Impact of traffic micro-structure control and ground-truth information on model development

ACM Reference Format:
. 2021. Impact of traffic micro-structure control and ground-truth information on model development. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Data-centric breakthroughs in other fields have not been achieved solely by more complex and computationally more powerful ML-methods, but have been equally reliant on a precise understanding of the corresponding data structure as well as corresponding datasets that provide researchers with richer information and enable them to analyse weak points and model failures. As an example, results in *automatic speech recognition (ARS)* were not achieved by immediately training then state-of-the-art models on large annotated datasets. Initial models were reliant on highly sanitised and structured speech snippets in order to isolate low-level structures such as phonemes or time-warping, before the understanding of these structures lead to the success of more layered models of FF and LSTMneural networks and more recently fully end-to-end trained models. Lately, datasets that contain labelled specialised speech characteristics with varying intensity enable researchers to better understand ASR weak points such as emotional speech (RAVDESS), accents (Speech Accent Archive), or background noise (Urban Sound Dataset).

In a similar fashion, several approaches to enhance the way information is collected and presented have been successful in improving understanding between data and detection systems in different areas of information security. Virtual machine introspection monitors and analyses the runtime state of a system-level VM, and the inclusion of threat reports to create behavioural feature labels enriches the way executables are described [10]. Recently, data provenance is being seen as a better representation of system executions [2].

However, such efforts have not been made in network intrusion detection yet, with the current benchmark datasets paying more attention to the inclusion of a wide variety of attacks rather than the close control and detailed documentation of the generated traffic structures. This has so far lead to researchers predominantly applying of a number of ML-models directly to general traffic datasets in the hope of edging out competitors without analysing what traffic causes the model to fail and how design choices could prevent that.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

This overall lack of connection between the nature of intrusion detection data and the applied data-driven detection systems has been identified as a 'semantic gap' Paxson and Sommer [11].

Part of the reason for this gap is the lack of precise datasets that address particular traffic characteristics and contain additional information to allow for an in-depth evaluation. Clausen et al. insert citation, possibly blinded? have recently proposed a tool called DetGen that relies on containerisation to simulate network activity in a very controllable and deterministic way. The aim of the tool is to provide researchers with more information about traffic micro-structures by simulating and labelling a different factors such as corresponding computational activities, failure modes, or data input that impact traffic on a packet and flow level.

In this work, we demonstrate how this ground-truth information on traffic micro-structures can boost model inspection and improve both corresponding results and the overall understanding of traffic models to ultimately lead to more efficient improvements in their development. We provide three use-cases where existing models are ...

Advantages for network experiments

- *In-depth model evaluation*: Drawing on the extensive labelling of granular activities and reproducible traffic generation, researchers have new opportunities to examine the performance of an intrusion detection model in-depth. Packet-level structures and resulting false-positives can be better associated with activities, which helps correct models better for identified weaknesses. Granular activities can be studied in a less noisy environment due to isolation and reproducibility.
- *Focus and understand novel attacks and traffic types*: Instead of being restricted to a restricted set of attacks and traffic types, researchers using DetGen can easily embed novel attacks such as the eternal blue exploit or new traffic types such as QUIC in a given network setup without abandoning the overall network coherence of the data.
- *Reproducible, open research*: Scientific experiments should be reproduced to be considered valid, and the use of containers has recently been promoted to enable easy reproduction of computational work by reducing the need for plattform and library dependencies. Network researchers can use DetGen to allow for the easy reproduction of generated network settings, generated data, and deployed network intrusion solutions.

1.1 Outline

Outline of the coming sections.

.....
.....
.....

1.2 DetGen

Detgen is a container-based network traffic generation framework that we developed to enable repeatable, controllable, and informative network experiments. In contrast to the pool of programs running in a VM-setup, DetGen separates program executions and traffic capture into distinct containerised environments in order to shield the generated traffic from external influences and enable the fine-grained control of traffic shaping factors.

