Improving the Evaluation of Network Anomaly Detection Using a Data Fusion Approach

A thesis
submitted in fulfillment
of the requirements for the degree
of
Doctor of Philosophy
in
Computer Science

at
The University of Waikato

by
Andreas Löf

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Department of Computer Science
Hamilton, New Zealand

Abstract

Currently, the evaluation of network anomaly detection methods is often not repeatable. It is difficult to ascertain if different implementations of the same methods have the same performance or the relative performance of different methods. This is in part due to a lack of open implementations, the absence of recent datasets and a no common format to express results.

A common approach to evaluating a method is to use the Defense Advanced Research Projects Agency (DARPA) 1999 datasets, or a derivative of them, in combination with a different dataset or network capture. The DARPA datasets are relatively old and bear little resemblance to modern day traffic and the other datasets are unlabelled and typically publicly unavailable making it difficult to ascertain the validity of the research evaluated in such a way.

This thesis primarily contributes a new evaluation methodology that uses a data fusion based approach that allows for reproducible evaluations with modern datasets.

The new methodology incorporates three other contributions; A new way to capture network traces that are fully anonymised yet retains more information than any current network traces and a new trace annotation format and a method for verifying the correctness of the annotations.

The DARPA 1999 dataset was used to demonstrate the validity of the approach and an evaluation was performed on a new dataset that has been captured using the methods introduced. In the evaluation we find that methodology is a viable approach forward, but that it comes with a different set of drawbacks than the current state of the art.

Acknowledgements

I would like to thank my supervisory panel, Richard Nelson, Michel Mayo and Tony McGregor for their support, guidance and advice while undergoing the process of acquiring a Ph.D.

I would also like to thank the other members of WAND, Jamie Curtis in particular, who have supported me over the years, providing a sympathetic ear and help when needed.

Matt Jervis, Brad Cowie and Paul Hunkin have provided me with much needed distractions and teabreaks, keeping me sane.

Andrea Schweer has showed infinite patience and help throughout the process, and I would not have been able to complete it without her. Words cannot express how grateful I am.

Finally, I would like to thank my family in Sweden, who have always supported me from afar, and always believed in my abilities. Even through I am on the other side of the planet, you never feel far away.

Contents

List of Figures

List of Tables

List of Acronyms

AUC	Area Under Curve
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DoD	Department of Defence
DoS	Denial of Service
DPI	Deep Packet Inspection
FN	False Negative
FP	False Positive
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol Version 6
IDS	Intrusion Detection System
IP	Internet Protocol
JDL	Joint Directors of Laboratories
MAC	Media Access Controller
MAPE	Mean Average Percentage Error
NIDS	Network Intrusion Detection System
RAM	Random Access Memory

ROC	Receiver Operating Characteristics
RTT	Round Trip Time
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
TTL	Time To Live
TP	True Positive
UDP	User Datagram Protocol
URL	Uniform Resource Locator
UTC	Coordinated Universal Time

1 Introduction

Computer networks have become increasingly complex over the years since they were invented, and with the commercial availability of the Internet, this complexity has increased even faster. With the increased usage and complexity of the networks, they have also come to the attention of malicious users. To make networks less susceptible to both operator errors and malicious users, network anomaly detection has become a widely studied field. This thesis focuses on the evaluation aspect of network anomaly detection. As a part of this, a brief discussion of what constitutes science is needed.

The scientific method is a continuously evolving social construct. Aristotle's *Organon* was fundamental in forming an empirical framework for a scientific approach. When Francis Bacon set out an empiric approach in the *New Organon*, empirical observations was still at the core. Both Aristotle and Bacon had elaborate systems of how to deduct or induct knowledge from empirical experiments, but the empirical experiments are a core component. As the scientific method has advanced, the empirical experimentation has been a central part of it (Oldroyd, 1986). As such, the method is based on systematic testing and rigorous observation of an experiment to either prove or disprove a hypothesis.

To conduct research, (Popper, 1972, p. 36–37) argued that testability was key for conducting science. He said

" 4. A theory which is not refutable by any conceivable event is non-scientific. Irrefutability is not a virtue of a theory (as people often think) but a vice.

5. Every genuine *test* of a theory is an attempt to falsify it, or to refute it. Testability is falsifiability; but there are degrees of testability: some theories are more testable, more exposed to refutation, than others; they take, as it were, greater risks."

The summary of Popper's theory was:

"One can sum up all this by saying that *the criterion of the scientific status of a theory is its falsifiability, or refutability, or testability.*"

Popper's concept of falsifiabily, refutability has also become a central part of the scientific method as it asserts that science must be testable. To be able to refute a scientific theory it must be possible to replicate an experiment.

While taking Karl Popper's requirements for refutability into account, satisfying the peer-review process and be able to perform research that satisfies a scientific method, we can establish the following minimum criteria that the conducted research need to conform to:

1. Open datasets.
2. Open, well described, methods.
3. Well described, repeatable, experiments.
4. Clearly presented results.

In the field of network anomaly detection, there is a lack of modern open datasets that are suitable, which inhibits the ability ro refute theories. The last open dataset was made during the Defense Advanced Research Projects Agency (DARPA) Intrusion Detection Challenge in the year 1998, 1999 and 2000. Since then, other datasets have been created but they are not as widespread nor do they contain the truth data necessary to establish the performance of new network anomaly detection methods. Unfortunately, the DARPA datasets contains a number of flaws which McHugh (2000) pointed out. Additionally, the dataset is a synthetic dataset and the traffic patterns in the dataset are different from modern network traffic patterns. This makes it difficult to fulfil criterion 1 that was established above.

Criterion 2 is fulfilled by the publications, but it is unfortunately uncommon that the actual implementations of the published work are made public. As such, it can increase the difficulty with fulfilling criterion 3 depending on the datasets which the experiments used. If the datasets are not public, or the methods are evaluated only against live network traffic, a lack of open implementations will make it impossible to determine if the actual implementations would produce the same results.

Even if the datasets used for the experiments are available, lack of open implementation of the methods can make it difficult to verify if alternative implementations are in fact the same as the original implementation if criterion 4 is not fulfilled. Ideally, the results should be presented both as the performance of the method and as the raw output of the method.

1.1 The Problem

Network anomaly detection methods have been studied for a number of years now. However, the need to evaluate the methods has been obvious since the field of network anomaly detection was established, and there have been various attempts to establish different evaluation criteria and methodologies. The DARPA intrusion detection challenges were the first large scale evaluations, and were run as competitions.

These evaluations came to set the standard for how the evaluations of the network anomaly detection methods were performed and while other methodologies have been proposed since, the methodology established in the DARPA evaluations has been the only commonly adopted method (with some corrections based on criticism). Unfortunately, this methodology relies heavily on synthetic datasets that were created for the intrusion detection challenge, where the full truth is known.

The DARPA datasets used Receiver Operating Characteristics (ROC) (Witten et al., 2012) curves to establish the performance of the different methods, and ROC curves have become the de facto format that a method's performance is shown. There was however never a common format established for the actual output of an anomaly detection technique. Without a common format it is very difficult to repeat the experiments conducted by a method's authors.

As the DARPA datasets have aged, they have come to resemble actual network traffic less and less, causing researchers to start to use other datasets that are not always commonly available and in most cases unlabelled.

As such, new datasets have been created, but they have lacked any proper ground truth. In fact, most datasets created have not had any labelling done at all, with one dataset having had expert labelling applied to it. Instead, the authors of a new method have labelled the output of the methods evaluated. Examples of this are Lakhina et al. (2005) and Tartakovsky et al. (2013).

Expert labelling has been shown to be problematic for other fields, and the sheer quantity of data makes it unsuitable for the datasets in the field of network anomaly detection. This is discussed in greater depth in Chapter 2

Therefore the only way to be able to supply full truth data when a dataset is created is to make the dataset completely synthetic. Unfortunately, a synthetic dataset is not going to have as high quality traffic as a dataset containing real traffic, or might contain artefacts that make it unsuitable by skewing the methods. On the other hand, a perfect dataset will be based on a live network capture. This kind of dataset will not have complete truth data associated with it because of the inherent difficulty of detecting all of the anomalies in a network capture with complete certainty.

There are different compromises to creating dataset that fall between the two extremes presented in the previous paragraph, but the fact remains that it is a trade off between coverage of the truth data versus the quality of the data in the dataset.

The problem is further exacerbated by a lack of open implementations of the methods proposed in the field. While readily available open source implementations would be ideal, a common output format is a more pragmatic solution since it allows researchers to keep their implementations secret while still making it possible for other researchers to verify the results of their alternative implementations. This however also requires common, readily available, datasets.

This thesis sets out to establish a new methodology that takes a different approach to these problems than any of the existing methodologies.

1.2 Thesis Structure and Contributions

The main contribution of this thesis is a new evaluation methodology for network anomaly detection methods that uses a data fusion based approach. The new methodology takes a different approach to network anomaly detection evaluation than previous methods. This does not suffer from the same set of problems as the existing methodologies and allows for reproducible evaluations. The evaluation methodology is built on top of two additional contributions.

Firstly, a new method is introduced that allows for the creation of anonymised network traces that contain more information than previously as it is preserved throughout the anonymisation process.

Secondly, a new annotation format supporting multiple ways of expressing annotations and a method to verify the correctness in the annotations is presented.

1.2.1 Thesis Structure

This thesis is divided into seven chapters, with chapters three to six each focusing on a specific contribution.

Chapter 2 contains the necessary background material for the thesis as well as the related work. The chapter also explores the problem introduced in this chapter in further depth.

Chapter 3 contributes an improved network capture process that preserves more data than the current network capture processes.

Chapter 4 introduces a new network trace annotation format and the process for verifying that the annotations are correct. Both the format and the process are contributions to the field.

Chapter 5 contains a new methodology that incorporates the work presented in the previous two chapters.

Chapter 6 demonstrates how the new methodology can be applied, and demonstrates the viability of the approach.

Chapter 7 contains the thesis conclusions and outlines the further work.

1.3 Ethical Consent

All work covered in his thesis has been approved by the ethics committee at the University of Waikato. Appendix C contains copies of the documents outlining the consent given by the committee.

2 Thesis Motivation

The previous chapter briefly introduced the thesis and motivation for the project. This chapter delves deeper into the problem, and introduces the approaches that have already been taken to solve the problem.

First is a brief overview of approaches to network anomaly detection. This shows that there are several different approaches, they do not always use the same type of data, and that different approaches are able to detect different types of anomalies, but that no method is able to detect all types of anomalies.

Then the problem is described from a top-down perspective. The purpose of this is to clearly define the problem this thesis is addressing. Possible implications on the field are also explored briefly, but it is not to be considered the main purpose of the problem description.

The problems with acquiring truth data for network anomaly detection are then briefly discussed. This section discusses the different ways that truth data can be created, and explores the different quality that the truth data can have. Quality in the context of truth data means the reliability of the data. The reliability is mainly impacted by the correctness and coverage of the truth data.

Finally, the last section of the chapter outlines the existing work that has been done to address the problem, and demonstrate why this work is insufficient in addressing all aspects of the problem.

2.1 A Brief Overview of Network Anomaly Detection

An anomaly in a network can be several types of events. It can be malicious traffic, a sudden traffic spike, a complete lack of traffic or more subtle errors like a high latency. Section 2.1.2 defines these events in more details and divides them into several categories.

There are many different methods that can be used to detect network anomalies. All of these methods can however be divided into two fundamental approaches, active and passive methods.

2.1.1 Active and Passive Measurement and Anomaly Detection Methods

An active measurement involves sending data through the network and recording how the network reacts to the sent data. There are many different types of data that can be used for active measurements. These are referred to as probes. An example of data that is commonly collected using active measurement is network Round Trip Time (RTT)s. This can be collected by sending Internet Control Message Protocol (ICMP) echo requests and measuring the time until an ICMP echo reply is received.

Passive network measurements are made at certain locations in the network, known as capture points. At the capture points, all of the user network traffic passing through that location is collected. The actual collection can be made in several different ways, depending on the guarantees sought that no data will be dropped.

The most common way to capture the traffic is to have a separate computer that is connected to the network via a span port on the local network equipment. A span port copies all of the data going through the network equipment to that port. This data, or a portion thereof, is then serialised to disk in one or several files. When the data is stored to several files, the data needs to be spooled while new files are created or the data will be lost.

Active methods use active network measurement and passive methods use passive measurement.

2.1.2 Types of Events seen as Anomalies

An *event* on the network is defined something of interest happening on the network. These events are divided into two main categories: network faults and security events.

Network Faults

Network faults are considered to be non-malicious in their origin but can still have an adverse effect on the network. We divide network faults into two main categories: hardware failures and configuration errors.

Hardware faults cause an abrupt cessation of traffic over the affected area. Examples are faulty network interfaces or a physical link being severed.

Configuration errors are more difficult to detect than hardware faults. They are generally caused by the operator choosing an improper setting for some or all of the networking equipment. This can cause symptoms such as higher-than-normal latencies, over-saturated links, underused links and inefficient routing.

Security Events

Security anomalies are malicious in nature. No distinction is made between Internet background radiation (Pang et al., 2004) and specific attacks or probes (Barford and Plonka, 2001). The reason for this is that many network probes have become so prevalent that they are considered background traffic by network administrators.

Security events can be divided into two types: probes and attacks.

Probes are attempts to map the network or the services running on a specific node in a network. A well known example of these are port scans.

Attacks attempt to disrupt one or more nodes on the network or attempt to saturate the links in the network itself. Examples are worms (Singh et al., 2004) and Denial of Service attacks (Kim and Reddy, 2008).

2.1.3 Approaches to Anomaly Detection

There are many different approaches to anomaly detection. This section is not an exhaustive list but is rather an attempt to describe the broader approaches undertaken by the research community.

Time Series Analysis

Time series analysis and forecasting is used when the network data can be aggregated into a time series. Examples of data that can be measured might be the traffic volume or the level of entropy.

In the easiest approach the data is measured over a single link (Brutlag, 2000) or it might be measured concurrently over multiple links in the network (Lakhina et al., 2004).

There are different ways to perform the actual anomaly detection. One common approach is to perform a forecast of the time series data and register an anomaly when the predicted value differs too much from the actual value in the time series. A common value for the difference is greater than $2 * \sigma$, where σ is the standard deviation of the time series data, this has been observed in (Brutlag, 2000; Logg et al., 2004; Cottrell et al., 2006).

A different approach to dealing with time series data is using principal component analysis Lakhina et al. (2004). The data is divided into orthogonal principal components, and a decision is made as to which of these principal components contain the anomalies in the original data.

Deep Packet Inspection

Deep packet inspection is the method that is most commonly applied in industry to detect security anomalies. Snort (Roesch, 1999) and Suricata (Suricata, n. d.) are examples of two open source deep packet inspection systems.

Deep packet inspection works by intercepting the individual packets in the network and inspecting both the packet headers and the payloads of the packets. When the system inspects the packets it attempts to match predefined signatures with the packet contents. The signatures need to be defined for a system to successfully be able to detect an anomaly, otherwise the system will not detect it. However, under the assumption that the signature has been created carefully, their false positive rate will be low.

Entropy Measurements

Entropy measurements measure the entropy of the packets in the network link. Normally this is done over packet headers or selected fields in the headers since a complete content inspection is costly. By establishing an entropy measure over the link it is possible to detect changes in the entropy levels that will occur during certain types of attacks on a network.

Flow Measurements

Flow measurement is not an actual anomaly detection approach but is rather a pre-processing step to allow other approaches. By performing a feature extraction on the different types of network flows it is possible to data mine or perform machine learning on the extracted attributes.

Clustering

There are two different approaches to clustering that are primarily used in network anomaly detection. Centroid based clustering and density based clustering (Witten et al., 2012) attempts to cluster the different flows in the network according to their behaviour while attempting to establish one or more clusters as anomalous as the same time.

Density based clustering searches for outlying network flows with the expectation that outlier flows are anomalous.

Classification

There are two types of classification methods that can be applied to network anomaly detection from the field of machine learning: multi class classification and one-class classification. Both are described by Witten et al. (2012).

Any type of classification has two stages, a learning stage and a prediction stage. The learning stage requires pre-classified data that the classification method can deduce classification rules based on the different labels. The prediction stage applies the deduced rules to unlabelled (or test) data and labels it.

Multi class classification requires the data to be split into two or more classes in the learning stage, and rules are created for all of the classes. Single class classification only requires one class to be pre-labelled and only extracts rules for that one class.

The difference between these two approaches is that the single class classifier is better able to cope with data that may not be anomolous or novel whereas the multi class classifier requires examples of all classes beforehand.

2.1.4 Comments

Not all of the approaches introduced here are specific to network anomaly detection, but the presented approaches have been used to detect network anomalies. Chandola et al. (2009) provide a good overview of anomaly detection techniques that are not only restricted to network anomaly detection.

The section introduced several different techniques, and they all work on different levels of data. A deep packet inspection method works on a per-packet-basis whereas a machine learning based method relies on attributes extracted from a network flow. Time series methods rely on yet another level of aggregated data. Combined, these different approaches might be able to capture all different aspects of the anomalies that can be present in a dataset. On its own, it is unlikely that any particular approach will be able to capture all types of anomalies. An example of this would be a deep packet inspection system that is unable to detect a sudden spike in malicious traffic without any malicious data in the packets. An example of such an attack would be a carefully constructed Distributed Denial of Service (DDoS) attack.

2.2 The problem

As described in the previous chapter, modern science is based on a scientific method. In the previous chapter we described a set of minimum criteria on our experiments that need to be followed in order to satisfy the scientific method. These criteria were:

1. Open datasets.
2. Open, well described, methods.
3. Well described, repeatable, experiments.

4. Clearly presented results.

The field of network anomaly detection currently suffer from a lack of modern open datasets, shared implementations of implemented anomaly detection techniques and the output from the anomaly detection techniques. The lack or partial fulfilment of these criteria makes it extremely difficult to repeat the experiments conducted by the original researchers or to perform independent evaluations of the published methods.

The experimental setup is often well defined when a method is introduced, so it is not considered a part of the problem. The other aspects are discussed in more detail below.

2.2.1 Datasets

A dataset consists of two parts. A network trace and the truth data. The DARPA datasets are the only available datasets that contain a full truth data.The DARPA datasets are dated and are unrealistic compared to a real network trace (Brown et al., 2009). Unfortunately, there are no other datasets that have full truth data, which has the implication that these datasets are still in use as a common reference point.

Because of the issues with the DARPA datasets the community has adopted other datasets or live captures to use in addition to the DARPA datasets when evaluating a method. The majority of these datasets are unlabelled, with a minority having had an expert label them. Examples of this would be the DARPA datasets (Lincoln Laboratory, n. d.) and the UMass dataset (University of Massachusetts, n. d.) .

The ideal dataset should be established with full truth data and real live network traces. Unfortunately a live network trace itself might contain anomalies that make this unfeasible. Instead, creating a new dataset is a trade off between quality of generated traffic and the full truth data. To date, the community has relied on using the DARPA datasets with various method authors deciding on the validity of the results against a different dataset. Examples of this is given in Section 2.4.1.

This thesis suggests a different methodology for evaluating network anomaly detection techniques that iteratively relies on the results from previous evaluations and expert consensus on the reliability of these results.

Chapter 3 contains an in depth exploration of longitudinal datasets for network anomaly detection and the difficulties with them. The chapter also establishes the criteria for creating new dataset.

2.2.2 Open Methods

Very few of the published methods have also had the original implementations or results made public. This leaves any researcher with the need to request access to the original implementations, which may not be granted, or to create alternative implementations. Without open datasets or the fine grained output from the initial methods, it is very difficult to ascertain whether the alternative implementations are correct.

2.2.3 Results

The level of results being shared at the moment is in fact the performance of a specific method. The performance is often presented as either graphs, tables or on occasion specific measures like the Area Under Curve (AUC). While this gives enough data for the reader to evaluate the method, it does not provide sufficient data for a different researcher to make an alternative implementation.

The lack of raw results is most likely a direct result of the fact that the space is limited in the publication venues and that it is more important to report on the performance of a method than to provide the actual results of a specific method against a dataset. Open implementations would negate the need for a the raw results, but there is little evidence that the field is moving in that direction. Instead, this thesis proposes a common output format that can be associated, and distributed, with the specific datasets that a method has been run against.

2.2.4 Recognition in the Field

Athanasiades et al. (2003) calls for a new methodology of evaluating intrusion detection systems. They survey the tools and methods that were available at the time, and acknowledge that there are privacy issues surrounding datasets. When they call for a new methodology, they mainly focus on the datasets, which consists of captured, unprocessed, live data and canned attacks that can be mixed with the network captures. However, 9 years after the recommendation, there does not appear to be such a a system in use. In fact, Camp et al. (2009) wrote a wish list calling for a new data set.

Additionally, Athanasiades et al. did not take the Internet background radiation (Pang et al., 2004; Wustrow et al., 2010) into account. Some of the background radiation would be anomalous and thus be labelled by the system under test and skewing the results.

Camp et al. (2009) published a wish list for the security research community. As a part of the wish list, they requested a process for data sharing, and labelled data sets. Explicitly, the process for data exchange calls for "data that is customised to a particular experiment, since the data collected in the absence of an experimental method is typically insufficient to conduct sound experiments" (Camp et al., 2009, p. 3). Additionally they call for "Annotated network traffic traces, such as network traffic from infected hosts". and that the "Traffic traces should be clearly labelled with their respective malicious or benign components, and they should include attacks that current detecting systems have failed to detect" (Camp et al., 2009, p. 4). The work presented in this thesis suggest a method for incrementally creating this type of data for both existing and new datasets.

2.3 The Challenge of Truth Data

There are many different ways to evaluate anomaly detection methods. They all have one elements in common; they want to establish how well a method behaves given a specific metric. All of these methods are dependant on having various datasets to evaluate them against, consisting of network traces and truth data. If the truth data is missing, or difficult to obtain, there are various approaches to attempt to remedy this. This section outlines these approaches.

2.3.1 Expert Labelling

When the data a method is tested against lacks truth data, an expert can label the data. Expert labelling is a good method for small datasets but rapidly becomes unfeasible when the size of the dataset grows. Depending on the nature of the data, it can sometimes be very difficult to distinguish an anomaly from normal data.

It does not appear that there have been any formal studies on expert performance in the field of network anomaly detection, but the findings from research performed on expert performance in other fields suggests that the experts might not be infallible.

Camerer and Johnson (1997) summarised that *"The depressing conclusion from these studies is that expert performance in most clinical and medical domains are no more accurate than those of lightly trained novices.'(We know of no comparable reviews of other domains, but we suspect that experts are equally unimpressive in most aesthetic, commercial, and physical judgements.) And expert judgement have been those of the simplest statistical models in virtually all domains that have been studied. Experts are sometimes less overconfident than novices, but not always."* (p. 349).

This summary shows the need for studies on experts performance in network anomaly detection, or anomaly detection in general.

Machine learning is the closest related field that has studied the expert performance when creating new datasets. A study by Snow et al. (2008) who evaluated the usefulness of the Mechanical Turk, found that when natural language clustered non-experts labels can compare to expert labels when using inter expert agreement to asses the similarity in the non-expert labels. The study used a very small data set, only a 100 instances, to create the labels, and require each instance to be labelled by 4 non experts to achieve the same performance as a single expert.

To label a network flow, the labeler needs to posses a certain level of domain knowledge, making solutions like the Mechanical Turk unsuitable as few labelers will posses that domain knowledge. There are however certain higher level of anomalies, that affect the trace as a whole, that labelers not possessing the domain knowledge could label. This is clearly not feasible given the sizes of the datasets and the requirements on the labelers, instead other means of creating labels are needed.

In network anomaly detection, a dataset will consist of millions of instances, each instance being a single network flow. In the trace set Waikato 8 (see Chapter 3), 2011-06-02 contains 349,717,701 packets and 13,837,822 flows. The dataset introduced in Chapter 3 consists of 18 days, each with similar amounts of packets and flows.

In order to establish an exhaustive set of labels, each and every instance would need to be reviewed by multiple labelers, ideally a combination of expert and non-experts (possessing sufficient domain knowledge) would be used to achieve a high level of agreement between the labelers. Without a high level of agreement, the produced labels would not be reliable.

To summarise, expert, or non-expert, exhaustive labelling is impossible to achieve for a modern network trace due to the volume of the data and the number of labelers needed to establish reliable labels.

Clustering of Expert Classification

In an attempt to remedy the problems of scale when an expert labels a dataset, it is possible to cluster on features extracted from the data. This is demonstrated by Zhong et al. (2007). Unless all of the data is examined by the expert, there will also be an uncertainty surrounding the exhaustiveness of the expert's classification. This is in turn increased by the need to choose a feature set that unambiguously captures the nature of the anomalies and the normal traffic in order to perform correct clustering.

2.3.2 Ground Truth

Establishing a complete set of ground truth for a data set can only be done with a synthetic dataset. Since all of the traffic is introduced by the researcher when creating a synthetic dataset, only the intended anomalies will be present. Generating a synthetic dataset that is representative of real world network is impractical because of the difficulties with generating background traffic that appears to be authentic and is guaranteed to not contain any undetected anomalies. Apart from that, the synthetic data needs to be merged with anomalies that are either synthetic or created on a real network. To date, the DARPA datasets are the only datasets for network anomaly detection that have had a complete ground truth established as a part of the creation of the datasets.

In Chapter 5 we introduce a new methodology to create partial ground truth, but there are also a number of closed source commercial systems that may be possible to use. Examples of these are ArcSight ESM (NDM Technologies, n. d.), Splunk (Splunk Inc, n. d.) and Autonomy(HP Autonomy, n. d.). These systems are rule or heuristics driven and the commercial aspect of them makes them less suitable as an open research platform in an academic environment.

2.4 Existing Methodologies

Puketza et al. (1996) introduced a methodology for evaluating intrusion detection systems. The methodology is primarily targeted towards host network intrusion detection systems, but is also presented in the context of network anomaly detection. The methodology introduces the need for recording anomalies so the evaluations can be re-run, but does not extend further than to record the anomalies. As such, it is an important paper as it shows the awareness of the need for repeatability. Unfortunately the approach falls short as it does not recognise the need to share the results or make it possible for other researchers to repeat the tests in the same conditions as the original tests.

Athanasiades et al. (2003) reviewed several of the then existing approaches to evaluate network intrusion detection methods and found them lacking. The authors recognises that the informal approach described in Section 2.4.1 is prevalent but makes it near impossible to fairly compare methods. They call for a need of a new methodology that is both open and repeatable, using synthetic or semi-synthetic data to generate the data for the evaluated method. To date, no methodologies have received widespread adaptation in the field of network anomaly detection, instead the field appears to be in the same state that Athanasiades et al. (2003) describes.

