Leveraging traffic micro-structure control for content-sensitive model development

***Abstract*—...**
...
...
...
...
...
***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The current development process of machine-learning (ML) based network intrusion detection (NID) models is mostly agnostic of specific traffic characteristics and lacks the ability to extensively explore model behaviour and limitations due to a lack of precise datasets with specifically curated characteristics and corresponding information. We demonstrate how the generation of traffic with controllable and labelled micro-structures enables researchers to probe a model and its reaction to various traffic phenomena to much greater detail in order to understand and develop the model's capabilities.

Machine-learning breakthroughs in other fields have often been reliant on a precise understanding of data structure as well as corresponding datasets that provide researchers with rich information to develop more powerful models that address the nature of the modelled data. Initial models in *automatic speech recognition (ASR)* for example 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 feed-forward and recurrent neural networks and more recently fully end-to-end trained models. Lately, datasets that contain labelled specialised speech characteristics 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 tools aim to improve the representation of system executions [1] over traditional logs.

However, such efforts have not been made in network intrusion detection yet, with the current quasi-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. Data containing ground-truth on the traffic generation process to link observable structures with corresponding computational activities is rare, which has so far lead researchers to predominantly apply a number of ML-models directly to traffic datasets in the hope of edging out competitors. and how the model design could be improved accordingly. 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], and is seen to be partly responsible for the lack of success machine-learning had in network intrusion detection. This has been supported and partly extended by Harang [5] in 2014 or by Liu et al. in 2019 [7]:

In this work, we aim demonstrate the usefulness of the control and information on traffic micro-structures for model validation and development. We showcase the inspection of two state-of-the-art network intrusion detection models with specially generated traffic to identify model flaws and subsequently boost corresponding results as well as the overall understanding of traffic models. Our motivation is to inspire other researchers to adopt this practice when developing new data-driven detection models.

A. Outline

Outline of the coming sections.

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

II. Motivation and Methodology

A. Motivation

Scientific machine learning model development requires both **model evaluation**, in which the overall predictive quality of a model is assessed to identify the best model, as well as **model validation**, in which the behaviour and limitations of a model is assessed through targeted probing. Model validation is essential to understand how particular data structures are processed, and enables researchers to improve their models accordingly.

We want to demonstrate how model validation can be performed for machine-learning-based network intrusion detection models such as traffic classifiers or anomaly-detection systems with the use of specifically generated traffic traces. We focus in particular on **traffic-microstructures**, which we

define as short-term atomic, sequential or cumulative structures in the packet or flow metadata stream, compared to aggregate structures visible over longer periods. Differences in observable traffic micro-structures are driven through factors such as the particular communicational activity, the choice and implementation of a communication protocol as well as external effects such as network reliability or available computation resources.

Training Probing

B. Generating controllable traffic micro-structures

We use a tool (citation currently blinded) that generates various types of traffic with a fine-grained control of traffic shaping factors. The tool provides a set of currently 29 traffic settings to cover a variety of traffic types and offers control and if necessary randomisation of the following traffic shaping factors:

a) Performed task and application: The conducted computational task and application ultimately drives the communication between computers, and thus hugely influences characteristics such as the direction of data transfer, packet rate, or the number of connections [12].

b) Transferred data: The amount and content of transferred data influences the overall packet numbers. Furthermore, the content of the data can potentially impact packet rates and sizes, such as shown by Biernacki [2] for streaming services.

c) Caching/Repetition effects: Tools like cookies, website caching, DNS caching, known hosts in SSH, etc. remove one or more information retrieval requests from the communication, which can lead to altered packet sequences, less connections being established [4].

d) Application layer implementations: Different implementations for TLS, HTTP, etc. can yield different channel prioritisation and can perform different handshakes in slightly different ways.

e) Host level load: Computational load (CPU, memory, I/O) on the host machine can affect the processing speed of incoming and outgoing traffic.

f) LAN and WAN congestion: Low available bandwith, long RTTs, or packet loss can have a significant effect on TCP congestion control mechanisms, which in turn influence frame-sizes, IATs, window sizes, and the overall temporal characteristic of the sequence.

Labels that describe the respective setting for each of the described factors are attached to each traffic sample after generation. Traffic is generated in a virtual network along with virtual software switches, Ethernet links and routers. The communication is mostly performed in a client-server setting, however some settings such as multi-host file-synchronisation involve more hosts.