Traffic is generated from a set of scripted *scenarios* (give examples here) that strictly control corresponding influence factors and offer the researcher to modify and label the conducted activity from a variety of angles and randomisations. Containers communicate in a virtual network created with Mininet along with virtual software switches, Ethernet links, routers, and firewalls.

In the following experiments, we rely on DetGen to generate traffic data with corresponding micro-structure information labels.

2 USE-CASES

2.1 Impacts of ground-truth information on model understanding

Extensive ground-truth labels on different traffic influences are arguably the most important contribution of the DetGen framework. We demonstrate the benefits of this additional information for model-understanding on two examples using a traffic classification model by Hwang et al. [6] and a highly regarded anomaly detection model by Casas et al. [3].

2.2 Improving traffic separation with congestion level information

Our first use-case looks at how descriptive ground truth information on traffic characteristics can improve a traffic classification through the analysis of data separation in dependence of different traffic features. For this, we use a recent traffic classification model by Hwang et al. [6] as an example, which aims at distinguishing various types of malicious activity from benign traffic. The model classifies connections on a packet-level using an LSTM-network[1], and is claimed to achieve detection and false-positive (FP) rates of **99.7%** and **0.03%** respectively.

In our use-case, we train the model on a set of different HTTP-activities in order to detect SQL-injections. Rather than providing an accurate and realistic detection setting, this use-case shows how traffic information can be linked to model failures and slumping performance. We use HTTP-traffic from the CAIDA data as background traffic (85% of connections) and the provided SQL-injection attack traffic (7.5%) as well as different HTTP-activities for analysis (7.5%) using the DetGen-framework. In total, we use 50,000 connections for training the model, or slightly less than 2 million packets.

The initially trained model overall performs relatively well, with an AUC-score[2] of **0.981**, or a detection and false positive rate[3] of **0.96%** and **2.7%**. However, we would ideally like to improve these rates to both detect more SQL-injections and retain a lower

[1]Long-short-term-memory neural network
[2]a measure describing the overall class separation of the model
[3]tuned for the geometric mean

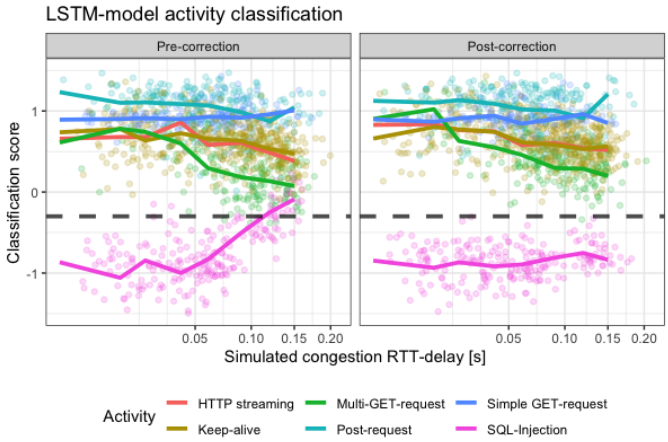

Figure 1: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold

false-positive rate. We therefore begin to explore which type of connections are misclassified most. potentially expand here to include influence of other traffic characteristics.... For this, we rely on the various ground-truth information provided by DetGen for both the included malicious traffic as well as part of the benign traffic, that allow us to relate classification scores to simulated traffic characteristics. Looking at the left panel of Figure 1, which depicts classification scores in dependence of the simulated network congestion, we learn that while classification scores are well separated for lower congestion, increased latency in a connection leads to a narrowing of the classification scores, especially for SQL-injection traffic. Since there are no classification scores that reach far in the opposing area, we conclude that congestion simply makes the model lose predictive certainty. A plausible cause would be that the increased amount of retransmission sequences decreases the overall sequential coherence for the model, i.e. that the LSTM-model loses context too quickly when processing retransmission sequences.