Chen et al. (2009) introduces such a methodology for testing intrusion detection systems that was described by Athanasiades et al. (2003). The methodology relies on having the system under test placed inside a controlled environment where the tester has control of all network traffic. The network traffic is played back from previous captured traces and anomalies are injected into the background traffic and the performance evaluated. The proposed methodology does however not consider the fact that the network traces used may already consider anomalies inside them, thus affecting the results. Nor does it emphasise the sharing of experiment parameters or results.

Dumitras and Shou (2011) introduce WINE, the Worldwide Intelligence Network Environment. WINE is focused on the collection of malware samples and the reputation of binary files and Uniform Resource Locator (URL)s, antivirus signatures, email spam, and malware samples. They do not address the issues surrounding network traces, but rather step away from it and deal with anomaly detection on an application level. However, Dumitras and Shou discuss the importance of reproducibility and explain that WINE will save all experiment setup. It is however unclear how this will allow for a comparison when the implementations can not always be shared amongst researchers.

2.4.1 Informal Methods

While none of the methods proposed in the papers above seem to be in use by the wider community, an informal approach inspired by the DARPA intrusion detection evaluations (Lippmann et al., 2000b,a) has been adapted by the community. This method consists of the following steps:

1. Create a new method.
2. Evaluate the method against the DARPA dataset.
3. Evaluate the method against a live traffic stream or other unlabelled capture, relying on expert knowledge to validate the performance.
4. Present the results in ROC curves or by presenting the detected anomalies in a different format.

Step 2 is sometimes omitted in favour of only relying on step 3. Examples of papers relying on these approaches are Lu and Ghorbani (2009) ,Kang et al. (2012), Mahoney and Chan (2003) and Paschalidis and Smaragdakis (2009).

Step 3 is used to have additional validation in place since the DARPA datasets are old and have well known flaws. These additional datasets are not always recorded (Lu and Ghorbani, 2009) and not always shareable (Seppälä et al., 2010) due to privacy reasons.

2.5 Discussion

All of the related work presented in this chapter fails to address the issues surrounding trace datasets and common formats for network anomaly detection. The work presented by Dumitras and Shou (2011) shows promise, but it does not address any of the issues about labelling of actual network data since the work focuses on more aggregated data than raw network traces.

Instead, the work presented in this thesis, and in particular in Chapter 5 and the following two chapters, address these issues surrounding datasets in a pragmatic manner. It is recognised that it is not an ideal solution, but it is however an improvement over the current state of affairs in the field of network anomaly detection.

2.6 Summary

This chapter has discussed the existing network anomaly detection evaluation methodologies and their shortcomings. We have shown the awareness in the field that the current approach has shortcomings, and we have identified the three major aspects that we focus on in this thesis. The need for a methodology that encompasses the sharing of results as its core, the need for a new dataset to use for network anomaly detection and a format to share the results in.

3

Collecting a New Dataset

In the previous chapter we established that there is a lack of proper methodologies and datasets that can be used to evaluate network anomaly detection techniques.

This chapter describes existing datasets and then contributes a new method to capture datasets, and a new dataset. The new method allows for the capture of a dataset that retains more information than a current anonymised dataset without violating the network users privacy without investing a significant amount of resources.

The new method of capturing datasets plays an important role in the overall methodology contribution since it makes it easy to create new datasets for network anomaly detection that will have more anomaly information associated with them than the current datasets.

3.1 Existing Datasets

There exist a number of datasets for network anomaly detection already. The ones we present here are in a considerable number of papers or other publications.

The only datasets that come with full truth data are the DARPA (Lincoln Laboratory, n. d.) and KDD (Hettich and Bay, 1999) datasets. UMass (University of Massachusetts, n. d.) has released one partially labelled dataset. MAWI (MAWI, n. d.) has started an initiative to label two of their historical datasets and the new data that is continuously captured. At the moment their labelling is also not exhaustive. Other datasets are completely unlabelled and only consist of various types of network traces.

3.1.1 DARPA

The three following datasets were released as part of three intrusion detection evaluations. The evaluations were performed by Lincoln Laboratories at the behest of DARPA in the USA. The evaluations were focusing on both host and network intrusion, but we shall only consider the network aspect of them for this thesis.

The first evaluation was released in 1998 and consisted of 9 weeks of network captures as well as truth data for the captures. During the original evaluation, the dataset was split up into a training and testing phase, with only the training truth data released at the time. After the evaluation was completed they released the truth data for the testing dataset as well.

McHugh McHugh (2000) wrote a critique that is mainly focused on the DARPA 1998 evaluation over the 1999 evaluation. The critique focuses on both issues with the dataset and how the evaluation itself was performed. In his paper McHugh discusses both the flaws and how to address them. In this chapter the focus is on the dataset aspects, but in Chapter 5 the other parts of his criticism will be discussed.

In the 1998 evaluation, McHugh points out that the artificial background data is devoid of any anomalies, the data rates are suspiciously low and that there is only a small subset of the machines in the data that are the target of any attacks. Another issue that McHugh highlighted is that there are no intentionally anomalous background traffic in the data generated.

It is difficult to ascertain how much the traffic patterns have changed since the studies due to the lack of any longitudinal studies in that timescale. However, the general usage patterns of the Internet have changed advent of services such as Youtube, bittorrent and other peer-to-peer protocols. The thread model on the Internet has also changed with the arrival of worms such as Slammer and Stuxnet, distributed Secure Shell (SSH) password bruteforcing, and botnets. Since the DARPA datasets feature none of these types of attacks or background data, it has become less relevant.

It is also well known that the attacks on the target systems have a different Time To Live (TTL) than the background traffic, making it easy to achieve good results on that particular dataset.

In Chapter 4 we will also discuss some issues with the truth data of the DARPA 1998 and DARPA 1999 datasets.

Year	Citations per year
2007	65
2008	76
2009	65
2010	59
2011	49
2012	49

Table 3.1: Citations for the DARPA 1999 dataset in Google Scholar

There have not been any studies performed that show the usage of the DARPA datasets. Table 3.1 show the number of citation of the DARPA1999 evaluation (Lippmann et al., 2000a). A small subset of these articles were examined to see whether they discussed the dataset or whether the work presented contains evaluations against the dataset, with the finding that the majority of examined articles had been evaluated against the dataset. These numbers are a strong indication that the DARPA 1999 dataset is still widely used and discussed. Even though its usage has declined slightly in the last two years, it can still be considered relevant.

DARPA 1998

The network aspect of the dataset consists of tcpdump captures that ran for 8-12 hours each day during the evaluation, 5 days a week. The dataset consists of a 7 week training period and a 2 week testing period.

The training period has few anomalies in it and slowly increases the number during the duration of the training phase. The testing phase has similar characteristics as the final week of the training phase.

Initially only the truth data was released for the training phase, but once the evaluation was completed the testing truth data was also released.

DARPA 1999

The evaluation performed in 1998 was repeated in 1999 with changes to address some of the criticism that had been raised. Most notably, the DARPA 1999 dataset introduces new attacks compared to the 1998 one and a different network setup. The network was partitioned into an internal and an external network, and a capture point established on each network.

The dataset consists of two phases, the training and testing phase. The training phase contains three weeks of data, with week 1 and 3 free of anomalies. The testing data consists of two weeks of data, both containing anomalies. The 1999 evaluation also introduced new attacks that were not present in the 1998 evaluation.

The DARPA 1999 dataset is the most commonly used dataset, as seen in Table 3.1.

DARPA 2000

The last of the DARPA dataset was created to address specific scenarios not present in the previous two datasets. The main new features introduced was the launch of a DDoS attack.

3.1.2 KDD 99

The KDD 99 dataset was released as the KDD CUP (Hettich and Bay, 1999) dataset in 1999. The dataset is derived from the DARPA 1998 dataset and a number of features has been extracted from the trace data.

Tavallaee et al. (2009) point out that the KDD 99 dataset suffers from two major problems. The first is that there are a large number of redundant records that will skew any evaluations relying on this dataset. The second problem is that even very simple methods will achieve an 86% success rate. In order to remedy this they have released a new version of the dataset, NSL-KDD (Tavallaee et al., n. d.). The new dataset has had the distribution of records adjusted to remove the redundancies and have had the ratio of anomalies to non-anomalies modified. These two modifications yield a more balanced dataset which allow researchers to perform less skewed evaluations.

3.1.3 DEFCON

The DEFCON conference is a conference that is held annually. The conference targets hackers of various backgrounds. During the conference there are normally different sorts of activities for the participants. One of there activities are the "capture the flag" contests. The capture the flags contests have a number of vulnerable hosts on a network that the attackers try to compromise. The network is only used for the contest, which means that no background noise or normal traffic exist that can be encountered on a real shared network.

There have been four datasets (Shmoo Group, n. d.) (DEFCON, n. d.) captured during the DEFCON capture the flag contests. The datasets contain full payload, but no truth. In addition to this, all of the traffic can arguably be considered malicious due to the nature of the contests.

3.1.4 Internet 2

Internet 2 is a backbone network that has been established primarily between universities. The network is mainly used for research purposes and was established as an attempt to make a new, better, Internet. There are flow records being collected from six core routers in the Cisco Systems (n. d.) version 5 format. The number of routers have changed over the years as the network has evolved.

The records contain metrics for every flow passing through the network as well as up to a 1% sample of the packets. The sample contains full payloads and have had the last octet in the IP addresses zeroed out as the anonymisation.

The benefits of using the Internet 2 dataset is that the data comes from many different links, which allows one to work with the behaviour of a network instead of just the behaviour over a single link.

The drawback is that it only contains NetFlow records for the majority of the data. This means that one can only use the flow attributes established by the NetFlow version 5 format and it is impossible to extract any additional attributes from the data.

The Internet 2 dataset has been used in multi-variate time series types of anomaly detection, such as the principal component analysis performed by Lakhina et al. (2004).

3.1.5 UMass

UMass has a network trace repository (University of Massachusetts, n. d.), where the Gateway Link 3 Trace has been used for network anomaly detection.

Gu et al. (2006, 2005) manually labelled parts of this trace for their studies in anomaly detection using maximum entropy. Their work focused on detecting SYN-floods and port scans, thus the labelling is only done for these two types of attacks. It is also important to note that it is not an exhaustive labelling, but rather that they chose from a subset of ports and hosts for their studies and only labelled the events concerning those criteria.

3.1.6 MAWI

MAWI (MAWI, n. d.) is a trace repository that is a part of the WIDE project. MAWILab (R.Fontugne et al., n. d.) is a repository where anomalies are stored that they have discovered in the traces from the MAWI repository. The anomaly detection methods used were run against two traces in the MAWI repository. It is not an exhaustive collection of anomalies because the anomalies are only collected after the capture. There has also not been an exhaustive attempt to detect all of the anomalies, but rather only a small subset of all possible anomalies.

The work by Fontugne et al. (2010) describes the data collected in the traces and the anomalies detected.

3.1.7 LBNL/ICSI Enterprise Tracing Project

The LBNL dataset (Laboratory and ICSI, 2005) is a four month long trace. The original trace has been split into two separate sets, one containing port-scan traffic and one set that contains the remainder of the traffic. Both sets have been anonymised with the tcpmkpub tool. The tcpmkpub tool and details about the anonymisation are described by Pang et al. (2006).

3.1.8 ITOC

The ITOC dataset (Sangster et al., 2009a) was created during a network warfare exercise and contains full packet payloads, but no truth data.

The most important contribution of the ITOC dataset is that Sangster et al. (2009b) introduces the concept of capturing additional information during the collection of the network trace. Unfortunately the focus of the paper is to only do this during a network warfare exercise and not when a generic network trace is captured.

3.1.9 Network Traces

The final part of the commonly used datasets used are from trace archives. There are numerous archives around the world, such as WITS (Waikato Internet Traffic Storage, n. d.) and RIPE (RIPE, n. d.).

The traces have a few attributes in common; They were captured over a single link, contains no or a heavily truncated payload and have had their IP addresses anonymised. The ways to anonymise addresses is discussed more in Section 3.6.9.

3.2 Problems With Existing Datasets

3.2.1 Privacy Issues

There are two different privacy related problems that cause difficulties. The first problem is not being able to distribute the dataset at all because of where it has been acquired. The second problem is that when the dataset is being distributed, it must be anonymised. There are in turn two steps of the anonymisation that have an impact on the dataset, Internet Protocol (IP)-address anonymisation and packet payload truncation. The packet truncation is the main problem because it makes it impossible to recreate all of the network events after the anonymisation.

3.2.2 The Lack of Ground Truth

Establishing the complete ground truth for a dataset is very difficult, and can only be done in a controlled environment mixing both synthetic and real network data. The synthetic data is used to increase the traffic volume and the real network data is generated by users inside the controlled environment.

Because connecting the controlled environment to the rest of the Internet will subject it to the Internet background radiation, and potentially malicious traffic, it should be kept in complete isolation.

To establish the ground truth in this controlled environment, all background traffic and traffic generated by users must be completely benign. The creators of the dataset can then introduce various forms of anomalies in the network, while recording which anomalies were introduced, and the time and location in the network.

Creating a dataset in this fashion is both costly and time consuming, and it will be very difficult to establish an equal amount of anomalies as can be found in ordinary Internet background radiation (Pang et al., 2004; Wustrow et al., 2010). While this approach was used for the DARPA intrusion detection challenges, it has never been repeated since.

There are two different approaches to creating new datasets, but neither approach can guarantee that all of the anomalies in the constructed dataset will be correctly detected and labelled.

Both approaches use existing network infrastructure and rely on passive monitoring of the network. As was discussed earlier, this will require the data to be anonymised. The two approaches can be described as follows, introducing anomalies into your own network or simply attempting to label all anomalies in the network. It is also possible to combine these two approaches. However, none of these approaches can guarantee that all anomalies will be labelled.

When introducing anomalies in a network already a part of the Internet, it will also have anomalies induced from external sources, some of which are difficult to detect (such as 0-day exploits).

3.2.3 Discussion

The difficulties and costs associated with a creating a new high quality dataset for network anomaly detection has led to a lack of new datasets that have full truth data. There have been attempts by researchers to establish new datasets, such as the UMass dataset, but they have not had as large coverage of truth data, if any. Their strengths have rather been that they are authentic network captures, but that comes at the cost of it being very difficult to label.

Instead, the hybrid approach of using multiple datasets for evaluations have been used.

The main drawback with all others of the previously described datasets are that they do not have an exhaustive labelling and that any labelling has been created after the data collection. As a result of that, it is much more difficult to establish whether an anomaly detection method is yielding correct results.

3.3 Anonymity of Users in a Dataset

A dataset containing real world network traffic has the potential to invade the privacy of the users of the network where the dataset was captured. To counteract this, it is an established practise to anonymise the datasets before they are released, and in many cases before they are examined by the researchers.

There are several pieces of information that need to be anonymised in a dataset, with the most obvious being the packet payloads. The other information that need to be anonymised is addresses in the packets since it can be used to identify specific computers.

Synthetic datasets do not need to be anonymised as they do not contain any incidental data that can be considered sensitive. All of the data in the dataset is a product of the researcher creating the dataset.

3.3.1 Anonymising Packet Payloads

Packet payloads can be anonymised by either truncating them or overwriting them with irrelevant data. Truncating packets is by far the most common method since it also decreases the size of the trace. Packets are commonly truncated after a certain number of bytes. This approach is likely to either leak payload data, or it is likely to truncate packet headers.

The dataset presented in this chapter uses a different method. By analysing the packets and headers, the capture system is able to determine what is the packet payload and truncate the packets at a dynamic interval, thus preserving the packet headers.

3.3.2 Anonymising Packet Addresses

Even if the payloads have been truncated, it is still possible to extract some information of what a user has been doing since addresses can be tied to specific network hosts. There are several ways that have been used to anonymise the packet addresses.

Enumeration

All of the hosts in the trace are placed in the same network. Each address that is encountered inside the trace is given a unique address inside that network in the order that they are encountered. The drawback with this method is that it does not preserve any network prefixes which leads to further information loss.

Encrypting the Addresses

Various encryption methods have been used to encrypt the network addresses. The benefit of this approach is that it can be used to only encrypt a part of the address, thus making the approach prefix preserving. The drawback is that the method is reversible if the encryption key is discovered or retained.

3.3.3 Implications

Information that is useful for network anomaly detection is removed when a dataset is anonymised. By removing packet payloads, the dataset is rendered unusable for Deep Packet Inspection (DPI) methods. Methods that work in the address space suffer similarity from the anonymisation of the packet addresses.

Both these drawbacks make the task of labelling a dataset after capture and anonymisation significantly more difficult.

3.4 Classifying Datasets

The following criteria are used in the classification of the datasets presented.

3.4.1 Access

Datasets access can be divided into two main categories: Public and Private. A public dataset is accessible for download and use without any restrictions. A private dataset is not accessible without prior consent and an agreement of the terms that the private dataset may be used under.

3.4.2 Creation

There are three ways to create datasets: Synthetic, Semi-synthetic and real network traffic. A synthetic dataset is created by using a network traffic generator to create both anomalies and background traffic. A semi-synthetic dataset mixes both real network traffic with synthetic network traffic generated by a traffic generator. A dataset created from real network traffic has had network data captured from one or more network links.

Each method suffers from different drawbacks. Synthetic traffic is dependant on having an accurate and realistic traffic generator. If the generated traffic is not realistic the dataset will provide bias or incorrect results network anomaly detection.

Semi-synthetic traffic relies even more on the synthetic data being indistinguishable from the real network data that it is mixed with. If anomalies are combined with generated data, it is also difficult to say whether the anomalies would be present in a real scenario, and in which volumes. Failure to take proper care when presenting this kind of dataset would result in very artificial datasets, from which results might translate poorly to a real world scenario.

Real traffic datasets do not suffer from any artefacts from the capture process, assuming that sufficient resources are available at the capture point and the capture software is bug free. Instead, this kind of datasets suffer from a lack of truth data, it is very difficult if not impossible to fully label a real world dataset.

3.4.3 Labelling

The labelling of a dataset can be done during the creation, or after the creation of the dataset. The labelling of a dataset during the creation is easiest on a synthetic or semi-synthetic dataset since the anomalies in the dataset will be known (unless they are inadvertently introduced by the traffic generator). No real world dataset exists that have been labelled during capture.

Post creation labelling can done either by hand or by automated tools. Both methods suffer from the same drawback, the reliability of the methodexpert that assigns the labels. Additionally, experts find it difficult to cope with big datasets whereas automated systems do not suffer the same scalability issues.

3.4.4 Overview

Table 3.2 contains a classification of the different datasets discussed in this chapter and a breakdown of how the dataset was created (type), how the data is anonymised and the state of the truth data associated with the dataset. Additionally, it also contains a note about how the data has been created or derived.

Table 3.2 show that the only datasets that with full truth data that have been created are the DARPA datasets. There are a multitude of potential other datasets, but none have been widely adopted for network anomaly detection.

3.4.5 Discussion

Synthetic and semi synthetic datasets can have full packet contents and complete truth data, but suffer from not being representative of real networks. The effort and inherent costs involved with creating such datasets means that they are not frequently updated.

Real traces are inherently realistic but they need to be heavily anonymised to be shared. The fact that they are real network data makes it impossible to have verified complete truth data.

The rest of this chapter presents the completion of a dataset through a new

Dataset Name	Type	Form of Anonymisation	Truth Data	Notes
DARPA 1998	Synthetic	None	Full	Network intrusion detection challange
DARPA 1999	Synthetic	None	Full	Network intrusion detection challange
DARPA 2000	Synthetic	None	Full	Network intrusion detection challange
KDD 99	Synthetic	None	Full	Derived from DARPA dataset
NSL-KDD 99	Synthetic	None	Full	Derived from KDD 99
Defcon	Network Trace	None	None	Network intrusion contest
Internet 2	Flow records	Address Truncation, 1% Full Packets	None	
Umass	Network Trace	Packet Headers/Address anonymisation	Partial, Expert labelling	Trace repository
MAWI	Network Trace	Packet Headers/Address anonymisation	None	Trace repository
LBNL/ICSI	Network Trace	Packet headers/Address anonymisation	None	Trace repository
Generic Network Traces	Network Trace	Packet Headers/Address anonymisation	None	Trace repository
ITOC	Network Trace	None	None	Network intrusion detection and defence exercise

Table 3.2: An overview of the different datasets that have been used for network anomaly detection

technique that allows retention of most of the packet data that is relevant to anomalies. This technique is the based on annotating the trace as it is captured.

Later chapters discusses techniques to further improve the annotations and truth data associated with a trace.

3.5 Dataset Quality

Evaluating a dataset's quality falls outside the scope of this thesis, but the following aspects of dataset quality have been inferred from the work conducted with dataset analysis, labelling and collection done in this thesis.

1. Completeness
2. Packet loss
3. Dataset duration
4. Timestamp precision
5. Accuracy of dataset labels
6. Quality of synthetic traffic
7. Coverage of truth data
8. Correctness of truth data

3.6 Collecting a New Dataset

To remedy the lack of a fresh dataset I decided to capture a new dataset. Capturing a dataset has the benefit of being real world traffic as opposed to creating a synthetic dataset. Unfortunately it will not have complete truth data. This is because the data is captured over the University of Waikato's primary connection. Because this is on a production network with real users there will be a large number of anomalies that are naturally occurring. As a part of the data collection, attempts were made to label as many of these anomalies as possible but it is impossible to verify how many of the anomalies were correctly detected.

The systems used to label the collected data during the traffic capture were

Snort Roesch (1999) and Bro(Paxson, 1998).

The following two sections explain the setup of the capture point and what type of data that was collected.

3.6.1 Motivation

A new network trace itself would not be very useful for network anomaly detection. There are already multiple institutions that collect network traces. However, a majority of the collected traces are not made publicly available or are too limited. The traces that are made publicly available are just network traces, with no additional data.

By collecting more data than normal during the network capture, we are able to alleviate the uncertainties present when attempts are made to establish the correctness of results from an anomaly detection method.When the data is captured, a first step towards this is to use well known methods that are considered reliable, such as deep packet inspection.

The aim of the work described in this chapter is to create a new dataset that comes with additional metadata that is only available during the capture. Although only two IDS systems ran at the capture point, it shows that it is possible to make a new dataset that contains more information than just a network trace consisting of packet headers. Since almost none of the network traces released contain packet payloads it would be impossible to recreate the results of an DPI system on the truncated packets.

Chapter 5 describes in detail how it is possible to extend on the results attached to a specific network trace. It is still important to gather as much information as possible during the actual capture of the network trace.

3.6.2 Methodology

The basic idea is simple: attach as many network probes as possible at the capture point, and make sure they receive exactly the same data. In this context, a network probe is something that inspects or measures the traffic at the capture point. Ideally, the probes should only contribute data that cannot be reconstructed after the capture. This means that IDS systems that perform deep packet inspections are better candidates as probes than something that measures bandwidth usage.

The probes should be tuned so that they have an low false negative rate, especially if it is relatively easy to filter out false positives later by selectively ignoring specific alerts. Using Snort as an example, we enabled many rules that generates alerts, that might only be of interest in very specific cases. The reasoning is that it is possible to filter the alerts out after the network capture, but it might be impossible to recreate the conditions that led to the alert.

All significant events that occur during the capture of the network traffic should be saved in such a fashion that it is uniformly presented and in correct temporal order. To do this, we created a binary file format and a classification of different types of events. The file format and the classification is described in Chapter 4 and appendix B. Chapter 4 also describes how different annotation formats can be converted into this file format and compared.

3.6.3 Capture Point Setup

Figure 3.1 shows the network setup at the Waikato University capture point. The capture point resides outside the firewall and is connected to the capture point via a span port on the gateway router. The capture point itself is also connected to an additional private network over a different interface over, which the traffic is transported to storage.

The capture point is a Inter Core 2 Duo E8200 with 8 GB of Random Access Memory (RAM). For the data collection it has an Endace DAG card, version 4.3*GE*. The DAG card uses GPS as the clock input to ensure that there is no clock skew or inaccuracies in the timestamps of the collected traffic. It uses the libtrace (Alcock et al., 2012) version 3.0.12 and WDCap(WAND Network Research Group, n. d.) version 3.1.12 for the data collection and

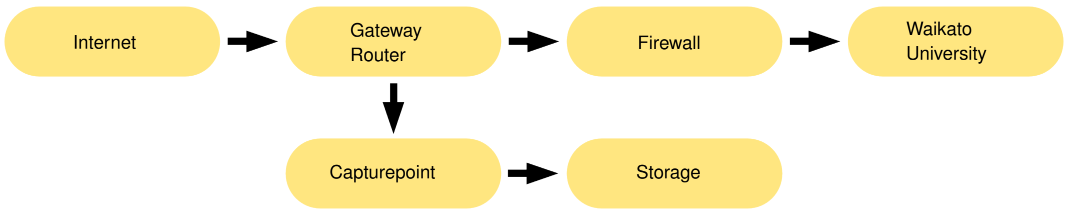


Figure 3.1: A network diagram showing the capture point setup at the University of Waikato

streaming of traffic for storage.

3.6.4 Data Collected

All data that is being transported over the link is collected. Section 3.6.9 details the steps taken to ensure the privacy of the users of the link.

The data collection has the approval of the University of Waikato's ethics committee. Appendix C contains a copy of the ethics agreement and the conditions that the data collection has to operate under.

While full packets were sent to the two network intrusion detection systems, only the anonymised packet headers were stored.

In order to be able to only store the packet headers and 4 bytes of payload, wdcap is protocol aware. This allows wdcap to dynamically alter the snap length based on the different protocols present in the packet. Wdcap dynamically changes the capture length to be the User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) header, complete with options, plus 4 bytes. Wdcap also supports ICMP and Internet Control Message Protocol Version 6 (ICMPv6). If a protocol is unsupported it will be treated as user payload and is truncated.

This dynamic snapping allows for the complete capture of link layer headers, network headers and TCP, UDP and ICMP headers.

3.6.5 Labelling During Capture

In order to perform partial labelling of the traffic during the capture we ran two different systems on the capture point during the data collection. These systems were chosen because they are mainstream deep packet inspection based network intrusion detection systems that see large scale deployment in many different organisations.

Figure 3.2 shows the data paths during the capture from the network. The data going into storage is completely anonymised and have the majority of the payload stripped. The data paths to Snort and Bro are identical. The packets in these data streams have their payload intact but have their IP addresses anonymised.

