PhD Research Proposal: Semantic Modelling of Network Traffic for Anomaly Detection

Henry Clausen

March 8, 2019

1. Research description

1.1 Intrusion detection

Sophisticated data breaches affect hundreds of million customers and inflict tremendous financial, reputational, and logistic damage. One reason for the recent rise of cyber crime is the increased use of advanced techniques for the attack of specific targets. Attackers use customised social engineering and custom-built malware that penetrate defensive barriers and potentially stay undetected in an infected system for an extended period of time.

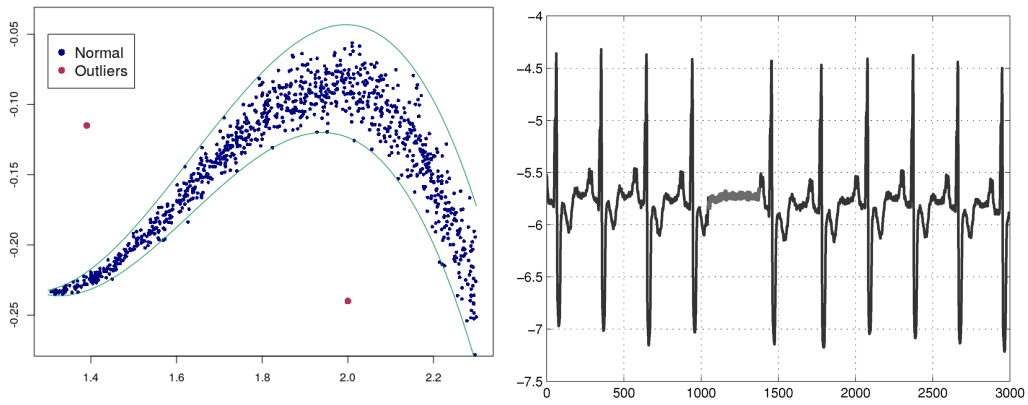
Cyber-security relies on a range of defensive techniques, including sophisticated intrusion detection systems and firewalls that try to detect and prevent attacks against software subsystems. Malicious software still remains the biggest threat to computer users, and its detection is of utmost importance.

The field of intrusion detection is concerned with the development of methods and tools that identify and locate possible intrusions in a computer network. An *intrusion detection system* (IDS) is typically a device or software application that detects malicious activity or policy violations in an automated way by scanning incoming data collected from one or more sources for patterns that a model or a set of rules classifies as potentially malicious.

Intrusion detection is a well researched area, with the first IDSs emerging in the late 1980's. Intrusion detection today comprises a variety of research areas in terms of different types of data sources, system architectures, detection sope, and so forth. Denning [10] in 1987 established the notion of two different types IDS implementations: a) host-based and b) network-based. *Host-based intrusion detection systems* (HIDS) monitor each host in a network individually, and rely on local data streams such as application logs or raw operating system calls as data source. *Network-based intrusion detection* refers to the detection of malicious traffic in a network of computers and uses network traffic captures as a data source. In this PhD project, I will mainly focus on network traffic as a data source due its universal nature, its resilience against modification, and my previous experience in the field. However, I will also investigate the possibility of relating both types of traffic. For more details on IDS implementations, I refer the reader to my more detailed literature review.

Current detection methods are predominantly based on the analysis of previously identified attack signatures, which provides great reliability and low false alert rates. However, these methods are dependent on an updated attack signature database and provide no protection against previously unseen attacks. With attackers having access to more resources than ever, custom built malware is becoming more common to circumvent signature-based detection.

Another approach that has recently gained traction in commercial deployment



(a) (b)

Figure 1.1: The left plot (a) depicts simple anomalies that deviate in distance from regular data. The right plot (b) shows a contextual anomaly where a group of data instances do not follow a repeating timely behaviour with respect to all other data points (corresponding to an *Atrial Premature Contraction*, taken from [5])

is based on detecting malware and other undesired activity as anomalous behaviour when compared to benign computer activity. In this approach, known as **network anomaly detection** after Denning [10], models must quantify the behaviour of normal network behaviour to be independent of a narrowly defined notion of malicious activity.

1.2 Anomaly detection

Anomaly detection refers to the problem of identifying data instances that have significantly different properties than previously observed "normal" data instances. Anomaly detection has its origins in statistical research where normal data instances are assumed to be generated as random variables by a probability distribution $P_N(X)$. New data is then identified as anomalous if its properties correspond to regions with vanishing likelihood, i.e. this particular data instance is highly unlikely to be generated by $P_N(X)$. Usually, the hard part in anomaly detection is to use observed data efficiently to build an estimated distribution $\hat{P}_N(X)$ that resembles $P_N(X)$ closely to identify anomalous events while asigning normal instances a non-vanishing likelihood. A variety of techniques exist to achieve this assuming comparably simple generating distributions. However, distributions for many interesting types of normal data can be complex and changing over time, and individual data points can have intricate interdependencies.

Anomaly detection has found wide application in areas where stability of critical systems or detection of their abuse is crucial. Examples of these include engine-fault detection in wind-turbines and fraud detection for credit cards. The assumption here is that the modelled system, such as the sensor-data stream of an engine or the buying behaviour of a customer, remains stable and generates consistent data. Detected anomalies in new observations then indicate a potential fault or abuse in this system. Obviously, it is not inherently clear that every abuse or fault generates data that differs from normal data. Therefore, it is important to choose data sources

that are able to reflect the unique nature of a particular system. .

Anomaly detection and NIDS

Anomaly detection has been applied to network security since the late 90's to assist IDSs. The behaviours of network computers are usually captured in the form of events logs such as network packets or flows, system calls, etc. The basic approach is to use collected training logs to infer a model of trustworthy activity, and then compare newly observed behaviour against this model. To prevent any anomaly model from learning malicious behaviour as normal, the used training data has to be free of any attacks[1].

As an example, an anomally large network flow involving an unusual pair of network computers could indicate that a hacked computer is extracting or sending out sensitive data to an unauthorized destination. An anomaly detection model does not make any assumptions about attack behaviour, but instead marks all deviations from the learned normal behaviour as potentially malicious. It is therefore better suited to find novel and unseen attacks. However, features mined from the data have to be able to reflect normal behaviour in a way that malicious traffic will look different in order to be detectable, which is not trivial. Furthermore, normal instances that only occur rarely or are diffuse will potentially also be flagged as potentially malicious, making an anomaly-based IDS prone to high false alert rates.

1.3 Research opportunities

Detection success and false alert rates of anomaly-based detection methods both depend on the area of application and the chosen trade-off between the two. As pointed out in my literature review, anomaly detection techniques currently perform best in the detection of attacks of larger volume, such as *DoS Attacks* or *Port Scans*, and currently are the most common application in commercial IDS systems such as *Hogzilla* or *CFEngine*. The most influential works can detect attacks in a network with an accuracy approaching 100% while false positives occur on a scale of about once per week or less [14, 12, 4]. In contrast, research on anomaly-based intrusion detection has been less convincing in the context of more targeted attacks that consist of only one or a few connections, with R2L (remote-to-local) and U2R (user-to-root) attacks currently being the least detected attack classes [19]. Furthermore, false positive rates in most proposed methods would generate thousands of false alerts per day when applied on an enterprise level without posterior alert-processing [26, 1, 4], with false alert rates between of 1% upwards not being uncommon.

These differing levels of success can in part be explained by a lack of understanding of how these smaller, more point-like attacks differentiate themselves from normal traffic. Attacks using a larger number of connections will inevitably disturb the distribution of one or multiple measures of the network, such as the byte-throughput or the port entropy [15]. However, an attacking connection inserting malicious code to gain control over a computer does not necessarily have different external properties[2] such as length, size, or port number, than the diverse range of

[1] Again, the assumption is that unauthorized and malicious activity will correspond to behaviour that differs from trustworthy one.

[2] also called *connection summaries*

normal connections. Most attempts to detect these types of attacks via anomaly detection rely exclusively on summary properties of individual connections or packets to draw a border between normal and malicious traffic. Very few approaches try to detect malicious connections as contextual anomalies, i.e. as traffic events that are not necessarily unusual on their own, but deviate from normal traffic in the context of their immediate temporal neighbourhood. Considering both the lack of literature in this area, this project will look more at the intrinsic nature of network traffic which is manifested in the *semantic structure* of packet and connection sequences.

Another pressing issue in anomaly-based intrusion detection is the problem of traffic evolution. An anomaly engine is usually trained once on a training set, and the learned notion of normal behaviour stays constant during the detection phase. There exist several approaches that can be trained in an online fashion, however none address the issue how new training instances are selected. This can lead to both high false alert rates when specific structures in the network change and to malicious traffic entering the updated training set. Sommer and Paxson identify this problem of non-adaptive models as one of the biggest problems in anomaly detection, naming it an area of tremendous research opportunities [26]. Only a few authors since then proposed comprehensive non-stationary models [20, 3, 28], and the proposed methods are specific to certain detection areas and do not solve the general problem of incorporating new data to adjust a model while excluding malicious instances.

More information about about current state-of-the-art methods and identified literature gaps can be found in my literature review in Sections 3 and 4.2.

1.4 Research aim

Chen, Aspinall, Gordon, Sutton, and Muttk [7, 6] from the University of Edinburgh recently demonstrated the effectiveness of state-based software models in the identification of malicious sequences actions in a stream of permission and API calls of Android applications. This usage of "semantic features"[3] allowed the discovery of new malware instances and improved the overall robustness of the classifier drastically.

This project will use the idea of semantic software models and apply it to network traffic data. It combines methods from machine learning and state-based software models, to automatically learn precise semantic models of network traffic generated by one or more machines. The semantic features of network traffic are here understood in terms of sequences of packet or connection metadata that correspond to fixed protocols of information exchange. These models together form a collection of normal semantic network behaviour of a network of machines, with malicious traffic being detected in an anomaly detection manner. Furthermore, a main aspect of this project will be to guarantee adaptivity and robustness of learned models to the evolution of software programs and their corresponding traffic.

One motivation for the use of semantic models in network traffic stems from the fact that current anomaly detection methods perform poorly in the identification of temporally isolated malicious connections, as described above. Attacks containing

[3]These features are based on a program's mathematical semantic meaning, rather than its syntactic structure.

such traffic often break into a machine by exploiting loopholes in the processing of program information, and may in some way break semantic structure of a communication. As an example, a session hijack inserts new packets in a communication and will likely alter the expected exit agreement. Traffic with malicious intent will therefore likely deviate in its intrinsic structure from benign traffic, and should be detectable when modelling this structure using semantic models.

Additionally, we expect that a semantic traffic model will offer more systematic and robust ways to evolve the notion of normal traffic over time. The safe incorporation of new traffic behaviours into an anomaly-based model remains an open research issue that we want to address.

Specific goals of this research project are:

- Build an understanding of how semantic structures manifest themselves in network traffic.
- Develop a semantic model of traffic structures that works on different levels of traffic abstraction (packets or flows) and enables the incorporation of semantic features from the packet level on a flow-level to extend available flow information.
- Introduce robustness to anomaly-based models by identifying gradual changes in network communication as traffic with common semantic substructures.
- Provide a framework to generate traffic data with a high level of ground truth about traffic origins and purposes.

2. Work so far

2.1 Data analysis and strategy

Although this project is motivated by previous successful applications of ML-driven semantic models for anomaly detection, the differences in the type of data used in this project means that different tools and modelling strategies are necessary. Network traffic usually contains more intrinsic variation and noise, with packets potentially being dropped, and data distribution being afffected by the level of network congestion or by varying input parameters. Therefore, significant amount of time has been spent on initial data analysis and reflection on good modelling strategies.

Hypothesis 1 *Application A running on a computer can be seen as a probability distribution $P_A(X)$ generating sequences X of network events, with X behaving as a random variable.*

Hypothesis 2 *Two applications A and B usually have different $P_A(X)$ and $P_B(X)$, which is applies especially to malicious applications. A computer running a set N of applications generates sequences X of network events with a distribution $P_N(X) = \sum_{n \in N} P_n(X)$. Consequently, a sophisticated and well enough trained anomaly model of a machine's traffic distribution $P_N(X)$ should be able to identify the existence of a new (and potentially malicious) application as traffic that did not originate from the set of existing applications N.*

An additional assumption about systems we are looking at is that the set of installed software is relatively constant throughout the network, with new software installations being rather rare and controlled.

Semantic structures and corresponding variations

Semantic behaviour of a program denotes how it transmits and reacts to information. This can be observed both on a packet and on a flow level: Programs follow specific protocols on key-exchanges etc, when a connection is established or how and with what frequency new bulks of data-packets are pushed. They also react to data retrieved in one connection with new connection requests such as establishing a HTTP connection to an address received by an earlier DNS request.

Variation and noise are introduced in several ways:

- Varying sizes of data means a varying number and sizes of packets to be transmitted, which can in turn influence the way the receiver responds. Similarly, requested content can be made up of several files of different sizes which are transmitted in a varying number of connections with differing sizes.

- Encryption can in a similar way introduce variation in packet or connection sizes and numbers.

- Variation in the computational load or on the network on a computer can translate into varying response times to retreived information or how much data is buffered and pushed at once. Packet loss or deviations from the established protocol by one side usually means that information has to be transmitted again.

I spent a significant amount gathering appropriate real-world data sources that can be used to train and to validate the developed methods. Due to its sensitivity, network traffic is not widely available. Furthermore, existing datasets are often subject sampling tools to reduce the amount of data collected, or are modified to preserve the privacy of the corresponding network users. Therefore, several datasets were analysed closely to establish whether they are suitable for both modelling semantic traffic structures and detection of malicious traffic. I emphasised most the completeness of the traffic capture. This include not dropping any packets or flows in a communication, but also that all or almost all traffic from a set of machines is captured. Furthermore, the consistency of the traffic with regular traffic originating from personal computers was another main concern. A summary of suitable datasets can be found in the attached literature review in Section 2.2.

2.2 Ground truth data generation

Building semantic models of network traffic means to build an understanding how different network interactions can be distinguished via their traffic trace. We want to use ML techniques to automatically extract meaningful sets of sequences that represent these different interactions. To ensure that this is actually true, that our extracted models are indeed representing real distinguishable interactions and not just nonsense, we need validation from *ground truth data*.

Network traffic datasets are already hard to obtain due to privacy concerns. However, as the correspondence between individual network traffic events and their particular purpose is virtually never recorded on a computer, very few datasets contain ground truth about the captured events. For that reason, a particular aspect of this project is to generate useful ground truth data with appropriate content myself.

Over the course of the last year, Nikola Pavlov[1] and I developed a framework that generates controlled and isolated network traffic from a variety of applications and tasks. For this, a virtual network was created using the virtualisation program *Docker*. In this network, two or more parties can communicate from containers, which are sandboxes containing programs inside a minimal virtualised operating system. The benefit of this design is that individual containers can only communicate with each other via the virtualised network while the host is in complete control of the parallel execution of tasks in multiple containers. To capture the traffic, every container in the network was complemented with a *tcpdump*[2] container hooked onto

[1] an LFCS summer inter

[2] A common packet capture utility

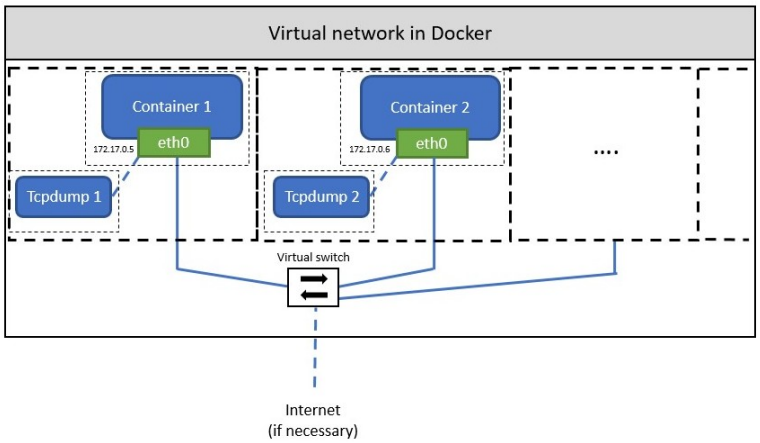


Figure 2.1: Visualisation of the virtual network in Docker

the network interface. The captured traffic can then be labelled according to the particular scenario it was generated by.

We implemented a variety of network service scenarios to capture a diverse set of network traffic. Most, but not all, are set up in a server-client manner. Implemented services so far include:

- A simple *icmp* ping service.
- A *nginx* server hosting a html-webpage with a client accessing it, both encrypted and unencrypted. Different client requests are implemented (*wget* and *Siege*).
- An *Apache* server hosting the same html-webpage with a client accessing it, also both encrypted and unencrypted. Again, different client requests are implemented (*wget* and *Siege*).
- A *Wordpress* server with a corresponding webpage and a client accessing it.
- Different versions of an ftp server and client.
- A mail transfer using *mailx*.
- An *ssh*-server and client.
- An *IRC* chat server and two clients.
- A file synchronisation service with multiple clients.
- A web-crawler gathering a larger volume of traffic from the internet.

For each service, a number of different scenarios are implemented. For example, the *ftp* client could pull, push, or remove one or multiple files (that might not exist), or create a directory. Furthermore, randomisation is introduced on parameters like the file-sizes, request times, etc. in order to explore the dimensional variation of the traffic from individual actions.

