreover, low-interaction honeypots can be deployed to lure attackers away from production machines, to detect and disable worms,

distract adversaries, or prevent the spread of spam e-mail [32, 402]. Low-interaction honeypots can also be combined into a network, forming a *low-interaction honeynet*.

In the context of the project *eCSIRT.net*, several European Computer Security Incident Response Teams (CSIRTs) set up a network of network sensors across Europe [183]. This network collected data about attacks in a central database for further analysis and helped in vulnerability assessment. After the project ended, some teams decided to continue the then established sensor network across Europe, which has been providing information about network attacks since September 2003. A similar project is *leurre.com* [401]. Several low-interaction honeypots are deployed on different networks and collect data about malicious network traffic in a central database. The collected information can be analysed and enables a comparison between several different parts of the Internet in terms of malicious network traffic.

High- and low-interaction honeypots enable a way to measure malicious network traffic. With low-interaction honeypots, the measurements are rather quantitative since the resulting data sets are typically statistics about the type of traffic, e.g., how much traffic has been observed at a particular port during a specified time frame. In contrast to that, high-interaction honeypots lead to qualitative measurements of malicious network traffic. The operator can learn about particular types of attacks or new attacking techniques. The collected data could be used to form a metric to classify traffic according to certain patterns. However, up to now there is no such metric, the research in this area concentrates currently on data collection. The following quantitative statistics could for example help to form metrics for this area of research:

- Most attacked network ports.
- Number of observed, unique IP addresses.
- Sequences of attacked ports.
- “Mean time between attacks”.

## Large-Scale Monitoring of Networks

Today, many solutions exist to observe malicious traffic on a large-scale base, e.g., on large parts of the Internet. These solutions often consist of monitoring a very large number of unused IP address spaces to monitor malicious activities. Several names have been used to describe this technique, such as *network telescopes* [81, 358], *blackholes* [104, 454], *darknets* [112], or *Internet Motion Sensor (IMS)* [34]. All of these projects follow the same approach: they use a large piece of globally announced IPv4 address space and passively monitor all incoming traffic or – to a very limited extend – also respond to incoming packets. For example, the network telescope run by the University of California, San Diego, uses  $2^{24}$  IP addresses. This is 1/256-th of all IPv4 addresses. The telescope contains almost no legitimate hosts, so inbound traffic to nonexistent machines is always anomalous in some way, i.e., the principle of honeynets is also used in this context. By analysing all packets, the operator is able to infer information about attackers. Since the network telescope contains approximately 1/256-th of all IPv4 addresses, it receives roughly one out of every 256 packets sent by an Internet worm with an unbiased random number generator. Thus the monitoring of unexpected traffic yields a view of certain remote network events, the so called *backscatter*. This can for example be used to study the threats posed by Denial-of-Service attacks [358].

Another approach in this area is to passively measure live networks by centralising and analysing firewall logs or IDS alerts [239, 517]. Especially the *Internet Storm Center* (ISC) / *DShield.org* [239, 240] is a well-known project in this area. In this project, the collected data is simple packet filter information from different sources all around the world and no “high-level” data is included. Reports are published on a daily basis. They include information about attack patterns and take a closer look at unusual events. A report combines 8 – 20 million records per day with 200,000 – 400,000 source and 300,000 – 450,000 target IP addresses per day. The results are statistics like “Most Attacked Port” or for each port the number of observed source addresses. However, the data contains no detailed information about the source which has collected the packet since this kind of information is anonymised. Therefore a comparison of different attacks is not easily possible. Nevertheless, the huge amount of collected data could enable a way to form metrics. Since the collected information can be compared on different scales, it would be possible to measure the impact of certain malicious events or other metrics could be applied.