Both systems received separate data streams of the anonymised traffic, but

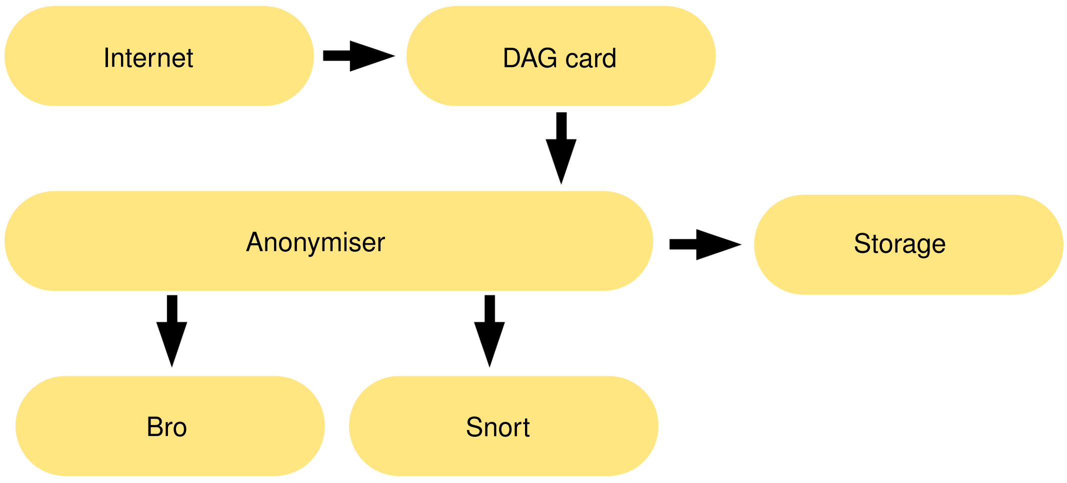


Figure 3.2: The different data paths used during the network capture

with full packet payloads. The data streams were buffered during the time that they had been opened until the systems had started processing packets in order to minimise packet loss. If the systems were not connected to the data streams no buffering was done due to the large amounts of data.

3.6.6 System Configuration

Both systems were monitored by watchdogs that restarted them when the systems terminated prematurely. This proved to be necessary for both Bro and Snort, but for different reasons. Snort was unable to restart gracefully when the binary rules were updated, instead Snort attempted a restart and then aborted. Bro suffered from stability issues during our research and would sometimes segfault.

Unfortunately, this problem led to some of the captured data not being analysed by the systems, but by using buffering of the data and the watchdogs we were able to mitigate this as much as possible. Tables 3.3 and 3.4 contain the upper bounds of unprocessed packets.

Snort

Snort is a network intrusion detection system that mainly relies on deep packet inspection to for deciding whether to raise an alarm or not. It also contains a number of pre-processors the most useful features being the ability to do stream tracking, detect port scans and detect protocol violations.

Snort was running both the Sourcefire VRT[1] and the Emerging Threats[2] open rule sets. A cron job ran at midnight each night to update the signature files to the latest versions and snort was restarted after the update. The Snort version used during the data capture is 2.9.0.5 and the VRT rule sets starting with the 2011-04-28 release and ending with the 2011-05-17 release. The EmergingThreats rules are updated on a daily basis and were also updated on a daily basis at midnight.

The main drawback to using the open rule sets compared to the commercial ones is that the open rule sets are lagging behind by 30 days compared to the commercial ones. This means that attacks that were less than 30 days old at the time of capture will not have been detected.

Bro

Bro is a network intrusion detection system that relies more on user configuration in its Turing complete language than on signatures of specific attacks. It can however use a subset of the snort signature description language and can thus also load signatures that only relies on the subset of the language.

The Bro installation was configured using the scripts that was distributed with the Bro source. There were no attempts made to write custom detectors for Bro, but rather to see what it could detect using the policies and detectors distributed with the system.

Bounds on the Number of Unprocessed Packets

Table 3.3 and Table 3.4 shows an upper bound of the packets not processed by the respective system when they were not running. The last and first ts columns contain the timestamps in Coordinated Universal Time (UTC) for the events reported by the system before they shut down and after they were restarted. The two systems do not report the first and the last packet captured, only the first and last event detected. By correlating the first and last reported events with the packets that triggered the alerts we can establish an upper bound on the number of packets that were not

[1]http://www.snort.org/vrt
[2]http://rules.emergingthreats.net/

Last TS	First TS	Packets
1307059200.908943	1307059201.041437	6573
1307145600.912754	1307145601.095437	3953
1307232000.823158	1307232001.570763	4931
1307318400.914954	1307318401.100041	3839
1307404800.770291	1307404801.069437	7356
1307491200.998200	1307491201.010361	6776
1307577600.979166	1307577601.090223	9223
1307664000.842506	1307664001.097906	6532
1307750400.937954	1307750401.100407	3541
1307836800.991659	1307836801.826832	5109
1307923200.737272	1307923201.175639	5884
1308009600.621993	1308009601.164393	7436
1308096000.786078	1308096001.985541	12850
1308182400.997815	1308182401.040782	7654
1308268800.922332	1308268801.054057	7262
1308355200.782837	1308355201.039382	2633
1308441600.973966	1308441601.111439	2290
1308528000.979519	1308528001.044631	4883

Table 3.3: Packets between first and last event in Snort

Last TS	First TS	Packets
1307581477.438300	1307582214.325630	4753957

Table 3.4: Packets between first and last event in Bro

processed.

3.6.7 Volumes of Data Collected

The data collection ran from 2011-06-02 to 2011-06-20. This collection happened between semesters at the University of Waikato. The impact of this is that the volumes of traffic is smaller than during a semester because fewer students were at campus and using the university's computer facilities. It is also worth to note that the "World IPv6 Day" occurred on 2011-06-08, which is a day where network operators where encouraged to enable IPv6.

The dataset averages 2.3 MB/s with a peak throughput of 31.7 MB/s. The lack of students does not have a significant impact on the diurnal variations compared to a time during the year when teaching is in progress at the

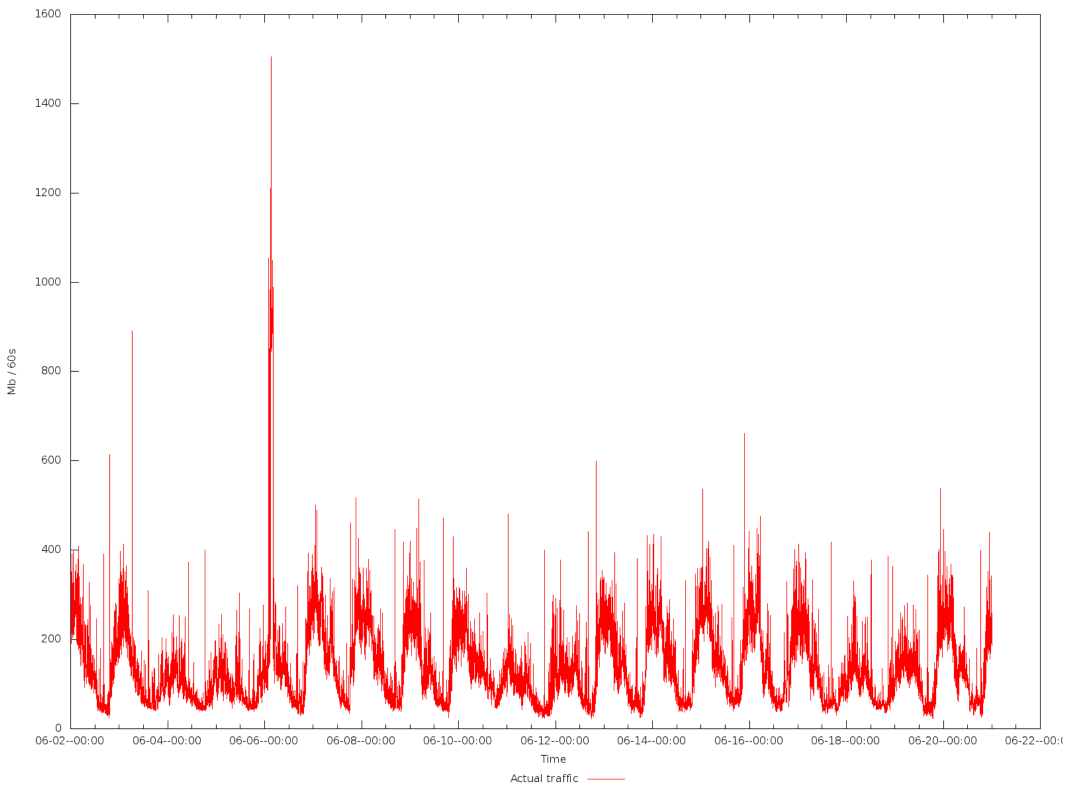


Figure 3.3: Total Traffic During Capture

university. This can be seen in Figure 3.3. The large spike is caused by an increased rate of outgoing traffic over the link.

There were no packets dropped during the collection of this network trace.

3.6.8 Privacy Concerns

As can be seen from the ethics agreement in Appendix C there are strict conditions in place to ensure the privacy of the users of the network. This section outlines the steps taken to ensure the privacy.

3.6.9 Identification of Individuals

The IP addresses of the individual packets can be used to pinpoint specific computers. When combined with the timestamps in the collected traces it can be used to identify specific individuals. To make this impossible, the addresses present in all IP packet (both IPv4 and IPv6) have been anonymised with Crypto-PAn[3] which implements the techniques de-

[3]http://www.cc.gatech.edu/computing/Telecomm/projects/cryptopan/

scribed by Fan et al. (2004).

Crypto-PAn is based on the rijndael cipher. It is prefix preserving, which means that the prefix relationships of the IP addresses are still usable after the anonymisation. While Crypto-PAn is cipher based, and can be reversed if the key is retained, no key is retained. A new key is randomly generated when the network capture starts and is rotated every seven days. The key rotation helps to ensure that it is impossible to identify specific hosts based on long running traffic patterns.

3.6.10 Packet Payloads

Even if the addresses of an IP packet has been anonymised, the payload can still contain sensitive information such as e-mail data or HTTP data. This is information that obviously must be kept private. Thus, all payload except for the first 4 bytes are stripped from the packets. While retaining any payload at all might be contentious, 4 bytes was approved by the University of Waikato's ethics committee. 4 bytes of payload is sufficient to be able to be able to do protocol identification and similar types of research without posing a too big risk of network user privacy.

The technical details of the payload anonymisation can be seen in Section 3.6.4.

3.7 Discussion

While the capture described in this chapter could have been longer, it should be viewed as a proof of concept. It is possible to create a semi-labelled dataset during capture without a significant overhead in time or resources. Specifically, it allows us to combine deep packet inspection systems with post-capture behavioural methods without violating the privacy of any users.

Unfortunately the lack of packet payloads mean that the dataset is not useful for research into deep packet inspection systems, but the focus on the research described in this thesis is on behavioural anomaly systems and the two deep packet inspection systems used are mainly used for a reference purpose.

While it can be argued that the accuracy of the annotations for the dataset could be improved by launching attacks against the network inside the capture point during the capture, this approach faces some drawbacks on its own.

The legality of attacking a network you do not own is dubious, only a few hosts inside the network could be attacked thus creating the same predictability for some of the attacks as present in the DARPA datasets, doing so would also increase the cost of the capture.

3.7.1 Weaknesses

The major weakness of the new dataset that was collected is that it is impossible to know about all of the attacks. There is most likely attacks in the captured traffic that were not known during the time of capture by the systems monitoring the network. It is possible that these attacks will be noticed by other systems during future evaluations that use the dataset, but it will be impossible to verify these attacks with complete confidence.

3.7.2 Achievements

We have created a new procedure for collecting network traces that yields more information about the network trace than other methods. Combined with the already robust data collection methods that is afforded by wdcap and libtrace, this allows us to collect datasets that are more useful for network anomaly detection.

3.8 Summary

This chapter has introduced datasets that have been used to date for network anomaly detection, and discussed the strengths and shortcomings of these datasets. Additionally, the different techniques to create a dataset has been and the strengths and weaknesses of the three approaches (synthetic, semi synthetic and real network traces) where we draw the conclusion that only real network traces are realistic, but suffer from the need to anonymise them.

To alleviate the loss of data during the anonymisation the chapter introduced a new network capture technique that attempts to collect relevant network anomaly annotation data during the capture so that the information can be retained as a part of the dataset.

4 Annotation Formats

As described in Chapter 2, evaluating network anomaly detection methods need to have a standardised way of sharing the results to be able to perform direct comparisons between methods. This chapter outlines a new, extensible, network annotation format that can be used in conjunction with network traces to share the results when sharing the implementations of network anomaly detection methods is impractical.

However, just another data format would be insufficient to make evaluating network anomaly detection methods easier, when we cannot ascertain the correctness of the produced annotations. To remedy this, this chapter also introduces a method to verify that the annotations produced are correct and that the annotations can be correlated with the correct events in the network trace. The chapter then brings the new annotation format and the verification method together in an example scenario where the truth data from the DARPA datasets are converted into the new format and verified for correctness.

When the new annotation format and verification process is used in conjunction with the network trace process described in the previous chapter it is possible to collect anonymised real world datasets that have more information associated with them than previously.

4.1 Motivation

There are many different ways to express annotations that are useful from a network anomaly detection point of view. When anomaly annotations from different sources and in different formats are brought together, it is difficult to know if these annotations can be trusted.

To establish that the annotations are correct and can be trusted, a process was created that match each specific set of anomalies to the network trace that contains the traffic where the annotations originate from, and verify that a packet or network flow can be associated with the annotation . By matching anomalies, to the largest extent possible, to the network trace, it is also possible to improve on the accuracy of the annotations.

In this chapter, accuracy is defined by how unambiguous an annotation is. Events are the actual cause of the annotations. By verifying that the trace contains the traffic that the annotations refer to, the events will also be present in the trace unless the annotation is the cause of a false positive.

4.2 Why Network Annotations Matter

Annotation of a network trace allows network researchers to share information about a network trace. Currently, due to a lack of suitable formats to store annotations in, this is usually passed as notes that are loosely tied to the network trace. A better method to share annotations would allow for more efficient sharing of information as well as the ability to share results from anomaly detection methods. There are currently two different formats that are discussed in more detail in Section 4.4. These formats are not flexible and extensible enough to allow an efficient sharing of information.

Section 4.5 describes the attempts to create a new annotation format that is closely tied to the network traces without having to change any of the existing trace format. This format has been designed with flexibility and extensibility as two of the design goals and can easily accommodate different extensions as well as be extended into a complete trace format with detailed annotations as a part of it.

4.2.1 About Network Trace Formats

There are a wide number of different formats for network traces, and with one exception they do not support inline annotations. The most prevalent format is the PCAP (Tcpdump/Libpcap, 2000) format. Microsoft Network Monitor(Microsoft, 2010) supports limited annotations in its native format, but only in a limited scope and the format has not gained any wide use. This is discussed more in Section 4.4.2.

Network trace formats can be generalised to the following; they contain complete or parts of raw frames as captured over the link layer, and meta data about these frames. What link layers are supported and what the meta data contains depends on the format itself. The two formats that are used in the research described in this thesis are PCAP and ERF (Endace, n. d.).

PCAP is the most common packet trace format and is supported by a wide range of application and libraries. ERF is mainly used for network traces captured by the Endace DAG cards.

4.2.2 Network Events

The events that we have an interest of are network faults and security events. Network faults can be either configuration errors or hardware faults. Security events are either network probes or network attacks.

A more comprehensive description of the network events is in Section 2.1.2.

Categories of Events

Events can be put into three different categories. Trace wide events, flow specific events and packet specific events. These events need to be labelled in such a way that they can be traced back to the originating event. An example of this would be a Denial of Service (DoS), which we would classify as a tracewide event with the attacker as the source and the victim as the destination. In the event of a DDoS attack we would only use the victim as the identifier.

Identifying Events

Each type are straightforward to identify. The trace wide annotations do not require any further identification than the timestamp they happen since they encompass all of the trace. Packet level events can be annotated using the information inside the packet or frame as well as the timestamp. A network flow is best identified by using the information present in the first packet or frame in the flow.

There is also a need to support for identifying events that affect specific hosts or services. To identify an event associated with a specific host, the timestamp and IP-address is used. To identify a server, the timestamp, IP address, protocol, and port are used.

4.3 Requirements on Annotations

Annotations primary purpose is to facilitate communication and sharing of information between both researchers and end users. To be able to accommodate this, they need to be expressible and flexible. To be useful for information sharing, the annotations need to provide both outstanding accuracy for identifying specific events as well as allow for a wide range of descriptions of what the annotation signifies.

To be useful to researchers, the annotations also need to be able to show how much confidence the researchers have in a particular annotation. Additionally, the annotation format should be extensible in such a way that different versions of the format can co-exist without any difficulties.

These simple use cases can be used to derive the following requirements:

1. Support high granularity timestamps.
2. Support an unique identifier for each annotation.
3. Be closely correlated to the network trace.
4. Support annotating different scopes of events.
5. Allow for easy comparison between different sources of annotations.
6. Flexible description format.
7. Support mixing annotations from several sources.
8. Extensible while retaining backwards compatibility.

4.3.1 Different Levels of Accuracy

There are two different aspect of accuracy, temporal and spatial. Temporal accuracy is concerned with locating a specific event in time. Spatial accuracy refers to the ability to pinpoint a specific event based on the network traffic in the trace. An example of a spatial coordinate within a network trace is the 5-tuple that describes a flow, containing the source and destination addresses and ports and the protocol. Both temporal and spatial accuracy is important to be able to correctly identify a specific event in a network trace. The source of the event can either be an anomaly detection method or an expert evaluating the network trace, or external sources of events such as syslog or SNMP traps.

4.4 Existing Annotation Formats

There are two existing annotation formats with the express purpose of identifying events within a network trace.

4.4.1 ADMD

ADMD (Fukuda et al., 2008) has been developed with the purpose of sharing scores for anomaly detection methods. It is an XML based format. ADMD supports annotations of packets or of slices. Packets are identified by a hash of the packet. A slice, as used by ADMD, is a complete or partial network flow between two hosts. The slice is identified by source and destination IP addresses, protocol, source and destination ports, and a duration.

Shortcomings

ADMD does not support annotations that are trace wide. Out of the three categories of events defined in Section 4.2.2, only packet and flow are supported. This does not fulfil requirement 4.

One annotation file only supports annotations from one specific scope, thus failing requirement 7. The format is not extensible due to it's implementation, failing requirement 8.

Strengths

ADMD allows researchers to add a description of the annotation file as well as information about the algorithm that created the annotations, including parameters. The annotation format outlined in this format does not support this additional meta data because this would decrease the extensibility of the format.

4.4.2 Microsoft Network Monitor

Microsoft Network Monitor has support for annotations when it is using its own custom annotation format. However, it is not possible to annotate a captured trace without saving and reopening it beforehand.

Shortcomings

The format only supports annotating packets and it has to be a manual procedure. This means that it is impossible to add annotations that are not tied to a specific packet. This makes it unsuitable for network anomaly detection since it is impossible to automatically label the trace, or read back the labels, as well as annotate anomalies that are not tied to a specific packet.

The format does fulfil requirements 4, 5, 7 and 8.

4.4.3 Network Logs

For the scope of this thesis, results from a network anomaly detection system, for example snort, which produces logs are not considered to be an annotation format as they are created with the express purpose of communicating the output of the method.

4.5 Creating the Waikato Annotation Format

To rectify the shortcomings in the two formats discussed in the previous section, a new annotation format was created. The goal was to fulfil all of the requirements in Section 4.3 and make it straightforward to extend the format without invalidating any created annotations in the future.

The decision was made to make the format binary, with all data stored in network byte order. The primary reason for a binary format was speed and uncompressed storage size. An implementation of the necessary functions to read and write the data will be released as open source software together with this thesis. This version submitted with this thesis is available at http://hdl.handle.net/10289/8041 and future versions will be available from http://research.wand.net.nz.

4.5.1 Annotation Format

Based on the requirements in Section 4.3 a new format was created. This format supports a time resolution as fine as the trace format, the ability to identify trace wide events, hosts, services, flows, and packets.

Extensibility

The current version of the format only supports annotations, but the design of the format is such that it is easy to extend it. Each different record is identified by an Internet Assigned Numbers Authority (IANA) enterprise number. This number can be used to identify the implementation of the format that created that specific record, which also means that the records should conform to the format as defined by that organisation. This would allow different institutions to add additional information in their own implementations of the records without explicitly breaking any existing design.

4.5.2 Format Outline

Figure 4.1 shows an annotation as it will be stored in the file. Each actual annotation is encapsulated in a container that identifies the content type, the enterprise that created the annotation and the total length of the record. An annotation, as encapsulated by the container, actually consists of three parts; the annotation, the identifier and the description.

The annotation part defines what type of event signified, the timestamp, a confidence number and a divisor. In addition to this, it also tells what type of identification is used for the annotation. The confidence number defined the belief held in the annotation's accuracy and the divisor is used when serialising annotations to disk in order to not lose precision.

The supported event types are: trace wide, flow, packet.

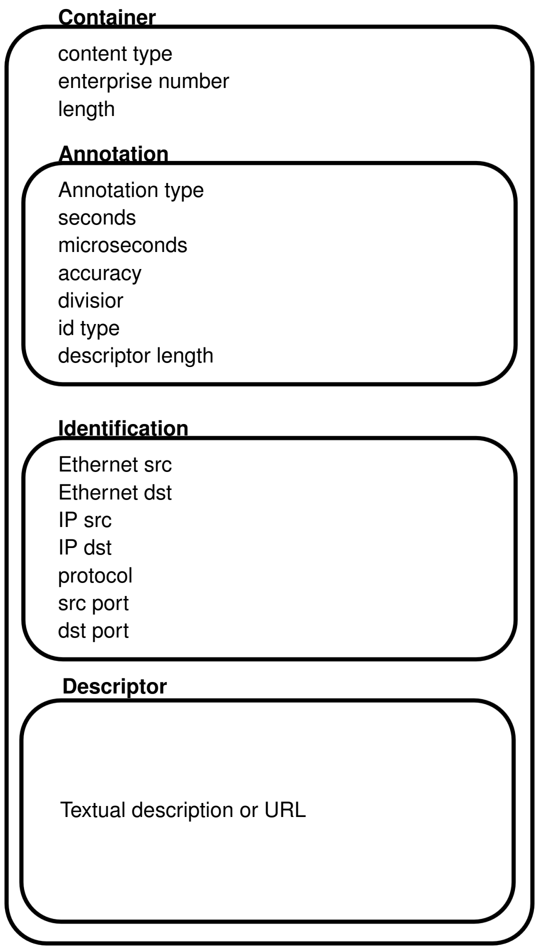


Figure 4.1: An Annotation record

Appendix B contains the C header files that the implementation relies upon.

Identification Types

There are currently four supported identification types. The most descriptive is Ethernet, the second most is Service, then there is Host and finally None. Figure 4.2 shows how these identification types are related and what each type contains.

The Ethernet type is used to identify packets and flows captured over an Ethernet link. It is possible that an additional identification type for ATM links, or not using the link layer addresses at all might be added in the future.

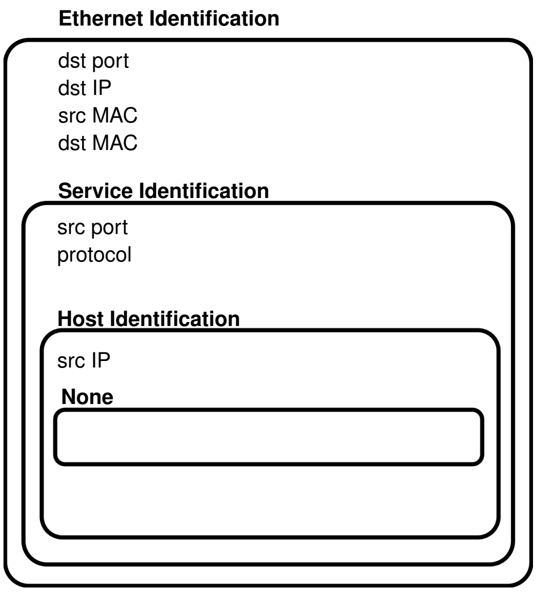


Figure 4.2: The relationship between annotation identification types

4.5.3 Descriptions

Normally, the descriptions for an annotation is made in the style of an semantic URL. The URL should directing the user of the annotation format to a web page with more information about the specific annotation. A semantic URL is structured in such a way that information is conveyed by the URL itself. The main benefit of having a semantic URL is that it still conveys information about the annotation even though the resource it points to might be missing.

An example of a semantic URL is as follows:

```
http://www.wand.net.nz/annotations/waikato8/snort/sid/128/revision/1
```

This URL consists of several parts. **www.wand.net.nz** tells us which organisation created the annotations. The name of the dataset the annotation is associated with is **waikato8**. The source of the annotation if **snort** and the Snort rule with the **sid 128** was triggered. The final part of the URL shows us that revision 1 of the rule was used.

4.6 Requirements on the Annotation Process

In order to verify that the annotations were closely coupled to the network trace, we established the following requirements on the process:

1. Sufficient temporal resolution to match the network trace's resolution.
2. Tag both packets and network flows in the traffic.
3. Allow for easy comparison between different sources of annotations.

In Section 2.1.2 the different types of events that can be detected in a network trace was established. Not all of these will easily satisfy the last requirement due to detection latency in the algorithms that produced the original set of annotations.

An example of this is how Snort detect port scans. Snort uses a specific pre-processor that investigates the traffic and looks for port-scan like patterns. The pre-processor does not report the port scan with the timestamp of the first flow, but rather when the ongoing port scan crosses a detection threshold, thus creating suffering from reporting latency.

4.7 Implementation of the Matching Software

To verify the correctness of the annotations created from data sources, custom software was written. The main purpose of the software is to correlate the annotations with the network traces, and report any lack of successful matches between annotations and the network traces.

There are four different variants of the software available at the moment. Each variant is capable to read a specific input format and match it against the network trace.

The supported formats are: Snort's unified2, Bro's log format, DARPA 1998 truth format. The final variant also reads the DARPA 1998 format but does fuzzier matching to allow for the sparser information present in the 1999 format.

All four variants are based on the same code base and uses the same data flow. However, each version has been tweaked to handle different detection delays present in Snort and Bro whereas the DARPA version corrects for the lack of temporal accuracy .

Figure 4.3 outlines the path that the data takes during the conversion of input annotations to the Waikato annotation format. The first step is to read the input from the specific format. The second step involves converting the input into a common format that is similar across all variants of the software. Finally the events are matched to specific packets or flows in the trace. This is discussed in detail in Section 4.7.1. Once a successful match has been made, an annotation is written with the correct details.

4.7.1 Matching

All variants use the same approach to matching the input to the trace files. The various input sources do not provide the same level of accuracy, which means that the matching mechanisms must compensate.

Snort provides the most complete input in the unified2 format. In this format, for each event, a header providing data about the event and the packet that caused the event to be logged is included.