To be scientifically consistent, the data generation should obey two basic principles:

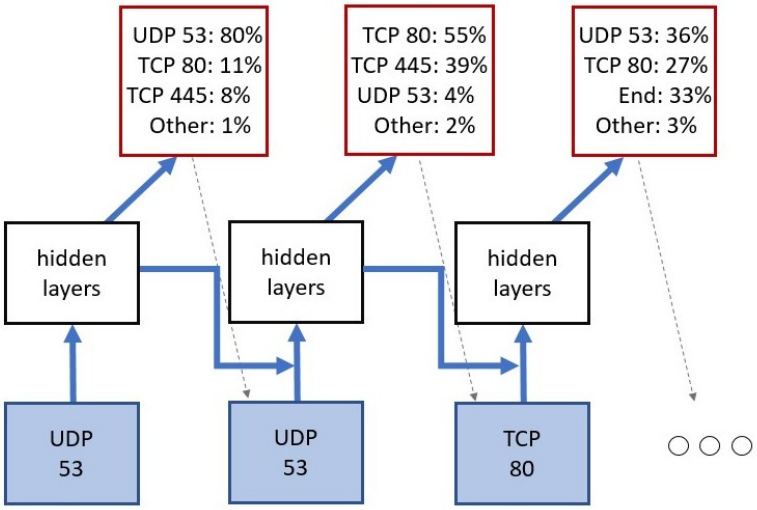


Figure 2.2: Visualisation of the RNN-method used

1. The repeated execution of individual scenarios should provide consistent data.

2. The data should be completely free from external influence, i.e. properties of or activities on the host machine should in no way disturb the data generation.

Both properties were tested for selected scenarios with positive results. For the first, randomisation of the input was avoided as much as possible and the results of several scenario executions were compared. The second was tested by executing scenarios under differing workload and on different machines, and summaries over the different packet properties were tested using a Kolmogorov-Smirnov test.

This work will expanded further during my research project, and I plan to present the framework in a publication.

2.3 Testing initial approaches on realistic data

In the context of the overall research goal of this project, some exploratory work on semantic anomaly detection for network events, conducted before my PhD started, by Marc Sabate, Gudmund Grov, Wei Chen, and David Aspinall. This work uses three different machine learning algorithms[3] from the area of *representation learning* to capture meaningful sequences of *NetFlows* and reflect reccurring patterns in a model. For that, recorded NetFlows are grouped according to the generating host. Furthermore, to filter out sequences of flows that are unrelated to each other, a squence of flows that are close in time is grouped into what is called a "session" as an approximation of the true relation. Each session then serves as a training or test sequence for a behavioural model. Learned semantic behaviour is reflected through the capability of the model to predict traffic protocols and network ports of flows in a session from a smaller subset of flows, with more accurate predictions being rewarded in the training process. Sessions which deviate from previously observed behaviour are then predicted poorly by the model and flagged as potentially malicious.

[3]A Markov Chain model, Finite-State Automata, and a Recurrent Neural Network

The described methods were tested on the CTU-13 dataset [11], a laboratory generated dataset available to study botnet detection. The great benefit of this data is the inclusion of both benign background traffic, on which the models were trained, and malicious traffic used to test their detection capabilities, and the availability of corresponding labels. Furthermore, it consists of more than ten million flows on multiple machines, thus containing much variation for testing purposes.

Anomaly-based methods in intrusion detection are often criticised for working only in controlled environments while failing in the real world. To counter this criticism, it is crucial to provide evidence of success on sufficient data gathered under realistic conditions. For the CTU-13 dataset, the generation of both types of traffic was done in a controlled laboratory environment, making it a so called "synthetic" dataset. This does not necessarily mean that it does not reflect realistic behaviour, however there is no guarantee that it contains all variations encountered in an operational environment.

Toextend the findings of this work, I applied the same methods on currently available real-world datasets which I am already familiar with through my MSc thesis [8]. The particular datasets come from the enterprise network of the *Los Alamos National Laboratory* [13] and from a Spanish ISP [16]. I selected appropriate machines (IP addresses) that contain the variety of traffic behaviour found on personal computers, in contrast to less diverse traffic sources such as IP-phones, printers or servers. Furthermore, the selected machines should be subject to attacks so that the effectiveness of the results can be verified. I made sure the data is in the right format to be passed to the implemented models and supervised the training process by adjusting important parameters and hyper-parameters. Finally, I evaluated the results and added them to the existing manuscript. The results are a promising extension of the existing work, with low false-alert rates for both on benign and attacked machines, and the reliable detection of nine out of 10 attacks conducted on two machines.

2.4 Other activity

A significant part of the first year was spent on reviewing literature. The results were used to produce both a comprehensive literature review and this document.

Besides this, I attended several events that are thematically relevant for my PhD. I attended a one-week data study group at the Alan Turing institute in the context of knowledge exchange and skill development. While there, I contributed in the creation of a comprehensive report on *Developing data science tools for improving enterprise cyber-security*. I also participated in a poster competition at the conference *Big Data in Cyber Security* at Napier University. Additionally, I attended the two-day *Tommy Flowers* conference on "repurposed innovation" in Ipswich, hosted by my PhD-industry sponsor *British Telecom*, and took part in a workshop at Edinburgh University on *Scalable Bayesian Inference*.

3. Thesis Proposal

3.1 Research questions

Section 2.1 described the broader motivation and ambition for this project. To achieve a coherent scientific progress, this PhD-project will explore the problem of modelling the semantic behaviour a computer from different angles, and hopefully find answers to the most important questions that arise in the process. Below I formulated four questions with a number of sub-questions that address different parts of this project. These questions will drive the direction of my research during this project. I do not expect to address all of them comprehensively, but I believe that I will be able to find satisfying answers to the majority. In Chapter 3, I will propose more specific sub-projects that will pave the way to find scientific results for questions in each of the different areas.

Data analysis and exploration

Network traffic is a form of communication between computers, but in our case it is also an observed signal of unobserved computational actions on those computers. This signal does not contain all semantic information of those actions and is subject to further variation steming from network congestion or similar sources. Consequently, it is not clear yet how much structure of the original action can be traced back from the semantic structures observed in the traffic. However, we assume that a reasonable amount of distinct semantic structures will be translated to the corresponding traffic generated by an action. As the space and structure that the data collected from traffic will form is of immense importance for anomaly detection, I wish to answer the following questions:

Research Question 1
How well-structured is the space of semantic behaviours observed in the traffic of a machine or a network structured?

1. *How can we scientifically quantify closeness between individual actions, and does it translate into the similarity of corresponding traffic structures?*

2. *How much does noise or input variation blur the observable semantic differences between clearly distinct actions?*

Modelling scope

Anomaly-based detection systems aim to find deviations from normal behaviour in as many ways as possible. For that, it is desired that any semantic model of network

traffic should represent common network traffic with as much information as possible. However, a too detailed model runs the risk of overfitting the structure found in the training data and consequently yielding high false-alert rates. Therefore, we need to address the following questions:

Research Question 2
To what degree can semantic structure in network traffic be captured in a model from a training dataset, and how can we achieve this?

1. *Which parts of network traffic both contain important information and can be represented by a model?*

2. *Can be combine models that act at different traffic levels to enhance the amount of context given by the data?*

3. *Can we efficiently disentangle overlaying network flows to isolate otherwise distorted flow groups corresponding to similar actions?*

4. *What is an efficient and realistic way to incorporate other data sources into the modelling procedure? How can this input enhance the learning process and the representation detail of a traffic model?*

5. *How can a model adapt to changes of normal semantic structures?*

In order to capture the diversity of network traffic, large quantities of training data will need to be processed. The choice of modelling methods will therefore depend on several aspects such as computational efficiency, retraining capabilities, and interpretability.

Ground truth data and evaluation of semantic understanding

Identified semantic structures in network traffic should represent semantic behaviour of programs on a computer to be meaningful and applicable for the detection of malicious network behaviour. A ground truth about the program or service that generated particular network events is practically non-existent in available datasets. In order to evaluate the representation of traffic structures a given model provides in a more comprehensive way than via the detection rate of known malware instances in a dataset, I want to answer the following question:

Research Question 3
What is a meaningful representation of traffic structures?

1. *What requirements must a labelled traffic generation framework fulfill to provide realistic data?*

2. *Can we evaluate the traffic representation of a model through its ability to identify semantic closeness between traffic instances correctly?*

3. *How can we quantify the capability of a given traffic model to identify new computational actions on a machine?*

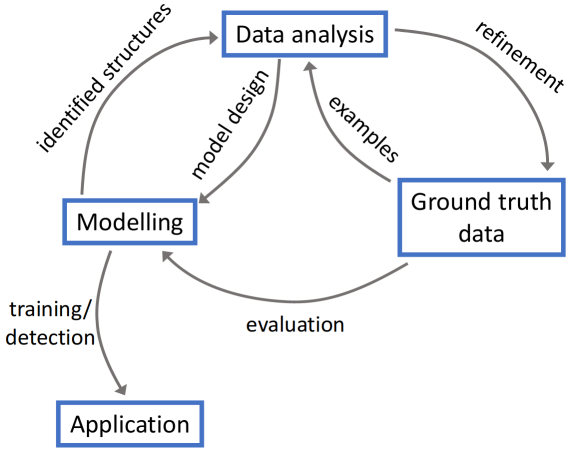


Figure 3.1: The formulated research questions are strongly interconnected and are mutually dependent in the answering process.

Application to realistic Intrusion Detection

Finally, this work should make an impact on future intrusion detection methods. Although anomaly-based methods are meant to be agnostic of a notion of malicious activity, we have to ask how and where this work will offer improvements in the detection of modern attacks. In order to effectively increase detection rates, semantic anomalies have to occur within attacks that are not reliably detected yet and are not trivial for an adversary to disguise or hide.

Research Question 4
What will a semantic model be able to prevent?

1. *What kind of attacks will necessarily show semantic anomalies, and which will not?*

2. *Can an adversary adapt his attacks to avoid detection? How can we prevent this?*

Challenges

The quest to answer these questions is coupled with a variety of challenges that come with both the data and the chosen method of detection. These demand careful consideration of the best steps to be taken and make this project a interesting, non-trivial, and demanding task.

Network traffic, both in the context of packets and network flows, is a complex and usually severly unclean source of data. Several very diverse aspects, from network topology to software malfunction, can introduce faults or disturbances in the data on different levels. Additionally, large quantities of network traffic have to be processed for an effective model building, meaning that data processing and modelling has to be both robust and efficient.

Secondly, the problem of extracting the desired structures from traffic event sequences can benefit from successful machine learning applications in *Natural Language Processing* (NLP. However, methods from NLP have rarely been applied to

network traffic before, and the unique features of network packets along with very different sources of noise and the interdependence of flow- and packet-level events mean that any modelling algorithm have to be specifically taylored and trained to be successful.

Another concern is interpretation and evaluation of model results along with the embedding of those results in an effective anomaly-based detection framework. I already mentioned this problem directly in the above given research questions, specifically question 3.2 and 3.3. The difficulty here lies in the fact that semantic structures in contrast to specific properties are hard to quantify, and the creation of labels or other validation tools requires a careful design and creativity. The unsupervised and anomaly-based setting furthermore complicates this problem as accurate measures of the semantic structure are needed to identify anomalous events without corresponding data to design this measure on.

The last big challenge that I see in this project is the training and adaptivity of methods in order to capture sufficient normal behaviour and reduce false alerts. This is a problem inherent in anomaly-based intrusion detection as diversity and evolvement of network traffic make it hard to fully capture all normal behaviour. Making a system adaptive while still retaining effective detection capabilities is a non-trivial task that will require both creativity and rigorous testing for any meaningful advancement.

3.2 Learning representations of packet exchanges

The distinct semantic behaviour of an applications manifests itself most clearly in the first few packets exchanged in a connection. This is the stage in a connection during which a specific client or server request is transmitted, encryption keys are exchanged, users authenticate themselves with passwords, ports for successive connections are defined, etc. This distinction goes far beyond individual traffic protocols, as even a single application usually can exert different actions which translate to different semantic behaviours in this initial negotiation phase.

The significance of the initial negitiation phase is bolstered by two publications on the area of traffic classification that share the same conclusion and train clusters and classifiers on the metadata of the first five to ten packets to distinguish different applications [2, 9].

Learning precise semantic representations of this initial negotiation phase from applications running on a computer has several main benefits for this project:

- Several types of attacks exploit loopholes accessible at the negotiation stage. A precise semantic representation model could detect anomalous deviations in particular packet parameters from the learned behaviour. A simple example of such a deviation from expected behaviour would occur during a *buffer overflow attack*, which overflows an input buffer (such as for a password) to directly modify memory locations. This should be observable as a significant deviation in the corresponding packet size.

- Newly installed malware will most likely exhibit different than normal behaviour in this stage, from very different negotiation procedures to just slight deviations in the packet interarrivals.

(a) (b)

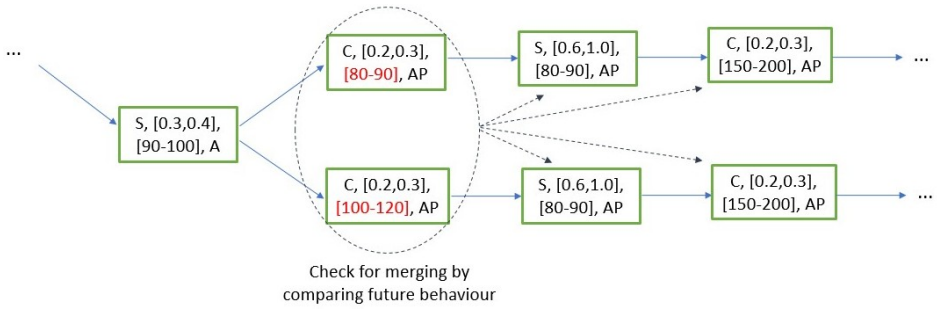
Figure 3.2: Depiction of user-ID and password exchange in an FTP-connection (a) and an SSH-connection (b) during the negotiation phase.

- The translation of semantic negotiation structures into different behaviour groups could be used to associate the corresponding flow with particular actions, i.e. to give the flow one or more labels. To be meaningful, the identified groups have to correspond to actual semantic behaviours and should be consistent along similar actions. Such a flow labelling could be very helpful for understanding semantic structures of flow traffic, and consequently for anomaly detection on a higher level, such as the described work by Chen et al.[6]. I will describe this more in Section 3.4.

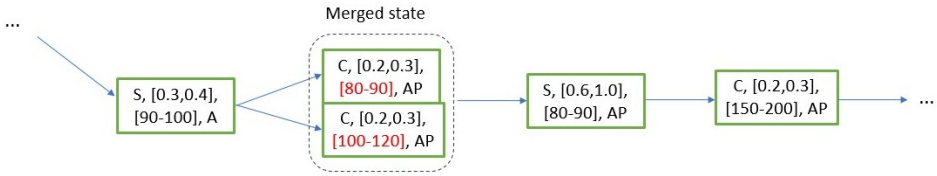
- As applications are updated, their particular traffic generation distribution $P(X)$ can also gradually change, causing corresponding traffic to be potentially flagged as anomalous by current anomaly detection methods. A representational model could identify common semantic substructures between traffic instances and thus indicate that they originate from an updated application. More detail will be given in Section 3.3.

Focusing exclusively on this initial phase, i.e. a fixed number of the first packets in a connection, has several benefits: We are dealing with a fixed and comparibly small input size, which lower computational cost and is beneficial for modelling techniques with a higher complexity. Our model is tainted by large data exchanges, which usually do not contain as much useful information but are spread across many more packets, which could create a high imbalance in the traffic representation. And we do not have to wait until a connection is finished for its evaluation.

A model that offers enough representation of semantic structure has to go further than the above mentioned work, which was intended for traffic classification instead of cyber-security. Therefore, I propose two unsupervised modelling approaches that are intended to encapsulate significantly more information.



(a)



(b)

Figure 3.3: Depiction of the state-merging process for two input sequences

3.2.1 Probabilistic Real-Time Automata

The *non-deterministic finite-state automaton (NFA)* is a widely used model for discrete event systems and can be seen as a mathematical model of computation. It is a type of Markov chain with a finite number of states and a set of Markovian transition probabilities from each state to the others. A recent variant of the of the NFA is the *probabilistic real-time automaton (PRTA)*, which models the event interarrival times as a discrete distribution that is dependent on the current state.

Automata are learned from a set of input sequences via state-merging. In this procedure, a trie with all observed event sequences is built, with each path in the trie representing one input sequence. Pairs of paths that are close to each other are then merged, thus generalising over the input data and learning the structure of the target machine. Closeness is determined by heuristic measures, with a merge considered good if the future behaviour after reaching state p in one path is similar to that after reaching q. Figure 3.3 depicts this process.

As the exchange of network packets is a symbolic sequences of a computational process, they are a very suitable candidate to be modelled by probabilistic automata: The tree-branching in an automata is well suited to represent the different avenues a negotiation can take while the probabilistic state-merging ensures that potential variation within one avenue are accounted for. With enough traning data, automata can therefore provide a much more refined representation of a given set of packet sequences than previously explored techniques without overfitting. Anomaly detection to identify potentially malicious traffic can then be done by computing the likelihood of a input sequence trajectory and flagging it if it false a likelihood-ratio test.