Coarse-grained interface counters and more fine-grained flow analysis tools such as *NetFlow/cflow* offer another readily available source of information. A *flow* is an abstraction of individual packets and a summary of packet data between two sites. It can be defined as IP traffic with the same source IP, destination IP, source port and destination port, since this quadruple describes the IP traffic between two devices on the Internet. A flow record typically also contains some additional data, e.g., the number of bytes sent, the duration of the flow, or the timestamp of the first packet. However, the actual payload of the connection is not included within a flow. This is mainly due to logistical reason: if also the payload would be stored, this would quickly result a unmanageable amounts of data. A router which is capable of monitoring flows will only output a flow record when it determines that the flow is finished, e.g., either by explicit connection shutdown or timeout. The flows are stored in a central database and can be analysed from a high-level point of view. With this aggregation of data, it is often possible to draw conclusions about unusual events within a network. Moreover, this concept is often used for visualisation of network traffic. Spikes at certain network ports or certain anomalies can be detected via flow-based analysis techniques. And again, the collected data can enable metrics to analyse the current state of a network in terms of malicious network traffic. However, also this area has up to now no real formalism or model as foundations of metrics.

## 13.2 Metrics for Intrusion Detection Systems

An *Intrusion Detection System* (IDS) generally tries to detect unwanted manipulations to information systems resources. An IDS is required to detect as many types of malicious network traffic and computer usage as possible. Each of these malicious events can be described with the help of an *attack vector*, i.e., a path or means by which an attacker can gain access or modify an information system resource. So the basic task of an IDS is to classify an event as normal or malicious. In general, there are two fundamentally different approaches to build an IDS:

- *Misuse detection systems* (or *signature-based Intrusion Detection Systems*) identify malicious attacks by comparing actual network traffic or executing flow of an application with *patterns of malicious attacks*. Therefore, this kind of systems needs to know the signature of an attack in advance. The main drawback is that these signatures have to be updated regularly to adapt to new attack vectors. In addition, a *zero day attack*, i.e., an attack vector for an unknown vulnerability, can not be detected with such an approach.
- *Anomaly-Based Intrusion Detection Systems* identify malicious attacks by detecting network traffic or executing flow of an application that deviates from “normal” network or system activity. In most cases, this “normal” state is learnt during a training period in which the IDS observes the regular behaviour of the information system resource, the *baseline/threshold*. Afterwards, the IDS can detect whether the current behaviour differs from the learned behaviour.

Orthogonal to this classification is the differentiation between *host-based IDS (HIDS)* and *network-based IDS (NIDS)*. A HIDS monitors and analyses the behaviour of a computer system and is typically also installed on this system. The HIDS could for example monitor system calls, modification of the filesystem, or other changes in the operating system or application. In contrast to that, a NIDS examines the network traffic within a computer network. It tries to detect malicious activity, e.g., Denial-of-Service (DoS) attacks or exploitation attempts, at the network layer. Moreover, both approaches can be combined to build a *Hybrid Intrusion Detection System* to enhance the effectiveness.

A survey on intrusion detection systems can be found in the paper by Hervé Debar and Wespi [214] or Axelsson [29]. A preliminary taxonomy of IDS and attacks is a result of the MAFTIA project [327].

## Binary Classification

In order to compare different IDS solution, we need metrics to evaluate them. Firstly, these metrics should be able to objectively measure the effectiveness of the solution in terms of its ability to correctly classify a certain behaviour as normal or malicious, i.e., measure the *accuracy* of the IDS. Secondly, an IDS can also be evaluated on the basis of its *performance*, i.e., how many packets per second can be examined or how large the memory usage is. For more information on this kind of metrics, please see Part IV of this book. Thirdly, an important metric is the *resilience* of an IDS. The resilience measures how the IDS reacts on stress tests or attacks against the IDS itself. In this area, there are currently no established metrics since this is rather an arms-race between attackers and defenders and this area is changing quickly. So we will focus in the following on metrics to measure the accuracy of an IDS solution.

The most simple — and most often used — accuracy metrics for IDS come from the area of binary classification. Since the IDS has the task to classify an event or behaviour as malicious or normal, we can use the results obtained in the area of statistics. The first metrics we introduce are thus *False Positive Rate (FP or Type I Error)* and *False Negative Rate (FN or Type II error)*. The FP is the probability that the IDS outputs an alert although the behaviour of the system is normal. This means that the IDS incorrectly outputs an alert. In contrast to that, the FN is the probability that the IDS does not

output an alert although the behaviour is malicious. FP and FN can be computed as the proportion of false positives from the number of negatives, and vice versa:

$$FP = \frac{\text{number of false positives}}{\text{number of negatives}}$$

$$FN = \frac{\text{number of false negatives}}{\text{number of positives}}$$

Consequently, we can also define *True Positive Rate (TP)* and *True Negative Rate (TN)* as metrics for an IDS. TP is defined as the probability that the IDS outputs an alert when there is an intrusion and can be determined as  $TP = 1 - FN$ . Similarly, TN is defined as the probability that the IDS outputs no alert when the behaviour is not malicious. This can be expressed as  $TN = 1 - FP$ .