From this it is possible to extract a 8-tuple that will be unique across the entire network trace. The tuple consists of:

1. timestamp

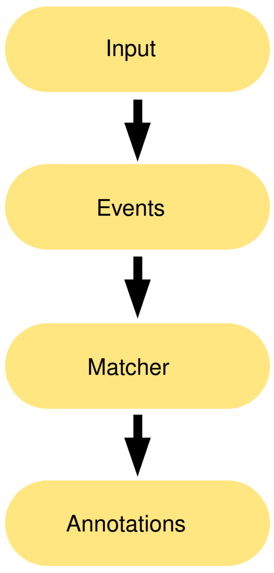


Figure 4.3: The datapath during the conversion of input annotations to the Waikato annotation format

2. source Media Access Controller (MAC) address
3. destination MAC address
4. source IP address
5. destination IP address
6. transport level protocol
7. source port
8. destination port

In this tuple, the timestamp can suffer from clock skew between the capture point and the computer running Snort, clock drift and detection delay. This can be solved by running Snort in off-line mode and sending it the captured traffic in real-time in the PCAP format. When Snort is working in offline-mode, it uses the packet timestamps present in the network trace it reads.

Snort Variant

When matching the unified2 data to the network trace, each of the fields described above were compared.

For efficiency reasons, the matcher starts by comparing timestamps. If the timestamps have a discrepancy larger than one second, the packet is not considered.

The following checks are then performed. If any of the checks fail, the matching is aborted for that particular packet.

1. source and destination MAC addresses.
2. source and destination IP addresses.
3. transport level protocol type.
4. source and destination ports.

The final step, verifying the source and destination ports, are only done for the transport level protocols that uses ports.

If a packet successfully matches all of the criteria in the input annotation, there are two annotations written. The first is a packet level annotation for the particular packet. The second is a flow level annotation, that is using the 8-tuple for the first packet present in the flow as its identifier.

By having both flow and packet annotations it is easy to model both packet and flow characteristics without having to do multi-pass processing.

Bro variant

Unlike Snort, Bro has the detection delay added to the timestamps. Bro also does not log the MAC addresses in its log formats. This makes it more difficult to match the input to the network trace.

By tuning Bro, the detection delay was changed from the default 30 minutes (for some types of events) to 5 minutes. While this is still a relatively large time window to perform the matching on, it is such that it is unlikely that we will encounter two flows with exactly the same parameters.

The actual matching process performed is almost identical to the one for snort, with the exception that a larger temporal window is considered and that no MAC addresses are compared.

Due to the lack of exact timing, the Bro variant only creates annotations for flows, not flows and packets.

4.7.2 DARPA 1998 variant

The DARPA 1998 truth data has a granularity of 1 second for the timestamps. There are no MAC addresses present in the truth data.

The matcher used is based of the matcher for Snort, but with more flexibility for the temporal matching and no comparison of MAC addresses.

The lack of consistent labelling in the truth data between packets and beginnings of flows makes it impossible to create correct annotations for the packets in the trace. Instead, only annotations are created for the network flows.

4.7.3 DARPA 1999 variant

The DARPA 1999 truth data uses a different format from the 1998 truth data. The main difference is that the DARPA 1999 truth data lacks the source port in many occasions. The differences will be explained in greater detail in Section 4.8.1. The 1999 truth format was converted to the same format as the 1998 truth data using a simple Perl script.

The matching works exactly the same way for the 1998 data, except that if the source port is missing, only one unique flow is matched to that event.

4.8 Investigation of the DARPA Dataset

This section outlines the different errors and their frequency as they were discovered in the DARPA 1998 and DARPA 1999 datasets. There were few errors found in the 1998 truth data, but we discovered that 14.42 % of the instances in the 1999 dataset are incorrectly labelled. This number includes three annotations in the truth data that will be discussed in depth in Section 4.8.1.

4.8.1 Truth Formats

DARPA 1998

The 1998 training data come with the network data being annotated. The annotations list each individual flow, regardless of it being an attack or not. As a pre-processing step to matching the network data to the supplied attack truth we filtered out all negative instances as well as the identification number assigned to every flow.

Figure 4.4 show the original truth data and Figure 4.5 the filtered version.

DARPA 1999

The 1999 dataset differs slightly from the 1998 in the way that the network was configured. Lincoln Laboratory created an internal and an external network with a capture point in each network. The supplied truth data, however, are not divided into an internal and an external network. To simplify the matching of the network data we made the decision to merge the two different network captures using tracemerge, which is a part of libtrace (Alcock et al., 2012). Tracemerge allowed us to merge the two network captures without modifying the data stored in the original network traces. We then used the merged traffic to match the truth data to the network data.

The truth data distributed with the 1999 dataset is in a different format compared to the 1998 dataset. There are two different formats available,

```
06/02/1998 00:00:07 00:00:01 http 2127 80 172.016.114.207 152.163.214.011 0 -
06/02/1998 00:00:07 00:00:01 http 2139 80 172.016.114.207 152.163.212.172 0 -
06/02/1998 00:00:07 00:00:01 http 2128 80 172.016.114.207 152.163.214.011 0 -
06/02/1998 00:00:07 00:00:01 http 2129 80 172.016.114.207 152.163.214.011 0 -
```

Figure 4.4: Original DARPA 1998 truth data

```
06/01/1998 08:05:07 00:03:06 telnet 1941 23 135.008.060.182 172.016.112.050 1 format_clear
06/01/1998 08:07:13 00:05:07 telnet 2064 23 135.008.060.182 172.016.112.050 1 ffb_clear
```

Figure 4.5: Processed DARPA 1998 truth data

the one used for this work is the **master_identification.list**. A manual comparison between the annotations present in that file and the detections lists showed that the master identification list was more complete.

The three main differences between the data in the master identification list and the DARPA 1998 truth data are:

- Attacks are listed, not instances.
- Source ports missing for almost all attacks.
- Different formatting.

An example of an attack record in the original truth data for the 1999 dataset is in Figure 4.6.

To cope with these differences we started out by converting the formatting in the 1999 data set to the 1998 formatting. This was done by replicating the attack instance for each attacker / victim pair by the number of times a specific port or jump packet was attacked. On the attack instances where there was not a one-to-one or one-to-many mapping for attackers and victims, we manually inspected the network traffic and created separate one-to-many attack instances.

The attack in Figure 4.6 is converted into three instances, each with an unknown source port. The source and destination addresses are the same for each instance, but the destination port will either 20, 21 or 513 in each

```
ID: 41.135830
Date: 03/29/1999
Name: ftpwrite
Category: r2l
Start_Time: 13:58:16
Duration: 00:05:45
Attacker: 194.027.251.021
Victim: 172.016.112.050
Username: 513/tcp , anonymous
Ports:
At_Attacker:
At_Victim: 20-21{1}, 513{1}
```

Figure 4.6: Example of an DARPA 1999 attack record.

.

instance. Timestamps and durations are retained from the original record.

Wild Cards in the Truth Data

The truth data for the 1999 dataset includes a number of wild cards. These wild cards were interpreted in a inclusive fashion. As an example, an address of 192.168.0.* would be interpreted as all valid addresses in that subnet, which are 192.168.0.1 to 192.168.0.255. If there were only one actual host present in the trace, it would count as one correct annotation and 253 incorrect annotations.

This approach was not taken for a smurf attack present in the last week of the testing data. Rather, the correct number of flows was extracted and they were all classified as mislabelled in the original truth data.

Classes of Errors

- Incorrect timestamps on attacks.
- Wrong datestamp on attacks.
- Erroneous source and destination ports.
- Erroneous source and destination addresses.

4.8.2 Frequency of Errors

This section lists the errors that we found in the truth data supplied with each of the data sets we examined. We also classified the types of errors based on our findings in the network captures.

1998 Dataset

Table 4.1 outlines the errors found in the 1998 training and testing data. The week and day fields show which part in the dataset is affected by the mislabelled truth data. The attack field indicates the name of the attack, the number of instances in the total number of instances affected by the errors. The final field, Error, contains a brief description of the error.

Table 4.2 shows the attacks with affected instances. The # Attacks column shows the total amount of attacks for that day. The following column shows how large a percentage of these attacks suffer from incorrect instances.

Table 4.3 shows the total number of instances for each day of the data

Part	Week	Day	Attack	#Instances	Total #Instances	Error
Training	Week 4	Friday	loadmodule	3	6	Src and destination ports swapped
Training	Week 5	Monday	satan	12	5203	Wrong destination address
Training	Week 6	Tuesday	portsweep	491	1804	Timestamp off by 24 hours
Testing	Week 2	Thursday	ignore	2	601	Src and destination addresses swapped
Testing	Week 2	Friday	httptunnel	10	99	Timestamp off by 24 hours

Table 4.1: Errors in the DARPA 1998 truth data

Phase	Week #	Day	# Attacks	% of Attacks with errors
Training	Week 4	Friday	8	12.50
Training	Week 4	Monday	4	25.00
Training	Week 6	Tuesday	5	20.00
Testing	Week 2	Thursday	19	5.26
Testing	Week 2	Friday	23	4.35

Table 4.2: Erroneous attacks in the DARPA 1998 data

Phase	Week #	Day	# Instances	% of Instances with errors
Training	Week 4	Friday	600	0.50
Training	Week 4	Monday	12085	0.10
Training	Week 6	Tuesday	1850	26.54
Testing	Week 2	Thursday	242602	$\simeq$0.00
Testing	Week 2	Friday	37392	0.03

Table 4.3: Erroneous instances in the DARPA 1998 data

containing errors as well as how large a percentage of these instances are erroneous.

1999 Dataset

This section contains listings of the errors in the 1999 training and testing data.

Table 4.6 shows the errors present in the truth data for the training data. Week 2 was the only week with errors in the training phase of the evaluation.

Table 4.7 shows the errors that were discovered in the two weeks of test data for the 1999 data set.

Day	# Attacks	% of Attacks with Errors
Monday	7	28.57
Tuesday	10	40.00
Wednesday	5	60.00
Thursday	9	33.33
Friday	9	55.56

Table 4.4: Erroneous attacks in week 2 in the DARPA 1999 data

Table 4.4 show the total number of attacks and how large a percentage of these attacks are affected in week 2 of the training data. Table 4.5 shows

Day	# Instances	% of Instances with Errors
Monday	64	31.25
Tuesday	3314	0.15
Wednesday	12680	0.02
Thursday	24297	0.01
Friday	20248	0.02

Table 4.5: Erroneous instances in week 2 in the DARPA 1999 data

the total number of instances for each day and how large a percentage of these instances contains errors.

Week	Day	Attack	# Inst.	Total # Inst.	Error
Week 2	Monday	pod	2	2	Wrong src address
Week 2	Monday	ntinfoscan	18	18	Wrong timestamp
Week 2	Tuesday	phf	1	1	Wrong timestamp
Week 2	Tuesday	httptunnel	1	1	Wrong timestamp
Week 2	Tuesday	eject	3	3	Not in network trace
Week 2	Wednesday	perl	1	3	Wrong timestamp
Week 2	Wednesday	crashiis2	1	1	Wrong timestamp
Week 2	Thursday	secret	1	1	Wrong timestamp
Week 2	Thursday	perl	1	1	Wrong timestamp
Week 2	Thursday	land	1	1	Wrong timestamp
Week 2	Friday	crashiis	1	1	Wrong timestamp
Week 2	Friday	loadmodule	1	1	Wrong timestamp
Week 2	Friday	perl	1	1	Wrong timestamp
Week 2	Friday	eject	1	1	Wrong timestamp
Week 2	Friday	phf	1	1	Wrong timestamp

Table 4.6: Errors in the DARPA 1999 training data

Table 4.8 and Table 4.9 shows the number of attacks for week 4 and week 5 of the testing data as well showing how big the percentage of these attacks contain errors.

4.8.3 Correcting the Erroneous Truth Data

Using the methodology described earlier in this chapter, it was possible to pinpoint exactly which of the annotations were incorrect. This allowed for a careful manual investigation of the trace files to find the correct attributes. Based on the successful examination of all of the errors, it was possible to create a set of corrected annotations that are closely tied to the network

Week	Day	Attack	# Inst.	Total # Inst.	Error
Week 4	Monday	crashiis	1	1	Wrong timestamp
Week 4	Monday	portsweep	7	23	Wrong timestamp / Not in trace
Week 4	Monday	sshtrojaninstall	1	2	Wrong timestamp
Week 4	Wednesday	portsweep	1	3	Not in trace
Week 4	Wednesday	smurf	270	398	Not in trace
Week 4	Thursday	ipsweep	9	30	Not in trace
Week 4	Thursday	netbus	3	4	Wrong timestamp / Wrong src ip
Week 4	Thursday	teardrop	1	1	Wrong timestamp
Week 4	Friday	netbus	2	4	Wrong src ip
Week 4	Friday	smurf	253	253	Not in trace / Wrong timestamp
Week 5	Monday	portsweep	3	17	Wrong timestamp
Week 5	Monday	smurf	5998	5998	Wrong src ip
Week 5	Monday	udpstorm	253	254	Not in trace
Week 5	Tuesday	fdformat	1	7	Not in trace
Week 5	Tuesday	ftpwrite	1	3	Wrong timestamp
Week 5	Tuesday	syslogd	1	1	Wrong timestamp
Week 5	Tuesday	tcpreset	1	1	Not in trace
Week 5	Tuesday	udpstorm	253	254	Not in trace
Week 5	Wednesday	ffbconfig	1	2	Wrong timestamp
Week 5	Wednesday	netbus	2	3	Wrong src ip
Week 5	Thursday	resetscan	254	254	Not in trace
Week 5	Thursday	satan	11302	11302	Wrong timestamp
Week 5	Friday	arppoison	1	1	Wrong date
Week 5	Friday	land	1	1	Wrong timestamp
Week 5	Friday	mscan	5005	5005	Wrong date/not in trace
Week 5	Friday	netcat	1	2	Not in trace

Table 4.7: Errors in the DARPA 1999 testing data

Day	# Attacks	% of Attacks with Errors
Monday	13	23.08
Wednesday	16	12.50
Thursday	14	21.43
Friday	12	16.67

Table 4.8: Erroneous attacks in week 4 in the DARPA 1999 data

Day	# Attacks	% of Attacks with Errors
Monday	19	15.79
Tuesday	22	22.72
Wednesday	13	15.38
Thursday	17	11.76
Friday	22	18.18

Table 4.9: Erroneous attacks in week 5 in the DARPA 1999 data

trace. The corrected truth data will be made available as both plain text files and in the Waikato Annotation Format at http://hdl.handle.net/10289/8041when this thesis is handed in.

Examples of the corrections would be to correct the timestamps of the annotations, or to correct the source or destination addresses that had been reported.

4.9 Discussion

The different levels that the annotations can be expressed upon, packet, flow, tracewide, allow us to differentiate between different types of events and how wide they affect a network. It is also possible to have different granularity in the tuple describing the event, thus making it easy to describe what an event affects.

The major difficulty that this method creates is the different timestamps that detection delays create, making it important to be aware of the detection delays associated with the origin of a set of annotations.

When combined with the dataset method from the previous chapter, we now have the capability to capture a new dataset and retain all of the information acquired in a application agnostic format. We also have the tools and methods needed to verify that the created annotations are correct.

The DARPA dataset show how difficult it is to create correct network annotations. Because of the difficulty there is a need to have a reliable and precise way to record annotations that can be refined and improved over time.

4.10 Summary

This chapter has explored the existing network annotation format and then outlined a new format. The chapter outlines a new method for verifying that the created annotations are correct, in the sense that there are no ambiguities in how the relate to the network trace.

The chapter then uses the DARPA datasets and the truth data as a study to prove the effectiveness of the new format, and shows that using a more accurate format and the method to verify the correctness of annotations decrease the likelihood of errors in the created annotations.

4.10.1 Contributions

In this chapter there are two contributions. The first is a common method to verify and in some cases increase the precision of annotations for network anomaly detection. The second contribution is an outline of the errors found in the DARPA 1998 and DARPA 1999 truth data. The corrected annotations will be made available at WITS (Waikato Internet Traffic Storage, n. d.) as a part of this thesis.

5

A New Data Fusion Based Methodology

Chapter 2 established the fact that the currently used methodologies for network anomaly detection allow for reproducibility, openness, and sharing of results. This chapter brings together Chapters 3 and 4 and uses the contributions from those two chapters to propose a new evaluation methodology that uses a novel approach with regards to datasets and truth data.

5.1 Motivation

The current evaluation methods, described in Chapter 2 for anomaly detection methods suffer from several drawbacks, no matter the method. Different methods are outlined in section , but they all suffer from these flaws:

- Lack of openness.
- Lack or repeatability.
- Improper treatment of datasets.

The methodology proposed in this chapter remedies these shortcomings. The following section describes existing evaluation methods that have been put forward in publications. No trace of any widespread adaptation of any these methodologies have been found, instead the most prevalent methodology is the informal one.

All of the related work presented in Section 2.4 fail to address the issues surrounding trace datasets and common formats for network anomaly detection. The work presented by Dumitras and Shou (2011) shows promise, but it does not address any of the issues about labelling of actual network data since the work focuses on malware samples and aggregated network information than raw network traces.

Instead, the work presented in this thesis, and in particular in this and the next two chapters, address these issues surrounding datasets in a pragmatic manner. It is recognised that it is not an ideal solution, but it is however an improvement over the current state of affairs in the field of network anomaly detection.

5.1.1 Expressing Results

The results achieved by a specific method are commonly only presented in the paper where the method is presented, on specific datasets. By having a common format, such as the one introduced in Chapter 4, it would express the actual output of the method. The output can then easily be expressed in the appropriate formats for the method and compared to other methods than the ones in the original paper.

5.2 The Problem With Existing Methodologies

Chapter 2 describes the existing network anomaly detection evaluation methods.

These existing methodologies only place an emphasis on the experiments themselves, and the results of those experiments. There is no focus on selecting or sharing the datasets, the openness of said datasets, the repeatability of the experiments and how to share the raw results instead of a metric achieved by the method.

As such, it is possible to improve on the current state of evaluating network anomaly detection methods by introducing a new methodology that takes all of these factors into account. This is done by defining a methodology that takes a novel and pragmatic approach with relation to sharing datasets and the results, and incorporating previously shared results into the evaluation.

5.3 The new Methodology

5.3.1 Approach

In an ideal world, there would be open implementations of all anomaly detection methods. However, the reality is that few researchers are willing to share the implementations freely and are thus withholding the implementations. By making the datasets and the results available it will be possible to see whether alternative implementations have the same performance as the original researcher's implementations.

By assuming that datasets and results will be available, a new approach can be created that does not rely on complete truth data. The format described in Chapter 4 allows for the certainty surrounding each specific anomaly to be established and it is also possible to establish an overall trust for the anomaly detection method.

This allows for the use of a Data Fusion approach to establish the data that a new method will be evaluated against. By carefully choosing different approaches of anomaly detection methods, it is possible to establish a broader spectrum of detected anomalies than any single method could.

Thus, by carefully selecting the methods and combining them using data fusion it is possible to correctly identify a sizeable portion of the anomalies. The performance is evaluated in Chapter 6. This consensus can then be treated as the results that a new method is evaluated against. Each new method that is used can then later be incorporated into the consensus, incrementally improving the quality of the comparison data. While this has the drawback that the comparison data changes over time, the updated consensus can be used when comparing the original, unchanged, output in the future.

However, at the time the evaluation is made, the most up-to-date information is used, and this will be a significant advantage over the current state of the art.

5.3.2 Steps

By adding these things to the existing methodologies, the new steps are as follows. Step 2 corresponds to evaluating the method or approach under the existing methodologies.

1. Choose datasets for the evaluation.
2. Evaluate against datasets.
3. Verify results against the dataset.
4. Use fused previous results as truth data.
5. Establish new method's performance.

Step 4 is explained in greater detail in Section 5.3.3. This is the most important part of the additions, since only by having a well established and agreed upon set labelled anomalies is it possible to evaluate against existing methods.

Choosing Datasets

For evaluation purposes, a method should be evaluated against a known dataset that is publicly available. As a part of the proposed methodology, this would be a dataset that has existing partial data associated with it.

Evaluating Against Datasets

The informal methodology, described in Chapter 2, could be used for the actual evaluation. However, as the truth is not completely known it is possible that the performance of the method could be deemed different in the future when the dataset has been more thoroughly mapped. This would however only affect the false positive rate of the method.

Verification Against the Dataset

As a part of making the results more reliable, they should be verified to ensure that they correspond to the network trace. This will ensure that programming errors or human errors will not lead to any ambiguity in the output of a method. As described in Chapter 4, the results should be expressed in a common output format that in turn is closely connected to the network trace.

Chapter 4 establishes an annotation format for network formats that is both extensible and generic. This format forms a cornerstone in the methodology because it allow different anomaly detection methods to express their output in the same format. This in turn allow for simple comparison of the methods.

The format also introduced the concept of semantic URL's as descriptors of network events. This is very important to allow for automatic comparison and data fusion of output results.

Use Fused Truth Data

By using fused truth data from algorithms that detect the same kind of anomalies it is possible to use datasets that would not have any existing truth data. The fused truth data is established by one of the methods described in Section 5.3.3, and Chapter 6 explores the impact of the different methods using the DARPA 1999 dataset.

Establish Performance

The performance of a method is established against the fuse base data from other methods. The methods that are used for the fusion process should be chosen carefully to provide as much coverage as possible. The methods used in the fusion must be the same as the methods that any competing methods are evaluated against.

5.3.3 Creating Fused Truth

Data fusion is a method of information processing that combines the results of several sensors to achieve a greater clarity of what the sensors are observing. A simple analogy is how the human body can observe a bird with its eyes while listening to the bird with its ears in an attempt to make a more accurate assessment of what species of bird it is.

Data fusion can be applied to many different problems, and has shown to increase the accuracy of anomaly detection methods (Ciza, 2009). The types of methods outlined below have been established in previous publications performing data fusion in network anomaly detection.

By fusing the results of several different anomaly detection to establish a labelling of a data set that has a significantly higher certainty than any of the methods would achieve on their own. In the next chapter such an approach is compared to the truth data established for the DARPA dataset.

Expressing Uncertainty

To be able to perform data fusion with the results from an anomaly detection method, the detector need to express its certainty in the output. There are many different ways to express certainty, or uncertainty, with one of the more common ones being probabilities. Halpern (2003) establish the following measures:

- Probability measures
- Lower and upper probabilities
- Dempster-Shafer belief functions
- Possibility measures
- Ranking functions
- Relative likelihood
- Plausibility measures

Probability measures and Dempster-Shafer belief functions are the most relevant ones for the purpose of this thesis and shall thus be elaborated on. Informal definitions of these two are as follows:

A probability measure is the likelihood that something will happen. This is the measure introduced in elementary statistics with the most common example being which side will land up in a coin flip or what number will face upwards in a dice roll.

A Dempster-Shafer belief function consists of two parts. The first part is a numerical value expressing how likely an event is to occur, similar to probabilities. The second part express how plausible the first part is considered to be. This allows for the expression of both the likelihood and the belief of how likely the likelihood is to be correct.

Formal definitions of these uncertainty measures can be found in Chapter 2 of Halpern (2003)'s book.

Types of Data Fusion

(Hall and McMullen, 2004) published contains a thorough description of different data fusion methods, and what they are best suited for. The book describes the Joint Directors of Laboratories (JDL) data fusion model and the different types of data fusion that can be used. The JDL model was created for the Department of Defence (DoD) in the United States, and is hence geared towards very specific types of situations.

Nakamura et al. (2007) published a survey paper, which in addition to describing different fusion models, also describes alternative models for data fusion.

The data fusion methods described below have been used in previous works for network anomaly detection. All these methods fall into Level 1 (Identity Fusion) of the JDL model.

Averaging Barford et al. Barford et al. (2004) introduces the concept of averaging of multiple Network Intrusion Detection System (NIDS)es as an attempt to fuse the results together and increase their detection accuracy. In the paper they describe a scheme where they average the input from different anomaly detection sources.

By averaging the inputs, they are able to increase the detection accuracy compared to what the detectors would achieve on their own.

Bayesian Data Fusion Bayesian data fusion Koks and Challa (2005) is based on Bayes theorem. Thus, the process is based on a probability between 0 and 100% that an event is an anomaly. As discussed in section 5.3.3 there are different ways of representing a belief in an event and the Bayesian approach uses a probabilistic model.

Dempster-Shafer Fusion Dempster-Shafer fusion is similar to Bayesian fusion, but in the fundamental approach it leaves room for uncertainty surrounding an event. For a data fusion approach, this allows us to assign a level of plausibility to our anomaly detectors, or expressing how much faith we have in their decisions.

Unlike in the Bayesian fusion process, the Dempster-Shafer process also allows for the belief assigned to an hypothesis to be non-exclusive. The belief can then be fused together via a set of rules and intervals on the belief established. Since the Dempster-Shafer fusion allows for uncertainty in the input data it is possible to take account for unknown sources in the input.

Fusing for Partial Truth

By fusing the data from several sources it is possible to create partial truth data. Chapter 6 contains an evaluation of the different techniques described in this chapter where they are applied to both the DARPA 1999 and the new Waikato 8 datasets. These methods are chosen because they have been applied previously in literature with promising results for anomaly detection. The addendum described in this chapter is not aimed at detecting anomalies, it rather adopts existing data fusion research to establish a partial truth data for a dataset and the belief surrounding the accuracy of this partial truth.

The main consideration to be creating a partial truth data by running multiple anomaly detection methods against a dataset is the choice of the methods. The initial establishment of the partial truth should be done by a variety of different methods that are from different approaches of anomaly detection. There should also be an overlap in the coverage of the different methods where possible to allow for the refinement of the truth data. Expert labelling should also be treated as a source, not as the truth. Chapter 6 show how this is done for the new Waikato 8 dataset.

Once an initial set of partial truth data has been established, any future evaluations of a new anomaly detection methods against that dataset can be improved in the corpora of results that are fused together.

5.4 Implementation

The methodology described in Section 5.3 requires a number of tools to be feasible. This section initially outlines the criteria that were established on the tools in order to increase the ease of adopting the new methodology. It then describes the different tools that were created to support the methodology.

5.4.1 Criteria on the Implementation for the new Methodology

A new methodology that address the shortcomings of existing methodologies should fulfil the following criteria in addition to the shortcomings.

Simplicity

Simplicity will help with adaptation of the new methodology. If the methodology adds a significant amount of overhead to the researcher it is less likely it will be used since can be considered to be too much additional work.

Openness

All of the tools, formats, and datasets must be open, and publicly available. This ensures that researchers can adopt the tools to their platforms and that the tools can be extended and improved upon by all researchers.

Repeatability

Repeatability is achieved by being open about results and implementations, but also by using datasets that are available, and will continue to be available in the future.

5.4.2 Tools

The different tools are made to support the different stages of the methodology where there are no sufficient tool support at the moment. The design of the tools is such that they are as simple as possible in their approach, only performing one specific step at a time where possible. The different tools are then to be invoked from various shell scripts to chain them together in such a way that the different steps in the methodology are performed.

The tools are discussed below and are made available as open source software and are licensed under the GNU General Public Licence, Version 3.