We try to correct the existing model with a simple fix by excluding retransmission sequences from the model input data, which leads to significantly better classification results during network latency, as visible in the right panel of Figure 1. SQL-injection scores are now far-less affected by congestion while scores for benign traffic are also less affected, albeit to a smaller degree. The overall AUC-score for the model improves to **0.997** while tuned detection rates and false positives improved to **99.1%** and **0.045%**.

2.3 Refining the notion of benign traffic for anomaly detection

Our second use-case looks at how ground-truth traffic information can help produce more coherent clusters and thus refine the capture of more detailed benign traffic structures in anomaly-detection. In particular, we will examine an anomaly-detection model by Casas et al. [3] that produces state-of-the-art detection results for access attacks according to a survey by Nisioti et al. [9]. The model takes a number of flow summary statistics as input, which include such as

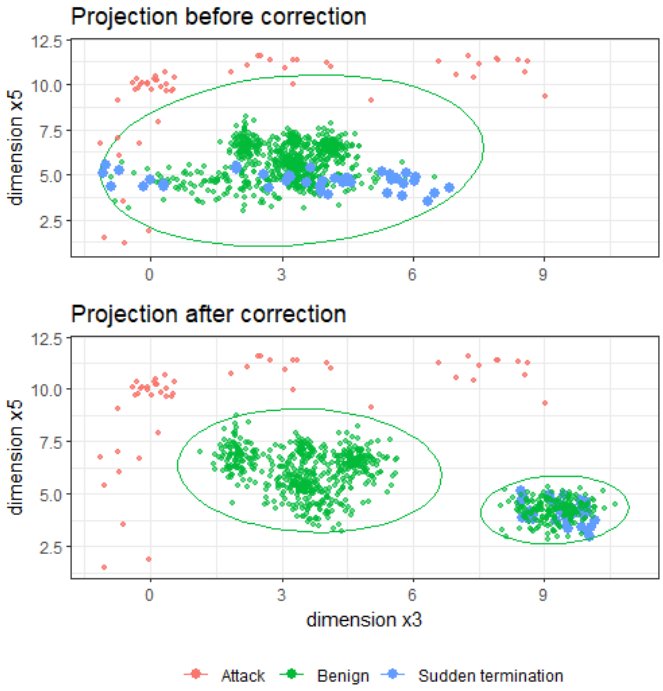

Figure 2: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold

packet size and interarrival statistics, number of idle and transfer periods, flag occurrencies etc. as input and projects it into different subspaces, where the connections are clustered. Anomalous outliers are detected by accumulating the Mahalanobis-distance from the cluster centers from each subspace. The identified clusters therefore serve as structural enclosures of benign behaviour, with the cluster borders acting as separators to abnormal behaviours. Benign traffic should ideally be distributed evenly around the cluster centres to allow a tight borders and good separation from actual abnormal behaviour.

Unstructured datasets such as the CAIDA traffic traces assumably contain too much abnormal behaviour to train an anomaly-detection model, which is why we train the model on benign traffic from the CICIDS-17 intrusion detection dataset (80%). Again, we also add traffic generated with the DetGen tool (HTTP, FTP, SSH, and SMTP, 20%) using a wide spectrum of settings for examination purposes. Attack data for the evaluation was again provided through the CICIDS-17 dataset, and includes access attacks such as SQL-injections, Heartbleed, Brute-Force attacks etc. We train the model with in total 15,000 connections.

When examining false-positive and corresponding anomaly scores, we noticed that in the CICIDS-17 dataset the model often classifies Brute-Force Web attacks as benign and some HTTP-traffic as anomalous. When examining the projected location of the corresponding connections, we see that most of this HTTP-traffic as well as the Brute-Force attack traffic lie near a particular cluster, depicted in Fig. 2. A significant portion of traffic in that cluster seems to be spread significnatly more across the cluster axis than the rest of the traffic in that cluster, leading to an inflated radius that partially encompasses Brute-Force traffic.