Using probabilistic automata has further benefits in the context of this project. In contrast to many methods, they are interpretable by domain experts when check-

ing, which is a priority for many researchers in order to process the inevitably[1] high occurrencies of false alerts. Furthermore, the branching structure in automata provides a natural grouping of sequences, which can be used to engineer labels for similar sequences in a similar manner to the *hierarchichal clustering method*. Such labels can be very useful in future ambitions discussed in Section 3.4.

PRTAs have been applied to network traffic before by Pellegrino et al. [21], however in a different context and scale. To apply automata to this project, network packets would have to be transformed to discrete, finite states. Interesting properties to be included are

1. the direction of the packet,
2. TCP-flags,
3. the interarrival time,
4. the size of the payload,
5. and potentially the offered window size.

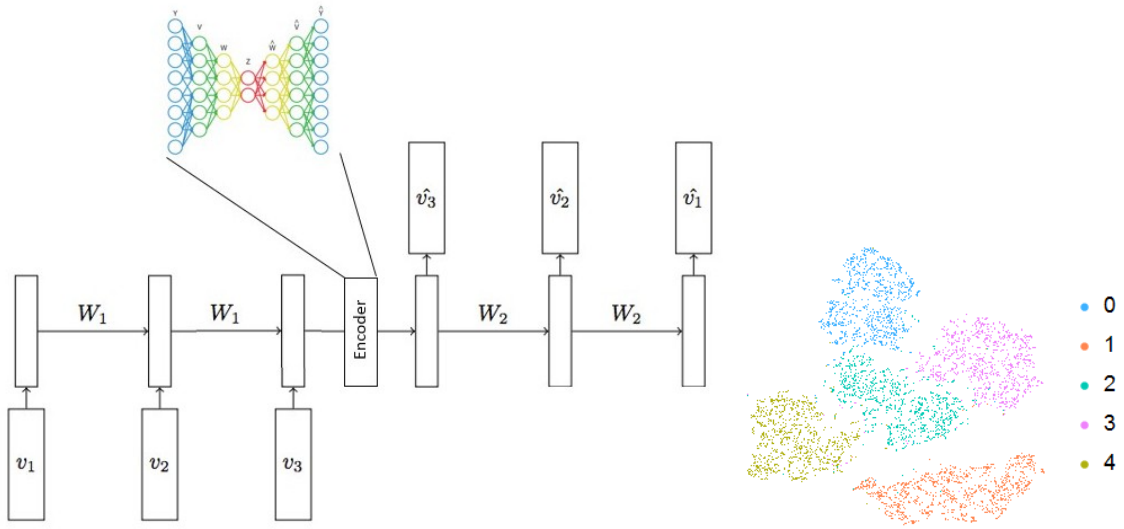
To create states, continouos variables have to be transformed into a set of finite non-overlapping intervals. For a better resolution, a rescaling onto a logarithmic scale is very sensible. The combination of all attributes into one set of states is then straightforward.

One challenge in this project is the large amount of training data needed to represent the variability of network traffic. Schmidt and Kramer [22] have recently proposed an online learning approach for PRTAs that enables the effective processing of massive datasets, which could be very useful for this project. A further thing to consider is that the chosen similarity measure for state-merging should be able to account for additional *ack*-packets inserted in the connection or potential packet loss. A possible solution is to introduce non-vanishing probabilities for both scenarios, i.e. it should be possible from every state to jump to an empty *ack*-packet and back, or to skip a packet. Another solution which could also reduce the potential computational burden is to learn multiple automatas from the input using rolling windows over, with each automata learning a specific interval of the negotiation phase, in a similar manner as in [21]. This could increase the stability of individual automata and decrease the size of the trie. Furthermore, as each automata would correspond to sub-behaviours in the connection, it would be possible to identify common subsequences in traffic from updated software and thus create a relation to previous traffic.

3.2.2 Autoencoder-based

Autoencoders are the most popular deep learning-based method for unsupervised representation learning. They can learn a compressed representation of input data by self-supervised training using the reconstruction error of the input data. Although autoencoders themselves do not produce a defined model of the input data, they translate complex structures into a more simple space in which they can be described by conventional density estimation methods.

[1] due to the inbalance between benign but irregular traffic compared to actual attacks



(a) (b)

Figure 3.4: Visualisation of the design of an LSTM Autoencoder (a), taken from https://machinelearningmastery.com/lstm-autoencoders/, and the embedding of different types of handwritten digits by an autoencoder.

Long short-term memory (LSTM) autoencoders are a recent version of autoencoders that use the structure of RNNs to capture the sequential nature of the input data before the encoding-process, first introduced by Srivastava et al. [27]. Since then, they have been successfully used for artificial creation of facial videos, music generation, or for text compression. LSTM autoencoders can be trained using backpropagation, and the application of an LSTM autoencoder to packet data is straightforward, however again time and size variables should be logarithmically rescaled. Anomaly detection can then either be performed by thresholding the reconstruction error of new input sequences [17].

LSTM autoencoders are an excellent alternative to PRTAs for the modelling of packet data due to their superb ability to discover non-linearities in sequential data, and the fact that they can be trained on massive datasets. They have been used to discover semantic structures before both for natural languages [29, 25] and computational sequences [30]. Furthermore, the embedding of input data into a lower dimensional continuous space means that clustering of similar traffic instances to create higher level flow labels can be performed, see Figure 3.4b.

3.2.3 Data and result validation

The above described methods should be tested for three properties:

1. Goodness of fit,
2. detection capabilities of malicious behaviour,
3. correspondance of engineered high-level labels to actual semantic behaviour.

Testing of the first two properties is straight-forward given a suitable dataset. Goodness of fit could be tested with any packet data collected from a suitable

environment and could even be generated for the experiment. To test detection capabilities, a packet dataset containing labelled malware traffic is necessary. Two widely used datasets, the CICIDS 2017 and the UNSW-NB 2015 dataset are suitable candidates [18, 23].

The third property is not as straight-forward to test, as hard ground truth about the purpose of individual connections is required to analyse the labelling consistency. Fortunately, such ground truth data is now available to us for a variety of services by our data generation tool, described in Section 2.2. To test the labelling consistency appropriately, a given dataset could be superimposed with labelled ground truth traffic, and split into training and test data. If individual scenarios in the test set are both identified consistently while associated with a different label than significantly different scenarios, the labelling can be seen as meaningful.

3.3 Software evolution and drift

Networks and their corresponding traffic are not static, but gradually change over time. Consequently, making intrusion detection systems adaptive to such changes in normal behaviour is an increasingly important issue in the research community [26, 20].

In our context, such changes in normal behaviour would appear in the observed semantic structures and could be induced by new software being installed on the observed machine, or by installed software changing through updates or similar modifications. As new software could equally be of malicious nature, there is no way to make our system robust against such changes without introducing additional notions of overall malicious software.[2]

Software updates however should correspond to more gradual changes as only small parts of the software are modified. Hence, it is reasonable to assume that in the process only parts of the semantic traffic structure will change, while larger parts surrounding it stay the same. A logical approach to distinguish this gradual software evolution from more abrupt changes would therefore rely on the comparison of semantic sub-features to detect an overall closeness between traffic instances. However, more analysis of traffic evolution under software updates has to be conducted to make reliable statements about precise modelling methods. Yet, a few ideas are presented here.

Chen et al. [6, 7] introduced the concept of common sub-automata between similar malware instances in order to improve the robustness of detection methods. This notion could be used in this project in a similar way: Traffic instances deviating from the learned traffic automata are examined for sub-automata that are within the learned range of normal behaviour, depicted in Figure 3.5. If enough common sub-automata exist, the traffic is marked a potential software update and either authorised by an administrator or directly incorporated into the existing model.

Depending on the exact implementation of methods proposed in Section 3.2.1 and 3.2.2, there are several ways to design such a comparison. In case a rolling window based approach to learn traffic automata is chosen, a comparison between automata from different time-intervals is straightforward, with heuristic measure to decide on an overall closeness to be chosen. If instead complete traffic automata

[2] which is why we are assuming that new software is not installed on a regular basis

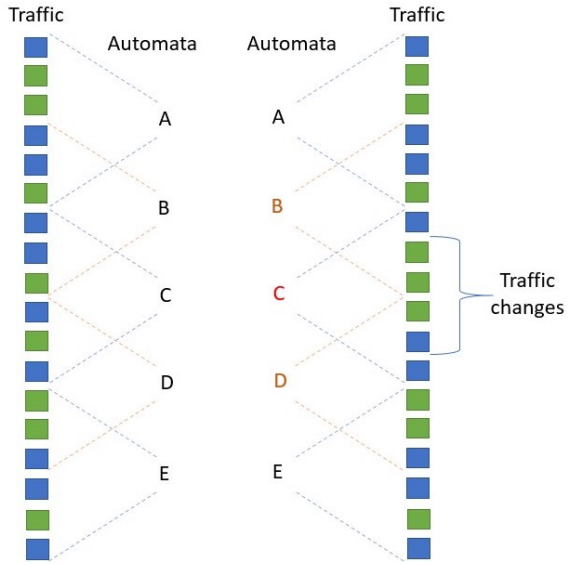


Figure 3.5: Depiction the comparison of sub-structures between two traffic instances.

are learned, Chen et al. proposition for a machine-learning-centred algorithm to efficiently choose salient sub-automata could be used. For an autoencoder-based approach, successive traffic sub-intervals can be mapped into lower-dimensions, with their distance to the normal region indicating if each interval corresponds to normal behaviour or not.

3.3.1 Evaluation

In order to gather data and to test developed methods, I will again make use of our ground-truth traffic generation framework. For that, containers with different versions of the same software will be installed and the corresponding traffic will be gathered. Next to initial data analysis, this traffic can then be injected into real-world datasets in a similar fashion as described in Section 3.2.3, with traffic from older versions being injected into the training data while that from newer versions into the test data. Such a dataset can then be used as a benchmark to test the ability of the above described methods to identify traffic from newer software version as such.

The data produced this way does not necessarily represent the influence of software updates on network traffic as a whole. However, it can give good indications of the amount of change introduced and is a good testing possibility for our methods in a very novel area of cyber-security until more extensive datasets become available.

3.3.2 Adaptive learning

Despite the identification of similar traffic instances, their incorporation into the existing model without introducing concept drifts or catastrophic interference is an important task. Online or incremental learning methods exist for both autoencoders and PRTA and could be used with new traffic gathered over time. However, special attention has to be paid to the fact that once traffic from updated software is identified, it should be included into the model immediately to avoid a flood of

traffic alerts from the same software. How this this specific problem can be solved will depend on specific models and is yet to be determined.

3.4 Flow-level based modelling

In Section 2.3, initial work by Aspinall et al. on modelling semantic traffic structures on a flow level is described. This work demonstrates that semantic structures in a machine's network traffic can be observed on a flow level, and how these can be used to identify malicious behaviour as deviations from normal behaviour.

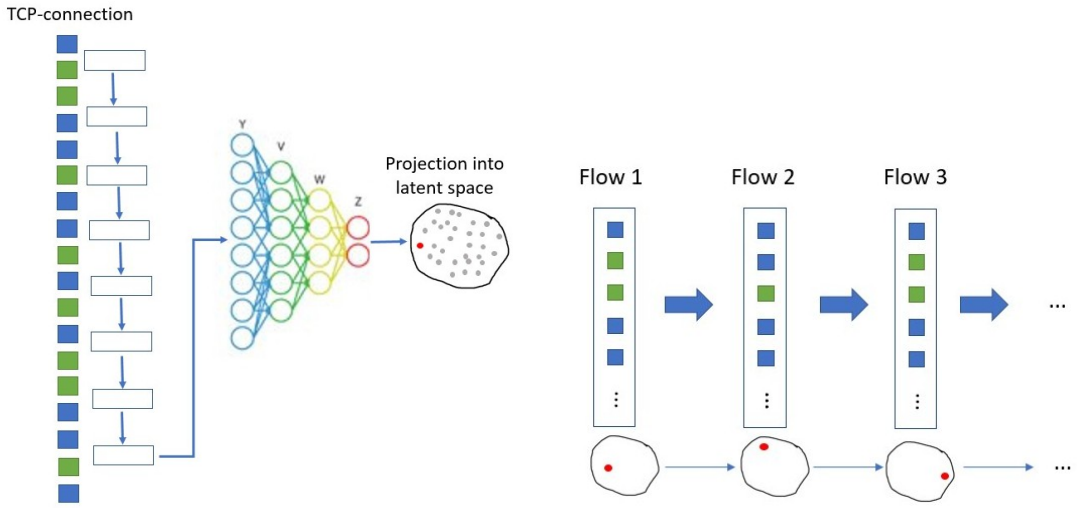
As this work is an initial step towards a flow-based semantic models, there is room for further developments and improvements which are congruent with the general aim of this PhD-project, in particular with . I therefore propose the following measures to create a more detailed and precise traffic model:

3.4.1 Combination of packet and flow-level information

The current traffic models by Aspinall et al. only incorporate protocol and port numbers of successive flows as semantic features. Network flows additionally include the IP addresses of contacted machines, the direction of the connection, and strongly correlated information about the duration, size, and packet number of a flow. Network flows additionally include the IP addresses of contacted machines, the direction of the connection, and strongly correlated information about the duration, size, and packet number of a flow. Both the direction and the duration or size of a flow are readily available quantities that could help identify more detailed substructures. However, all these parameters describe a flow on a quite high level of abstraction, with a lot more information about the type of traffic not being captured. This information gap is widely recognised in traffic classification research, with a number of approaches generating additional connection information such as interarrival or packet size statistics. Including such information would be a fairly straightforward choice to improve existing methods. I however am more interested in examining how semantic information about a connection could be used to give sequences on a flow level more context.

This task is in direct correspondance to the posed research question 2.2: "How can we combine models at different traffic levels to enhance the amount of context given by the data?" This question is very important in regards to extending a systems capability to detect anomalies effectively. Events in a network can be disturbed by many factors, and the space of events accepted has to included many rare, albeit possible events as normal. This could for example mean that a TCP hijacking attack is for a model indistinguishable from a connection malfunction. However, such an event at packet level will likely trigger some effect on the flow level, like the opening of a similar follow-up connection etc, which might not necessarily be the same if the event was malicious. If the context of a malfunction or similar event inside a connection would be available to models on the flow level, their expected effect could be compared to the observed one to receive a greater level of detection capabilities.

How this combination of semantic packet and flow level information would be implemented will depend on the outcome of the project described in Section 3.2. A



(a) (b)

Figure 3.6: Visualisation of the use of extracted semantic information from the packet level (a) to be used in modelling sequences of flows (b).

fully connected model that learns information structures at both levels simultaneously would be desired as learning and updating would be more natural and cause less problems in the information linking. This is however not the only possible way to proceed as a stacked combination of two models where training is performed independently could be an easier solution to implement and test.

In Section 3.2, I described how a model of packet exchanges can be used to automatically create connection labels for different groups of semantic behaviour, both using PRTA and autoencoders. These labels would describe a greater subdivision of traffic types in similar way to common network ports which already give a broad indication about the type of traffic. As the currently used methods already work with port numbers, it would be straightforward to extend them on these artificial flow labels.

3.4.2 Traffic separation

The described current models are representing semantic structures as the prediction accuracy of the next element in a sequence, as depicted in Figure 2.2. This probabilistic method is quite effective when semantic sequences are isolated, even under noise, as can be observed in current language models [24]. This is however not always the case in a stream of network flows as two or more independent actions generating traffic can have temporal overlap, causing their corresponding semantic structures to overlay each other. Subsequent separation of the different signals is a hard problem and at this stage likely infeasible. Since these overlays can occur between completely independent actions, it is hard for our current models to accurately predict the next event in an overlayed session. Although these traffic overlays appear to not be the norm in a machine's network stream, the decrease in prediction certainty in a session depletes the possibility to detect anomalous behaviour and thus opens a door a door for adversaries to hide attacks in normal traffic.

The problem of separating traffic from different services itself is computationally not trivial, and there may not exist complete solution. It is still however still worth

to consider possible ways to isolate significant traffic instances, or how to at least improve robustness of existing solutions. The traffic generation framework described in Section 2.2 can be very beneficial for this taks as we can generated overlayed traffic with the ground truth labels of the different traffic signals.

Training a classifier

One possible idea I would like to explore is to generate a larger quantity of overlayed traffic sessions with corresponding signal labels and a realistic level of variation to train a classifier. This classifier should try to put flow events into groups according to their generation source based on group properties observed in their IP-addresses, sizes, etc. As labels will be provided, this task can be done in a supervised manner. The learned distinction distinction properties should be general enough to be applied to different traffic tasks.

In order to support the learning process, data from other sources such as application logs or operating system calls could and should be used. More information on how these could be generated can be found in Section 3.5.

If the classification of network events into groups is too hard, the classifier could instead be trained to just identify if a session contains overlayed traffic or is generated by one source, which could inform a detection system of the ambiguity of the observed session.

How exactly an appropriate classifier could be designed is still to be determined.

Robustification

If the separation of signals turns out to be too difficult, it is still possible to improve the existing RNN model and increase prediction accuracy of overlayed events in a session:

- Instead of predicting one successive event in a sequence, prediction accuracies of an RNN can be improved by extending the prediction to multiple unordered events that are supposed to occur within a window. This will shift the focus from one event which is relatively unpredictible under overlay to a larger group in which particular events are relatively certain to occur. This way, a model can learn semantics easier and can provide a higher certainty under overlays.