Data Flow

The tools are made for three different use cases:

1. When there is existing data in a different format that needs to be converted.

2. When multiple data sources need to be fused.
3. When a new method need to be evaluated.

Figure 5.1 demonstrates the data flow for use case 1. There exists existing data that needs to be converted into the annotation format. A corresponding matcher is either created or reused, which is creates annotations from the truth data if it can locate the corresponding data in the network trace.

The data may then be merged with a different annotation file, and if it is, it should be sorted using the annotationsorter.

Figure 5.2 shows the data flow for fusing several sets of annotations, corresponding to use case 2. The different annotations files are fed into the fuser, which fuses the data and then creates an output file.

Figure 5.3 contains the data flow when a method is evaluated against truth data. The method output and the fused truth are given to the comparer,

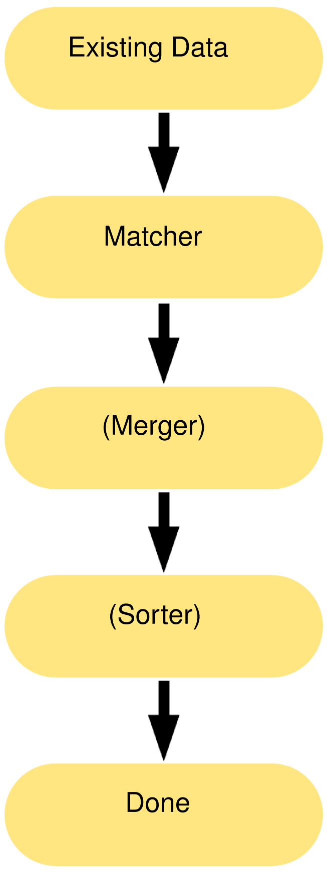


Figure 5.1: The data flow for when existing data exists and need to be converted into the annotation format using a corresponding matcher.

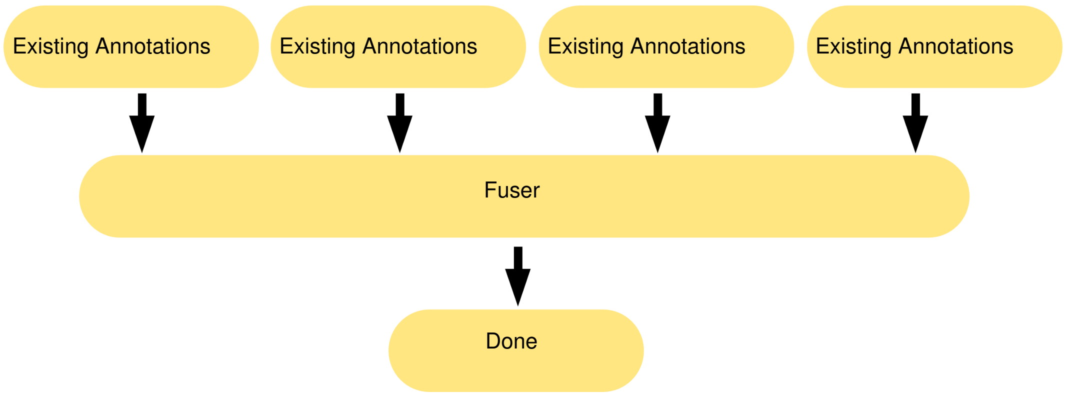


Figure 5.2: The data flow for when several sets of annotations need to be fused together.

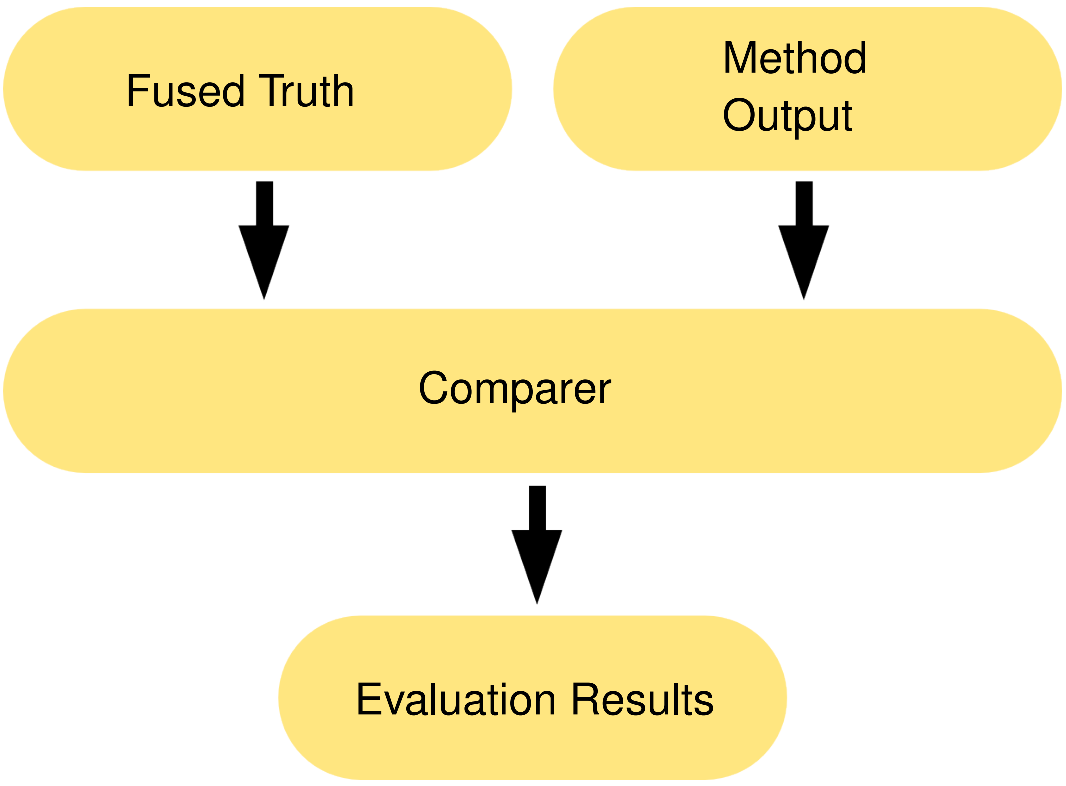


Figure 5.3: Data flow for performing an evaluation of existing annotations against truth data

which evaluates the method's output against the fused truth. The evaluation results are presented as a set of true positives and false positives.

Data Format

The data format is described in detail in Chapter 4. The goal of the data format is to support different levels of annotations while being extensible enough that it could encapsulate or add a network trace format.

To work with the data format, a number of additional tools were developed.

annotationprinter is a tool that converts the binary annotations into a terse text based printout for human reading.

annotationsorter processes an annotation file and makes sure that all entries are in chronological order.

annotationmerger merges several annotation files into one output file.

Matchers

A suite of of programs were created to match an entry in a different file format with the corresponding packet or flow and create a proper annotation record. The workings of the matchers are described in detail in Chapter 4, but they are presented with a brief description of their function.

bromatcher reads Bro's notice.log and matches it with the corresponding network trace. It creates entries at the lowest level possible, but often only a host is identified or a tracewide event declared.

DARPA1998matcher Reads the truth data as shipped with the dataset and converts it to the annotation format, improving the accuracy of the matches in the process.

DARPA1999matcher works similarly to the DARPA1998matcher, but the lack of details in some cases in the truth data supplied with the dataset requires a slightly different approach in the matching logic.

Snortmatcher reads Snort's alert.log (in the unified2 format) and converts it to flow records in the annotation format.

Comparator

The comparator is used to compare one annotation file with another. One of the compared files is assumed to be truth data and it compares the number of identical records in the two files. For flows and packets, this is a straightforward comparison.

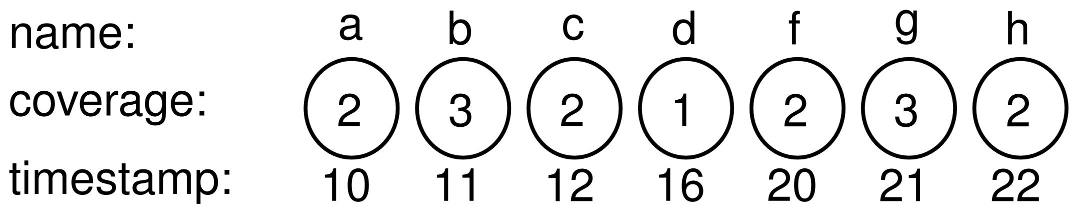
For higher levels of events than flows and packets, there is a need to take detection delay into consideration as well. The detection delay is used as the maximum absolute value the difference between the truth data and the tested entry may be. In other words $|t_{truth} - t_{other}| \leq delay$.

The output of the comparator is given as a truth matrix containing the number of true positives, false positives, true negatives and false negatives.

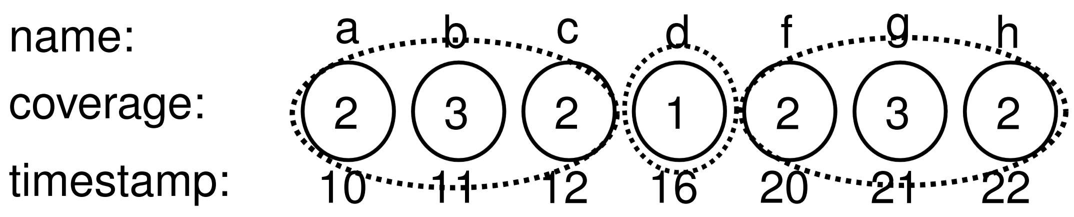
Fuser

The fuser is an utility that reads annotations from multiple files, clusters them and then applies data fusion to the events inside the clusters.

The clustering is done on a tempo-spatial basis, only items with the same spatial characteristics are placed in the same cluster. On the temporal side, there is room for a detection delay. Figure 5.4 shows a number of possible events that might be fitted in a cluster. Each node has the number of nodes that would be inside the cluster if that specific node was the centre node



(a) The different parameters used to create the clusters



(b) The created clusters

Figure 5.4: An example of the clusters obtained by using the clustering algorithm. Three clusters are created from the data shown in figure a, with each cluster being the largest possible cluster given a fuzziness factor of 2

of the cluster. A greedy approach is then used to maximise the sizes of all clusters. From the possible events in Figure 5.4, the outcome would be three clusters, centred on nodes b, d, g when the detection delay defined above is set to a distance 2.

Finally, all nodes in the cluster are fused together into one output event.

5.5 Summary

This chapter motivated and introduced a new methodology for evaluating network anomaly detection methods. The methodology relies on data fusion to create partial truth data for captured datasets, and used false positives and false negatives as the primary metrics.

The tools necessary to use the methodology have been created as a part of this thesis, and are described in this chapter. The tools are available from http://hdl.handle.net/10289/8041.

5.5.1 Contributions

This chapter describes a new evaluation methodology that combines the current informal method that is widely adopted in the field with an approach that increases openness and repeatability. The same approaches can however be applied to any of the methodologies described in Section 2.4 and will bestow the same benefits on those methodologies.

5.5.2 Discussion

The methodology in this chapter does not rely on any specific dataset that can become outdated. Instead, it relies on the fact that it is easy to capture datasets with the method described in Chapter 3 and the strengths in data fusion to improve upon detection accuracy. Chapter 6 contains a case study where the potential coverage of the truth data is established with a minimum of methods.

Because the methodology also relies heavily on the partial truth data established by existing trusted anomaly detection methods, it is impossible to describe accurately how large a portion of the anomalies are accurately described in the truth data, which may affect detection results.

5.5.3 Difficulties

Methods suffering from different detection delays cause difficulties when they report anomalies that are not tied to a specific packet or flow. The differences in detection delays lead to the situation when the same anomaly is reported at different times by different methods.

To remedy this, each method need to have a deterministic delay, and this delay need to be taken into account when fusing against output from that method.

5.5.4 Strengths and Weaknesses

The new methodology makes it easy for researchers to have an evolving dataset and continuously improve upon it. The methodology incorporates sharing of results as a cornerstone since the same results also make the foundation of the truth data for a dataset. Unlike previous methodologies, the tools will also be provided as a part of this thesis to facilitate adaptation by other researchers, available at http://hdl.handle.net/10289/8041.

The new methodology amends all of the problems outlined in Section 5.1. It cannot do so without making a trade since it is impossible to estimate the amount of false negatives in the dataset without having access to the ground truth.

6

A Case Study on The Partial Truth Coverage

This chapter uses the DARPA 1999 dataset to examine the individual performance of several network anomaly detection methods and the fused output of all of these methods. The purpose of this chapter is to establish the minimum coverage of partial truth data one can expect from a case where there is a minimum of network anomaly detection methods used. The chapter demonstrates that the fused performance is better than that of any individual method and shows that Dempster-Shafer fusion is a viable method to establish partial truth.

The chapter also performs an evaluation on the Waikato 8 dataset using the time series based method described in this chapter with traffic volume and T-Entropy as their inputs.

6.1 Description of Methods Used in This Chapter

The methods described in this section have been used for both datasets. The methods have been chosen for simplicity over performance and are representative for the different categories of anomaly detection.

6.1.1 Time Series Based

By decomposing the network stream into time series data, both in packets and throughput we can use forecasting methods to make predictions based on the data. By comparing the forecast value with the actual value it is possible to detect abrupt changes in the time series data. There is a fine line that needs to be maintained in the fit between the forecasted values and the actual data. If the fit of the forecast data is too good, no anomalies will be detected since the forecasting method will react too quickly for the anomalies to be detected.

Normally network traffic amounts and packet counts are described as time series data.

6.1.2 Forecasting Methods

Adaptive Single Exponential Smoothing Exponential smoothing algorithms are relatively simple methods that can be used to either smooth (i.e. eliminate burstiness in data), or forecast data. This is demonstrated in Figure 6.1. The figure shows the number of megabytes transmitted over the University of Waikato's ingress and egress points, aggregated over five minute intervals. The dotted line shows the actual measured traffic and the dashed line

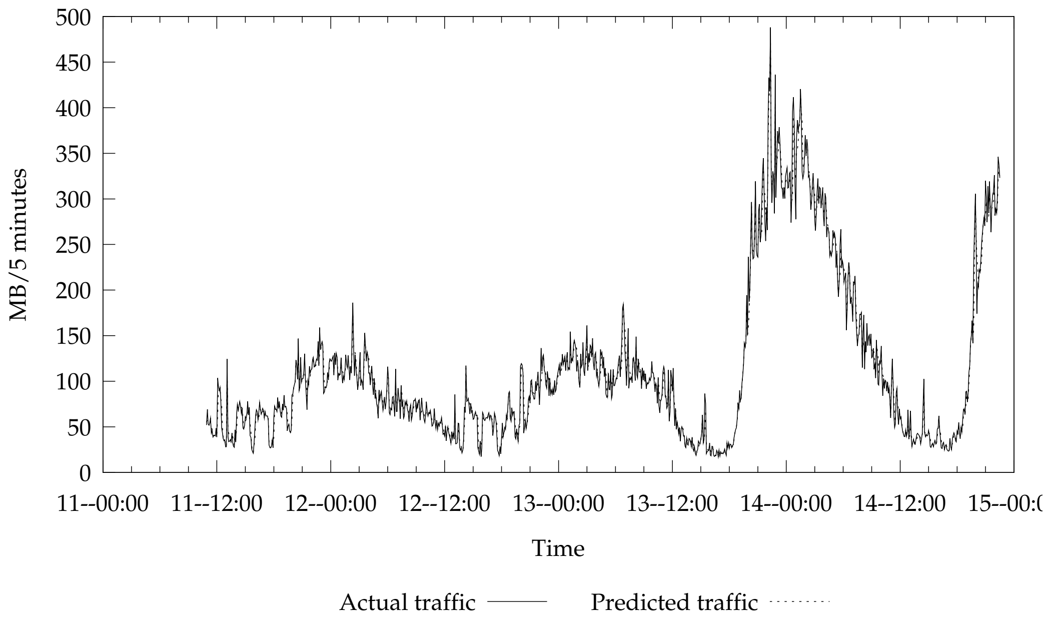


Figure 6.1: An example of Adaptive Single Exponential Smoothing

shows the smoothed traffic. It is important to note that the smoothed traffic does not have the same extreme peaks and drops, but still follows the same pattern.

The basic function for a single exponential smoothing function is

$$F(t) = \alpha X_t + (1 - \alpha)F_{t-1} \tag{6.1}$$

where α is the a weighting constant used to determine how much weight the actual value (X_t) should be given.

The weighting can be automatically adjusted to make the method adaptive , which allows the method to adjust for rapid changes in the data, giving a better fit than what would otherwise be achieved.

Single exponential smoothing does not deal well with seasonal trends or patterns diurnal variations, which is why the method has been made adaptive. This was added done by adding a second parameter, β that adjusts the value of α.

$$\alpha_{t+1} = |(E_t/M_t)| \tag{6.2}$$

where E_t is defined as

$$E_t = \beta * e_t + (1 - \beta)E_{t-1} \tag{6.3}$$

and M_t is defined as

$$M_t = \beta * |e_t| + (1 - beta)M_{t-1} \tag{6.4}$$

and finally e_t is defined as

$$e_t = X_t - F_t \tag{6.5}$$

Combining Equations 6.1 and 6.5 result in an adaptive response rate single exponential smoothing.

Holts-Winters is a triple exponential smoothing forecasting method. Brutlag (2000) introduced the method for network anomaly detection. The method works similar to the single exponential smoothing method de-

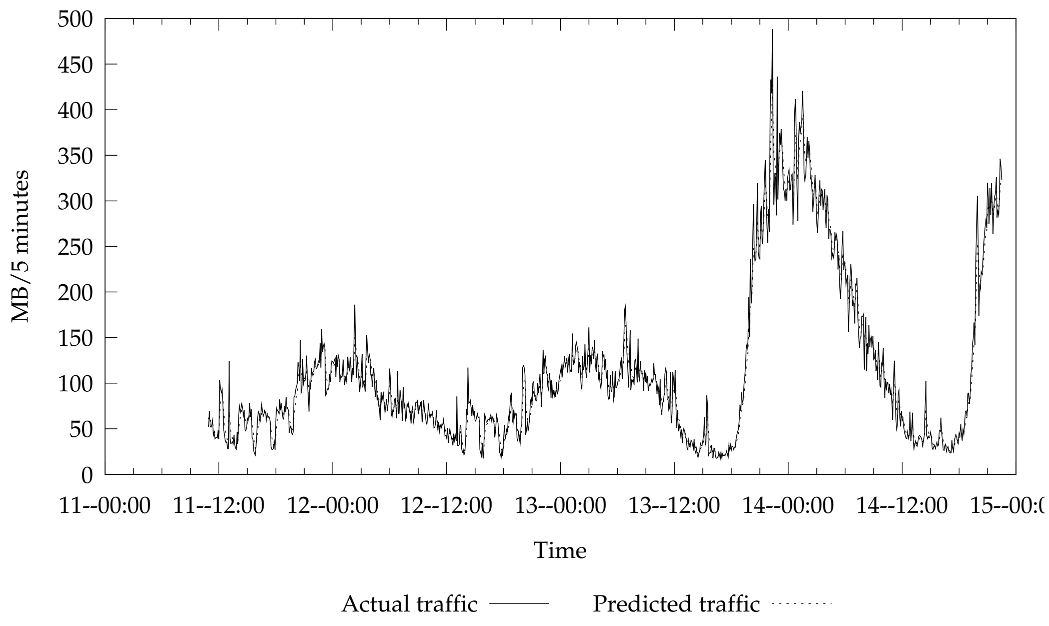


Figure 6.2: An example of Holts-Winters forecasting

scribed above, with the difference that it also takes seasonal variations into effects and is able to cope with long term trends without being an adaptive method. This makes the method better suited to detect repeated patterns when they are first occurring, but later adapt to them. An absence of a repeated pattern can also be detected since the forecast will contain the repeated pattern inside its predictions.

Figure 6.2 demonstrates this on the bandwidth at Waikato University's ingress and egress points. The traffic has been aggregated over five minute intervals. The dotted line shows the actual measured traffic and the dashed line shows the smoothed traffic. It is important to note that the smoothed traffic does not have the same extreme peaks and drops, but still follows the same pattern.

6.1.3 Detectors

Both detectors described in this section are used with the three forecasting techniques described in Section 6.1.2.

Standard Deviation A simplistic detector, introduced by Brutlag (2000) compares the forecasted value with the actual value at time t. A sliding window of the last n samples is used, where n corresponds to the number of samples in the smallest seasonal variation observed in the measured data. In network traffic this is typically a 24 hour period, since usage patterns vary over the day.

An alert is raised if the forecasted value is not within 2 σ, computed from the last n samples, of the actual value. This threshold is the same as presented by Brutlag (2000).

Modified Plateau Logg et al. (2004) created the modified plateau algorithm as an anomaly detection technique for network traffic presented as time series data. The method is based around a trigger buffer that is filled every time the forecast differs from the standard deviation by a value of 2. An alert is raised when the trigger buffer is full.

The approach of using a trigger buffer causes a detection delay that is proportional to the sample interval $\times$ buffer size. By having the buffer, no alerts are raised from spikes in the actual data, since they will not be lasting long enough for the trigger buffer to fill.

6.1.4 Deep Packet Inspection

Deep packet inspection systems examine the full contents of a network packet, both headers and payload.

Snort

(Roesch, 1999) is a deep packet inspection system which uses signatures to perform various forms of regular expression matching on packet contents. Snort also has various preprocessors that allows it to analyse certain protocols such as Simple Mail Transfer Protocol (SMTP), and detect anomalies in protocol behaviours as well as other preprocessors to detect some anomalies that cannot be detected by signatures (such as port scans).

Bro

Bro (Paxson, 1998) is a different deep packet inspection system from Snort. It comes with a Turing complete configuration language which can be used to implement different anomaly detection methods. By default the system comes with rules to detect stepping stone attacks and port scans. The system can also load Snort rules, but this feature was never used in the experiments since Snort was being used at the same time.

By virtue of the configuration language, Bro is more flexible than snort, and by default detects different types of events. Bro however have an as large library of pre-defined signatures or detectors, hence the greater flexibility is not taken advantage of.

6.1.5 T-Entropy

T-entropy (Eimann, 2008) is a method that uses the T-information measure to calculate the amount of entropy in all packet headers over time. The method is designed to be run over a full packet trace and produces a measure as an output. The output data can then be put through the time-series methods to detect network anomalies.

6.1.6 Phad

Mahoney and Chan (2001) created the Packet Header Anomaly Detection (PHAD) tool. Phad examines 33 fields in the packet header and assigns a probability for each packet that it considers to be anomalous. Mahoney trained the method against week3 of the DARPA 1999 dataset and used the two weeks of testing data.

6.1.7 Alad

Mahoney and Chan (2002) presents an Intrusion Detection System (IDS) that incorporates Phad and a new method called Alad. While Phad is responsible for monitoring all packets in the link, Alad is a application layer detector. Alad examines whole connection flows for anomalies, and reports the flows it finds anomalous.

6.1.8 Comparison of Methods

This section described the different methods mentioned earlier, what types of events they detect and what their inputs respective outputs are.

Table 6.1 clearly demonstrates that the different methods used capture different aspects of the potential anomalies that can develop in a network. The methods have not been selected because the represent the state of the art, rather they have been selected because they are relatively simple and have laid the foundations for more advanced methods. Methods that require

Method	Type	Inputs	Output	Level	Types of Anomalies
Holts Winters	Time series	Time series data from trace	annotations	Tracewide	Deviations from forecast
Adaptive single exponential smoothing	Time series	Time series data from trace	annotations	Tracewide	Deviations from forecast
T-Entropy	Entropy	Network Trace	time series data	Tracewide	Changes in entropy - packet floods
Snort	Deep packet inspection	Network Trace	annotations	Flow	Non-novel malware, attacks
Bro	Deep packet inspection	Network Trace	annotations	Flow	Network probes

Table 6.1: Breakdown of the Anomaly Detection Methods Used in the Case Study

training have been disregarded in favour of on-line methods.

6.2 Evaluation Setup

6.2.1 Dataset

A part of the DARPA 1999 dataset has been used for this comparison. The main reason for this is because we have truth data for the dataset, and can establish the individual coverage of the different network anomaly detection methods.

The part used is the test phase of the dataset, which consists of two weeks of full payload packet captures. The dataset is discussed in more detail in Chapter 3.

Total Flows	Anomalous Flows	Background Flows
2537762	91722	2446040

Table 6.2: Number of flows in the testing phase of DARPA 1999 before changing some anomaly types to trace-wide

The truth data used is the corrected truth data derived from the verification process described in Chapter 4.

Table 6.2 show the number of network flows present in the testing phase of the DARPA 1999 dataset. These numbers were extracted from the network traces and the modified truth data obtained from the process shown in Chapter 4. These numbers have not been modified to allow for the tracewide events described below as the network flows are still in the network trace used in the evaluation.

The following anomalies in the DARPA 1999 datasets are considered to be tracewide and only present once, instead of being reported once for every flow taking parts in the anomalies. The reason the anomalies have been changed from a per-flow basis to a tracewide event is because many of the existing network anomaly detection methods are unable to report the specific flows participating in an attack, but instead only reports that the attack has occurred.

The following anomalies have been converted to tracewide events:

- ipsweep

- nmap
- portsweep
- smurf
- mscan
- satan
- neptune

By converting attacks of these types into tracewide events these flows are no longer tagged as anomalous. Table 6.3 show the number of the flows and the distribution between anomalous and non-anomalous with these events considered tracewide.