When cross-examining the traffic in this cluster with the DetGen traffic, we see that HTTP-traffic with the label "Sudden termination" are distributed across the cluster axis in a similar fashion, also depicted in Fig. 2, suggesting the conclusion that this type of traffic causes the inflated cluster radius. DetGen generates traffic with the label "Sudden termination" as half-open connections which were dropped by the server due to network failure. One defining characteristic of such connections are that they are not closed with a termination handshake using FIN-flags. To better capture this defining characteristics in the modelling process, we included an additional feature that indicates a proper termination with FIN-flags in the modelling process.

The newly trained model now projects "Sudden termination" connections into a different cluster, which leads to a far better cluster .

2.4 Empirical evaluation of traffic sequence embeddings

Recently, the embedding of sequences into a vectorised representation has gained traction in the network science community [4, 7]. Embeddings are (often low-dimensional) vectors that aim to characterise a sequence through its position in a vector space. This numerical representation can then be used in various ways, such as to classify or cluster traffic, profile hosts or users, or generate new artificial traffic traces.

Since the embedded position of a traffic relative to similar or dissimilar traffic crucial ...

2.4.1 Position consistency evaluation. It is absolutely crucial for a represenation model that sequences with the same characteristics receive a similar embedding, i.e. traffic traces that correspond to the same activity and settings should be projected very close together.

To do so, we generate traffic from settings within which all controllable influence factors are held constant. Traffic samples from each setting should then be similar up to small deviations in IATs, additional ACK-packets etc., such as depicted in Fig. **??**.

2.4.2 Similarity evaluation.

3 FIDELITY CONFIRMATION EXPERIMENTS

3.1 Traffic control and generation determinism

We now assess the claim of control over the outlined traffic influence factors, and how similar traffic generated with the same settings looks like. We also demonstrate that this level of control is not achievable on regular VM-based NIDS-traffic-generation setup.

To do so, we generate traffic from settings within which all controllable influence factors are held constant, both with DetGen framework and with a regular VM-based setup. Traffic samples from each setting should then be as similar as possible to provide sufficient experimental determinism. To measure how similar two traffic samples are, we devise a set of similarity metrics that measure dissimilarity of overall connection characteristics, connection sequence characteristics, and packet sequence characteristics:

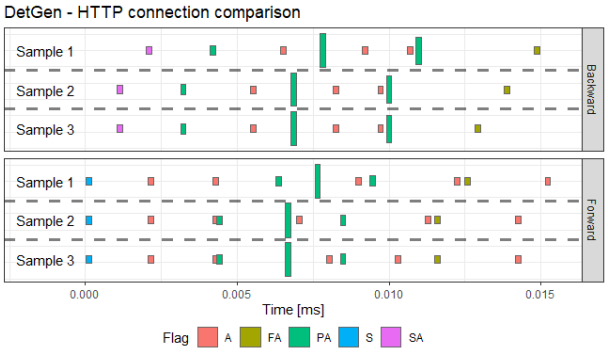

Figure 3: Packet-sequence structure similarity comparison for HTTP-activity under constant settings generated by the DetGen framework (left) and in a regular setting (right). Colours indicate packet flags while the height of the packets indicates their size. Note that in addition to more differences in the timing, the packet sizes vary more in the regular setting.

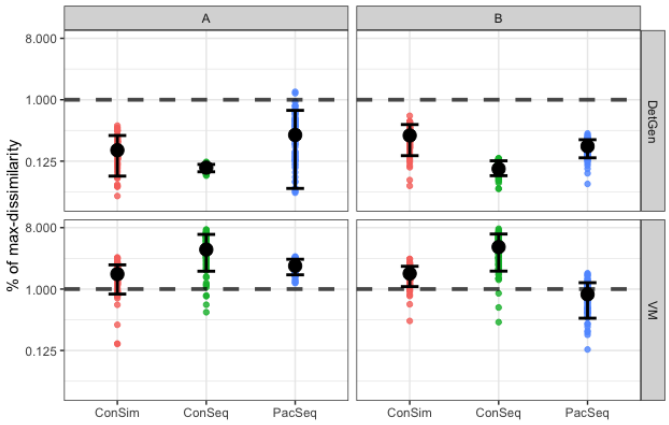

Figure 4: Comparison of HTTP-group dissimilarity scores for the DetGen-framework and a regular VM-setup, on a logarithmic scale. Samples from the VM-setting are consistently more dissimilar, in particular for flow-based metrics, where the average dissimilarity is more than 30 times higher than for the DetGen setting.