C. Methodology

Before the probing, we have to identify which types of characteristics the model should be probed on, and generate the corresponding data.We then train the model mainly on the datasets that is used for the general evaluation, but also attach a sufficient amount of the data dedicated for model probing to the training set. This is to ensure that the model is able to see and learn the structures in the probing data before the probed, even though the overall type of traffic in the probing data should be similar to the evaluation data to provide a consistent model.

After training and general evaluation, the model probing is done by feeding the model data samples with the desired descriptive labels, monitoring the output or behaviours in dependence of these labels, and comparing them to the expected output or behaviour. Since each traffic sample contains multiple descriptive labels, it is possible to monitor the model response to multiple characteristics in parallel.

III. Improving traffic separation for a classifier 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 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 2, 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

[1]Long-short-term-memory neural network

[2]a measure describing the overall class separation of the model

[3]tuned for the geometric mean

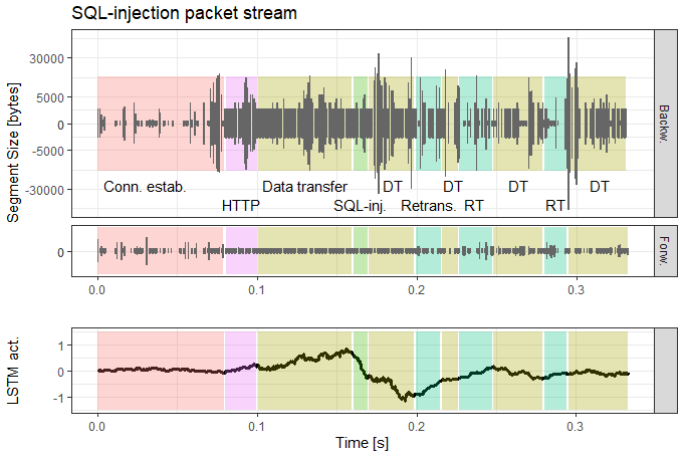

Fig. 1: LSTM-output activation in dependence of connection phases. Depicted are packet segment streams and their respective sizes in the forward and backward direction, with different phases in the connection coloured and labelled. Below is the LSTM-ouput activation while processing the packet streams.

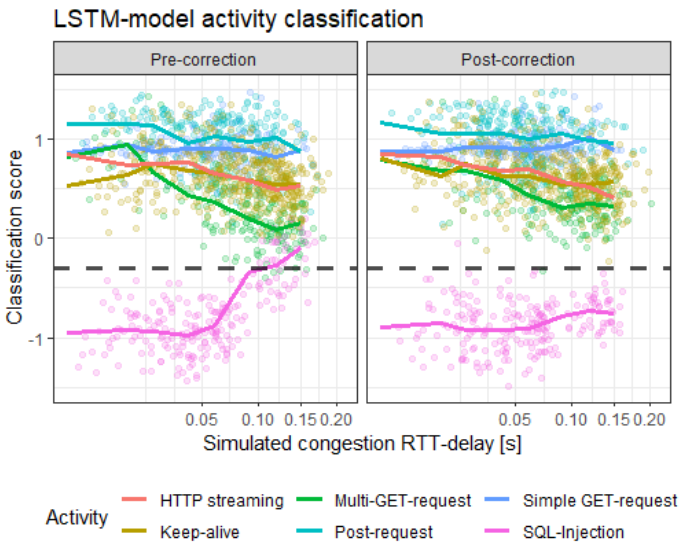

Fig. 2: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold.

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.

To examine the exact effect of retransmission sequences on the model output, we generate two connections with the exact same characteristics except that one connection is subject to moderate packet loss and reordering while the other is not. We then compare how the LSTM-output activation is affected by retransmission sequences. Fig. 1 depicts the evolution the LSTM-output layer activation in dependence of difference connection phases. As visible, initially the model begins to view the connection as benign when processing regular traffic, until the SQL-injection is performed. The model then quickly adjusts and provides a malicious classification after processing the injection phase and the subsequent data transfer. The negative output activation is however quickly depleted once the model processes a retransmission phase, and is afterwards not able to relate the still ongoing data transfer to the injection phase. When comparing this to the connection without retransmissions, we do not encounter this depletion effect, instead the negative activation persists after the injection phase.

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 2. 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%**.