- For longer sessions, a rolling window approach should be introduced. This way, hidden neurons can be reset easier and adjust to the current behaviours in a session. This is helpful when one behaviour is directly followed by a different independent one.

3.5 Traffic generation and data fusion

To build meaningful semantic models, it is essential to compare discovered traffic features with the corresponding actions the traffic represents, for which ground truth traffic is necessary. Our data generation framework described in Section 2.2 is to our knowledge the first of its kind in the way that every generated packet or connection can be linked to a specific application and the specific task that

application completed. This framework might not only be beneficial fo us, but also for other researchers interested in data with a high level of explanatory labelling.

In order to harness the unique opportunities this framework is offering on ground truth data generation, I hope to extend it in the following ways:

Simulate real network traffic distributions

As many network intrusion detection methods are designed for application to enterprise network traffic from personal computers, it is reasonable to evaluate them on traffic that has similar characteristics. As network traffic[3] consists of traffic originating from a set of applications performing different tasks, we can try to simulate network traffic on a machine by generating sequentially traffic instances from each scenario from a probability distribution specific to that scenario, and pooling it into one traffic stream. For that, the following questions are interesting to us:

1. What kind of applications and services are typically responsible for the generation of typical enterprise network traffic on individual machines?
2. Into which generalised sub-tasks can these services be broken down?
3. How can we quantify the frequency of each application generating traffic?
4. What other underlying traffic properties such as service correlations or network congestion did we not capture?

Data fusion

Apart from network traffic, applications often produce additional events captured in log files. These are usually intended to control whether a service is working properly, but are also increasingly used for cyber-security purposes. The fusion of multiple events streams such as log files and network traffic is a very promising direction in current intrusion detection as the combination of multiple indicators for malicious activity has the potential to significantly decrease the rate false positives, and some effort has been made to provide synchronised data sources, notably from the Los Alamos National Laboratory [13]. However, it is traditionally very hard to link specific log events with corresponding network traffic events due to the lack of ground truth traffic. Due to the separation of traffic from individual applications and programmable execution of different scenarios, our framework can provide the unique opportunity to examine and model this relationship.

[3] at least on an IP level

4. Programme of work and potential risks

Section 3 presented a number of different sub-tasks this PhD-project will be split into. I will now give an outline of my workplan on this series of interdependent work packages, which will be carried out in parallel. The outlined plan of work is relatively conservative to accomodate enough time for different tasks. I expect that several projects will be completed sooner and that additional room for further work on the research questions in Section 3.1 will be conducted.

First year

Much of the work conducted during the first year is described in Section 2. As those task are in priniciple completed, I also anticipate to start the work on learning representations of packet exchanges, as described in Section 3.2, in month 10 or 11 of my first year.

Second year

The first few months of the second year will be dedicated to my objectives on learning packet communication representations. The work will essentially consist of implementing, training, and evaluating the proposed methods, but I anticipate that several adjustement steps on the design will be necessary once initial evaluations are available. I anticipate that this project will be concluded after four months.

Upon conclusion of this project, significant time will be spent on the work on detection adaptivity to software evolution and semantic substructures, as described in Section 3.3. This work will directly be tied to the results of the previous task. Implementation of specific ideas will be quite straightforward and not as time-consuming, but developed methods again will almost certainly be adjusted in several steps to an optimal state. The specific time allocation to this task will depend on how much data will be available for the evaluation, and how much work can consequently be done on the testing of online learning methods.

In order to provide the necessary data for the intended task, the data generation tool will be extended throughout the second year. Depending on potential collaboration opportunities, concepts for data fusion might be implemented already in the second year.

I believe that substantial progress will be made and that both of the main tasks will be concluded by the end of the second year.

Third year

Time allocation in the third year will depend on the progress achieved in year two, but as for now I anticipate that both the extension and publication of our traffic generation framework, and the combination of flow-based and packet-based methods with the results from year two will be the focus of year three.

As more applications will be added to our *Docker* environment during year two, the third year is a good time to develop automatised execution scripts that are able to emulate normal network traffic distributions with extensively labelled traffic from real applications. As this traffic generation framework should be available for other researchers to use, the framework has to be tested for errors, reliability, and documentation. In addition to that, other datasets have to be analysed to infer statistical traffic distributions.

The other main project in year three will be the work on overlaying traffic instances and correspondingly on the incorporation of other data sources. I outlined two approaches how to address this problem in Section 3.4.2. These should be relatively quickly explored, however it is hard to anticipate how much success they will provide, and whether further work has to be done. The time allocation on this task can be adjusted according to progress on other intended tasks.

Fourth year

The remaining six months of year four will be used to write up the project thesis.

4.1 Risks

Because this project is novel and ambitious, the exact time necessary for different tasks is difficult to anticipate, and it is possible that some planned goals will not be achievable. In this proposal, I therefore did not propose a detailed, tightly-organised schedule, but a broad workplan with the expectation to dynamically reallocate time spent on different tasks unitl the faced difficulties and opportunities are clearer. Yet, time is obviously a constraint to this project that can potentially limit the results acquired in the process.

One potential risk to keep in mind is the scalability of estimated models. The proposed algorithms themselves are scalable to process massive amounts of data. However, the data we are dealing with is not only large in size, but is also very diverse and potentially contains a large number of distinct sub-behaviours, and it is not clear yet if this diversity will be immediately reflected by the chosen models. This does not mean that the set goals are unachievable as models can be enlarged and traffic can be subdivided to train multiple models, but it can potentially slow my progress down.

Another aspect to keep in mind is the overall scarcity of appropriate attack data in intrusion detection datasets. As our methods are directed to more specific attack types, it will be helpful to test their detection capabilities on corresponding attack data to demonstrate their effectiveness. There exist datasets with some appropriate attack data, and my supervisor and I are talking to other potential data providers at the moment, however it is not completely clear how much high quality attack data we will have access to.

Further concerns regard both computational and storage resources. This project will deal with large data quantities, which will both require sufficient computational power and storage capacitites that we currently do not have access to. It should be far from impossible to acquire access to these resources in the future, however it might be possible that additional funding has to be acquired to pay for the access.

Bibliography

[1] M. Ahmed, A. N. Mahmood, and J. Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[2] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, 2006.

[3] D. A. Bodenham and N. M. Adams. Adaptive Change Detection for Relay-Like Behaviour. In *2014 IEEE Joint Intelligence and Security Informatics Conference*, pages 252–255, Sept. 2014. 00008.

[4] A. L. Buczak and E. Guven. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, 2016. 00294.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[6] W. Chen, D. Aspinall, A. D. Gordon, C. Sutton, and I. Muttik. More semantics more robust: Improving android malware classifiers. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 147–158. ACM, 2016.

[7] W. Chen, D. Aspinall, A. D. Gordon, C. Sutton, and I. Muttik. On Robust Malware Classifiers by Verifying Unwanted Behaviours. In E. brahm and M. Huisman, editors, *Integrated Formal Methods*, volume 9681, pages 326–341. Springer International Publishing, Cham, 2016.

[8] H. Clausen, M. Briers, and N. M. Adams. Bayesian activity modelling for network flow data. *Data Science For Cyber-security*, 3:55, 2018.

[9] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic classification through simple statistical fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37(1):5–16, 2007.

[10] D. E. Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987.

[11] S. Garcia, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

[12] D. Jiang, Z. Xu, P. Zhang, and T. Zhu. A transform domain-based anomaly detection approach to network-wide traffic. *Journal of Network and Computer Applications*, 40:292–306, 2014.

[13] A. D. Kent. Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory, 2015.

[14] A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-wide Traffic Anomalies. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '04, pages 219–230, New York, NY, USA, 2004. ACM. 01230.

[15] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM, 2005.

[16] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón. Ugr 16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers & Security*, 73:411–424, 2018.

[17] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[18] N. Moustafa and J. Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, Nov. 2015. 00074.

[19] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys & Tutorials*, 2018.

[20] J. Noble and N. Adams. Real-Time Dynamic Network Anomaly Detection. *IEEE Intelligent Systems*, 33(2):5–18, Mar. 2018.

[21] G. Pellegrino, Q. Lin, C. Hammerschmidt, and S. Verwer. Learning behavioral fingerprints from netflows using timed automata. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 308–316. IEEE, 2017.

[22] J. Schmidt and S. Kramer. Online induction of probabilistic real-time automata. *Journal of Computer Science and Technology*, 29(3):345–360, 2014.

[23] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani. Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2018(1):177–200, 2018.

[24] Y. Shi, P. Wiggers, and C. M. Jonker. Towards recurrent neural networks language models with linguistic and contextual features. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[25] C. Silberer and M. Lapata. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 721–732, 2014.

[26] R. Sommer and V. Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, May 2010.

[27] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

[28] T. Wüchner, M. Ochoa, and A. Pretschner. Robust and effective malware detection through quantitative data flow graph metrics. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 98–118. Springer, 2015.

[29] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139*, 2017.

[30] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula. Autoencoder-based feature learning for cyber security applications. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3854–3861. IEEE, 2017.

Anomaly Detection in Computer Networks - Literature Review

Henry Clausen

November 28, 2018

1. Overview

1.1 Network attacks

Sophisticated data breaches affect hundreds of million customers and inflicts tremendous financial, reputational, and logistic damage. One reason for the recent rise of cyber crime is the increased use of sophisticated techniques for the attack of specific targets. Attackers use customised social engineering and custom-build malware that penetrate defensive barriers and potentially stay undetected in an infected system for an extended period of time.

In 1980, James P. Anderson, a member of the *Defense Science Board Task Force on Computer Security* at the U.S. Air Force, published the first report to introduce the notion of automated intrusion detection [4]. In it, he defines an **intrusion attempt** or a threat as

> *"... an unauthorized and deliberate attempt to access or manipulate information, or to render a system unreliable or unusable."*

Such attacks can be very diverse in their nature: They can be used to achieve different goals, and correspondingly exploit different types of tools and vulnerabilities. Very often, intrusive attacks involve some sort network communication between the victim machine(s) and a malicious agent. As this work is focusing primaly on network intrusion and corresponding defense systems, we will take a closer look at this type of communication. A recent survey covering intrusive attacks and defense systems distinguishes five classes of malicious network traffic [62]:

1. *DoS-attacks*: A denial-of-service attack is an attempt to remove ability of a particular computer to communicate with other machines over an extended period of time. Such attacks are usually targeted at network servers in order to disrupt the service it is providing. All major types of DoS-attacks achieve this by overwhelming the target server with service requests, which are usually corrupted in a way that causes the server to bind resources unnessecarily long for each request, and thus losing its capability to process other requests. The most prominent type of DoS attacks are SYN-floods. They exploit the TCP-handshake protocol by sending many SYN-requests to a server while ignoring the SYN-ACK response packets sent in return by the server. This causes the server to keep waiting for a response for each of the attacker's requests, and thus binds the resources of the server while being computationally very cheap for the attacker. After a certain treshold, the server will not be able to process any more requests, rendering it unusable for actual client requests.

2. *Network probing/Reconnaissance attack*: The purpose of network probing attacks is to gather information about computers in a network and possibly find vulnerabilities which can be exploited in further attacks. This typically involves sending specific service requests to other computers in the network in order to gather information about this system, such as open ports or the operating system running on a machine, contained in the corresponding response packets.

   A common type of network probing attacks is *port scanning*. Its aim is to gather knowledge of computers in the network than run vulnerable services, such as HTTP servers, mail servers, and so on. A port scan achieves this by sending queries to one or more network ports on one or more computers in the network. A computer on which the contacted network port is open will respond to the query and thus reveils himself. A port scan can either be vertical, during many ports on one computer are scanned, or horizontal, where the attacker scans a small number of ports on many computers in the network.

   Network probing is often an integral part in the spreading mechanism of*computer worms*.

3. *Access Attacks*: These are attacks that attempt to gain unauthorized access to a machine. This could both be an individual from outside gaining access to the network, or a user from inside the network accessing services or privileges outside of their authority. Access attacks are often divided into *Remote-to-Local* (R2L) where a remote attacker gains access on a system over the network, and *User-to-Root* (U2R), where a user illegally gains administrator access to a machine. However, many attacks fall into both categories.

   Access attacks can be very diverse in their nature. A simple example are brute-force attacks where an attacker guesses the passwort of a user over a network service such as SSH. Other prominent and more sophisticated cases include *SQL injections*, where nefarious SQL statements are passed to an entry field for execution, or *buffer overflow*, in which more data is put into a buffer of a service than it can hold in order manipulate data in the memory past the buffer.

4. *Data Manipulation Attack*: Also known as "man-in-the-middle", these attacks typically involve an attacker reading and manipulating information in a data stream that is not addressed to him by exploiting vulnerable or missing authentication mechanisms in the IP protocol and related applications. A common form of such an exploit is *IP spoofing* where an attacker pretends to be a trusted computer by sending packets with a spoofed trusted source IP address. Similarly, vulnerabilities in digital certificates that serve as a unique identifier of a trusted computer can be used to create fake certificate and thus trick a victim into trusting the intruder carrying the faked certificate.

   Two examples of data manipulation attacks are *session replay* and *website impersonation*. In a session replay, the intruder captures packet sequences exchanged between two parties and modifies part of the data before forwarding it to the receiver. Here, both parties are unaware of the data manipulation and trust the authenticity of the connection. In website impersionation, a user is unknowingly redirected to a perfect copy of the website he requested. The user is then tricked to enter confidential information into a web form, which is then sent to the attacker instead of the trusted party operating the original page.

Data manipulation attacks are often used pass malicious code to a victim in order to gain access on its machine. An impressive example of such behaviour was demonstrated by the malware *Flame*[1]: An infected host in a network sends messages to other machines running Windows advertising itself as a Windows update provider, using spoofed IP addresses and a fake Microsoft certificate and thus defeating Microsoft's authentication mechanism. Other computers were consequently tricked into receiving malicious updates from the infected host, which would then infect their machine [81].

5. *C&C traffic*:

   C& stands for *"Command and Control"* and denotes the communication between an infected host and a rogue agent, called the C&C server. The data transmitted in a C&C channel is usually exfiltrated information about the environment of the infected host, or commands from the C&C server for the victim for further operations. Typically, C&C communication is used for the control of one or more so called *bots*, computers that can perform tasks such as network scanning, establishing connections to other machines, or participating in DoS attacks. The communication between a bot and the C&C server is therefore extended and continuous over time. However, C&C communication can also be used for the request and transmission of an encryption key needed in a ransomware attack, in which it is limited in time and size.

   C&C communication is usually sent over the HTTP or HTTPS protocol as it is widely available and allows the attacker to hide their communication in the large volume of diverse traffic sent over this channel [46].

An additional class of networking threats is the *unauthorized surveillance* of network traffic. A local network of computers is usually separated from the outside, with a router establishing the connection between computers in the network and ones outside the network. The traffic exchanged between machines inside the network therefore does not leave the network and is not visible for outsiders. Unauthorised access captures of internal network traffic can give an intruder significant information about the network topology, and even access to sensitive information if a connection is not encrypted. As network surveillance usually does not leave any visible traces in the network, I did not include it in the above listed types of malicious traffic.

1.2 Intrusion Detection Systems

The field of intrusion detection is concerned with the development of methods and tools that identify and locate possible intrusions in a computer network. An *intrusion detection system* (IDS) is typically a device or software application that detects malicious activity or policy violations in an automated way by scanning incoming data collected from one or more sources for patterns that a model or a set of rules classifies as malicious.

Intrusion detection is a well researched area, with the first IDSs emerging in the late 1980's. Intrusion detection today comprises a variety of research areas in terms of different types of data sources, system architectures, detection sope, and so forth. Figure 1.2 provides a broad yet uncomplete overview of these different areas.

[1] also known as *Skywiper*

Figure 1.1: A broad overview over different aspects in an IDS

Implementation

Denning [19] in 1987 established the notion of two different types IDS implementations: a) host-based and b) network-based. *Host-based intrusion detection systems* (HIDS) monitor each host in a network individually, and rely on local data streams such as application logs or raw operating system calls as data source. A relatively new form of host-based data sources are biometrik user data such as keystrokes or eye movement, please see [70] for more information.

Network-based intrusion detection refers to the detection of malicious traffic in a network of computers. A network intrusion detection system (NIDS) monitors network traffic within in a network and/or between the network and external hosts for malicious activity or policy violations. Network traffic data can take different forms, a more detailled explanation will be provided in Section 2.1.

Host-based systems have the advantage of working with high quality data that are

typically very informative [48]. NIDS have the advantage of being plattform independent and more resiliant to attacks as detection of an infection is not done on the infected system. In this review, I will focus on work done in the area of network intrusion detection.

Architecture

The architecture of an IDS can affect its overall performance, and is therefore an important design aspect in the operational deployment of IDSs. The decentralised topology of computer networks usually means that data is collected from multiple sensors or hosts across the network. The processing of the collecting can be done in three different ways:

Centralised IDSs gather all collected data to a central processing unit, where the analysis of the collected data and the detection take place. This unit then has complete information about the network, however the lack of scalability makes these systems less common.

Decentralised systems follow a hierarchical structure. Sensors sent data to the closest processing unit, which pre-processes the data before sending it to the main processing unit. This way, workload is reduced at the central unit, and the scalability problems can be avoided, but not fully overcome.