Label	Congest.	HTTP	File-Sync	Mirai-C&C
A	Low	Get-req. NGINX	Two hosts	Command 1
DetGen		0.20%, 0.03%, 0.18%	0.12%, 0.01%, 0.19%	0.21%, 0.0%, 0.22%
B	Low	Multi-req. NGINX	Four hosts	Command 2
DetGen		0.27%, 0.30%, 0.17%	0.11%, 0.1%, 0.24%	0.19%, 0.02%, 0.17%
C	High	Post-req. Apache	Two hosts	Command 3
DetGen		0.41%, 0.01%, 0.29%	0.39%, 0.01%, 0.21%	0.29%, 0.02%, 0.23%
D	High	Multi-req. Apache	Four hosts	Command 4
DetGen		0.55%, 0.39%, 0.20%	0.31%, 0.15%, 0.38%	0.24%, 0.0%, 0.32%

Table 1: Outline of the traffic settings used for the determinism evaluation, along with the average dissimilarity percentages for each setting (red=overall connection similarity, green=connection sequence similarity, blue=packet sequence similarity)

- **Overall connection similarity** We collect 80 flow summary statistics (IAT and packet size, TCP window sizes, flag occurrences, burst and idle periods). We compress this information using PCA to 8 significant dimensions, and measure the cosine similarity between connections, which is also used in general traffic classification [1].
- **Connection sequence similarity** To quantify the similarity of a sequence of connections in a retrieval window, we use the following features to describe the window, such as used by Yen et al. [12] for application classification: The number of connections, average and max/min flow duration and size, number of distinct IP and ports addresses contacted. We then again measure the cosine similarity based on these features between different windows.
- **Packet sequence similarity** To quantify the similarity of packet sequences in traffic captures, we assign packets a discrete state according to their flags, direction, sizes, and interarrival times (insert citation). We then calculate the Markovian probability of each packet state conditional on the previous packet. We do this for sequences of 15 packets at the start, the middle, and the end of a connection, and use the average sequence likelihood of each group as a similarity measure. If connections are completely similar, the conditional probabilities and thus the likelihoods should converge to one.

We normalise all dissimilarity scores by dividing them by the maximum dissimilarity score measured for each traffic type in our experiment in Section **??**, so that the reader can relate the measured scores to the traffic type.

As a comparison, we use a regular VM-based setup, were applications are hosted directly on two VMs that communicate over a virtual network bridge that is subject to the same NetEm effects as DetGen. To compare the amount of traffic control and the corresponding generative determinism of DetGen and the VM-setup, we generate three different types of traffic (HTTP, file-syncing, and botnet) from four different settings, within which all generative parameters are kept constant. For each setting and traffic type, we generate 100 traffic samples and apply the described dissimilarity measures to 100 randomly drawn pairs sample pairs. Fig. 3 depicts the calculated dissimilarity scores for DetGen and the VM-setup, while Table 2 describes the different settings and the corresponding average dissimilarity scores.