When developing an IDS, there is always a trade-off between false positive rate and false negative rate: we can make the IDS more sensitive at the risk of introducing more false positives, or can deploy it more restrictively at the risk of rejecting false negatives. The risk of false positives must be balanced against the risk of false negatives when selecting the best IDS configuration.

Similar to the above metrics, we can also use the *sensitivity* of an IDS as a metric. The sensitivity is defined as the proportion of normal behaviour.

$$\text{Sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

A sensitivity of 1 means that all malicious events are detected. But this is not very meaningful since this can be trivially achieved by classifying all behaviour as malicious. Therefore, another metric that we need to determine is the *specificity*. This is the proportion of true negatives of all the negative behaviour examined:

$$\text{Specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

Here, a specificity of 1 means that all normal behaviour is classified as such. Again, specificity alone does not help us much since specificity of 1 can be trivially reached by classifying all behaviour as normal. To combine these two metrics, we can use the *F-measure*. This is the harmonic mean of sensitivity and specificity:

$$F\text{-measure} = \frac{2 \times \text{sensitivity} \times \text{specificity}}{\text{sensitivity} + \text{specificity}}$$

Moreover, there are additional metrics that are similar to the metrics presented up to now, but which bear some slightly different information:

- *Positive Predictive Value (PPV or Bayesian Detection Rate)* is the probability that there is an intrusion when the IDS outputs an alert.
- *Negative Predictive Value (NPV)* is the probability that there is no intrusion when the IDS does not output an alert.

Both *PPV* and *NPV* can be calculated similar to specificity and sensitivity:

$$PPV = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}}$$

$$NPV = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false negatives}}$$

In contrast to specificity, *PPV* yields a measurement of actual normal behaviour in the whole observation set. The important difference between both concepts is that specificity and sensitivity are independent from the total number of samples in the sense that they do not change depending on the fraction of malicious traffic in the whole observation set. In contrast to this, *PPV* and *NPV* are sensitive to this fraction. *PPV* is called Bayesian detection rate [28] since it can also be expressed by using the Bayes theorem (accordingly for *PNV*):

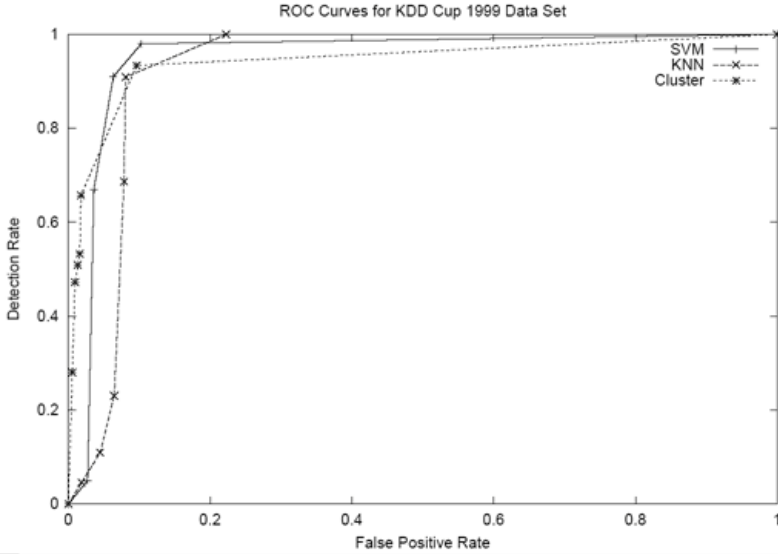
$$PPV = P(\text{actual intrusion} | \text{IDS alert})$$

To determine these metrics for an IDS system, the common way is to use some standardised data set. An example of such a set is the 1998 DARPA Intrusion Detection Evaluation Program [302]. The whole DARPA data set is a test bed that contains normal traffic data similar to that of an organisation with hundreds of users and thousands of hosts. In addition, it contains more than 300 instances of 38 different attack vectors. There is also the 1999 DARPA Off-Line Intrusion Detection Evaluation during which a similar test bed with even more data was generated [303]. These two test beds were used for the most comprehensive evaluation of research in intrusion detection systems. Nevertheless, the results and the methodology are not without controversy, see for example McHugh [343] for a criticism of that work. The main criticism for these two data sets is that they contain *artifacts* due to the way the data was generated.