Distributed IDS consist of multiple autonomous units that process incoming data. Processing and evaluation for malicious activity are done independently from a main unit. Individual units can communicate with each other to identify network-wide anomalies.

Detection methods

Detection methods are the core of an IDS, and are therefore the most important design choice. Traditionally, two broad types of detection approaches are identified: a) Misuse detection and b) anomaly detection.

Misuse detection aims at detecting a particular and well known reoccurring characteristic or pattern of a malicious behaviour. Two simple examples of such a characteristic are the large number of SYN packets sent by a host in a DoS attack, and the synchronised connection of many hosts to one server in a botnet. In misuse detection, abnormal or malicious behaviour is therefore defined first before developing a model to distinguish the defined behaviour from other traffic.

In contrast, anomaly detection aims at building a model of normal system behaviour that is accurate enough to spot any malicious behaviour as traffic that deviates from the estimated model. Anomaly detection is principally more difficult than misuse detection since the traffic model has to incorporate potentially very heterogenuous traffic behaviours. However, it is generally acknowledged that anomaly detectionhas is more suitable to detect new and previously unseen malicious behaviour as it makes no definite assumptions on the anomalous behaviour. Misuse detection is robust against evolution of malware as long as defined malicious behaviours do not change.

In reality, anomaly and misuse detection are not necessarily mutually exclusive, and there is a fluent passage between the two. This is because many anomaly detection approaches choose a particular set of features to be modelled with a particular threat in mind. For instance, models for the number of connections of a machine are naturally suitable for detecting DoS attacks, port scans, or Worm attacks.

As misuse detection detection methods are aimed at detecting very specific behaviour, they usually only detect one type of malicious traffic. Areas that have been researched particularly well include botnet C&C channel detection, DoS attack detection, and port scan detection. Areas that lack a comprehensive body of research are different types of R2L and U2R attacks. These are also currently the least detected attack classes [62].

1.3 Scope of this work

As my PhD-project is aimed at developing more general methods that are capable of detecting new types malicious behaviour, this review will focus mainly on anomaly detection methods. Specifically, I will look at **anomaly-based network intrusion detection**. I will however cover some areas of misuse detection if the employed methods have a potential applications to anomaly detection methods.

I will identify different classes of feature generation from network traffic and examine the benefits and drawbacks of each class. I will then proceed to review significant methods in each of the identified classes. Additionally, I will examine existing network traffic datasets that are suitable for testing anomy-based methods, and I will also give a brief overview of existing surveys and taxonomies. I will then conclude my results and briefly discuss implications for my PhD-project.

2. Data

2.1 Network traffic

Computers in a network mostly communicate with each other by sending *network packets* to each other, in which the transmitted information is encapsulated. Each network packet is split into the control information, also called packet header, and the user information, called payload. The packet header contains the necessary information for the correct transmission of the packet, including the transmission protocol layer (such as TCP, UDP, or ICMP), and the source and destination IP addresses and network ports[1]. It furthermore contains error-checking fields suchs as the size of the packet and a checksum over the payload, and protocol-specific fields.

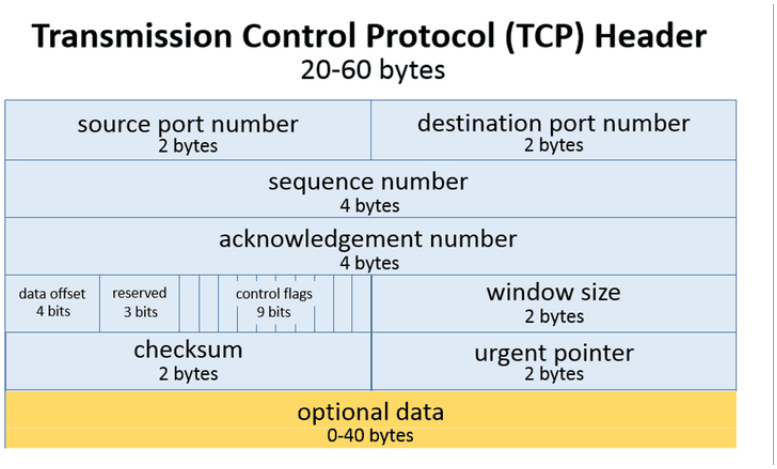


Figure 2.1: Typical format of a TCP packet. Source: https://www.lifewire.com/tcp-headers-and-udp-headers-explained-817970

The payload of a packet is in general the data that is carried on behalf of an application. It can be of variable length, however, it cannot exceed the maximum packet length set by the network protocol. In contrast to the packet header, the payload can be encrypted, a technique that makes it unreadable to any third parties and that is becoming increasingly common in modern computer networks.

While traversing between to parties, a network packet can pass multiple connecting devices which direct the packet in the right direction. Any device in the immediate circuit traversed by a packet can capture and store it. In a monitoring setting, packets are usually captured by network routers and stored in the widespread *pcap* format. In

[1] A network port is a number that identifies which service or application is responsible for the processing of incoming packets.

case of space shortage or privacy concerns, the payload of a packet can be dropped in the saving process.

Another more structured way of capturing network traffic is based on connection summaries, or **network flows**. RFC 3697 [10] defines a network flow as a sequence of packets that share the same source and destination IP address, IP protocol, and for TCP and UDP connections the same source and destination port. A network flow is usually saved containing these informations along with the start and duration of the connection and the number of packets and bytes transferred in the connection.

```
Date flow start        Duration Proto  Src IP Addr:Port      Dst IP Addr:Port   Packets  Bytes Flows
2010-09-01 00:00:00.459   0.000 UDP    127.0.0.1:24920   ->  192.168.0.1:22126      1     46     1
2010-09-01 00:00:00.363   0.000 UDP    192.168.0.1:22126 ->  127.0.0.1:24920        1     80     1
```

Figure 2.2: A typical network flow output

Raw packets grant full information about the connection, but take a lot of space when stored, whereas network flows give a more structured and lightweight overview over the traffic in a network. Both data formats are used widely by NIDSs.

2.2 Existing datasets

In order to evaluate their ability to model the behaviour of a network and to identify malicious activity and network intrusions, new methodologies have to be tested using existing datasets of network traffic. This network should ideally contain realistic and representative benign network traffic as well as a variety of different network intrusions. However, as network traffic contains a vast amount of information about a network and its users, it is notoriously difficult to release a comprehensive dataset without infringing the privacy rights of the network users. Furthermore, the identification of malicious traffic in network traces is not straightforward and often requires a significant amount of manual labelling work. For that reason, only a handfull of datasets for network intrusion datasets containing real world traffic exist. There have also been some efforts to artificially creating such datasets and thus bypassing any privacy concerns. However, up to today, no artificial dataset truly resembles real network traffic in every aspect [62].

As described in a recent survey by **Ahmed et al.** [3], we can generally distinguish four different types of datassets containing network traffic:

1. **Real network data containing known intrusions**:
2. **Real network data containing injected intrusions**:
3. **Real network data containing no intrusions/untruthed real network data**:
4. **Synthetic network data with/without injected intrusions**:

I will now describe the properties of existing datasets suitable for network intrusion detection. As this review is primarily concerned with anomaly detection models that require benign network traffic, I did not include datasets such as honeypots that mostly contain malicious traffic. A description that includes such datasets can be found in the below described survey by Ahmed et al. [3].

Los Alamos National Laboratory, 2015 - Comprehensive, Multi-Source Cyber-Security Events [38][37]

In 2015, the Los Alamos National Laboratory (LANL) released a large dataset containing **network flow** traffic from teir corporate computer network, which contains about 17600 computers. The data was gathered over a period of 58 days with about 600 million events per day. The data only contains internal network connections, i.e. no flows going to or coming from computers outside the network are included. IPs and ports were de-identified (with the exception of the most prominent port), but are consistent throughout the data. Since the data stems exclusively from one corporate network, it can be assumed that it shows more homogeneity in the observed traffic patterns than general network traffic.

Additionally, the dataset also contains other event sources which were recorded in parallel in order to give a more comprehensive look at the network, and could be very useful when investigating a detection approach that correlates multiple event sources. These sources include process events and authentication events from Windows-based computers and servers, and DNS lookup events from the DNS servers within the network.

The dataset furthermore contains a labeled set of redteam events which should resemble intrusions. However, these events are not part of the network flow data and only contain information about the time of the attack and the attacked computer. These events apparently resemble remote *access attacks*, are not described further and appear to be artificial or injected into the dataset. It is thus not certain how well they resemble actual network intrusions.

LANL released another dataset containing network flow traffic from their network in 2017 [83]. This dataset is similar to the one from 2015, but spans over a longer period of time, 90 days. Furthermore, it contains no labeled malicious activity, however that does not mean that the data is completely free of malicious activity.

CTU 2013 [1, 23]

The *Stratosphere Laboratory* in Prague released this dataset in 2013 to study botnet detection. It consists of more than 10 million labeled **network flows** captured on lab machines for 13 different botnet attack scenarios. Additionally, the raw packets for the botnet activity is also available for attack analysis.

The labelling in this dataset is different from other datasets as each flow in the list is labeled based on the source IP address. In the experiments, certain hosts are infected with a botnet and any traffic arising from such a host is labeled as Botnet traffic. Traffic from uninfected hosts is labeled as Normal. All other traffic is Background, as one cannot classify it.

A criticism of this dataset is the unrealistically high amount of malicious traffic contained in the dataset, which makes it easier to spot it while reducing false positives. Furthermore, the way normal or background traffic is generated is described only poorly and leaves the question how representative it is of actual network traffic.

UGR 2016 [51]

The UGR'16 dataset was released by the University of Grenada and contains **network flow**[2] data from a spanish 3-tier ISP. This ISP is a cloud service provider to a number of companies, and thus the data comes from a much less structured network than the LANL data. It contains both client's access to the Internet and traffic from servers hosting a number of services. The data therefore contains a very wide variety of traffic patterns, an advantage emphasised by the authors. IP-adresses are consistently anonymised while network ports are unchanged. However, it is not ensured that the traffic capture is complete, i.e. that all traffic coming from and going to a particular machine is captured.

A main focus in the creation of the data was the consideration of long-term traffic evolution and observable periodicity in order to enable the testing of so called *cyclostationary* traffic models. The dataset correspondingly covers a very long period, spanning from March to August of 2016, and containing about 14 GB of traffic per week.

The data is split into a training set and a test set, with the latter containing labeled attack data. This attack data does not stem from rogue agents but is in part generated in controlled attacks on victim machines, and in part injected from previously observed malware infections. The attack data is therefore does not truly correspond to actual attacks, but achieves a high degree of similarity. The implemented attacks contain:

- DoS attacks (controlled attacks),
- Port scanning (controlled attacks),
- C&C traffic from a botnet (injected).

The authors also acknowledge that the background traffic is not necessarily free from further attacks. In fact, three real attacks have been observed and labeled, corresponding to IP-scanning and a spam mail campaign.

UNSW-NB 2015 [58]

The dataset realeased by the *University of New South Wales* in 2015 contains real background traffic and synthetic attack traffic collected at the "Cyber Range Lab of the Australian Centre for Cyber Security". The data is collected from a small number of computers which generate real background traffic, and is overlayed with attack traffic using the *IXIA PerfectStorm tool*. The time span of the collection is in total 31 hours.

An advantage of the collected dataset is the inclusion of both **raw packets** and **network flows** along with two other data formats containing newly engineered features. This allows a more detailed analysis of the data and possibly a better distinction between attack and benign traffic. In total, the data contains 260 000 events.

Another advantage of the data is the variety of attack data, containing a number of DoS, reconnaissance, and access attacks. However, due to the synthetical injection of these attacks, it is unclear how close they are to real-world attack scenarios.

Since this dataset is collected from a relatively small number of machines and during a limited period of time, it is furthermore unclear how suitable for capturing both the temporal evolution and the heterogeneity of real background traffic.

[2] netflow v9

CICIDS 2017 [25][76]

This dataset, released by the *Canadian Institute for Cybersecurity* (CIC), contains 5 days of network traffic from 12 computers. These computers all have either different different operating systems such as Windows, OSX, or Ubuntu, or different versions of the same operating system in order to enable a wider range of attack scenarios. The network furthermore contains switches, routers, a web server, a modem, and a firewall in order to ensure a realistic network topology. The traffic data itself consists of **labeled benign and attack traffic**, and is available as 11 GB per day of **raw packets** with payloads, or as **network flows**.

It was ensured that the data contains all traffic coming and going from individual machines. However, in contrast to other datasets, the background traffic is not directly generated through user interactions on the machine, but by using a method to profile abstract user behaviour in different traffic protocol. The purpose of this is to make the traffic more heterogenuous and to ensure that different types of behaviour are present in the data during the comparably short time span. This However, it is not completely clear how much of the underlying structure of real traffic is lost in the process, and therefore how suitable this data is to build models of benign user activity.

The attack data of this dataset is one of the most diverse among NID datasets, as it contains a variety of up-to-date attacks, such as different types of DoS attacks, SQL-injections and Heart-bleed attack, network scanning, or botnet activity. These are not always successful in order to reflect actual attack scenarios. However, the authors did not describe very well how the data from these attacks is generated and combined with the background traffic as it is also processed through a form of profiling engine.

The CIC released another very similar dataset to this one in 2012.

DARPA 1998 [50]

The *Defense Advanced Research Projects Agency* released the first major dataset to test network intrusion detection systems. The data stems from two experiments at the *MIT Lincoln Laboratory* were multiple victim hosts running Unix and Windows NT were subject of over 200 attacks of 58 different types. The data spans three weeks of training and two weeks of testing data and contains *raw packets* that are labeled. It was since then heavily used as a benchmark to test new detection methods.

Also due to its prominence, it was heavily scrutinised and received a lot of criticism for its lack of realistic background traffic, which was generated through simulation procedure, and the presence of artifacts from these simulations in the data that could heavily skew any model relying on benign traffic. Also, the high percentage of attack traffic in the data is described as unrealistic.

Furthermore, since the dataset is now more than 20 years old, it is remarked that both the benign and attack traffic does not resemble modern network traffic anymore.

KDD Cup 1999 [17, 18]/NSL-KDD 2012 [80]

The *MIT Lincoln Laboratory* created this dataset in 1999 by processing portions of the 1998 DARPA dataset with new labels for a competition at the conference on *Knowledge Discovery and Data Mining*, and is the most widely used dataset in intrusion detection. It contains 2 million connections summaries in a new format and in total 38 attack types. This new format is essentially a form of **network flows** with a greatly increased number

of features, 46 in total, which give additional details about the origin of the connection. The availability of these features in a real-world application however is in my opinion unrealistic as most of them could be mined due to the availability of parallel surveilance of the host, a Solaris-based system. However, we can see significant differences in today's operating systems which barely resemble Solaris. In addition, parallel host system surveilance usually cannot be taken for granted in a realistic network environment. Naturally, as the KDD'99 data stems directly from the DARPA dataset, it also faces the same problems and criticism.

The *Canadian Institute for Cybersecurity* postprocessed the KDD'99 data in order to address some of its shortcomings. This includes removing redundant records, balancing the size of the training and test data, and adjusting the proportion of attack traffic in the data. However, the biggest criticism from the KDD'99 and the DARPA data, the unrealistic generation of background data, still prevails.

LBNL 2013 [66]

This dataset released by the *Lawrence Berkeley National Laboratory* in 2005 is the first one to examine internal network traffic inside a modern enterprise. It contains more than 100 hours of *packet headers* from several thousand internal hosts.

This dataset contains no known attack traffic, and is therefore only suitable for traffic analysis and model fitting analysis. Furthermore, as being the first dataset containing enterprise traffic, privacy concerns caused the authors to remove any possibilities to identify individual IP addresses.

In 2011, Saad et al. [74] combined this dataset with existing botnet traffic to create a dataset containing both benign and attack traffic.

UNIBS 2009[84]

This dataset was collected on the campus network of the *University of Brescia* on three consecutive days in 2009. The dataset contains in total 79000 anonymised TCP and UDP *network flows*.

This dataset is not directed towards intrusion detection research, but was made as *ground truth data* for traffic classification. It therefore contains labels which indicate which of in total six applications generated the corresponding traffic flow. It might however still be of interest for model assessment in intrusion detection that is relying on traffic classification.

CAIDA 2016 [86]

The *Center for Applied Internet Data Analysis* started collecting network traces from a high-speed backbone link in 2008 with the collection still ongoing. The data is available in anonymised yearly datasets containing one hour of **packet headers** for each month.

Since the traffic is collected from a backbone link, it is very unstructured and heterogenuous. It is furthermore not necessarily free from attack traffic. Although this dataset has been used for intrusion detection before, it is more suitable for general internet traffic analysis.

MAWI 2000 [78]

Similarly to the CAIDA dataset, this dataset contains **packet headers** from the WIDE backbone. It is therefore similarly unstructured, anonymised, and not free from attack traffic. Since this dataset was already collected and released in 2000, it can also be remarked that the contained traffic is too old to represent modern traffic.

ADFA 2013/2014 [14, 15]

The ADFA datasets, released by the *University of New South Wales*, focuses on attack scenarios on Linux and Windows systems as well as **stealth attacks**. To create host targets, the authors installed web servers and database servers , which were then subject to a number of attacks.

The dataset contains both attack traffic and benign traffic. However, the dataset is directed more towards attack scenario analysis and is criticised as being unsuitable for intrusion detection due to its lack of traffic diversity. Furthermore, the attack traffic is not well separated from the normal one.