As visible, the scores yield less than 1% of the dissimilarity observed on average for each protocol. Scores are especially low when compared to traffic groups collected in the VM setting, which is also visible in Fig. **??** for the HTTP-traffic. Dissimilarity scores for the VM-setting are most notably higher for the flow-metric,

Label	Overall Setting	HTTP	File-Sync	Mirai-C&C
A	Low congest., high load	Get-req. NGINX	Two computers	Command seq. 1
DetGen		0.20%, 0.03%, 0.18%	0.12%, 0.01%, 0.19%	0.21%, 0.0%, 0.22%
VM		1.5%, 3.8%, 1.3%	0.9%, 1.6%, 0.38%	1.1%, 1.8%, 0.7%
B	Low congest., no load	Multi-req. NGINX	Four computers	Command seq. 2
DetGen		0.27%, 0.30%, 0.17%	0.11%, 0.1%, 0.24%	0.19%, 0.02%, 0.17%
VM		1.3%, 2.9%, 0.9%	1.1%, 4.9%, 1.2%	1.1%, 1.1%, 0.6%
C	High congest., no load	Post-req. Apache	Two computers	Command seq. 3
DetGen		0.41%, 0.01%, 0.29%	0.39%, 0.01%, 0.21%	0.29%, 0.02%, 0.23%
VM		1.9%, 1.3%, 1.5%	1.3%, 1.4%, 1.2%	1.1%, 1.3%, 0.9%
D	High conges., high load	Multi-req. Apache	Four computers	Command seq. 4
DetGen		0.55%, 0.39%, 0.20%	0.31%, 0.15%, 0.38%	0.24%, 0.0%, 0.32%
VM		1.9%, 4.2%, 1.4%	1.6%, 4.8%, 1.4%	0.9%, 1.2%, 0.9%

Table 2: Outline of the traffic settings used for the determinism evaluation, along with the average dissimilarity percentages for each setting (red=overall connection similarity, green=connection sequence similarity, blue=packet sequence similarity)

caused by additional background flows frequently captured. While sequential dissimilarity is roughly the same for the DetGen- and the VM-settings, overall connection similarity for the VM-setting sees significantly more spread in the dissimilarity scores when computational load is introduced.

4 CONCLUSIONS

In this paper, we proposed DetGen, a framework aimed at improving researchers understanding of traffic micro-structures and their respective effect on traffic models in order to close the *semantic gap* described by Sommer and Paxson [11]. DetGen allows reproducible traffic generation experiments that allow the control and monitoring different traffic shaping aspects while delivering data that is truthful to real-world structures. Our framework achieves this through containerised applications and corresponding process and traffic separation, meticulous attention to the corresponding generating activities and their facets, and the careful emulation and control of external effects. Currently, DetGen produces traffic for 29 different activities.

We verified the improvements regarding reproducibility and traffic control of DetGen compared to traditional VM-setups in an experiment as well as the fidelity of the generated traffic to real-world characteristics in another experiment. Especially regarding the exclusion of background traffic and corresponding connection sequence structures, DetGen outperformed traditional setups significantly. In terms of traffic diversity, DetGen achieved significantly better results than current state-of-the-art NIDS-datasets.

DetGen offers strong insights into traffic micro-structures and their effect on traffic models, which we demonstrated in three use-cases. By allowing researchers to analyse the particular characteristics of events that lead to false-positives or model failure as well as their effect on model training for three distinct NID-models, we were able to understand where the design of those models is flawed and how to improve them to boost detection performance.