### Receiver Operating Characteristic (ROC) Curves

Another possible approach to define a metric for intrusion detection systems is to use a *Receiver Operating Characteristic (ROC)* curve and plot the true positive rate (*detection rate*) versus the false positive rate. ROC curves can be used to evaluate the results of different IDS systems. An example of such a curve is given in Fig. 1 (taken from Eskin et al. [144]). Three different algorithms perform unsupervised anomaly detection on a training data set taken from the KDD Cup 1999 Data [9]. This data set consisted of a subset of the 1998 DARPA Intrusion Detection Evaluation Program [302]. The three curves were obtained by varying the baseline of the underlying algorithms and plotting the corresponding false positive/true positives rates.

But which of these three IDS systems is now the best one? Since the ROC curves of the systems cross, there is no easy way to compare them. If the ROC curve for one IDS is always above (i.e., closer to the top left corner) the ROC curve of another IDS, then this means that the first IDS performs better than the second one: for every false positive rate, it has a higher detection rate. But if both curves cross, it is unclear which IDS has the better overall performance. Regarding the figure, we can tell that in certain areas



**Fig. 1.** Example of ROC curve for three different IDS setups

each of the three algorithms has its advantages, e.g., if a false positive rate of about 0.1 is tolerable, then KNN is worse than the two others.

A possibility to extend a metric based on ROC curves is to measure the *Area Under (ROC) Curve (AUC)* and use this as a metric. An area of 1 represents a perfect detection rate of malicious attacks. And an area of 0.5 represents a worthless IDS since the false positive rate always corresponds to a true positive rate. Again, this metric has some limitations since it measures the overall performance of an IDS at all baselines. In practice, however, the IDS would be deployed with the best baseline possible.

### Cost-Based Approaches

In order to integrate the notions of false positive rate and true positive rate, it is possible to assign a *cost* to each of them. This *cost-based analysis* yields a metric that considers the trade off between false negative rate and false positive rate in terms of a (possibly estimated) cost measure. This cost measure can be individually adjusted to differentiate between the damage caused by a successful intrusion or the costs corresponding to a false alarm. For example, a company with lots of sensitive information will presumably prefer a very low false negative rate even if the false positives rate could be high. Ulvila [476] and Gaffney [176] use such a cost-based analysis to combine ROC curves and cost-based estimations to determine the expected cost of several IDS baselines. The expected costs can be used as a metric to identify the best baseline and also to compare different IDS implementations. The difficulties and caveats to assign a cost to false negative rate and false positive rate will be further examined in Sect. 15.

## Information-Theoretical Approaches

It is also possible to transfer concepts from another research area to build a metric for intrusion detection systems. The concept of information theory can be used to motivate an *information theoretic metric* [201] for IDS. This metric is based on the following observation: at an abstract level, the purpose of an IDS is a binary classification of the input data  $X$  (i.e., events that the IDS observes) as normal or malicious. From an information theoretic point of view, this means that we should have less *uncertainty* about the input given the IDS output  $Y$ . This metric – called *Intrusion Detection Capability* ( $C_{ID}$ ) – is the ratio of the mutual information between IDS input and output  $I(X, Y)$ , and the entropy  $H(X)$  of the input:

$$C_{ID} = \frac{I(X, Y)}{H(X)}$$

This metric provides a normalised measurement of the amount of certainty gained by observing IDS outputs. Besides the information given in this paragraph, more information can be found in the technical report by Gu et al. [201].

Another metric for intrusion detection systems is proposed by Helman and Liepins [213]. They model network activity as generated by two stationary stochastic processes, one being malicious and the other legitimate. They formally demonstrate that the accuracy of an IDS is bounded by a function of the difference of the densities of the two processes over the space of transactions. The according metric is called *prioritisation*.