ICT datasets [2]

The *Impact Cyber Trust* releases cyber security oriented data. Its repository includes many datasets, synthetic as well as real captures, from different sources. Many datasets focus on observed attack data and thus are not directly applicable to intrusion detection. Furthermore, there is in general very little information provided that describes a dataset's origin, which makes it hard to investigate the network topology.

Among the more useful datasets are the *USC datasets*[3], which contain network traffic (both **packet headers** and **network flows**) from academic networks in the US between 2008 and 2010. The datasets are very large, with the largest one covering 48 hours and containing 357 GB of packet headers.

[3]DS-062, LANDER Data, and DS-266

3. Methods and Systems for Network Anomaly Detection

3.1 Feature generation

Anomaly-based intrusion detection moved into the focus of researchers at the end of the 1990s, with many advances and new ideas being implemented between 1998 and 2005. Anomaly detection methods often rely heavily on tools from the area of machine learning and statistics in order to generate quantitative models for normal traffic from large amounts of data. A challenge for researchers is that network traffic data, both in the form of packet headers or network flows, is a mix of continuous and categorical variables, with the latter often having an immense number of categories. Furthermore, the data has a temporal aspect, which is however only incorporated by a few authors.

Many existing surveys [3, 11, 9] distinguish anomaly-based approaches directly according to the area the method is borrowed from, e.g. into statistical, supervised, unsupervised methods and so forth. However, this neglects an aspect that should be the primary decision factor for which method or model to use: Feature engineering. As network traffic is itself not a data format, it is necessary to generate meaningful features from it. The type of features is crucial for accurateness of traffic representation and ultimately for the types of malicious activity that can be discovered.

While surveying existing literature, I identified four different classes of detection methods, depending on how the authors incorporate traffic features:

- Traffic aggregation-based methods,
- event-based methods,
- and methods that incorporate the temporal structure of network events.

Iwill now explain the nature of the different classes and analyse anomaly-based methods in each class. It is important to mention that I neglected methods that are based on **payload inspection**, which form themselves another class of detection methods. However, payload-based methods are dependent on unencrypted traffic and are often seen as to computationally expensive for a successfull deployment in real networks. As encryption becomes more and more the standard in the way we communicate over the internet, I will not discuss payload-based methods in this work and instead generally assume that no unencrypted traffic is available for my PhD-project.

3.2 Approaches based on Volume or Traffic Aggregation

Traffic aggregation is a simple way of creating a contextual format of network traffic in contrast to isolated events. Usually, all events in a certain time interval are binned and summary statistics over the different traffic parameters are calculated, such as the sum of events of bytes transferred, or the number of distinct IP addresses/ports communicating. Events in each time interval therefore stand in a contextual relationship as they together influence the different summary statistics. Aggregation of traffic can be done for individual hosts as well as for groups or entire networks, while the computational load depends on the choice of summary statistics.

Methods relying on traffic aggregation are most suitable for attacks in which events form a collective instead of a point anomaly. Examples of attacks forming collective anomalies are DoS and port scanning attacks as events like a half-open connection are not anomalous themselves, but are anomalous when occurring in larger numbers. Attacks that contain less volume however are less likely to be detected in aggregated traffic as individual features of the events would have to be large or unusual enough to influence the summary statistic of the entire group.

3.2.1 Subspace projection/PCA-based

Principal Component Analysis is a statistical form of *orthogonal coordinate transformation* to convert a set of observations (or feature vectors) into a set of linearly independent variables[1]. These variables uncorrelated variables each account for differing amounts of the variation contained in the data. By projecting an observation only onto the components that account for the most variation, it is possible to retain most of the information while operating in much lower dimensions.

Lakhina et al. [44, 43] introduced a PCA-based anomaly detection method for network traffic in 2004. In their approach, they aggregated the network flows for each OD[2] pair into 5-minute bins, with the number of transferred bytes, packets, and flows being the features for each bin. Each 120 consecutive bins were then treated as a an observation (with $3 \cdot 120$ variables), and PCA was then applied to the collection of observations. The first 5 principal components are then identified as the dominant temporal patterns. Anomalies were then identified as observations that could only very poorly reconstructed using the first 5 principle components. Since then, this approach has been adopted to several other datasets without much methodological advances. Camacho et al. [12] proposed an improvement to the existing PCA-based approach with a more natural implementation of spotting anomalies.

This approach can be applied to individual OD pairs, or on a network-wide basis by using q-statistics to spot multivariate anomalies. The approach was tested on data from the Abilene backbone network, and worked well to identify significant episodes such as DoS attacks, fast spreading worms and other large-scale scanning activity, alpha-flows, or power outages.

Naturally, since the traffic is aggregated into bins and the temporal behaviour of these bins are examined, this approach is aimed towards identifying attacks with a comparably large volume of traffic, even if they are isolated in time. It is however

[1] the *principal components*

[2] Origin-Destination

unlikely that it is capable of spotting smaller U2R and R2L attacks or C&C traffic. Another possible criticism is that anomalies are not spotted immediately, but in the worst case after hours.

Ringberg and Rexford [73] provided a discussion of PCA-based approaches to traffic anomaly detection. They concluded that PCA is very sensitive to small differences in the number of used principal components and to the level of aggregation of the traffic measurements. Furthermore, the training data has to be absolutely free of any traffic anomalies, otherwise the projection onto the first principle components can change drastically.

3.2.2 Entropy-based

The entropy is a measure for the degree of disorder a system is in. Applied to network traffic, a popular quantity to measure is the dispersion of events onto different source or destination IP addresses. A high entropy would correspond to all events being evenly distributed among the all existing IPs while a low entropy corresponds to the majority of events being concentrated between a small number of IPs. Entropy-based approaches are usually emplyed more in misuse detection, but can also have reasonable applications in anomaly detection.

Wagner and Plattner [85] measured the entropy of network flow distribution across source and destination IP addresses and across source and destination ports on a network-wide basis. The entropy is measured in a sliding window of 5-minutes with 1-minute shifts and monitored continuously. Anomalies are flagged as sudden changes in one or more of the mentioned entropy sources with tresholds that were determined from empirical judgement. The described measures were applied to network data from a swiss internet backbone which contained data from two large-scale worm outbreaks[3]. The characteristics of these worms in terms of entropy changes[4] were then analysed as an evaluation of the technique.

Lakhina et al. [45] used entropy in a similar, albeit more sophisticated way to detect anomalies. Instead of using entropy on a network-wide basis, it used to monitor src and dst IP and port distributions on individual hosts over time. The obtained values are then bundled in a three-way matrix $H(t, p, k)$ where t is the current time, p is the particular host, and k is one of the four monitored traffic features. This data matrix is then converted into a two-way matrix and similar to other work by Lakhina, a PCA-like subspace projection is applied to mine temporal features as orthogonal components. However, here these components reflect correlations in simultaneous entropy changes on different hosts and features. Anomalies are again detected as poor reconstructions by via the most dominant components, and the performance is examined using untruthed data from the Abilene backbone network. Another clever addition of this paper is the use of unsupervised learning to identify different types of observed anomalies. This is done by applying hierarchichal clustering with a fixed number of clusters to the residual vector of H. However, contains some obvious flaws that in my opinion prevent a generalised grouping of anomalies.

[3] *Blaster worm* and *Witty worm*, both more than 50 000 infections

[4] Source IP and destination port entropy decreases drastically, destination IP entropy increases moderately

3.2.3 Wavelet-based

Wavelet modelling is a frequency-based signal processing approach. Amplitudes of most signals can be described as a finite sum of wavelets with different frequencies. These frequency coefficients can then be used as a measure to describe the signal's generalised behaviour, and to compare with future data from the same signal. Three significant papers applying wavelet modelling to intrusion detection can be found:

Both **Barford et al.** [7], and **Thottan and Ji** [82] introduced wavelets to network anomaly detection in 2002/2003. Both approaches are fairly simple, as they only look at the network flow numbers from multiple machines at different points in the network, aggregated into 5-minute intervals. Using enough anomaly-free training data[5], this volume signal can be described by a set of frequency-components. Barford et al. then compare the frequencies of any future traffic episodes against this set, and marked as anomalous if a treshold is exceed. The approach is directed towards detecting network-wide flash crowds, DoS attacks, and outages, which is evaluated using proprietary data. Thottan and Ji detected anomalies by looking at the reconstruction error of such traffic episodes using the estimated frequency-components instead comparing different estimates. The reconstruction error is assumed to follow a gaussian distribution, and anomalies can be detected using a hypothesis test.

Jian et al. [35] proposed a refined wavelet-based model in 2014 which is applied to individual OD pairs. All observed OD pairs in the network are grouped into q[6] groups. To each group, an S-transformation (a modified version of a wavelet-transformion) is applied. The signal for each OD pair is then reconstructed using only the estimates of the high-frequency components since these correspond to any bursty behaviour. Now, the reconstructed signal is free from any slow variation and contains only fast and bursty behaviour. The assumption of the authors this degenerated signal must be heavily correlated between individual OD pairs. Using a sliding window, the pairwise correlations of each OD pair is computed. If the correlation for any pair falls below a certain treshold, this pair is marked as an anomaly. The approach is designed to detect volume-intensive attacks on busy servers, the exact motivation for this approach however is not described very well by the authors. The evaluation is done using data from the *Abilene backbone*.

It is clear that any approach that looks exclusively at the traffic volume either between individual hosts or in a network-wide fashion will only detect attacks with sufficient attack volume, such as DoS attacks. Additionally, wavelet-based approaches do not provide a probabilistic framework for anomaly detection, which makes the seperation of true anomalies from false positives difficult, especially in larger networks.

3.2.4 Other

Kind et al. [39] proposed a network-wide detection approach that is based on traffic histograms and clustering to identify and model substructures in normal traffic for better anomaly detection. A number of different traffic quantities are divided into a fixed number of subgroups, such intervals of IP-addresses or network ports, bins of packet sizes or connection durations, or the different TCP flags in a connection. For every different feature, authors create histograms measuring the number of events

[5]It is crucial that this data is absolutely free of anomalies that are to be detected

[6]number depends on network topology

occuring in each subgroup during a 5-minute interval. Histograms are monitored over a period of time without any malicious activity in order to gather a collection of training histograms. After removing subgroups that remain, each traffic feature is then divided into clusters using the Mahalanobis-distance measure as a similarity metric between histograms and either hierarchical or k-means clustering. Anomalies in individual traffic features can then be detected as histograms with distances from the nearest cluster that exceed a certain treshold. Evaluation is conducted with an unnamed dataset containing 15 labeled attack, containing DoS attacks, worm propagation and network scanning, and network-wide system fingerprinting, and mail bombs, of which 13 were detected. Unsurprisingly, the two undetected attacks addressed fewer machines and thus consisted of less traffic. In general, this approach indivates a great improvement from the previous approaches as it is the first significant attempt at discovering substructures in aggregated traffic, which generally grants a better detection of non-trivial anomalies. An interesting improvement would be to correlate simultaneous outliers in different features using a variant of hypothesis tests as a lower anomaly treshold could be used.

Heard et al. [31, 30] recently proposed a rather different approach: The authors here monitor the number of network-internal connections and authentication events on each machine over 5-minute invervals. The authors observed network-wide a strong *power-law-like* distributions of the number of events per host, i.e. the number of events is very large for some hosts an declines proportional to cx^{-k} where x is the number of the host. The authors then model the behaviour using two related probabilistic methods, the *Dirichlet process* and *latent Dirichlet allocation*, where the model parameters were estimated from attack-free training data in a Bayesian fashion. Event numbers which deviate strongly from the estimated model were then scored according to their unlikeliness. Both approaches were tested using the LANL dataset and asigned high scores to known inected computers. Furthermore, the authors claim to have possibly detected an unlabeled machine as being infected through manual investigation after it was asigned the highest score of all machines. As this approach is solely looking at the number of incoming and outgoing connections and events, this approach similarly to many entropy-based approaches covers a narrow area of possible malicious activity, and it is concerning that the ratio of benign machines with high anomaly scores (which can be seen as false alerts) is very high. However, this approach marks a step into a more probabilistic anomaly asignement which is beneficial for a more adaptive approach to model estimation and a quantification of detection certainty.

3.3 Event-based

The majority of network anomaly detection approaches are based on point anomaly detection, in other word they identify individual events as malicious solely on the observed characteristics of this event. Such events are usually either individual network packets or flows. In contrast to approaches on aggregated traffic, which can usually only detect attacks with a certain traffic volume, an event-based approach is independent of the traffic volume and therefore more suitable to identify activities consisting of only a few events, such as data exfiltration, R2L attacks, or C&C communication.

3.3.1 Statistics-based

Mahoney and Chen [53] were one of the first to develop statistical methods to identify anomalous events in network traffic. Their approach consists of two separate scoring stages.

The first stage is the *packet header anomaly detection* (PHAD). Here, the 33 different fields of an Ethernet-transmitted packet are converted from their one to four bytes to an integer value. The gathered values for each field are then clustered in a simple agglomerative fashion, and the clusters are updated each time a new packet arrives in order to keep the number of clusters below a treshold. The anomaly score of a packet is then proportional to the number of fields in which the clusters had to be updated.

At the second stage, the *application layer anomaly detection* (ALAD), scores are asigned to the packet according to a frequency table build using previously collected packets. These frequency tables address the several combinations of a variable conditional on another variable. These variables include source or destination IPs, destination port, TCP flags, or the first word of the payload. An interesting factor considered by the authors is the inclusion of the time since last observance for each of these frequency tables in the anomaly score.

The approach was tested on the DARPA'98 dataset and detected 70 out of 180 attacks while raising 100 false alerts. When the unrealistically high number of malicious packets in the data is considered, the number of false alerts is alarmingly high. Another issue with this publication is that the authors claim that they are building nonstationary models, yet give little information on how these models should adapt over time.

Kruegel et al. [42] developed an approach that is aimed fitting individual models for each of the different services generating network traffic. Their assumption is that by concentrating on only one type of traffic, statistical data with lesser variance can be collected.

The approach works as following: Once a connection is openened, the packet processing unit reads the first packets of a connection and extracts the specific service, such as a get request for a HTTP request. It is then asigned an anomaly score based on the different aspects: The type of service, the length of the request, and the payload contained in the request.

The anomaly score associated with the type of service is proportional to the negative logarithm of the service frequency observed in the training data. Thus, rare services receive a higher anomaly score.

To score the length l of the request, the mean μ and standard deviations σ of request lengths in the training data is estimated using maximum likelihood. The score is then proportional to $(l - \mu)/\sigma$.

Finally, the payload is scored according to a frequency distribution of the letters occuring in the training data. The deviation of a payload from the distribution can easily be estimated using a χ^2 test. By scoring the payload of a service request, the authors hope to detect malicious requests that try to disrupt the reciver through a corrupt combination of non-printable or replaced characters. Kruegel et al. [41] later greatly improved the payload scoring specifically for HTTP traffic by using a *Markov model*.

In their evaluation, the authors only considered DNS traffic due to lack of resources and space. Testing was done after a calibration of the anomaly tresholds by attacking their own DNS servers with 5 different attacks, all of which have been detected. However, evaluation of other services and on independent data would shed more light on

the actual performance. Another important issue not addressed by the authors is the possible temporal drift of the estimated distributions.

Both of these papers introduced introduced new concepts of how to model the distributions of individual event features which take into account the nature of network traffic. However, there is a lot to criticise about these approaches. The developed estimation and scoring methods lack a broader probabilistic foundation and can be improved greatly. Furthermore, these papers do not address any possible interdependence of features, which could lead to serious mismodelling of behaviour observed as anomalous. Also, it is unclear if these models will provide behaviour over time.

3.3.2 Classifier-based

Due to the availability of datasets such as DARPA'98 or KDD'99 which are labeled and rich in both benign and malicious traffic, it is tempting to apply existing machine learning approaches directly onto the event features provided in the data. Additionally, the KDD'99 data contains significantly more features for each event than normal network flow data. Consequently, there exists a large body of literature drawing from this apparent opportunity and applying a variety of classifiers such as decision trees, support vector machines, Naïve Bayes, or Neural Networks to these two datasets. However, such approaches often lack a greater understanding of the utilised data and the overall problem of intrusion detection, and achieved test results can normally not translated into good performance in realistic environments. Also, even if labels for both normal and anomalous data is available, the imbalance of normal traffic to malicious one generally makes learning a classifier hard.

Two illustrative examples are the works of **Barbará et al.** [6] on the DARPA 98 data, and of **Javaid et al.** [34] on the NSL-KDD data, both highly cited papers. Barbaá et al. apply a Bayesian network classifier on the available features[7] of individual packets while Javaid et al. train two layers of neural networks on the numerical features of the KDD flow events. Both approaches achieve high detection accuracies on the used data. The overall assumption here is that the classifier can generalise the learned differences that separate benign from malicious traffic in the dataset to a broader set of malware classes or even most of them.

However, there is no valid reason to assume that this is true. I would instead argue against it and claim that since there are only a relatively small number of different malware activity in every available dataset, the malicious traffic in a dataset only contains very little variation, which makes it fairly easy to overfit the malicious traffic with the available features and achieve high detection accuracy. This is especially true for the DARPA and the KDD data as they contain artifacts and duplicate entries, which are known lead to over-estimation of anomaly detection performance, according to recent work by **Ahmed and Mahmood** [3]. Another important point is that benign traffic is very heterogenous, and a classifier would have to be trained for every network to learn how to separate it from malicious traffic. Truthed data containing both attack and benign traffic is however very rare and notoriously hard to get, so it is far from realistic to assume that such data is available for every network. It is also important to repeat that event data is generally not as rich in features as the KDD data, which will further impact the ability of detecting malware.