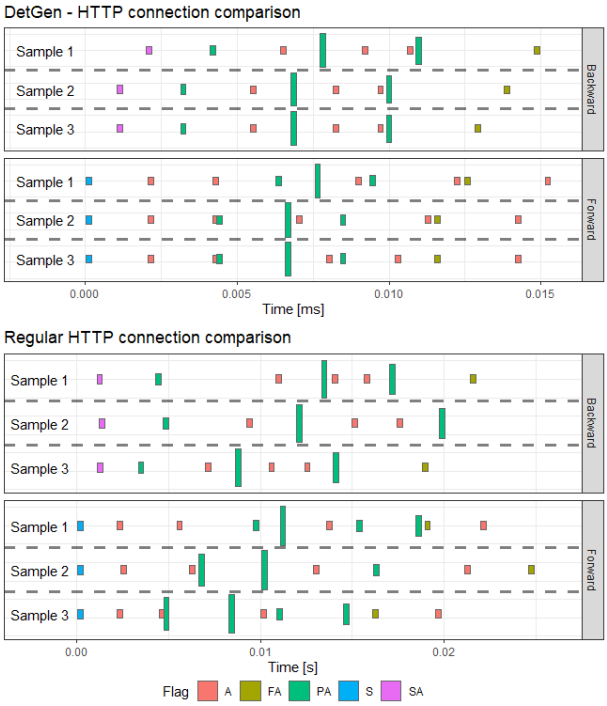

Figure 5: Packet-sequence structure similarity comparison for HTTP-activity under constant settings generated by the DetGen framework (left) and in a regular setting (right). Colours indicate packet flags while the height of the packets indicates their size. Note that in addition to more differences in the timing, the packet sizes vary more in the regular setting.

4.1 Difficulties and limitations

DetGen is building network traffic datasets from a small-scale level up by coalescing traffic from different fine-grained activities together. While this provides great insight into traffic microstructures, our framework will not replicate realistic network-wide temporal structures, such as port usage distributions or long-term temporal activity. These quantities would have to be statistically estimated from other real-world traffic beforehand to allow our framework to emulate such behavior reliably. Other datasets such as UGR-16 use this approach to fuse real-world and synthetic traffic and are currently better suited to build models of large-scale traffic structures.

We paid meticulous attention to enable control over as many traffic impact factors as possible. However, DetGen is currently only offering insufficient control over underlying application-layer implementations such as TLS 1.3 vs 1.2. In theory, it should be unproblematic to provide containers with different implementations for each scenario to provide this control. However we faced difficulties to compile containers in a suitable manner and are currently investigating, how to improve DetGen on this shortcoming.

Working with Docker containers can sometimes complicate the implementation of individual scenarios compared to working with VMs. Although several applications are officially maintained Docker containers that are free from major errors, many do not. For instance, in the *BitTorrent* scenario, most common command line tools, such as `mktorrent`, `ctorrent` and `buildtorrent`, failed to actually produce functioning torrent files from within a container due to Docker's union filesystem. Furthermore, due to the unique way in which we are using these software packages, unusual configuration settings are sometimes needed.

4.2 Future work

Our traffic generation framework is designed to be expandable and there are many avenues for future work. The continual development of scenarios and subscenarios would improve the potential realism of datasets generated using the framework. The addition of more malicious scenarios would enable a more detailed model evaluation and improve detection rate estimation. Another future improvement for framework is to add scripts that emulate the usage activity of individual scenarios by a user or a network.

Although ground truth for particular traffic traces is provided by capturing `.pcap`-files for each container individually, we have not implemented a labelling mechanism yet for the dataset coalescence process. Though not technically difficult, some thought will have to be put how such labels would look like to satisfy different research demands. Furthermore, the Docker platform provides the functionality to collect system logs via the `syslog logging driver`. We plan on implementing their collection in the future, where they could act either as traffic labels providing more ground truth details, or act as a separate data source that complements the collected traffic.

We wish to publish this framework to a wider audience, allowing for further modification. This will be done using a GitHub repository, which contains both the implemented capture scenarios as well as the corresponding container images.

5 BACKGROUND AND MOTIVATION

5.1 Misuse and machine learning

Network intrusion detection is the field of detecting intrusions in a network by analysing captured traffic traces exchanged between computers in the network. Most commonly used are misuse detection systems identify known signatures of bad behaviour in traffic such as malicious packet payloads or rule-based patterns concerning port usage and/or packet sequences. Although very efficient, these methods are reliant on precise details on known attacks in the form of signature databases. Significant efforts have been invested in developing machine-learning based methods that are trained on large amounts of traffic to develop a more generalisable distinction between benign and malicious behaviour to remove the need of attack signatures and enable the detection of zero-day attacks.