Total Flows	Anomalous Flows	Background Flows
2537762	1702	2536060

Table 6.3: Number of flows in the testing phase of DARPA 1999 after changing some anomaly-types to trace-wide

Figure 6.3 shows the different steps that the data goes through. All steps use the network trace data as their input. The time series based methods

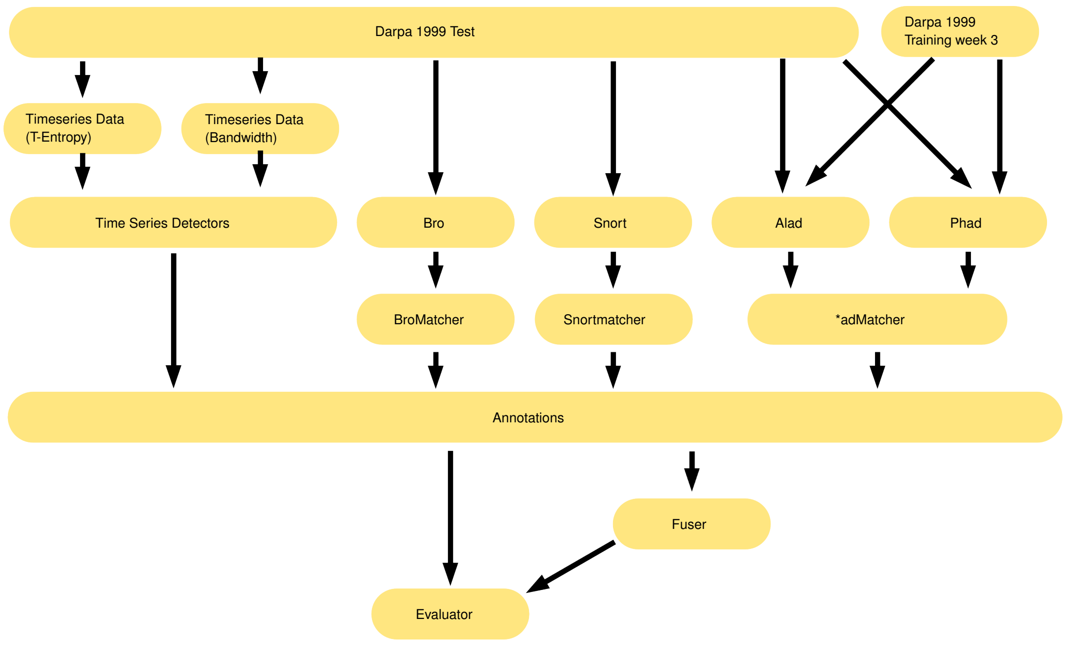


Figure 6.3: The different steps in performing the annotations. The parameters for each method is showed in their respective section.

have bandwidth and t-entropy data extracted from the trace data and the time series data is used for input to these methods. All other methods operate directly on the network trace data. Alad and Phad relies on week 3 of the training data before they are evaluated on the testing data.

The output of Alad, Bro, Phad, and Snort are run through the respective matchers and converted into annotations. The annotations are then fused using the methods described in this chapter and both the unfused and fused annotations are compared against the modified truth data obtained in Chapter 4.

6.2.2 Methods

Section 6.1 introduces the methods. The parameters that the methods have been run with are described in the table below, with the exception for Snort and Bro. Snort and Bro are described in their respective sections.

Alad

Alad requires the trace files to be processed into specific flow records before they can be evaluated. For each run of Alad, all relevant network trace files was processed with that tool.

Alad was run with the data from week 3 of the DARPA 1999 training period as the training data. It was then given one day at a time from the testing period to evaluate.

Bro

Bro version 1.5.1 has been used for this case study.

Bro has been configured to detect scans and sensitive access to machines. This has been done by enabling the **hot, scan, trw** modules that are shipped with Bro 1.5.1.

Phad

Phad was run in a exactly the same manner as Alad, with the main difference that Phad does not require an intermediate step.

Snort

Snort version 2.9.0.4 has been used, with the open VRT ruleset published on 2011-04-14.

Snort has been specifically tuned to the DARPA 1999 dataset. This is to resemble the fact that any snort running on a production network will have been tuned to the traffic that is seen on that network. The snort performance tuning by Tjhai et al. (2008) has been used as a starting point and has then been improved upon. The results for snort shows which signature has triggered on every true positive and false positive.

T-Entropy

The T-entropy command *pcap_map* was run with the parameters *-t1000000 -F5000 -Te -m27* on each trace file in the dataset. The output was saved as a time series data and used as input to the time series methods, which then generated events.

Time Series

There are two sets of aggregated data that the time series methods have been applied to; the traffic volume and the t-entropy measure.

The time series based methods have been fitted to the fitted to the network data and the T-entropy output and the methods with the lowest mean average percentage error have been chosen. It is assumed that any other evaluators or network administrators would make an effort to likewise tune their methods to the network.

The tuning was performed by exploring the parameter space for all possible permutations to find the best fitting set of parameters on the network data.

6.2.3 Establishing Performance

The performance is established using the corrected ground truth data presented in Chapter 4. Performance metrics are the number of true positives, false positives, and false negatives. True negatives are not considered to be an important metric since there is no limitation on the amount of negative data that could result in true negatives for some methods.

True Positive

A True Positive (TP) is an accurate detection of an event in the network trace. Depending on the level of event, as defined in Chapter 4, there are different measures of similarity that are considered. For a flow or packet level event, there has to be a full match of the complete 8-tuple used to identify a trace on that level. For a tracewide event, the timestamps have to match, compensating for detection delay in the method that detects the event.

False Positive

A False Positive (FP) is caused by a method that raises an alert for an event that is not considered to be of interest.

False Negative

A False Negative (FN) is caused by the method under evaluation failing to detect an event which it should raise an alert for.

6.3 Creating Partial Truth Data

There are a number of common steps that is used to create partial truth data, no matter which fusion method is used. These steps are outlined here, with the implementation of each fusion method described in more details in their respective section.

Each method reports a list of events. These events are then clustered around a central event where the cluster will contain up to *methodcount* events. *methodcount* is the number of methods used in the evaluation, in this case 4. In the case where there are events missing from one method, the event is assumed to have reported as a negative from that particular method. The events are then fused with the fuse method used, and the outcome is reported as an anomaly if the resulting score is higher than a threshold.

The results presented in this section have been achieved with a value of 0.4 or higher in the score of the fusion process.

6.3.1 Majority Voting

Majority voting is done by allowing each method one vote each whether each detected anomaly is an anomaly or not. If a simple majority of the methods decides that an input is an anomaly, that anomaly is flagged in the output.

6.3.2 Dempster-Shafer Fusion

The Dempster-Shafer fusion relies on the pyds (Reineking, n. d.) library and the fusion process uses a conjunct combination of all inputs. The resulting fused data is then converted into a pignistic probability containing the anomaly and the not anomaly probabilities. The anomaly probability is assigned as the output score.

There are three different states used in the fusion process, $\{$ *Anomaly*, *NotAnomaly*, *Unknown* $\}$. If a method has triggered an anomaly, we assign it's score to the *Anomaly* state and $1 - score$ to the *NotAnomaly* state. If the method has not flagged an event, all mass is assigned to the *Unknown* state since the method may not be able to detect the type of event.

6.3.3 Averaging

Each method is assigned a base score of 0, where 0 indicates a strong certainty that it is not an anomaly, and 1 that there is a strong belief that there is an anomaly. If the method has assigned a score to the event being considered, that score is used. The average score is then calculated across all methods for the specific flagged event, and the average is assigned to the output.

6.4 Results

This section establishes the performance of the methods described above when run against the dataset. The results presented in this chapter is an aggregate of all days of the two weeks of testing data of the DARPA 1999 dataset. Appendix A contains a breakdown of the results per day for each method.

Table 6.4 contains a summary of all methods used on the DARPA 1999

Method	TPs	FPs	FNs
fused voting	0	7731	90065
alad	113	1510	89984
bro	10	79	90046
fused average	94	7637	89971
snort	840	3736	89225
phad	167	2293	89928
fused ds	963	6768	89103

Table 6.4: Aggregated results for all methods on DARPA 1999 testing, showing the performance of the methods. The performance is measured by the number of true positives and false negatives

Method	TPs	FPs	FNs
fused voting	0	7731	1654
alad	99	1524	1603
bro	24	65	1624
fused average	57	7674	1597
snort	793	3783	880
phad	66	2394	1621
fused ds	867	6864	788

Table 6.5: Aggregated results for all methods on DARPA 1999 testing, showing the performance of the methods. The performance is measured by the number of true positives and false negatives

dataset, as well as the fused output. Each method has the number of false positives and false negatives presented. Table 6.5 show the results on the same dataset but with the trace-wide events converted from flows annotations to trace-wide annotations in the truth data.

Snort itself provides a strong baseline by discovering many of the anomalies, as seen in Table 6.5.

The other methods does not detect nearly as many events, and there is not a significant overlap in the methods that are detected. This clearly demonstrates that the different approaches has different benefits and will not all detect the same anomalies.

The Dempster-Shafer data fusion method is the best performing method, which can be seen by comparing the data fusion methods in Table 6.5. The performance difference between averaging and fusion is relatively small, but it can be noted that Dempster-Shafer based fusion has a somewhat better performance.

The case study shows that it is viable to use existing network anomaly detection methods but that there will always be a certain level of uncertainty surrounding the number of false positives and false negatives. By using data fusion we can achieve a greater coverage of the dataset than any of the methods would have on their own, but until there is a sufficient overlap in detected anomalies this uncertainty will be greater.

A sufficient overlap would ideally be when there are three or more methods detecting the same anomaly.

6.4.1 Discussion

Majority voting is not a suitable method unless there is a significant overlap in the types of methods used to detect the anomalies. It is best used to resolve conflicts where there are methods that have almost the same output.

Averaging performs similarly to Dempster Shafer fusion for lower values of belief. When a threshold is applied the number of positively identified fused processes decrease drastically.

Dempster Shafer fusion performs the best of the three fusion methods used in this chapter. The outcome of the fusion process is however highly dependant on the initial mass values that are assigned to the belief that a specific event is an anomaly.

6.5 Evaluation of the Case Study

Dempster-Shafer fusion provides the best results out of the fused data. It is apparent that not all anomalies will be detected until there is a large set of methods that have an overlap in the detected methods.

It is worth noting that Alad and Phad's performance differs significantly from the original papers. The cause of this discrepancy comes from the different methodologies applied. Mahoney et al. counted an attack as correctly detected if a packet or flow that made up the overall attack was detected. In this evaluation, only the correctly detected flow is considered correctly labelled.

Strengths

Using Snort we can establish a baseline of the dataset except for a few specific attacks. The evaluation demonstrates that adding additional methods increases the number of attacks correctly identified by the fused output.

Weaknesses

The number of methods used in this case study is what can be considered a bare minimum to achieve a performance that is suitable for creating partial truth data.

6.5.1 Uncertainty Surrounding Confidence

In a dataset collected from a real network, such as Waikato 8, it would be very difficult to ascertain whether all attacks have been detected or not. There is also the possibility of a bias when events are incorrectly flagged as a false positive by several methods.

6.5.2 Suitability of Data Fusion Approach

The data fusion approach appear to be a viable method to achieve partially fused data. The fused output has a combined better ratio of true positives than the stand alone methods, as can be seen from Table 6.5. Unfortunately, the false positive ratio is also increasing.

There are however are three main issues affecting the study in this chapter.

Firstly, the data fusion methods are more simplistic than they should be, since the purpose of this case study is to show the coverage that could be expected. Using a more sophisticated method is likely to yield better results, but studying the suitability of the different approaches is outside the scope of this thesis.

Secondly, the DARPA 1999 truth data marks multiple flows that are a part of the same attack. An example of this would be the *ssh* or *netbus* attacks, where snort would only mark the one flow containing the binary being downloaded to the attacked system, but the truth data marks multiple flows. The truth data would mark one flow for the initiating remote session that installs the backdoor, one for the download of the binary and one for the connection being initiated to the backdoor. In the example of Snort, only one of these three flows would be flagged and the other two classified as false negatives.

Finally, the synthetic background data is causing more modern IDSs to suffer from false positives due to the unrealistic nature of the synthetic data.

6.6 Applying Time series and T-Entropy on Waikato 8

This section demonstrates the performance of the time series based methods on Waikato 8 using the T-entropy and traffic volume as the input data.

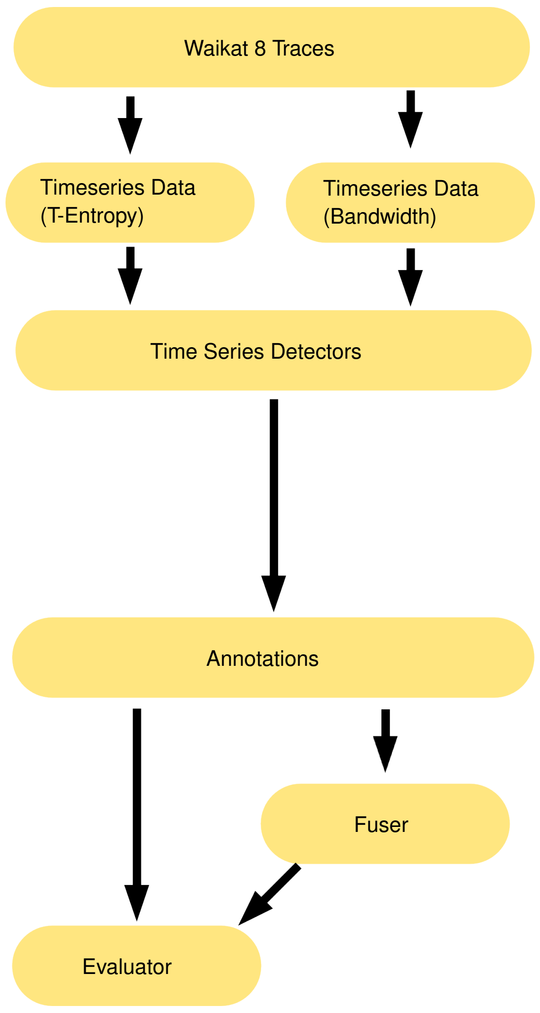


Figure 6.4: The data transformation steps for Waikato 8

6.6.1 Evaluation Setup

Figure 6.4 show how the data have been treated. The original network captures were used as input to the two aggregators, the T-entropy computer and traffic volume calculator. The output of these two aggregators were then used as input to the time series based methods. The forecasts with the best Mean Average Percentage Error (MAPE) was used as the basis for creating the truth data. The traffic volume and T-Entropy data is extracted into time series data with one minute between each data point.

Figure 6.5 show the traffic volume over the network trace. The graph contains at least one significant peak that may be considered a bandwidth anomaly.

Respectively, Figure 6.6 show the t-entropy measure for the dataset.

Choosing Methods for the Partial Truth Data

For both sets of input data, holtswinters with the modified plateau detector is evaluated. The other five variants are used to produce the partial truth

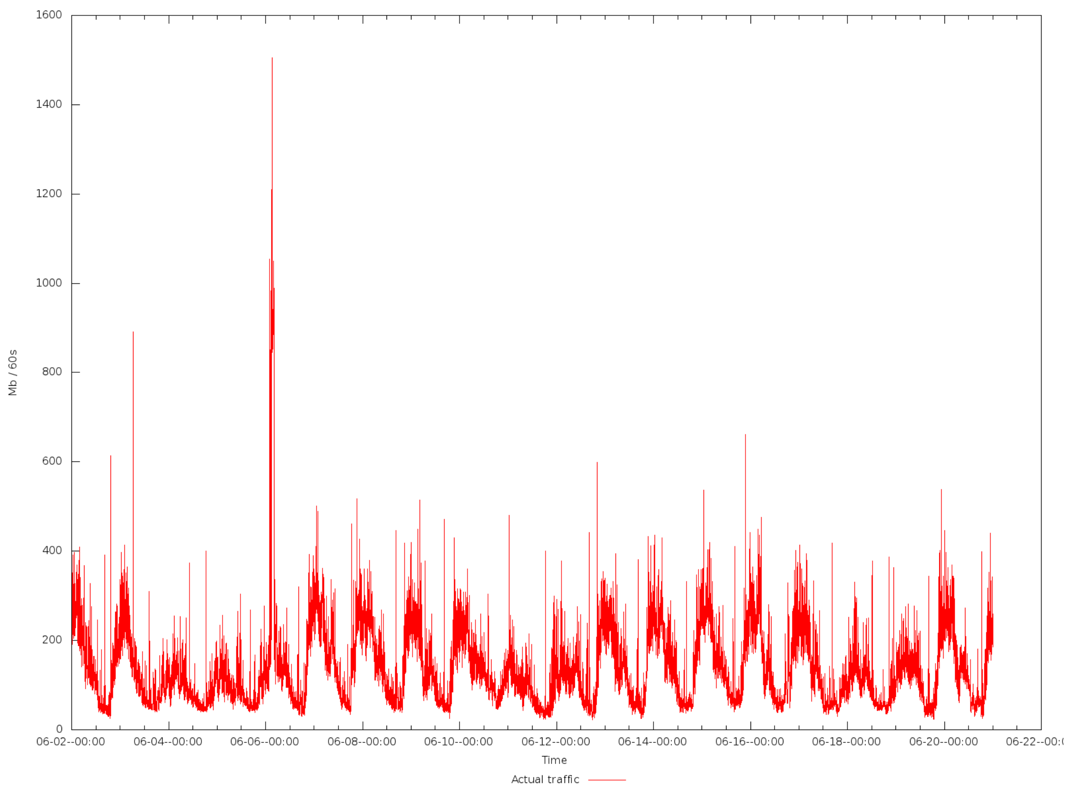


Figure 6.5: Waikato 8 traffic volume

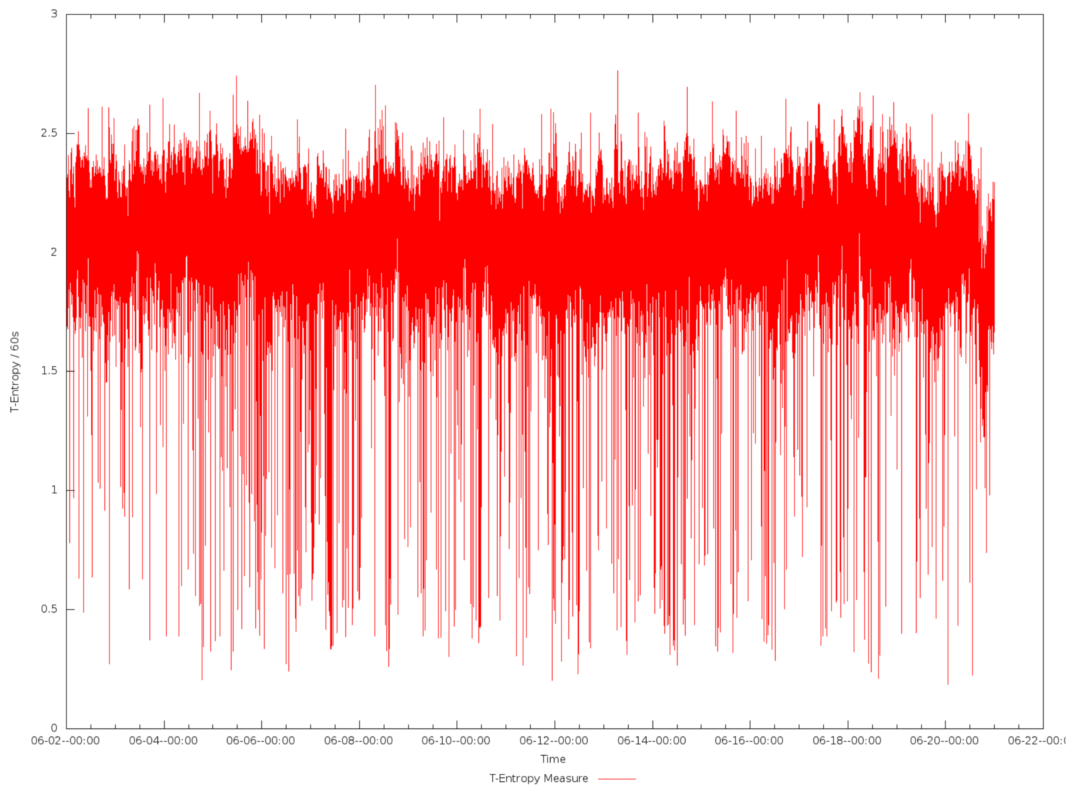


Figure 6.6: Waikato 8 T-Entropy measure

data.

This creates a scenario where a range of methods have already been used to establish partial truth data and another method is introduced to be evaluated against fused output from the previously acquired fusion output.

Choosing Thresholds

One of the more critical aspects is the choosing of the thresholds used for the belief in an annotation. Both the partial truth data and the events generated by a method have a belief associated with them. By choosing a low value there will be many fused or weak events that will be incorporated in the evaluation of the method and might affect the FP rate.

These thresholds will vary between different datasets, and it is assumed that the creators of an particular method are responsible for tuning the method to the dataset used. Suggesting the appropriate approach to tune method's thresholds is beyond the scope of this thesis.

6.6.2 Results

Table 6.6 shows all the anomalies evaluated against all of the fused data from all input methods. The thresholds were set to 0 for both the truth and the methods during the evaluation.

Table 6.7 shows the results where the method evaluating against a fused truth had the belief threshold of 0.6 and the truth threshold was 0.7.

The importance of fair and accurate thresholds is one of the weaknesses of this approach as it clearly affects the results, and should be studied further. However, by carefully choosing an appropriate threshold based on the beliefs expressed by the fused data, many weakly labelled instances will disappear from the evaluation. Table 6.6 shows that only two of the time series methods using T-Entropy as an input contributed any events.

6.7 Summary

This chapter demonstrates that using data fusion to create a fused truth data is a viable alternative to the current state of the art when it comes to evaluating network anomaly detection methods. It is demonstrated that simple data fusion approaches can create better performance than simply combining the results of the different methods.

Method	TPs	FPs	FNs
t-entropy-adaptive-plateu	0	0	2940
t-entropy-adaptive-stdev	0	0	2940
t-entropy-holtswinters-plateu	0	0	2940
t-entropy-holtswinters-stdev	2940	3800	0
t-entropy-singleexp-plateu	0	0	2940
t-entropy-singleexp-stdev	0	1	2940
timeseries-adaptive-plateu	0	0	552
timeseries-adaptive-stdev	0	0	552
timeseries-holtswinters-plateu	422	365	130
timeseries-holtswinters-stdev	61	10	491
timeseries-singleexp-plateu	105	59	447
timeseries-singleexp-stdev	0	0	552

Table 6.6: The results for traffic volume forecasting and T-entropy on Waikato 8 with no thresholds applied

Method	TPs	FPs	FNs
t-entropy-adaptive-plateu	0	0	0
t-entropy-adaptive-stdev	0	0	0
t-entropy-holtswinters-plateu	0	0	0
t-entropy-holtswinters-stdev	0	0	0
t-entropy-singleexp-plateu	0	0	0
t-entropy-singleexp-stdev	0	0	0
timeseries-adaptive-plateu	0	0	10
timeseries-adaptive-stdev	0	0	10
timeseries-holtswinters-plateu	1	13	9
timeseries-holtswinters-stdev	8	7	2
timeseries-singleexp-plateu	2	0	8
timeseries-singleexp-stdev	0	0	10

Table 6.7: The results for traffic volume forecasting and T-entropy on Waikato 8

6.7.1 Discussion

The relatively few methods in the study makes it difficult to draw any clear conclusions from the results. A large portion of the anomalies are detected, but there are also a number of anomalies that are not correctly detected, even accounting for issues with the truth data. When adding more methods the results are likely to improve.

7 Conclusions

This thesis has introduced a novel network anomaly detection methodology that creates partial truth data from existing anomaly detection techniques by using data fusion. More importantly, by following the methodology, the experiments that a researcher does will be testable, which fulfills Popper's criterion of refutability that was introduced in Chapter 1. The results will be open, datasets readily available and methods and parameters well described.

7.1 Thesis Summary

Creating datasets for network anomaly detection is a difficult problem because of the conflicting requirements of completeness and privacy. This has created a situation where the most commonly used datasets are either unrealistic and outdated or lacking truth data and can be difficult to share.

This situation is compounded because open implementations of proposed anomaly detection techniques are rare and further there has been no way to accurately compare the outputs of such techniques.

This thesis addresses these problems individually and combines the solutions to create an overall methodology.

First we have introduced an improved new way to capture live network traces. This synchronously performs network anomaly detection on the captured datastream and creates network annotations containing the results. The resulting datasets are still anonymised and publicly sharable. This technique has been demonstrated with a new dataset.

To be able to create these annotations, a new network trace annotation format was introduced. The trace format is extensible that currently is solely targeted towards annotating network traces, at either at a high level or at specific packets or flows within a trace. The new annotation format increases the accuracy of existing annotations. The advantages of this have been demonstrated by converting the Defense Advanced Research Projects Agency (DARPA) truth data and uncovering and correcting errors.

The new dataset collection technique and the new annotation format are then combined with a new data fusion based evaluation methodology to create an overall solution to the problem of evaluation network anomaly detection techniques. The viability of the new solution is demonstrated by assembling a small corpora of network anomaly detection methods and fuses the results together. Two datasets are used to demonstrate the overall solution and the validity of it.

7.2 Key Contributions

The work presented in this thesis shows a new approach to evaluating network anomaly detection techniques, where data fusion is utilised to create data that new methods can be evaluated against.

7.2.1 A Novel Evaluation Methodology Using a Data-Fusion to Create Partial Truth Data

Chapter 5 contributes a new methodology that is based around creating partial truth data for collected datasets and a common annotation format. The methodology incorporates datasets derived from actual network data, common formats for all output and the ability to share the results without having to share the implementation of the anomaly detection methods. This is the primary contribution of the thesis and is built upon the other contributions.

7.2.2 An Improved Network Capture Process

Chapter 3 introduced how to capture a suitable dataset to use with the methodology, where as much information as possible is preserved throughout the capture process while still protecting the network users privacy. By capturing the additional information before it is erased by the anonymisation process we can create datasets that are more useful for anomaly detection research than the existing datasets. The approach is easy to set up, and cheap, making the creation of new datasets easier than before.

7.2.3 A Flexible Network Annotation Format

The network annotation format described in Chapter 4 relies on the same information as the network trace it annotates. This includes timestamps, addresses, ports, protocols and all other information that can be associated with a packet header. By utilising this information we have created a network annotation format that provides a higher degree of accuracy than any existing formats. By carefully designing the implementation of the format we have made it extensible so that the format can be used to include more information than just the network annotations while still being backwards compatible.

7.2.4 A Method to Verify Network Trace Annotations

Chapter 4 introduced a method to verify the correctness of the annotations created for a network trace. By using this method on the DARPA datasets we were able to demonstrate that there are fallacies in the distributed truth data and correct these fallacies.

7.2.5 Open Implementations of Network Anomaly Detection Methods and Evaluation Tools

All network anomaly detection methods and supporting software used in this thesis are already open implementations or will have implementations made available as a compendium to the thesis. This software is available at http://hdl.handle.net/10289/8041. Updated versions of all bundled software and data may be released at http://research.wand.net.nz.

7.3 Conclusions

Our methodology, introduced in Section 5.3(p. 71), was demonstrated in Chapter 6. In the chapter the methodology was studied on the DARPA 1999 dataset with four methods, each taking a different approach to how the anomalies were detected.

By combining the output of the four methods, the performance and certainty surrounding the fused partial truth was improved upon, but it also highlights the importance of having a large corpus of initial methods that have a large overlap in the detected anomalies.

Looking at the performance of this case study, approximately 47% of the anomalies from the truth data was detected. Unfortunately, out of the 793 True Positive (TP)s there were 3783 False Positive (FP)s. Many of these are an artefact of the dataset, and the way that the DARPA 1999 evaluation was set up with regards to datasets and labels. For Snort (Roesch, 1999) and Bro (Paxson, 1998), the results are expected to have fewer FPs in a real network capture where there is no synthetic background data that will give false positives.