[7]Numerical features and dictionaries of pre-seen IPs and ports

Other notable examples can be found in the references [27, 59, 52, 68, 72]. An interesting contribution was made by **Hu et al.** [33]. While having the same methodological flaws described above, the paper is more noteworthy for the proposal of a distributed learning framework that allows the large-scale training of a host-based model on many machines simultaneously.

3.3.3 Clustering based

Clustering is a techniques designed to identify and parametrise areas of higher probability density in the feature space of a dataset and can thus discover underlying structures in unlabelled data. These areas are identified as cumulations of datapoints that lie close to each other according to a distance metric. This can be very helpful in building an anomaly detection framework as it less trivial anomalies can be detected as datapoints outside of the identified data structure.

Clustering overcomes many drawbacks that other machine learning techniques face in the area of intrusion detection: As it does not perform a classification task, it is better suited to deal with unlabelled data that has a high imbalance between two classes, in our case normal and malicious traffic. Clustering furthermore usually has a low testing time and can be used for network data that does not labelled attack data. Robust techniques that can identify the existence of anomalies in the training data exist, however it is not properly tested how well translates to spotting malicious traffic. A drawback of clustering techniques are that only numerical variables can be used as input as a distance measure cannot be asigned to categorical variables.

Similarly to classification-based methods, the availability of the DARPA'98 and the KDD'99 dataset make the direct application of clustering techniques to raw data as an anomaly detection technique or as a pre-processing tool for classification based methods tempting. The described benefits of clustering can reduce some of the above described flaws of classification-based methods as it reduces overfitting and computation time. However, again little attention is paid to the fact that the features available in these datasets are unrealistic and flawed, and the identification rates achieved are not transferable to other datasets.

Lin et al. [49] recently developed a clustering based framework for intrusion detection, called *CANN*. They apply *k-means clustering* to the normalised numeric features of the KDD'99 data for $k = 5$. Anomalies are then identified as points exceeding a treshold distance from these centres using a specific distance metric that includes every cluster center, comparable with the *k-neirest neighbour* technique. Several other authors use clustering as a form of data pre-processing to reducing overfitting and computation time. Among them include **Wang et al.** [87] who use *ant-colony based clustering* combined with a neural network classifier with mixed results, or **Giacinto et al.** [26] who used k-means clustering with a *support vector machine* classifier, both on the KDD'99 data.

Traffic Classification and its relation to Clustering

Clustering of network events is related to traffic classification in that different traffic generator classes which influence the observed event features are assumed to exist. As I mentioned above, raw network events such as flows are not overly rich in features and are thus limited in the amount of traffic structure they can convey. Anomaly detection techniques that rely on clustering can benefit from the data mining techniques developed

in the area of traffic classification, which is why I will include some noteworthy results here.

In 2004, **McGregor et al.** [54] published one of the earliest works on network traffic clustering. They use a probabilistic method, the *Gaussian mixture model* (GMM), in order to estimate the position and the width of a small number of clusters. As this approach is based on the likelihood of the datapoints in a Gaussian model, it is possible to estimate the number of clusters using GMMs, a big advantage over most other methods were the number of clusters has to be chosen manually.

An important assumptions the authors make is that there can be several quite distinct classes of traffic within individual protocols, and that these classes can themselve be spread accross more than one port or protocol. The usual numeric features of network flows, the number of bytes and packets and the duration of the connection, are extended by statistics[8] of the packet sizes and the interarrival times of packets and the idle time which is the cumulated interarrival times exceeding two seconds. Furthermore, the notion of transaction and bulk mode of a connection is established, with the latter denoting more than three successive packets being sent in one direction, and the number of transitions between the two are recorded. A GMM is then trained using standard expectation maximisation, and 6 different well-separated clusters of traffic are identified and described. It is however questionable how representative 6 clusters are for the variety of network traffic, and how accurate the assumption of Gaussian distributions for such a small number of clusters is. *Zander et al.* [94] in 2005 use a very similar approach with a slightly more sophisticated probabilistic clustering method. The concept of extracting packet size and interarrival statistics as additional flow features is also used in classifier-based appraoches to traffic classification, most notably **Moore et al.** [5, 57] with classification accuracies up to 99% using a Bayesian neural network. Remarkably, accuracy only decreases to 95% when testing applications eight months apart.

A slightly different approach is taken by **Bernaille et al.** [8]: Instead of gathering statistics over the whole connection, only the size of the first five packets of a flow are used, and the flows are then clustered using k-means for individual protocols. The intuition here is that the first few packets capture the initial negotiating phase, which is usually a pre-defined sequence of messages with relatively consistent packet sizes. The benefit of this approach is that a flow can be classified before it ended, however out-of-order packets will lead to very different cluster asignements. The authors were able to distuingish flows from 10 different applications relatively accurately. A similar, yet classifier-based and thus supervised approach for traffic classification was developed by **Crotti et al.** [16].

Yen et al. [92] propose a very interesting method to passively fingerprint different browser implementations. They observed that in order to speed up content retrieval, different techniques are used in different browsers, which can in part be seen by the fact that Firefox initiates more flows than the other browsers upon connecting to a website, Opera sends more packets in earlier flows, and Safari sends fewer packets overall. In order to construct a classifier, they extract similar statistics for the byte and packet counts in each flow as McGregor et al. [54]. Furthermore, they use the number of simultaneously open flows and the time passed since the last start of a flow as features. Furthermore, for each retrieved website [9], the total number of flows along with the

[8] mean, first five modes, minimum and maximum, and quartiles

[9] It is however unclear, how the authors identify all flows corresponding one website retrieval in

cumulative byte count and retrieval duration are identified as useful classification features. A SVM-classifier is then be trained using these features and a dataset of website retrievals with labels for the corresponding browser. For this, a labelled dataset was generated by the authors for four different browser[10] and 150 websites over the course of multiple months. Acchieved classification accuracy is between 71% and 100%, depending on how many websites are left unclassified due to uncertainty of the classifier. However, as this detection method is supervised and depends on timely separated website retrievals, it is unclear how well these results translate to untruthed real-world data more common in the area of anomaly detection.

3.3.4 Representation-learning based

Representation learning, also called *feature learning* is a set of techniques aimed at automatically learning underlying structures in raw and noisy data, and are in a broader sense a form of density estimation. These techniques are often based on learning lower dimensional representations of the data, similar to subspace-projections, and are therefore suitable for data with highly correlated variables. Existing methods are often based on neural-networks and backpropagation. Learning of normal traffic behaviour can be done directly using representation learning instead of deriving probability distributions and correlations of individual traffic variables first. However, current methods are only suitable for numerical variables and not for categorical ones.

Ramadas and Ostermann [71] in 2003 proposed the use of *self-organizing maps* (SOM) to learn the representation of individual types of network services. A self-organizing map projects input data onto a two-dimensional lattice, which is why they are often used for data visualisation. The projection is learned using groups of competitive neurons. Each generation drops neurons which have different representations from the group, which makes this approach particularly computation-intensive. Since the projected data lies densely together, the authors train the map with normal traffic and detect anomalous events via their distance to the nearest neighbour. The authors however only evaluate their approach using 6 numerical features from DNS and HTTP network flows which they collected themselves. This makes a performance evaluation difficult and also leaves the question open how much knowledge is gained by using only 6 different flow features. Kayacik et al. [36] later extend this approach to all 41 numerical features of the KDD'99 data. The evaluation showed most success in the detection of DoS and probing attacks.

A direct approach to outlier detection is provided by **Hawkins et al**. [28] in 2002. They applied a *replicator neural network*[11] to the numerical features of the KDD'99 data. A replicator network tries to accurately reconstruct any input data it receives after sending it through a lower-dimensional bottleneck. The difference to an SOM is that learning is based on error-correction. By training it on normal traffic, the authors build a model that can reconstruct any normal traffic from its lower-dimensional representation with small errors. Anomalies are then detected as input data which is not reconstructed well and therefore deviates substantially from the learned data structures. Supposedly, a replicator network is robust agains small numbers of outliers in the training data. However, it requires careful examination how well this assumption

overlapping traffic

[10] Firefox, Opera, Safari, Internet Explorer

[11] Also called *Autoencoder network*

translates onto network traffic.

Gao et al. [22] use a similar technique called *deep belief networks* (DBN) on the KDD'99 data. They have a similar structure to replicator networks, but training is more difficult since their hidden layers are probabilistic. The authors mainly focus on explaining the benefits of using probabilistic neurons and discussing possible ways how to train a DBN on network traffic while not providing a thorough discussion of their results.

In general, effective applications of representation learning require the availability of data rich in features, something that is in general not true for network flow data. Existing approaches based on representation learning benefited heavily from the availability of additional features in the KDD'99 dataset, which sets an unrealistic standard. Representation learning can offer a great advances to the area of intrusion detection, however new approaches have to address the issue of engineering features suitable for learning real representations of the data.

3.4 Temporal correlation/Semantics-based

Despite network traffic being a stream of events, most anomaly-based intrusion detection approaches neglect any temporal features. Nevertheless, malicious behaviour most often is composed of a series of related computer and network events and have a distinct temporal and semantic profile [91]. As an example, an intruder using a session relay to exfiltrate information or pass malicious code generates a strong dependency between incoming packets in one connection and outgoing packets in the other connection (and vice versa) albeit the individual packets or connections resemble perfectly benign behaviour[12]. Building a model that can that can understand relations and dependencies between individual events could potentially lead to great improvements in the detection of otherwise non-anomalous attacks.

Since a machine's network traffic is the collective stream of multiple processes accessing the internet, individual traffic sequences are mixed with others, which makes the modelling of temporal dependencies a non-trivial task. In comparison to the other identified categories, research in the area of anomaly detection using temporal correlation or semantic models is sparse.

In 2003, **Krishnamurthy et al.** [40] propose a rather simple, yet efficient method for network-wide monitoring of individual key occurrences in an online fashion. For that, a sliding window approach is used to asign all keys (which here stand for individual IP addresses or network ports) a value containing the number of its occurrency. For each key, the collected values are then used to train either an *ARIMA* or a *Holt-Winters* method, both popular and powerful time-series forecasting models which can be trained in an online fashion. Anomalies are then identified as values with a forecasting error exceeding a certain treshold and thus indicating a sudden change in occurrency-behaviour of that particular key. This is an improvement to summarisation measures such as entropy or histograms in two ways: It provides a better resolution of individual traffic channels, enabling the detection of attacks with far less volume, and enabling the modelling of more complex temporal patterns which become apparent on a lower level. And it makes attack attribution far simpler by indicating directly the key which is subject to an anomaly.

[12] as they are essentially two normal connections that are relayed by the intruder

Since traffic is usually arriving at a fast rate, it is a computationally hard and memory-consuming task to count the occurrencies of all keys simultaneously. Schweller et al. overcome this problem using a *sketch-based counting approach* which uses hash-function to direct the values of a key directly to a position in a hash table without the need to store actual key in the memory. As this process is not exact, the count value is subject to statistical variations and the authors propose an unbiased estimator. The inaccuracy of the count estimator is also preventing the authors from using a probabilistic approach to anomaly detection instead of simple tresholding. However, this approach is also only counting occurrencies and does not detect any correlated key behaviour.

A great problem of the proposed framework is the fact that the key translation works only in one direction, making it impossible to associate a detected change with the corresponding key. This problem is overcome by **Schweller et al.** [75] who propose a reversible sketch method.

Pellegrino et al. [69] last year proposed *BASTA*, a framework to mine behavioural fingerprints from network flows using *timed automata learning*. An automata encodes patterns of short-term interactions of a system and is a representation of symbolic sequences, often corresponding to state transitions, which it encodes in a state transition function. Applied to network flows, an automata represents sets of events[13] that can follow each other in the network trace of a system. To be more specific, they used a type of probabilistic automata that works with transition probabilities and thus works in a similar way to a *hidden Markov model*. Events are asigned a state corresponding to the protocol and the direction of the flow, and the quantile[14] its duration, size, and number of packets are lying in. The authors then use this framework to create malicious fingerprints by training it on malicious traffic intermixed with normal traffic. These fingerprints are then detected on an infected host if the difference between the expected and the observed state counts drops below a treshold.

This paper is itself not developing any anomaly detection techniques, and I will not discuss the flaws of its application for malware fingerprinting. It is interesting for us as the developed automata mining techniques can also find direct application in mining normal behaviour automata and thus be used in an anomaly detection model.

Noble and Adams [64, 65] have recently proposed *ReTiNa*, a tool that measures temporal changes in the correlation between individual events in order to find intrusions on individual hosts. In their approach, they estimate the correlation between the time passed between two events, also called *interarrival time*, of an OD pair and the associated size or number of packets of the involved events. For this, interarrival time and the size/packet number are modelled as a bivariate gaussian distribution, and the covariance matrix is estimated using maximum-likelihood-estimation. The authors use a sophisticated online-estimation method to adapt the estimates to changes in the correlation structure, which can then be identified by comparison to an offline estimate. Anomalies are then identified as a collection of changes happening across multiple OD pairs on one host or in the entire network by simple hypothesis testing, which decreases the false-positive rate. The assumption here is that different OD pairs are independent of each other.

A big advantage of this approach is that it is adaptive and does not need a training phase, i.e. it is not reliant on attack-free training data. The method was tested both on

[13] distinguished by port, protocol, etc.

[14] of the overall distribution of the individual parameters

the LANL network flow data as well as internal data from the *Imperial College Academic network*. The method found several anomalies that coincide malicious activity in the network, but a definitive conclusion whether they are related is difficult to make.

Whitehouse, Evangelou and Adams [89] modeling the number of network flow and *user authentication* events on individual hosts as a polynomial function of the time and day and its rarity. Anomalies are then identified using Fisher's product test statistic and the reconstruction error. The method was tested on the LANL data using the auth and the flow sources and wa able to identify persistent structures in the data.

3.4.1 Application to stepping stone detection

Especially in larger computer networks, attackers often try use relay-like command chains, also called *stepping stones*, to obfuscate their origin or access machines without external connection. In such a chain, no direct connection exists between the first and the last machine, commands are sent through one or multiple intermediate machines. Stepping stone connections are usually encrypted and are notoriously difficult to identify. Upon detection, pairs of stepping stones are a clear indication of anomalous activity.

As stepping stone detection falls much more in the field of misuse detection, I will only give a very brief overview over emplyed techniques.

Zhang and Paxson [95] proposed one of the first methods for detecting stepping stones in 2000. They model relayed key-stroke packet streams as a two-dimensional ON/OFF switching process, where state changes have to occur within a certain window between the two channels. Correlation is then simply detected if the number of switches lying in this window exceeds a treshold.

A common assumption made when trying to find stepping stones is that packet or flow streams between different hosts in a network are almost always independent of each other, which is shown by [60]. **He and Tong**[29] model normal packet arrivals in a connection as a Poisson Process, and use the assumption of independence to derive a probabilistic distribution for the similarity of packet numbers in two channels in a time interval. P-values are then used to identify when the number of similar intervals becomes unlikely. They also show that if the ratio of chaff-packets exceeds to necessary packets in a stepping stone becomes too large, correlation is impossible to detect under the asumption of normal traffic following a Poisson distribution.

Neil et al. [60] also model normal event arrivals along an edge as a *negative Binomial process* with two different rates, evolving as a hidden Markov model, and correlation in different time intervals in then used using a likelihood ratio test to obtain p-values. In their approach, instead of testing network wide correlation, which is computationally unfeasible, testing is done on two different forms of local subgraphs, a star shape and a path-shape on selected edges.

3.4.2 Semantic-based approaches using different data sources

Approachs that model semantic or temporal behaviour characteristics have been also been applied on host-based data streams such as *system call logs* or *process logs* as well. As these are mostly symbolic event streams, anomaly detection in principal works similarly as for network traffic. As these data sources have usually a more hierarchical structure of events following each other in a parent-child fashion and therefore do not

suffer from overlapping signals, and it is generally easier to identify semantic structures. Notable examples include the use of deterministic automata [88], Markov chains [91], hidden Markov models [93, 32], or *recurrent neural networks*[20].

4. Related work and Conclusion

4.1 Related work

Lazarevic et al. [47] provided the first survey on network anomaly detection in 2003. The authors focused primarily on generating a comparitive study by testing different anomaly-based approaches on the DARPA'98 dataset as well as a private collection of network data. In order to compare the gathered results, the authors introduced four different measures to capture the response time, the detection rate, the false positive rate, and the amount of detected malicious connections in an attack burst, of each tested method.

Estevez-Tapiador et al. [24] in 2004 introduced a first taxonomy of anomaly-based NIDSs, proposing the particular network features analyzed, the type of behavior model, and the temporal scale of analysis as the basic criteria to classify existing methods. The authors concluded that anomaly-based intrusion detection was still a hard problem, and gave several recommendations for future research.

Similarly, Garcia-Teodoro et al. [21] in 2009 classified anomaly-based NIDSs into statistical, knowledge-based, and machine-learning approaches and their respective subcategories. The authors furthermore discuss available platforms and systems under development, and identify the assessment of anomaly-based methods due to the scarsity of representative datasets as one of the main open issues for network intrusion detection.