5.2 Existing problems

Machine-learning based network intrusion detection has been subject to extensive criticism due to being unable to deliver sufficient detection rates at an acceptable false-positive rate in actual deployment. Two main causes for these failings have been identified particularly for network-based methods by Sommer and Paxson [11] in 2010, which have been supported and partly extended by Harang [5] in 2014 or by Liu et al. in 2019 [8]:

Semantic gap between between results and their operational interpretation. Arguably the biggest concern expressed by Sommer and Paxson is that methods lack a deep semantic insight into a system's capabilities and limitations and are instead treated as black boxes. The authors here draw comparisons to other areas of machine learning such as character recognition where the precise understanding of the data structure and how existing systems process it have lead to breakthroughs such as the convolutional layers that process the data in a more adequate way. In network intrusion detection, different methods are thrown at existing data without thorough analysis where the system performs well and where it fails or breaks, and what the reasons for this are. The authors recommend to researchers to narrow the scope to more specific applications and closely examine what types of traffic trigger which responses by the system in order to develop a better understanding of where and how future systems can be designed to better suit this particular type of data and application.

Fundamental difficulties for conducting sound evaluation. The semantic gap stems in part from persistent difficulties for researchers to evaluate their system thoroughly and in a comparable and reproducible manner due to a lack of appropriate public datasets. Privacy and security concerns discourage network administrators to release rich and realistic datasets for the public, leading to publicly available real-world datasets being the exception and missing informative features such as captured packets or consistent IP-addresses. This forces researchers to generate synthetic datasets using small virtual networks, and restricts the diversity and coverage of traffic researchers are able to examine.

Furthermore, the labelling process is significantly more difficult in network intrusion detection than in other domains with easier interpretable data. Often, only traffic directly involved in an attack is labelled manually, with all other traffic receiving the same 'Benign'

label. This lack of informative labels impedes researchers abilities to analyse different types of traffic and thus understand the properties of their system.

The lack of benchmark datasets often forces researchers to assemble their own data, which is mostly done in a non-reproducible way, leading to unverifiable detection rates and incomparable results.

Other problems identified by Sommer and Paxson include the diversity of network traffic, the high cost of errors, and lacking computational speed or detection systems.

REFERENCES

[1] Y. Aun, S. Manickam, and S. Karuppayah. A review on features' robustness in high diversity mobile traffic classifications. *International journal of communication networks and information security*, 9(2):294, 2017.

[2] M. Barre, A. Gehani, and V. Yegneswaran. Mining data provenance to detect advanced persistent threats. In *11th International Workshop on Theory and Practice of Provenance (TaPP 2019)*, 2019.

[3] P. Casas, J. Mazel, and P. Owezarski. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Computer Communications*, 35(7):772–783, 2012.

[4] R. Gonzalez, F. Manco, A. Garcia-Duran, J. Mendes, F. Huici, S. Niccolini, and M. Niepert. Net2vec: Deep learning for the network. In *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pages 13–18, 2017.

[5] R. Harang. Bridging the semantic gap: Human factors in anomaly-based intrusion detection systems. In *Network Science and Cybersecurity*, pages 15–37. Springer, 2014.

[6] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang. An lstm-based deep learning approach for classifying malicious traffic at the packet level. *Applied Sciences*, 9(16):3414, 2019.

[7] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. Generating high-fidelity, synthetic time series datasets with doppelganger. *arXiv preprint arXiv:1909.13403*, 2019.

[8] H. Liu and B. Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20):4396, 2019.

[9] 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.

[10] M. R. Smith, N. T. Johnson, J. B. Ingram, A. J. Carbajal, R. Ramyaa, E. Domschot, C. C. Lamb, S. J. Verzi, and W. P. Kegelmeyer. Mind the gap: On bridging the semantic gap between machine learning and information security. *arXiv preprint arXiv:2005.01800*, 2020.

[11] 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. IEEE, 2010.

[12] 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.