When applying the methodology to the Waikato 8 dataset, it performed as expected. The data fusion process combined the belief of the methods that were used as input and when a threshold was applied to the fused output, only the events with the strongest overall belief remained.

However, the main difficulty and weakness with the proposed methodology is the reliance of every instance having belief associated with it. Choosing a correct belief is difficult and greatly affects the outcome of both the data fusion process and the evaluation.

7.4 Discussion

There appear to be a general shift towards better quality datasets and evaluations in the field of network anomaly detection at the moment. The work of Fontugne with MawiLAB (Fontugne et al., 2010) is highly related and shows that there is awareness of the problem discussed in this thesis. Unlike MawiLAB, which is simply aimed at labelling traces post-capture, the work in this thesis encompass a more wide-fetching approach. In addition to perform post-capture labelling we also put forward a new method to collect new datasets, a new annotation format better suited to this form of data capture and an evaluation methodology that uses the labelled data to iteratively explore a collected dataset.

The main obstacles for widespread adaptation of the work proposed in this thesis is the lack of an open corpus of network anomaly detection methods or the lack of multiple results for one dataset. If this was available, the methodology and tools made available in relation to this thesis could easily be adopted by other researchers.

The Waikato 8 dataset used in this thesis has a relatively low throughput, especially compared to core networks. However, there is no technical reasons as to why the methods described in this thesis will not scale well. The slowest component of the current implementation is the clustering of events for the data fusion process, which can be improved upon by using a more efficient clustering algorithm. All other components should scale linearly with the number of events present in the dataset.

7.4.1 Future Work

This thesis lays the foundation for a pragmatic approach to evaluating network anomaly detection methods. There is however future work that would extend on the usefulness of the work outlined in this thesis and make it more relevant to the anomaly detection community.

One of the main limitations of the approach is the sensitivity to the thresholds and how to assign the correct belief in an event. Further study is needed to be able to improve upon the reliability.

Adding more anomaly detection methods and evaluating them would be a significant contribution to the field. At the moment there is little common ground for establishing the performance of an anomaly detection method. While this thesis lays the ground for future work in this area, it is beyond the scope to extend on this work.

Extending the annotation format to be both a network trace format and a network annotation format would enable researchers to embed a core set of annotations with a dataset. By doing this it would be easier to distribute specific versions of a dataset and facilitate better evaluations.

Adding support for the annotation format to exising anomaly detection systems, such as Bro and Snort would allow for easier adaptation of the work outlined in this thesis.

References

Alcock, S., Lorier, P., Nelson, R. (2012). Libtrace: a packet capture and analysis library. *SIGCOMM Computer Communications Review*, *42*(2), 42–48. doi:10.1145/2185376.2185382.
http://doi.acm.org/10.1145/2185376.2185382

Athanasiades, N., Abler, R., Levine, J., Owen, H., Riley, G. (2003). Intrusion detection testing and benchmarking methodologies. In *Proceedings of the First IEEE International Workshop on Information Assurance (IWIA'03)*, IEEE-IWIA '03, pp. 63–73. Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-1886-9.
http://dl.acm.org/citation.cfm?id=857183.857559

Barford, P., Jha, S., Yegneswara, V. (2004). Fusion and filtering in distributed intrusion detection systems. In *Proceedings of the 42nd Annual Allerton Conference on Communication, Control and Computing*.

Barford, P., Plonka, D. (2001). Characteristics of network traffic flow anomalies. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 69–73. New York, NY, USA: ACM. ISBN 1-58113-435-5. doi:http://doi.acm.org/10.1145/505202.505211.

Brown, C., Cowperthwaite, A., Hijazi, A., Somayaji, A. (2009). Analysis of the 1999 DARPA/Lincoln Laboratory IDS evaluation data with NetADHICT. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–7. doi:10.1109/CISDA.2009.5356522.

Brutlag, J. D. (2000). Aberrant behavior detection in time series for network monitoring. In *Proceedings of the 14th USENIX conference on System administration*, LISA '00, pp. 139–146. Berkeley, CA, USA: USENIX Association. http://dl.acm.org/citation.cfm?id=1045502.1045530

Camerer, C. F., Johnson, E. J. (1997). *Research on Judgment and Decision Making*. Cambridge University Press.

Camp, J., Cranor, L., Feamster, N., Feigenbaum, J., Forrest, S., Kotz, D., Lee, W., Lincoln, P., Paxson, V., Reiter, M., Rivest, R., Sanders, W., Savage, S., Smith, S., Spaffort, E., Stolfo, S. (2009). Data for cybersecurity research: Process and "wish list". Retrieved from webpage. http://www.cc.gatech.edu/~feamster/papers/data-wishlist.pdf

Chandola, V., Banerjee, A., Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, *41*(3), 15:1–15:58. doi:10.1145/1541880.1541882. http://doi.acm.org/10.1145/1541880.1541882

Chen, Z., Delis, A., Wei, P. (2009). A pragmatic methodology for testing intrusion prevention systems. *Comput. J.*, *52*(4), 429–460. doi:10.1093/comjnl/bxn043. http://dx.doi.org/10.1093/comjnl/bxn043

Cisco Systems (n. d.). Cisco netflow. Last accessed on 2013-03-19. http://www.cisco.com/go/netflow

Ciza, T. (2009). *Performance Enhancement of Intrusion Detection Systems using Advances in Sensor Fusion*. Ph.D. thesis, Indian Institute of Science.

Cottrell, R., Logg, C., Chhaparia, M., Grigoriev, M., Haro, F., Nazir, F., Sandford, M. (2006). Evaluation of techniques to detect significant network performance problems using end-to-end active network measurements. In *IEEE/IFIP Network Operations and Management Symposium NOMS*, pp. 85–94. IEEE. ISBN 1-4244-0142-9. doi:10.1109/NOMS.2006.1687541.

DEFCON (n. d.). Defcon 18 capture the flag contest. Last accessed on 2010-02-22. http://ddtek.biz/dc18.html

Dumitras, T., Shou, D. (2011). Toward a standard benchmark for computer security research: the worldwide intelligence network environment (WINE). In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, BADGERS '11, pp. 89–96. New York, NY, USA: ACM. ISBN 978-1-4503-0768-0. doi: 10.1145/1978672.1978683.
http://doi.acm.org/10.1145/1978672.1978683

Eimann, R. E. A. (2008). *Network Event Detection with Entropy Measures*. Ph.D. thesis, University of Auckland, Auckland.

Endace (n. d.). ERF. Last accessed on 2013-09-09.
http://wiki.wireshark.org/ERF

Fan, J., Xu, J., Ammar, M. H., Moon, S. B. (2004). Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, *46*(2), 253 – 272. doi:10.1016/j.comnet.2004.03.033.
http://www.sciencedirect.com/science/article/pii/S1389128604001197

Fontugne, R., Borgnat, P., Abry, P., Fukuda, K. (2010). Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pp. 8:1–8:12. New York, NY, USA: ACM. ISBN 978-1-4503-0448-1. doi:10.1145/1921168.1921179.
http://doi.acm.org/10.1145/1921168.1921179

Fukuda, K., Abry, B., Borgnat, P., Claffy, F., Claffy, K., Aben, E. (2008). ADMD. Last accessed on 2013-09-09.
http://admd.sf.net

Gu, G., Fogla, P., Dagon, D., Lee, W., Skorić, B. (2006). Measuring intrusion detection capability: an information-theoretic approach. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS '06, pp. 90–101. New York, NY, USA: ACM. ISBN 1-59593-272-0. doi:10.1145/1128817.1128834.
http://doi.acm.org/10.1145/1128817.1128834

Gu, Y., McCallum, A., Towsley, D. (2005). Detecting anomalies in network traffic using maximum entropy estimation. In *IMC'05: Proceedings of the*

Internet Measurement Conference 2005 on Internet Measurement Conference, pp. 32–32. Berkeley, CA, USA: USENIX Association.

Hall, D. L., McMullen, S. A. (2004). *Mathematical Techniques in Multisensor Data Fusion*. 685 Canton Street Norwood, MA 02062: Artech House, second edn.

Halpern, J. Y. (2003). *Reasoning About Uncertainty*. Massachsetts Institute of Technology.

Hettich, S., Bay, S. D. (1999). The UCI KDD archive. Irvine, CA: University of California, Department of Information and Computer Science.
http://kdd.ics.uci.edu

HP Autonomy (n. d.). Autonomy. Last visited on 2013-09-09.
http://www.autonomy.com

Kang, I., Jeong, M. K., Kong, D. (2012). A differentiated one-class classification method with applications to intrusion detection. *Expert Systems with Applications*, *39*(4), 3899 – 3905. doi:10.1016/j.eswa.2011.06.033.
http://www.sciencedirect.com/science/article/pii/S0957417411009286

Kim, S. S., Reddy, A. L. N. (2008). Statistical techniques for detecting traffic anomalies through packet header data. *IEEE/ACM Transactions on Networking*, *16*(3), 562–575. doi:http://dx.doi.org/10.1109/TNET.2007.902685.

Koks, D., Challa, S. (2005). An introduction to Bayesian and Dempster-Shafer data fusion. Tech. rep., The University of Technology,, Sydney. DSTO?TR?1436.

Laboratory, L. B. N., ICSI (2005). Lbnl/icsi enterprise tracing project.
http://www.icir.org/enterprise-tracing/

Lakhina, A., Crovella, M., Diot, C. (2004). Diagnosing network-wide traffic anomalies. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 219–230. New York, NY, USA: ACM. ISBN 1-58113-862-8. doi:http://doi.acm.org/10.1145/1015467.1015492.

Lakhina, A., Crovella, M., Diot, C. (2005). Mining anomalies using traffic feature distributions. *SIGCOMM Compututer Communications Review, 35*(4), 217–228. doi:http://doi.acm.org/10.1145/1090191.1080118.

Lincoln Laboratory, M. (n. d.). DARPA 1998, 1999, and 2000 datasets. Last accessed on 2013-02-06.
http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., Das, K. (2000a). The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks, 34*(4), 579 – 595. doi:DOI:10.1016/S1389-1286(00)00139-0. Recent Advances in Intrusion Detection Systems.
http://www.sciencedirect.com/science/article/B6VRG-411FRK9-4/2/e13b056e24ce249e712ba92a5a28948e

Lippmann, R. P., Fried, D. J., Fried, D. J., Graf, I., Graf, I., Haines, J. W., Haines, J. W., Kendall, K. R., Kendall, K. R., Mcclung, D., Mcclung, D., Weber, D., Weber, D., Webster, S. E., Webster, S. E., Wyschogrod, D., Wyschogrod, D., Cunningham, R. K., Cunningham, R. K., Zissman, M. A., Zissman, M. A. (2000b). Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *in Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, pp. 12–26.

Logg, C., Cottrell, L., Navratil, J. (2004). Experiences in traceroute and available bandwidth change analysis. In *Proceedings of the ACM SIGCOMM workshop on Network troubleshooting: research, theory and operations practice meet malfunctioning reality*, NetT '04, pp. 247–252. New York, NY, USA: ACM. ISBN 1-58113-942-X. doi:10.1145/1016687.1016690.
http://doi.acm.org/10.1145/1016687.1016690

Lu, W., Ghorbani, A. A. (2009). Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing, 1*. doi:10.1155/2009/837601.
http://asp.eurasipjournals.com/content/2009/1/837601

Mahoney, M., Chan, P. (2003). Learning rules for anomaly detection of hostile network traffic. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pp. 601–604. doi:10.1109/ICDM.2003.1250987.

Mahoney, M. V., Chan, P. K. (2001). Phad: Packet header anomaly detection for identifying hostile network traffic. Tech. rep., Florida Institute of Technology, Melbourne, FL 32901.

Mahoney, M. V., Chan, P. K. (2002). Learning nonstationary models of normal network traffic for detecting novel attacks. Tech. Rep. CS-2002-08, Florida Institute of Technology.

MAWI (n. d.). MAWI working group traffic archive. Last accessed on 2013-02-06.
http://mawi.wide.ad.jp/mawi/

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, *3*(4), 262–294. doi:10.1145/382912.382923.
http://doi.acm.org/10.1145/382912.382923

Microsoft (2010). Microsoft Network Monitor. Last accessed 2013-09-09.
http://www.microsoft.com/en-us/download/details.aspx?id=4865

Nakamura, E. F., Loureiro, A. A. F., Frery, A. C. (2007). Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, *39*(3), 1–55. doi:10.1145/1267070.1267073.
http://doi.acm.org/10.1145/1267070.1267073

NDM Technologies (n. d.). Arcsight esm. Last accessed on 2013-09-09.
http://www.ndm.net/siem/arcsight/arcsight-esm

Oldroyd, D. (1986). *The Arch of Knowledge: An Introductionary Study of the History of the Philosophy and Methodology of Science*. Kensington, Australia: New South Wales University Press.

Pang, R., Allman, M., Paxson, V., Lee, J. (2006). The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, *36*(1), 29–38. doi: 10.1145/1111322.1111330.
http://doi.acm.org/10.1145/1111322.1111330

Pang, R., Yegneswaran, V., Barford, P., Paxson, V., Peterson, L. (2004). Characteristics of internet background radiation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 27–40. New

York, NY, USA: ACM. ISBN 1-58113-821-0. doi:http://doi.acm.org/10.1145/1028788.1028794.

Paschalidis, I., Smaragdakis, G. (2009). Spatio-temporal network anomaly detection by assessing deviations of empirical measures. *IEEE/ACM Transactions on Networking*, *17*(3), 685–697. doi:10.1109/TNET.2008.2001468.

Paxson, V. (1998). Bro: a system for detecting network intruders in real-time. In *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*, SSYM'98, pp. 3–3. Berkeley, CA, USA: USENIX Association.
http://dl.acm.org/citation.cfm?id=1267549.1267552

Popper, K. R. (1972). *Conjectures and Refutations: The Growth of Scientific Knowledge:*. London, United Kingdom: Routledge and Kegan Paul, fourth edn.

Puketza, N. J., Zhang, K., Chung, M., Mukherjee, B., Olsson, R. A. (1996). A methodology for testing intrusion detection systems. *IEEE Transactions on Software Engineering*, *22*(10), 719–729. doi:10.1109/32.544350.
http://dx.doi.org/10.1109/32.544350

Reineking, T. (n. d.). pyds. Last accessed on 2013-09-09.
https://github.com/reineking/pyds/

R.Fontugne, P.Borgnat, P.Abry, K.Fukuda (n. d.). MAWILab. Last accessed on 2013-02-06.
http://www.fukuda-lab.org/mawilab/

RIPE (n. d.). RIPE data repository. Last accessed on 2013-03-13.
https://labs.ripe.net/datarepository

Roesch, M. (1999). Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, LISA '99, pp. 229–238. Berkeley, CA, USA: USENIX Association.
www.snort.org

Sangster, B., O'Connor, T. J., Cook, T., Fanelli, R., Dean, E., Adams, W. J., Morrell, C., Conti, G. (2009a). ITOC dataset.
https://www.itoc.usma.edu/research/dataset/

Sangster, B., O'Connor, T. J., Cook, T., Fanelli, R., Dean, E., Adams, W. J., Morrell, C., Conti, G. (2009b). Toward instrumenting network warfare competitions to generate labeled datasets. In *Proceedings of the 2nd conference on Cyber security experimentation and test*, CSET'09, pp. 9–9. Berkeley, CA, USA: USENIX Association.
http://dl.acm.org/citation.cfm?id=1855481.1855490

Seppälä, T., Alapaholuoma, T., Knuuti, O., Ylinen, J., Loula, P., Hätönen, K. (2010). Implicit malpractice and suspicious traffic detection in large scale ip networks. In *Fifth International Conference on Internet Monitoring and Protection (ICIMP)*, pp. 135–140. doi:10.1109/ICIMP.2010.26.

Shmoo Group (n. d.). Defcon 8, 10, 11 capture the flag contest. Last accessed on 2013-02-06.
http://cctf.shmoo.com/

Singh, S., Estan, C., Varghese, G., Savage, S. (2004). Automated worm fingerprinting. In *Proceedings of the 6th conference on Symposium on Operting Systems Design & Implementation*, pp. 4–4. Berkeley, CA, USA: USENIX Association.

Snow, R., O'Connor, B., Jurafsky, D., Ng, A. Y. (2008). Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pp. 254–263. Stroudsburg, PA, USA: Association for Computational Linguistics.
http://dl.acm.org/citation.cfm?id=1613715.1613751

Splunk Inc (n. d.). Splunk. Last visited on 2013-09-09.
http://www.splunk.com/

Suricata (n. d.). Suricata Website. Last accessed on 2013-02-06.
http://www.openinfosecfoundation.org/index.php

Tartakovsky, A., Polunchenko, A., Sokolov, G. (2013). Efficient computer network anomaly detection by changepoint detection methods. *Selected Topics in Signal Processing, IEEE Journal of*, *7*(1), 4–11. doi:10.1109/JSTSP.2012.2233713.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. (n. d.). The nsl-kdd data set. Last accessed on 2013-02-06.
http://nsl.cs.unb.ca/NSL-KDD/

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*, CISDA'09, pp. 53–58. Piscataway, NJ, USA: IEEE Press. ISBN 978-1-4244-3763-4.
http://dl.acm.org/citation.cfm?id=1736481.1736489

Tcpdump/Libpcap (2000). PCAP. Last accessed 2013-09-09.
http://www.tcpdump.org/

Tjhai, G. C., Papadaki, M., Furnell, S. M., Clarke, N. L. (2008). The problem of false alarms: Evaluation with Snort and DARPA 1999 dataset. In *Proceedings of the 5th international conference on Trust, Privacy and Security in Digital Business*, TrustBus '08, pp. 139–150. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-540-85734-1. doi:http://dx.doi.org/10.1007/978-3-540-85735-8_14.
http://dx.doi.org/10.1007/978-3-540-85735-8_14

University of Massachusetts (n. d.). UMass trace repository. Last accessed on 2013-02-06.
http://traces.cs.umass.edu/

Waikato Internet Traffic Storage (n. d.). WITS Website. Last accessed on 2010-02-22.
www.wand.net.nz/wits

WAND Network Research Group (n. d.). WDCap. Last visited on 2013-09-09.
http://research.wand.net.nz/software/wdcap.php

Witten, I. H., Frank, E., Hall, M. A. (2012). *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington, MA, USA: Morgan Kaufmann, third edn.

Wustrow, E., Karir, M., Bailey, M., Jahanian, F., Huston, G. (2010). Internet background radiation revisited. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pp. 62–74. New York, NY, USA: ACM. ISBN 978-1-4503-0483-2. doi:10.1145/1879141.1879149. http://doi.acm.org/10.1145/1879141.1879149

Zhong, S., Khoshgoftaar, T. M., Seliya, N. (2007). Clustering-based network intrusion detection. *International Journal of Reliability, Quality and Safety Engineering*, *14*(02), 169–187. doi:10.1142/S0218539307002568. http://www.worldscientific.com/doi/abs/10.1142/S0218539307002568

A Additional Results From the Case Study

This appendix contains a more detailed breakdown of the results achieved in Chapter 6. The results in that chapter are presented as an aggregate of both weeks of testing data. In this appendix those results are broken down into each day of the week.

A.1 DARPA 1999 Week 1

A.1.1 Alad

Day	Attack name	TPs	FPs	False Negative (FN)s
Monday	ftpwrite	3	0	1
Monday	sendmail	2	0	0
Monday	yaga	1	0	1
Monday	sshtrojanInstall	1	0	1
Monday	ps	0	0	1
Monday	snmpget	0	0	167
Monday	secret	0	0	2
Monday	guesstelnet	0	0	1
Monday	smurf	0	0	1
Monday	xsnoop	0	0	1
Monday	portsweep	0	0	21

Day	Attack name	TPs	FPs	False Negative (FN)s
Monday	guessftp	0	0	4
Monday		0	40	0
Tuesday	ps	2	0	1
Tuesday	phf	1	0	0
Tuesday	secret	2	0	1
Tuesday	sechole	2	0	1
Tuesday	mailbomb	1	0	27
Tuesday	crashiis	1	0	0
Tuesday	processtable	0	0	88
Tuesday	land	0	0	1
Tuesday	httptunnel	0	0	1
Tuesday	sqlattack	0	0	1
Tuesday	loadmodule	0	0	1
Tuesday		0	63	0
Wednesday	satan	7	0	1
Wednesday	netcat	4	0	1
Wednesday	warezmaster	3	0	0
Wednesday	ncftp	1	0	1
Wednesday	mailbomb	1	0	15
Wednesday	guessftp	1	0	13
Wednesday	processtable	0	0	126
Wednesday	named	0	0	1
Wednesday	guest	0	0	1
Wednesday	secret	0	0	1
Wednesday	guesstelnet	0	0	1
Wednesday	ppmarcro	0	0	4
Wednesday	smurf	0	0	1
Wednesday	snmpget	0	0	201
Wednesday	imap	0	0	1
Wednesday	portsweep	0	0	1
Wednesday		0	85	0
Thursday	guesspop	2	0	7
Thursday	phf	1	0	0
Thursday	sshtrojan	2	0	0
Thursday	mailbomb	1	0	64

Day	Attack name	TPs	FPs	False Negative (FN)s
Thursday	ntinfoscan	1	0	0
Thursday	sqlattack	0	0	1
Thursday	guest	0	0	2
Thursday	xlock	0	0	1
Thursday	teardrop	0	0	1
Thursday	sshprocesstable	0	0	123
Thursday	ppmarcro	0	0	3
Thursday	ncftp	0	0	1
Thursday	dosnuke	0	0	1
Thursday	netbus	0	0	4
Thursday		0	117	0
Friday	sshtrojan	2	0	0
Friday	mailbomb	1	0	47
Friday	sechole	3	0	1
Friday	netbus	2	0	3
Friday	named	0	0	2
Friday	ipsweep_unlikely	0	0	5
Friday	xlock	0	0	1
Friday	ipsweep	0	0	262
Friday	portsweep	0	0	11
Friday	loadmodule	0	0	1
Friday	smurf	0	0	1
Friday	ncftp	0	0	1
Friday		0	76	0

Table A.1: Alad on week 1 of the DARPA 1999 testing dataset

A.1.2 Bro

Day	Attack name	TPs	FPs	FNs
Monday	ftpwrite	1	0	2
Monday	ps	0	0	1
Monday	snmpget	0	0	167
Monday	sendmail	0	0	1
Monday	yaga	0	0	2
Monday	secret	0	0	2

Day	Attack name	TPs	FPs	FNs
Monday	guesstelnet	0	0	1
Monday	smurf	0	0	1
Monday	sshtrojanInstall	0	0	2
Monday	xsnoop	0	0	1
Monday	portsweep	0	0	20
Monday	guessftp	0	0	4
Monday		0	3	0
Tuesday		0	3	0
Wednesday		0	3	0
Friday	loadmodule	1	0	0
Friday	named	0	0	2
Friday	sshtrojan	0	0	1
Friday	ipsweep_unlikely	0	0	5
Friday	xlock	0	0	1
Friday	ipsweep	0	0	262
Friday	portsweep	0	0	9
Friday	smurf	0	0	1
Friday	ncftp	0	0	1
Friday	mailbomb	0	0	48
Friday	sechole	0	0	3
Friday	netbus	0	0	3
Friday		0	4	0

Table A.2: Bro on week 1 of the DARPA 1999 testing dataset

A.1.3 Phad

Day	Attack name	TPs	FPs	FNs
Monday	sendmail	2	0	0
Monday	yaga	2	0	0
Monday	ftpwrite	2	0	2
Monday	smurf	2	0	0
Monday	portsweep	30	0	1
Monday	sshtrojanInstall	1	0	1
Monday	ps	0	0	1
Monday	snmpget	0	0	167

Day	Attack name	TPs	FPs	FNs
Monday	secret	0	0	2
Monday	guesstelnet	0	0	1
Monday	xsnoop	0	0	1
Monday	guessftp	0	0	4
Monday		0	68	0
Tuesday		0	36	0
Wednesday	warezmaster	2	0	1
Wednesday	snmpget	1	0	200
Wednesday	imap	2	0	0
Wednesday	portsweep	1	0	0
Wednesday	processtable	0	0	126
Wednesday	named	0	0	1
Wednesday	guest	0	0	1
Wednesday	secret	0	0	1
Wednesday	satan	0	0	5
Wednesday	guesstelnet	0	0	1
Wednesday	ppmarcro	0	0	4
Wednesday	smurf	0	0	1
Wednesday	netcat	0	0	3
Wednesday	ncftp	0	0	2
Wednesday	mailbomb	0	0	16
Wednesday	guessftp	0	0	14
Wednesday		0	40	0
Thursday	teardrop	1	0	0
Thursday	ncftp	1	0	0
Thursday	guest	1	0	1
Thursday	dosnuke	1	0	0
Thursday	phf	1	0	0
Thursday	sqlattack	0	0	1
Thursday	ntinfoscan	0	0	1
Thursday	sshtrojan	0	0	1
Thursday	xlock	0	0	1
Thursday	sshprocesstable	0	0	123
Thursday	ppmarcro	0	0	3
Thursday	guesspop	0	0	8

Day	Attack name	TPs	FPs	FNs
Thursday	mailbomb	0	0	65
Thursday	netbus	0	0	4
Thursday		0	130	0
Friday	smurf	1	0	0
Friday	mailbomb	1	0	47
Friday	portsweep	5	0	6
Friday	named	0	0	2
Friday	sshtrojan	0	0	1
Friday	ipsweep_unlikely	0	0	5
Friday	xlock	0	0	1
Friday	ipsweep	0	0	262
Friday	loadmodule	0	0	1
Friday	ncftp	0	0	1
Friday	sechole	0	0	3
Friday	netbus	0	0	4
Friday		0	28	0

Table A.3: PHAD on week 1 of the DARPA 1999 testing dataset

A.1.4 Snort

Day	Attack Name	GID	SID	TPs	FPs	FNs
Monday	sendmail	1	648	1	0	0
Monday	secret	1	1882	1	0	0
Monday	snmpget	1	1417	167	0	0
Monday	xsnoop	1	1226	1	0	0
Monday	portsweep	1	621	10	0	0
Monday	guessftp	1	13360	2	0	0
Monday	ps			0	0	1
Monday	yaga			0	0	2
Monday	secret			0	0	1
Monday	ftpwrite			0	0	3
Monday	guesstelnet			0	0	1
Monday	smurf			0	0	1
Monday	sshtrojanInstall			0	0	2
Monday	portsweep			0	0	11