Patcha and Park [67] in 2007 produced a well-cited survey discussing existing anomaly-based solutions and future trends. The authors discuss the general benefits and drawbacks of anomaly detection for finding "zero day attacks", and proceed to focus on the technical realisability of individual methods and point. Additionally, several technical problems identified by the authors such as high false alert rates and scalability are discussed, and the authors give an outlook to future challenges and developments.

Sperotto et al. [79](2010) point out that the individual packet processing may not be possible at real-time due to the amount and speed of incoming traffic. They focus exclusively on flow-based detection methods using either anomaly or misuse methods. However, they discuss existing frameworks with less technical detail than other surveys.

Bhuyan et al. [9] (2014) examine a large number of existing network anomaly detection methods, and additionally provide a brief, but incomplete overview of existing datasets. They classify type of methods and systems in a detailed, however sometimes inconsistent manner. The authors also present capturing methods, different metrics, attack types

Ahmed et al. [3] (2016) provide a similar, yet less extensive survey of network anomaly detection techniques. However, the authors provide a more detail discussion of available datasets, pointing out their specific strengths and weaknesses. They also discuss some datasets that are more directed towards testing misuse-based methods rather than anomaly-based ones.

Nisioti et al. [63] this year produced a survey on feature selection methods unsupervised detection methods in network intrusion detection,. The authors furthermore discuss the potential evolution of current IDS architectures to include correlation of different data sources and to improve attack attribution. The authors also provide a new taxonomy of network attack classes which is adopted in this work. Lastly, the authors provide a discussion of available datasets similar to Ahmed et al.

Buczak et al. [11] (2016) produced a comparative survey of representative machine learning and data mining methods used in intrusion detection. Their evaluation is based on the citation number and the relevancy of a publication. The authors also give background information of the field and compare different methods. Finally, the authors also discuss the significant problems in intrusion detection, particularly the collection of labelled data for re-training of classifiers and misuse methods.

Several surveys discuss anomaly-based intrusion detection methods in specific environments such as wireless sensor networks, mobile ad hoc networks, or cloud service providers. Notable contributions come from Zhang et al. [90], Modi et al. [56], and Mitchell et al. [55].

Chandola et al. [13] (2009) provide a very-well cited and in my eyes the most comprehensive survey on general anomaly detection methods. They distinguish three classes of anomalies, namely point anomalies, group anomalies, and contextual anomalies. The authors then discuss a variety of techniques including the individual strengths and weaknesses, and compare them in terms of computational complexity and training time. Several applications of anomaly detection, including intrusion detection, are briefly discussed.

Sommer and Paxson [77] in 2010 provided a notable discussion of the application of anomaly detection and machine learning in network intrusion detection. In particular, the assumption that anomaly detection is suitable for finding novel attacks is examined and criticised. The authors identify the diversity and variability of network traffic, models lacking a semantic grasp of attacks, and a too wide scope for individual methods as problems that prevent the application of anomaly detection from being applied in operational settings with more success. They proceed to give recommendations for future research, among them to rely on the strength of machine learning in true classification for identifying evolutions of malware, and focusing on reducing false positives in anomaly detection.

Lastly, Nguyen et al. [61] (2008) provide a comprehensive survey of IP traffic classification using machine learning techniques. The authors provide context and motivation for the application of traffic classification, and discuss 18 significant works from the dominant period between 2004 and 2007. Methods are categorised according to their strategies, and are compared against a set of identifed requirements for the successful employment of ML in traffic classification.

4.2 Conclusion

Network intrusion detection is not a new field, and substantial literature exists with a variety of approaches having been proposed identify malicious network traffic. I identified three different categories of anomaly-based intrusion detection methods and reviewed the in my eyes most significant contributions in each category. I furthermore reviewed existing datasets that are suitable for anomaly detection.

In a nutshell, existing methods achieve mixed results upon detecting malicious activities as anomalous behaviour, especially when comparing the detection rates for different types of malicious traffic. Methods using aggregated traffic offer convincing models of normal behaviour in terms of the volume and frequency of different traffic features, and of their interdepence. They perform as expected best in the detection collective anomalies such as DoS attacks and network probing in moderate to large size, and are in my eyes a good supplementation of existing misuse frameworks in the detection of this type of malicious traffic.

Detection rates for other types of malicious traffic are however not as convincing. This most visible for event-based methods. As these are directed towards point anomaly detection, it should be expected that they are most suitable for the detection of intrusions that consist of individual malicious packets or flows like buffer overflow attacks or code injections. Most methods however were tested on the KDD'99 dataset, which has received significant criticism for allowing unrealistic detection rates. Those methods tested on more realistic datasets surprisingly achieved the highest detection rates for DoS attacks and network probing. Right now, methods from the area of misuse detection offer better and more reliable solutions for the detection of botnets and data manipulation attacks.

The two areas were I see the most room for significant advances are more sophisticated feature mining techniques for network flows, and models that capture temporal or semantic substructures in network events. Network flows are a structured quantity and give a better representation of the communication between two parties than individual packets. However, as I described in this work, currently generated flows are not rich in features and do not contain any information about the temporal or semantic structure of the communication.

Similarly, little attention has been paid so far at understanding the temporal correlation of individual flows for intrusion detection. However, it has already been shown [92] that programs create multiple flows in a distinct manner. Gathering information about this semantic structure of generated flows can be used to create narrow semantic profiles of host machines.

Both of the described areas have been overlooked by NIDS researchers and are in my eyes necessary to build more meaningful representations of modern network traffic.

Bibliography

[1] The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background traffic. 00001.

[2] Impact cyber trust: USC DS-062, USC DS-266, USC LANDER. `https://www.impactcybertrust.org/dataset_view?idDataset=62/` `https://www.impactcybertrust.org/dataset_view?idDataset=75/` `https://www.impactcybertrust.org/dataset_view?idDataset=265`, 2010. Accessed on 05 Nov. 2018.

[3] M. Ahmed, A. N. Mahmood, and J. Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[4] J. P. Anderson. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980.

[5] T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on neural networks*, 18(1):223–239, 2007.

[6] D. Barbara, N. Wu, and S. Jajodia. Detecting novel network intrusions using bayes estimators. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–17. SIAM, 2001.

[7] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 71–82. ACM, 2002.

[8] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, 2006.

[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014. 00357.

[10] N. Brownlee, C. Mills, and G. Ruth. Traffic flow measurement: Architecture. Technical report, 1999.

[11] A. L. Buczak and E. Guven. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, 2016. 00294.

[12] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, and G. Maciá-Fernández. PCA-based multivariate statistical network monitoring for anomaly detection. *Computers & Security*, 59:118–137, June 2016. 00027.

[13] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. 05458.

[14] G. Creech. *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks.* PhD thesis, University of New South Wales, Canberra, Australia, 2014.

[15] G. Creech and J. Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 4487–4492. IEEE, 2013.

[16] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic classification through simple statistical fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37(1):5–16, 2007.

[17] K. Cup. Data. knowledge discovery in databases darpa archive, 1999.

[18] K. Cup. Dataset. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 1999. Accessed on 05 Nov. 2018.

[19] D. E. Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987.

[20] M. Du, F. Li, G. Zheng, and V. Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298. ACM, 2017.

[21] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo. Anomaly detection methods in wired networks: a survey and taxonomy. *Computer Communications*, 27(16):1569–1584, Oct. 2004. 00208.

[22] N. Gao, L. Gao, Q. Gao, and H. Wang. An Intrusion Detection Model Based on Deep Belief Networks. In *2014 Second International Conference on Advanced Cloud and Big Data*, pages 247–252, Nov. 2014. 00046.

[23] S. Garcia, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

[24] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1):18–28, Feb. 2009. 00000.

[25] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. An evaluation framework for intrusion detection dataset. In *Information Science and Security (ICISS), 2016 International Conference on*, pages 1–6. IEEE, 2016.

[26] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, 9(1):69–82, 2008.

[27] K. K. Gupta, B. Nath, and R. Kotagiri. Layered Approach Using Conditional Random Fields for Intrusion Detection. *IEEE Transactions on Dependable and Secure Computing*, 7(1):35–49, Jan. 2010. 00170.

[28] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier Detection Using Replicator Neural Networks. In Y. Kambayashi, W. Winiwarter, and M. Arikawa, editors, *Data Warehousing and Knowledge Discovery*, Lecture Notes in Computer Science, pages 170–180. Springer Berlin Heidelberg, 2002. 00451.

[29] T. He and L. Tong. Detecting encrypted stepping-stone connections. *IEEE Transactions on Signal Processing*, 55(5):1612–1623, 2007.

[30] N. Heard, K. Palla, and M. Skoularidou. Topic modelling of authentication events in an enterprise computer network. 2016.

[31] N. Heard and P. Rubin-Delanchy. Network-wide anomaly detection via the dirichlet process. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*, pages 220–224. IEEE, 2016.

[32] J. Hu, X. Yu, D. Qiu, and H.-H. Chen. A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection. *IEEE network*, 23(1):42–47, 2009.

[33] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank. Online adaboost-based parameterized methods for dynamic distributed network intrusion detection. *IEEE Transactions on Cybernetics*, 44(1):66–82, 2014.

[34] A. Javaid, Q. Niyaz, W. Sun, and M. Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[35] D. Jiang, Z. Xu, P. Zhang, and T. Zhu. A transform domain-based anomaly detection approach to network-wide traffic. *Journal of Network and Computer Applications*, 40:292–306, 2014.

[36] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. A hierarchical som-based intrusion detection system. *Engineering applications of artificial intelligence*, 20(4):439–451, 2007.

[37] A. D. Kent. Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory, 2015.

[38] A. D. Kent. Cybersecurity Data Sources for Dynamic Network Research. In *Dynamic Networks in Cybersecurity*. Imperial College Press, June 2015.

[39] A. Kind, M. P. Stoecklin, and X. Dimitropoulos. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110–121, 2009.

[40] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247. ACM, 2003.

[41] C. Kruegel, G. Vigna, and W. Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5):717–738, 2005.

[42] C. Krügel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 201–208. ACM, 2002.

[43] A. Lakhina, M. Crovella, and C. Diot. Characterization of Network-wide Anomalies in Traffic Flows. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, pages 201–206, New York, NY, USA, 2004. ACM. 00439.

[44] A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-wide Traffic Anomalies. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '04, pages 219–230, New York, NY, USA, 2004. ACM. 01230.

[45] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM, 2005.

[46] P. Lamprakis, R. Dargenio, D. Gugelmann, V. Lenders, M. Happe, and L. Vanbever. Unsupervised detection of apt c&c channels using web request graphs. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 366–387. Springer, 2017.

[47] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. In D. Barbara and C. Kamath, editors, *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 25–36. Society for Industrial and Applied Mathematics, Philadelphia, PA, May 2003. 00843.

[48] A. Lazarevic, V. Kumar, and J. Srivastava. Intrusion detection: A survey. In *Managing Cyber Threats*, pages 19–78. Springer, 2005.

[49] W.-C. Lin, S.-W. Ke, and C.-F. Tsai. Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, 78:13–21, 2015.

[50] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 12–26. IEEE, 2000.

[51] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón. Ugr '16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers & Security*, 73:411–424, 2018.

[52] L. A. Maglaras and J. Jiang. Intrusion detection in SCADA systems using machine learning techniques. In *2014 Science and Information Conference*, pages 626–631, Aug. 2014. 00048.

[53] M. V. Mahoney and P. K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 376–385. ACM, 2002.

[54] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *International Workshop on Passive and Active Network Measurement*, pages 205–214. Springer, 2004.

[55] R. Mitchell and I.-R. Chen. A Survey of Intrusion Detection Techniques for Cyber-physical Systems. *ACM Comput. Surv.*, 46(4):55:1–55:29, Mar. 2014. 00164.

[56] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan. A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications*, 36(1):42–57, Jan. 2013. 00465.

[57] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 50–60. ACM, 2005.

[58] N. Moustafa and J. Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, Nov. 2015. 00074.

[59] Z. Muda, W. Yassin, M. N. Sulaiman, and N. I. Udzir. Intrusion detection based on K-Means clustering and Naïve Bayes classification. In *2011 7th International Conference on Information Technology in Asia*, pages 1–6, July 2011. 00104.

[60] J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie. Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics*, 55(4):403–414, 2013.

[61] T. T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys Tutorials*, 10(4):56–76, 2008. 01138.

[62] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys & Tutorials*, 2018.

[63] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos. From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. *IEEE Communications Surveys Tutorials*, pages 1–1, 2018. 00000.

[64] J. Noble and N. Adams. Real-Time Dynamic Network Anomaly Detection. *IEEE Intelligent Systems*, 33(2):5–18, Mar. 2018. 00000.

[65] J. Noble and N. M. Adams. Correlation-Based Streaming Anomaly Detection in Cyber-Security. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 311–318, Dec. 2016. 00004.

[66] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 2–2. USENIX Association, 2005.

[67] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, Aug. 2007. 01297.

[68] S. Peddabachigari, A. Abraham, and J. Thomas. Intrusion Detection Systems Using Decision Trees and Support Vector Machines. page 17, 2004. 00070.

[69] G. Pellegrino, Q. Lin, C. Hammerschmidt, and S. Verwer. Learning behavioral fingerprints from netflows using timed automata. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 308–316. IEEE, 2017.

[70] J. Peng, K.-K. R. Choo, and H. Ashman. User profiling in intrusion detection: A review. *Journal of Network and Computer Applications*, 72:14–27, Sept. 2016. 00043.

[71] M. Ramadas, S. Ostermann, and B. Tjaden. Detecting anomalous network traffic with self-organizing maps. In *International Workshop on Recent Advances in Intrusion Detection*, pages 36–54. Springer, 2003.

[72] V. Ramos and A. Abraham. ANTIDS: Self Orga nized Ant-Based C lustering Model for Intrusion Det ection System. In A. Abraham, Y. Dote, T. Furuhashi, M. Köppen, A. Ohuchi, and Y. Ohsawa, editors, *Soft Computing as Transdisciplinary Science and Technology*, Advances in Soft Computing, pages 977–986. Springer Berlin Heidelberg, 2005. 00075.

[73] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for Traffic Anomaly Detection. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '07, pages 109–120, New York, NY, USA, 2007. ACM. 00337.

[74] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian. Detecting p2p botnets through network behavior analysis and machine learning. In *Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on*, pages 174–180. IEEE, 2011.

[75] R. Schweller, A. Gupta, E. Parsons, and Y. Chen. Reversible sketches for efficient and accurate change detection over network data streams. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 207–212. ACM, 2004.

[76] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani. Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2018(1):177–200, 2018.

[77] R. Sommer and V. Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, May 2010. 00654.

[78] C. Sony and K. Cho. Traffic data repository at the wide project. In *Proceedings of USENIX 2000 Annual Technical Conference: FREENIX Track*, pages 263–270, 2000.

[79] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An Overview of IP Flow-Based Intrusion Detection. *IEEE Communications Surveys Tutorials*, 12(3):343–356, 2010. 00367.

[80] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. Nsl-kdd dataset. `http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html`, 2012. Accessed on 05 Nov. 2018.

[81] S. A. TEAM et al. skywiper: A complex malware for targeted attacks. Technical report, Technical Report.

[82] M. Thottan and C. Ji. Anomaly detection in ip networks. *IEEE Transactions on signal processing*, 51(8):2191–2204, 2003.

[83] M. J. M. Turcotte, A. D. Kent, and C. Hash. Unified Host and Network Data Set. *ArXiv e-prints*, Aug. 2017.

[84] UNIBS. Data sharing. `http://netweb.ing.unibs.it/~ntw/tools/traces/`, 2009. Accessed on 05 Nov. 2018.

[85] A. Wagner and B. Plattner. Entropy based worm and anomaly detection in fast ip networks. In *Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on*, pages 172–177. IEEE, 2005.

[86] C. Walsworth, E. Aben, K. Claffy, and D. Andersen. The caida ucsd anonymized internet traces 2012,", 2015.

[87] G. Wang, J. Hao, J. Ma, and L. Huang. A new approach to intrusion detection using artificial neural networks and fuzzy clustering. *Expert systems with applications*, 37(9):6225–6232, 2010.

[88] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*, pages 133–145. IEEE, 1999.

[89] M. Whitehouse, M. Evangelou, and N. Adams. Activity-based temporal anomaly detection in enterprise-cyber security. In *IEEE International Big Data Analytics for Cybersecurity computing (BDAC'16) Workshop, IEEE International Conference on Intelligence and Security Informatics*. IEEE, Nov. 2016. 00001.

[90] Yang Zhang, N. Meratnia, and P. Havinga. Outlier Detection Techniques for Wireless Sensor Networks: A Survey. *IEEE Communications Surveys & Tutorials*, 12(2):159–170, 2010. 00605.

[91] N. Ye et al. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, page 169. West Point, NY, 2000.

[92] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter. Browser fingerprinting from coarse traffic summaries: Techniques and implications. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 157–175. Springer, 2009.

[93] D.-Y. Yeung and Y. Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition*, 36(1):229–243, 2003.

[94] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 250–257. IEEE, 2005.

[95] Y. Zhang and V. Paxson. Detecting stepping stones. In *USENIX Security Symposium*, volume 171, page 184, 2000.