Day	Attack Name	GID	SID	TPs	FPs	FNs
Monday	guessftp			0	0	2
Tuesday	phf	1	1147	1	0	0
Tuesday	sechole	1	1292	1	0	0
Tuesday	sechole	1	16363	1	0	0
Tuesday	sechole	1	648	1	0	0
Tuesday	processtable			0	0	88
Tuesday	land			0	0	1
Tuesday	httptunnel			0	0	1
Tuesday	ps			0	0	3
Tuesday	sqlattack			0	0	1
Tuesday	secret			0	0	3
Tuesday	loadmodule			0	0	1
Tuesday	mailbomb			0	0	28
Tuesday	crashiis			0	0	1
Wednesday	named	1	648	1	0	0
Wednesday	ppmarcro	1	1394	1	0	0
Wednesday	netcat	1	1292	1	0	0
Wednesday	netcat	1	1394	1	0	0
Wednesday	snmpget	1	1411	201	0	0
Wednesday	imap	1	648	1	0	0
Wednesday	guessftp	1	13360	14	0	0
Wednesday	processtable			0	0	126
Wednesday	guest			0	0	1
Wednesday	secret			0	0	1
Wednesday	guesstelnet			0	0	1
Wednesday	ppmarcro			0	0	3
Wednesday	smurf			0	0	1
Wednesday	netcat			0	0	1
Wednesday	warezmaster			0	0	2
Wednesday	ncftp			0	0	2
Wednesday	mailbomb			0	0	16
Wednesday	portsweep			0	0	1
Wednesday	satan			0	0	5
Thursday	sshtrojan	128	4	1	0	0
Thursday	phf	1	1147	1	0	0

Day	Attack Name	GID	SID	TPs	FPs	FNs
Thursday	xlock	1	1226	1	0	0
Thursday	teardrop	123	3	1	0	0
Thursday	ppmarcro	1	1394	1	0	0
Thursday	dosnuke	1	1257	1	0	0
Thursday	netbus	1	1394	1	0	0
Thursday	netbus	1	110	1	0	0
Thursday	sqlattack			0	0	1
Thursday	guest			0	0	2
Thursday	ntinfoscan			0	0	1
Thursday	sshprocesstable			0	0	123
Thursday	ppmarcro			0	0	2
Thursday	guesspop			0	0	8
Thursday	ncftp			0	0	1
Thursday	mailbomb			0	0	65
Thursday	netbus			0	0	2
Friday	named	1	648	2	0	0
Friday	sshtrojan	128	4	1	0	0
Friday	xlock	1	1226	1	0	0
Friday	ipsweep_unlikely	1	368	5	0	0
Friday	sechole	1	1292	1	0	0
Friday	sechole	1	16363	1	0	0
Friday	sechole	1	648	1	0	0
Friday	netbus	1	1394	1	0	0
Friday	netbus	1	110	1	0	0
Friday	ipsweep			0	0	262
Friday	loadmodule			0	0	1
Friday	smurf			0	0	1
Friday	ncftp			0	0	1
Friday	mailbomb			0	0	48
Friday	portsweep			0	0	11
Friday	netbus			0	0	2

Table A.4: Snort on week 1 of the DARPA 1999 testing dataset

A.1.5 T-Entropy

A.2 DARPA 1999 Week 2

A.2.1 Alad

Day	Attack name	TPs	FPs	FNs
Monday	warezclient	2	0	1
Monday	ncftp	1	0	1
Monday	crashiis	2	0	0
Monday	selfping	0	0	1
Monday	portsweep	0	0	12
Monday	udpstorm	0	0	1
Monday	syslogd	0	0	1
Monday	portsweep_unlikely	0	0	2
Monday	neptune	0	0	20480
Monday	apache2	0	0	172
Monday	ipsweep_unlikely	0	0	5
Monday	dict	0	0	8
Monday	ffbconfig	0	0	1
Monday	loadmodule	0	0	1
Monday	smurf	0	0	1532
Monday	guesstelnet	0	0	1
Monday	pod	0	0	3
Monday	imap	0	0	1
Monday	dosnuke	0	0	2
Monday	ls	0	0	1
Monday		0	121	0
Tuesday	casesen	2	0	1
Tuesday	ftpwrite	3	0	0
Tuesday	fdformat	3	0	0
Tuesday	xterm1	3	0	1
Tuesday	ppmarcro	1	0	13
Tuesday	httptunnel	0	0	2
Tuesday	syslogd	0	0	1
Tuesday	eject	0	0	1
Tuesday	ps	0	0	1

Day	Attack name	TPs	FPs	FNs
Tuesday	queso	0	0	5
Tuesday	neptune	0	0	51200
Tuesday	back	0	0	1
Tuesday	yaga	0	0	2
Tuesday	perl	0	0	1
Tuesday	ipsweep	0	0	37
Tuesday	teardrop	0	0	1
Tuesday	udpstorm	0	0	1
Tuesday	selfping	0	0	1
Tuesday	ncftp	0	0	1
Tuesday	xsnoop	0	0	1
Tuesday	dosnuke	0	0	1
Tuesday		0	147	0
Wednesday	phf	1	0	0
Wednesday	netbus	2	0	2
Wednesday	selfping	0	0	1
Wednesday	processtable	0	0	129
Wednesday	snmpget	0	0	20
Wednesday	queso	0	0	11
Wednesday	back	0	0	15
Wednesday	perl	0	0	1
Wednesday	xlock	0	0	1
Wednesday	ffbconfig	0	0	2
Wednesday	netcat	0	0	1
Wednesday	apache2	0	0	89
Wednesday	portsweep	0	0	109
Wednesday		0	162	0
Thursday	casesen	2	0	2
Thursday	fdformat	3	0	1
Thursday	ntinfoscan	9	0	9
Thursday	phf	2	0	0
Thursday	warezclient	2	0	1
Thursday	satan	5	0	11279
Thursday	sechole	3	0	1
Thursday	mscan	2	0	3058

Day	Attack name	TPs	FPs	FNs
Thursday	httptunnel	0	0	2
Thursday	yaga	0	0	2
Thursday	ipsweep	0	0	9
Thursday	teardrop	0	0	1
Thursday	ls	0	0	1
Thursday	portsweep_unlikely	0	0	3
Thursday	snmpget	0	0	61
Thursday	resetscan	0	0	1
Thursday		0	603	0
Friday	warezclient	2	0	1
Friday	xterm	3	0	2
Friday	eject	3	0	1
Friday	sendmail	2	0	0
Friday	framespoofer	2	0	0
Friday	yaga	2	0	1
Friday	crashiis	3	0	0
Friday	land	0	0	1
Friday	guest	0	0	2
Friday	sqlattack	0	0	1
Friday	queso	0	0	5
Friday	syslogd	0	0	2
Friday	neptune	0	0	400
Friday	back	0	0	6
Friday	perl	0	0	1
Friday	guesstelnet	0	0	3
Friday	netcat	0	0	1
Friday	xsnoop	0	0	1
Friday	portsweep	0	0	11
Friday		0	96	0

Table A.5: Alad on week 2 of the DARPA 1999 testing dataset

A.2.2 Bro

Day	Attack name	TPs	FPs	FNs
Monday	ffbconfig	1	0	0

Day	Attack name	TPs	FPs	FNs
Monday	selfping	0	0	1
Monday	warezclient	0	0	2
Monday	portsweep	0	0	12
Monday	udpstorm	0	0	1
Monday	syslogd	0	0	1
Monday	portsweep_unlikely	0	0	2
Monday	neptune	0	0	20480
Monday	apache2	0	0	172
Monday	ipsweep_unlikely	0	0	5
Monday	dict	0	0	8
Monday	ls	0	0	1
Monday	loadmodule	0	0	1
Monday	dosnuke	0	0	2
Monday	smurf	0	0	1532
Monday	guesstelnet	0	0	1
Monday	pod	0	0	3
Monday	ncftp	0	0	2
Monday	imap	0	0	1
Monday	crashiis	0	0	1
Monday		0	10	0
Tuesday	ftpwrite	1	0	2
Tuesday	teardrop	0	0	1
Tuesday	casesen	0	0	3
Tuesday	fdformat	0	0	2
Tuesday	httptunnel	0	0	2
Tuesday	syslogd	0	0	1
Tuesday	eject	0	0	1
Tuesday	ps	0	0	1
Tuesday	queso	0	0	5
Tuesday	xterm1	0	0	3
Tuesday	neptune	0	0	51200
Tuesday	back	0	0	1
Tuesday	yaga	0	0	2
Tuesday	perl	0	0	1
Tuesday	ipsweep	0	0	37

Day	Attack name	TPs	FPs	FNs
Tuesday	udpstorm	0	0	1
Tuesday	selfping	0	0	1
Tuesday	ppmarcro	0	0	13
Tuesday	ncftp	0	0	1
Tuesday	xsnoop	0	0	1
Tuesday	dosnuke	0	0	1
Tuesday		0	15	0
Wednesday		0	5	0
Thursday	ntinfoscan	2	0	13
Thursday	fdformat	1	0	2
Thursday	mscan	2	0	3058
Thursday	casesen	0	0	4
Thursday	httptunnel	0	0	2
Thursday	warezclient	0	0	2
Thursday	phf	0	0	1
Thursday	yaga	0	0	2
Thursday	ipsweep	0	0	9
Thursday	teardrop	0	0	1
Thursday	satan	0	0	11284
Thursday	ls	0	0	1
Thursday	portsweep_unlikely	0	0	3
Thursday	snmpget	0	0	61
Thursday	resetscan	0	0	1
Thursday	sechole	0	0	3
Thursday		0	25	0
Friday	eject	1	0	2
Friday	warezclient	0	0	2
Friday	xterm	0	0	4
Friday	land	0	0	1
Friday	guest	0	0	2
Friday	framespoofer	0	0	1
Friday	queso	0	0	5
Friday	syslogd	0	0	2
Friday	sendmail	0	0	1
Friday	neptune	0	0	400

Day	Attack name	TPs	FPs	FNs
Friday	back	0	0	6
Friday	yaga	0	0	2
Friday	perl	0	0	1
Friday	xsnoop	0	0	1
Friday	guesstelnet	0	0	3
Friday	portsweep	0	0	11
Friday	netcat	0	0	1
Friday	sqlattack	0	0	1
Friday	crashiis	0	0	2
Friday		0	11	0

Table A.6: Bro on week 2 of the DARPA 1999 testing dataset

A.2.3 Phad

Day	Attack name	TPs	FPs	FNs
Monday	portsweep	14	0	0
Monday	syslogd	1	0	0
Monday	neptune	2	0	20478
Monday	apache2	3	0	169
Monday	guesstelnet	2	0	0
Monday	dosnuke	1	0	1
Monday	smurf	2	0	1530
Monday	udpstorm	1	0	0
Monday	pod	5	0	0
Monday	portsweep_unlikely	1	0	1
Monday	crashiis	1	0	0
Monday	selfping	0	0	1
Monday	warezclient	0	0	2
Monday	ipsweep_unlikely	0	0	5
Monday	dict	0	0	8
Monday	ffbconfig	0	0	1
Monday	loadmodule	0	0	1
Monday	ncftp	0	0	2
Monday	imap	0	0	1
Monday	ls	0	0	1

Day	Attack name	TPs	FPs	FNs
Monday		0	380	0
Tuesday	fdformat	2	0	1
Tuesday	syslogd	1	0	0
Tuesday	queso	3	0	3
Tuesday	neptune	6	0	51194
Tuesday	ipsweep	5	0	32
Tuesday	ftpwrite	1	0	2
Tuesday	udpstorm	2	0	0
Tuesday	teardrop	2	0	0
Tuesday	ncftp	1	0	0
Tuesday	dosnuke	1	0	0
Tuesday	casesen	0	0	3
Tuesday	httptunnel	0	0	2
Tuesday	eject	0	0	1
Tuesday	ps	0	0	1
Tuesday	xterm1	0	0	3
Tuesday	back	0	0	1
Tuesday	yaga	0	0	2
Tuesday	perl	0	0	1
Tuesday	ppmarcro	0	0	14
Tuesday	xsnoop	0	0	1
Tuesday	selfping	0	0	1
Tuesday		0	444	0
Wednesday	queso	6	0	6
Wednesday	xlock	2	0	0
Wednesday	portsweep	13	0	98
Wednesday	apache2	2	0	87
Wednesday	selfping	0	0	1
Wednesday	processtable	0	0	129
Wednesday	snmpget	0	0	20
Wednesday	back	0	0	15
Wednesday	perl	0	0	1
Wednesday	ffbconfig	0	0	2
Wednesday	netcat	0	0	1
Wednesday	phf	0	0	1

Day	Attack name	TPs	FPs	FNs
Wednesday	netbus	0	0	3
Wednesday		0	93	0
Thursday	portsweep_unlikely	2	0	1
Thursday	snmpget	1	0	60
Thursday	satan	1	0	11283
Thursday	teardrop	2	0	0
Thursday	mscan	4	0	3056
Thursday	casesen	0	0	4
Thursday	fdformat	0	0	3
Thursday	httptunnel	0	0	2
Thursday	warezclient	0	0	2
Thursday	ntinfoscan	0	0	15
Thursday	phf	0	0	1
Thursday	yaga	0	0	2
Thursday	ipsweep	0	0	9
Thursday	ls	0	0	1
Thursday	resetscan	0	0	1
Thursday	sechole	0	0	3
Thursday		0	727	0
Friday	syslogd	2	0	0
Friday	sendmail	1	0	0
Friday	queso	3	0	2
Friday	xterm	2	0	2
Friday	neptune	1	0	399
Friday	portsweep	11	0	1
Friday	warezclient	0	0	2
Friday	land	0	0	1
Friday	guest	0	0	2
Friday	eject	0	0	3
Friday	framespoofer	0	0	1
Friday	back	0	0	6
Friday	yaga	0	0	2
Friday	perl	0	0	1
Friday	xsnoop	0	0	1
Friday	guesstelnet	0	0	3

Day	Attack name	TPs	FPs	FNs
Friday	netcat	0	0	1
Friday	sqlattack	0	0	1
Friday	crashiis	0	0	2
Friday		0	347	0

Table A.7: PHAD on week 2 of the DARPA 1999 testing dataset

A.2.4 Snort

Day	Attack Name	GID	SID	TPs	FPs	FNs
Monday	apache2	119	20	172	0	0
Monday	ipsweep_unlikely	1	368	5	0	0
Monday	ffbconfig	1	1882	1	0	0
Monday	dosnuke	1	1257	2	0	0
Monday	imap	1	648	1	0	0
Monday	portsweep	1	621	12	0	0
Monday	selfping			0	0	1
Monday	warezclient			0	0	2
Monday	udpstorm			0	0	1
Monday	syslogd			0	0	1
Monday	neptune			0	0	20480
Monday	guesstelnet			0	0	1
Monday	ls			0	0	1
Monday	loadmodule			0	0	1
Monday	smurf			0	0	1532
Monday	dict			0	0	8
Monday	pod			0	0	3
Monday	ncftp			0	0	2
Monday	portsweep_unlikely			0	0	2
Monday	crashiis			0	0	1
Tuesday	casesen	1	16363	1	0	0
Tuesday	queso	1	621	1	0	0
Tuesday	queso	1	624	1	0	0
Tuesday	perl	1	1882	1	0	0
Tuesday	teardrop	123	3	1	0	0
Tuesday	xsnoop	1	1226	1	0	0

Day	Attack Name	GID	SID	TPs	FPs	FNs
Tuesday	dosnuke	1	1257	1	0	0
Tuesday	casesen			0	0	2
Tuesday	fdformat			0	0	2
Tuesday	httptunnel			0	0	2
Tuesday	syslogd			0	0	1
Tuesday	eject			0	0	1
Tuesday	ps			0	0	1
Tuesday	queso			0	0	3
Tuesday	neptune			0	0	51200
Tuesday	back			0	0	1
Tuesday	yaga			0	0	2
Tuesday	ipsweep			0	0	37
Tuesday	ftpwrite			0	0	3
Tuesday	udpstorm			0	0	1
Tuesday	ppmarcro			0	0	14
Tuesday	ncftp			0	0	1
Tuesday	xterm1			0	0	3
Tuesday	selfping			0	0	1
Wednesday	queso	1	621	2	0	0
Wednesday	queso	1	624	1	0	0
Wednesday	phf	1	1147	1	0	0
Wednesday	apache2	119	20	89	0	0
Wednesday	xlock	1	1226	1	0	0
Wednesday	snmpget	1	1411	20	0	0
Wednesday	portsweep	1	621	10	0	0
Wednesday	netbus	1	1394	1	0	0
Wednesday	netbus	1	110	1	0	0
Wednesday	selfping			0	0	1
Wednesday	processtable			0	0	129
Wednesday	queso			0	0	8
Wednesday	back			0	0	15
Wednesday	perl			0	0	1
Wednesday	ffbconfig			0	0	2
Wednesday	netcat			0	0	1
Wednesday	portsweep			0	0	99

Day	Attack Name	GID	SID	TPs	FPs	FNs
Wednesday	netbus			0	0	1
Thursday	casesen	1	1292	1	0	0
Thursday	casesen	1	16363	1	0	0
Thursday	ntinfoscan	1	973	3	0	0
Thursday	ntinfoscan	125	8	2	0	0
Thursday	phf	1	1147	1	0	0
Thursday	yaga	1	1292	1	0	0
Thursday	teardrop	123	3	1	0	0
Thursday	snmpget	1	1411	61	0	0
Thursday	sechole	1	1292	1	0	0
Thursday	sechole	1	16363	1	0	0
Thursday	sechole	1	648	1	0	0
Thursday	casesen			0	0	2
Thursday	fdformat			0	0	3
Thursday	httptunnel			0	0	2
Thursday	warezclient			0	0	2
Thursday	ntinfoscan			0	0	10
Thursday	yaga			0	0	1
Thursday	ipsweep			0	0	9
Thursday	satan			0	0	11284
Thursday	ls			0	0	1
Thursday	portsweep_unlikely			0	0	3
Thursday	resetscan			0	0	1
Thursday	mscan			0	0	3060
Friday	eject	125	2	1	0	0
Friday	sendmail	1	648	1	0	0
Friday	queso	1	621	1	0	0
Friday	queso	1	624	1	0	0
Friday	yaga	1	1292	1	0	0
Friday	netcat	1	1394	1	0	0
Friday	xsnoop	1	1226	1	0	0
Friday	portsweep	1	621	5	0	0
Friday	warezclient			0	0	2
Friday	land			0	0	1
Friday	guest			0	0	2

Day	Attack Name	GID	SID	TPs	FPs	FNs
Friday	eject			0	0	2
Friday	framespoofer			0	0	1
Friday	queso			0	0	3
Friday	syslogd			0	0	2
Friday	xterm			0	0	4
Friday	neptune			0	0	400
Friday	back			0	0	6
Friday	yaga			0	0	1
Friday	perl			0	0	1
Friday	guesstelnet			0	0	3
Friday	portsweep			0	0	6
Friday	sqlattack			0	0	1
Friday	crashiis			0	0	2

Table A.8: Snort on week 2 of the DARPA 1999 testing dataset

B

Annotation File Format

This appendix contains the actual C header files describing the annotation file format. The library together with supporting examples can be accessed from http://www.wand.net.nz/~andreasl.

Listing B.1: The Annotation Header

```
/*
    Copyright (C) 2012  Andreas Löf <andreas.
       lof@cs.waikato.ac.nz>

    This program is free software: you can
       redistribute it and/or modify
    it under the terms of the GNU General Public
       License as published by
    the Free Software Foundation, either version
       3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that
       it will be useful,
    but WITHOUT ANY WARRANTY; without even the
       implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR
       PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU
       General Public License
    along with this program.  If not, see <http
       ://www.gnu.org/licenses/>.
*/
/*
 *  annotation.h
 *
 *
 */

#ifndef __annotation_h_
#define __annotation_h_


#include <stdio.h>
#include <inttypes.h>
#include <netinet/in.h>

enum container_type {
  CONTAINER_ANNOTATION = 0
};




/*
 * The different types of annotations we support.
 * Generic events that cover the entire trace,
    like link faults and ddos, portscans.
 * Flow specific events, for marking flows as
   anomalous.
 * Packet specific, for marking packets as
   anomalous.
 */
enum annotation_type{
  ANNOTATION_TYPE_EVENT = 0,
  ANNOTATION_TYPE_FLOW = 1,
  ANNOTATION_TYPE_PACKET = 2,
  ANNOTATION_TYPE_EVENT_START = 3,
  ANNOTATION_TYPE_EVENT_END = 4
};

enum identification_type {
  ANNOTATION_IDENTIFICATION_ETHERNET = 0,
  ANNOTATION_IDENTIFICATION_HOST = 1,
  ANNOTATION_IDENTIFICATION_SERVICE = 2,
  ANNOTATION_IDENTIFICATION_NONE = 3
};



typedef struct annotation_t {
  annotation_type type;

  /* timestamp */
  uint32_t seconds;
  uint32_t microseconds;

  float accuracy; /* in percentatages, we aim for
      a 5 digit accuracy */

  identification_type id_type;
  void * identification;
  char* url; /* hierarchical url */
} Annotation;


struct annotation_identification_ethernet_t {
  uint8_t mac_src[6]; /* network byte order */
  uint8_t mac_dst[6];  /* network byte order */
  /* to support both ipv6 and ipv4 */
  struct sockaddr_storage ip_src;
  struct sockaddr_storage ip_dst;
  uint8_t protocol;
  uint16_t port_src; /*host byte order*/
  uint16_t port_dst; /*host byte order*/
} __attribute__((__packed__));

typedef struct
   annotation_identification_ethernet_t
   EthernetIdentification;

struct annotation_identification_ip_service_t {
  /* to support both ipv6 and ipv4 */
  struct sockaddr_storage ip;
  uint8_t protocol;
  uint16_t port; /*host byte order*/
} __attribute__((__packed__));

typedef struct
   annotation_identification_ip_service_t
   IpServiceIdentification;

struct annotation_identification_ip_host_t {
  /* to support both ipv6 and ipv4 */
  struct sockaddr_storage ip;
```

```
} __attribute__((__packed__));

typedef struct
   annotation_identification_ip_host_t
   IpHostIdentification;

typedef struct annotation_file_t {
  char* path;
  FILE* handle;
} AnnotationFile;


/* returns true if successfull, otherwise false
   */
bool write_annotation(AnnotationFile *af,
   Annotation* a,uint32_t en);

/* returns NULL on failure, always return a
   static buffer*/
Annotation * read_annotation(AnnotationFile *af);

AnnotationFile* open_annotation_file_in(char *
   path);
AnnotationFile* open_annotation_file_out(char *
   path);

void close_annotation_file(AnnotationFile* af);


#endif
```

Listing B.2: The Annotation File Format Header

```
/*
    Copyright (C) 2012  Andreas Löf <andreas.
       lof@cs.waikato.ac.nz>

    This program is free software: you can
       redistribute it and/or modify
    it under the terms of the GNU General Public
       License as published by
    the Free Software Foundation, either version
       3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that
       it will be useful,
    but WITHOUT ANY WARRANTY; without even the
       implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR
       PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU
       General Public License
    along with this program.  If not, see <http
       ://www.gnu.org/licenses/>.
*/

#include "annotation.h"

#ifndef __annotationfileformat_h
#define __annotationfileformat_h
```

```
/*********************
 |-------------------|
 | container         |
 |-------------------|
 | annotation        |
 |-------------------|
 | identification    |
 |-------------------|
 | url               |
 |-------------------|
 *********************/

/*
 * Contains everything else. At the moment we
    only do annotations though.
 *
 */

struct container_t {
  uint32_t content_type;
  uint32_t en; //enterprise number as define by
    IANA
  uint32_t length; // annotaion + identification
    + (url + \0)
}  __attribute__((__packed__));

typedef struct container_t FileContainer;




/* always saved in network byte order */
```

```
struct trace_annotation_t {
  uint32_t type;

  /* timestamp */
  uint32_t seconds;
  uint32_t microseconds;

  uint32_t accuracy;
  uint32_t divisor;

  uint32_t id_type;

  uint32_t url_length; // one null terminated url
      that should be hierarchical
}  __attribute__((__packed__));

typedef struct trace_annotation_t FileAnnotation;




#endif
```

C
Capture Ethics Agreement

The ethics agreements are on the following two pages.

School of Computing & Mathematical Sciences
The University of Waikato
Private Bag 3105
Hamilton
New Zealand

Phone +64 7 838 4021
www.scms.waikato.ac.nz

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

27th January 2009

To: Postgraduate Committee

PhD Proposal

Using Machine Learning to Detect Events in Network Flows

Lennart Andreas Löf (ID 1085967)

The Ethics Committee of the School of Computing and Mathematical Sciences has considered the PhD proposal as described in the document:

Löf, Lennart Andreas. *Using Machine Learning to Detect Events in Network Flows*
Full PhD Research

The passive capture at commercial internet providers is outlined in the proposal in Appendix A (Richard Nelson's letter of ethics approval). See also the ethics statement in Section 6.2.

The School of Computing and Mathematical Sciences Ethics Committee considers that this document constitutes sufficient evidence to indicate that the proposed studies will comply with the Human Research Ethics Regulations for the University of Waikato as described in the University Calendar and available online at:

http://calendar.waikato.c.nz/assessment/humanresearchethics.html

Additionally, the individual studies outlined in the proposal (or any others subsequently proposed) will be required to pass through the normal approval process of the School of Computing and Mathematical Sciences Ethics Committee.

Yours faithfully

[illegible]

Masood Masoodian
(on behalf of the)
Ethics Committee
School of Computing and Mathematical Sciences

School of Computing &
Mathematical Sciences
The University of Waikato
Private Bag 3105
Hamilton
New Zealand

Phone +64 7 838 4021
www.scms.waikato.ac.nz

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

15th October 2008

Richard Nelson
C/- Department of Computer Science
University of Waikato

Dear Richard

Request for ethical approval to conduct a research project involving human participants

I have considered your request for approval to conduct a research project involving human participants during October 2008 to passively capture data at Commercial Internet Services Providers.

The purpose of this project is to collect data which will be used as the basis for research into the performance of the public Internet and tracking usage patterns in terms of protocol operation and a range of studies are expected to be performed using the data. The experiment will consist of placing passive collection probes in the machine rooms of two New Zealand ISPs.

The procedure described in your application is acceptable. I note that the data collection details have been agreed with the two ISPs involved and will be part of a formal agreement with them. The agreement specifically allows the publication of statistics and other non identifying aggregations or analysis of the collect data and anonymised trace data. Release of any results to anybody outside the WAND group must be approved by the WAND group academic staff member leading the project responsible for the data collection.

The collected data will be transferred to secure servers in the WAND machine room and access to the data will be confined to staff and students who have signed the confidentiality requirement.

The prescribed details in your Application for approval under the Ethical Conduct in Human Research and Related Activities Regulations and attachments, comply with the requirements of the University's human research ethics policies and procedures.

I therefore approve your application to undertake the experiment.

Yours faithfully

[illegible]

Dave Nichols
Department of Computer Science
Human Research Ethics Committee
School of Computing and Mathematical Sciences