IV. 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 packet size and interarrival statistics, number of idle and transfer periods, flag occurrencies, number of flows in window, 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,

Congest.	HTTP	File-Sync	Mirai-C&C
Low	Get-req. NGINX	Two hosts	Command 1
1	0.14 , 0.45	0.19 , 0.27	0.03 , 0.06
Low	Multi-req. NGINX	Four hosts	Command 2
2	0.32 , 0.45	0.15 , 0.33	0.03 , 0.04
High	Post-req. Apache	Two hosts	Command 3
3	0.17 , 0.28	0.16 , 0.28	0.02 , 0.04
High	Multi-req. Apache	Four hosts	Command 4
4	0.53 , 2.51	0.71 , 1.31	0.03 , 0.05

TABLE I: Outline of the traffic settings evaluation, along with the average (blue) and maximal (red) Mahalanobis-distance for each projection.

Heartbleed, Brute-Force attacks etc. We train the model with in total 15,000 connections.

A. *Projection coherency evaluation*

Like many approaches that generate representations of benign traffic for anomaly detection, Cases et al. project traffic events into a vector-space where traffic clusters and similarities become more apparent. In order for the projection to accurately capture important traffic structures, this projection should be consistent, i.e. traffic events with similar origins and characteristics should be projected to similar positions rather than be dispersed throughout the vector space.

Verifying the projection consistency of a model is not straightforward as usually no or not enough ground-truth information about different traffic characteristics is available to asses if the model is projecting similar traffic to dissimilar positions, or if the traffic just bears some dissimilar characteristics.

Using DetGen, we are able to generate traffic from identical conditions to provide certainty on the expected traffic similarities, which is more suitable for the described task. We generate a small dataset that consists of traffic from 12 different settings, with the conducted activities and other traffic-shaping factors being held constant. Within each setting, we only allow for minor randomised effects such as packet reordering or slight variations in the activity input. Table I summarises the traffic from each setting.

We verify if traffic samples within each group are projected to similar areas by measuring the average and maximum Mahalanobis-distance to quantify the overall dispersion of the samples. The results are displayed in Table I and depicted in Fig. 3. First of all, the model projects all samples from each group within the same cluster, which confirms the capture of the coarse traffic structure. When looking at the traffic dispersion and the corresponding Mahalanobis-distance measurements, we notice that the *multi-request HTTP* traffic as well as the *file-synchronisation* between mutliple computers is much further dispersed than in the other settings, and that the corresponding axis with the most projected dispersion seems to be the same for each of the four settings, which suggests that

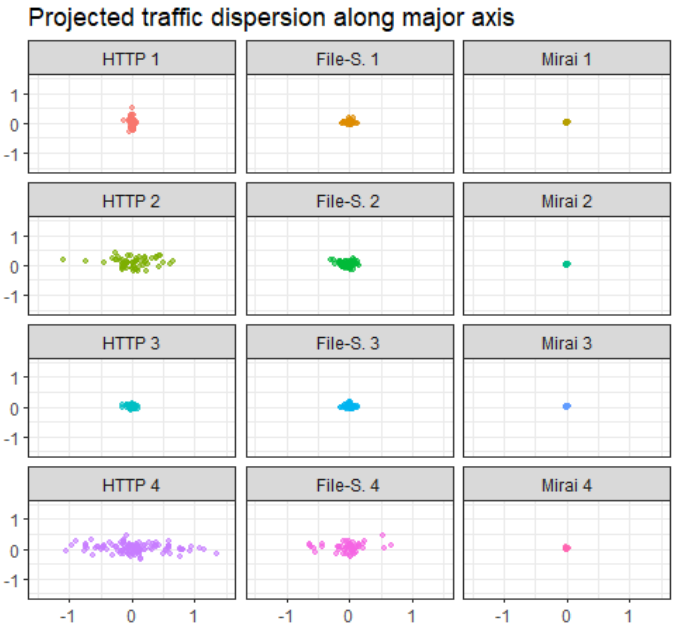

Fig. 3: Dispersion of projected traffic samples from each setting, plotted along the two most dispersed axes.

the cause for the dispersion is the same for the different traffic types. When we look at the the influence of the individual flow features on projected position, we notice that even slight differences in the IAT of the preceding and the subsequent flow impacts the projected position quite strongly, which is why only the settings that generate multiple flows are affected.

B. *Investigating individual cluster incoherences*

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. 4. 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. 4, 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.

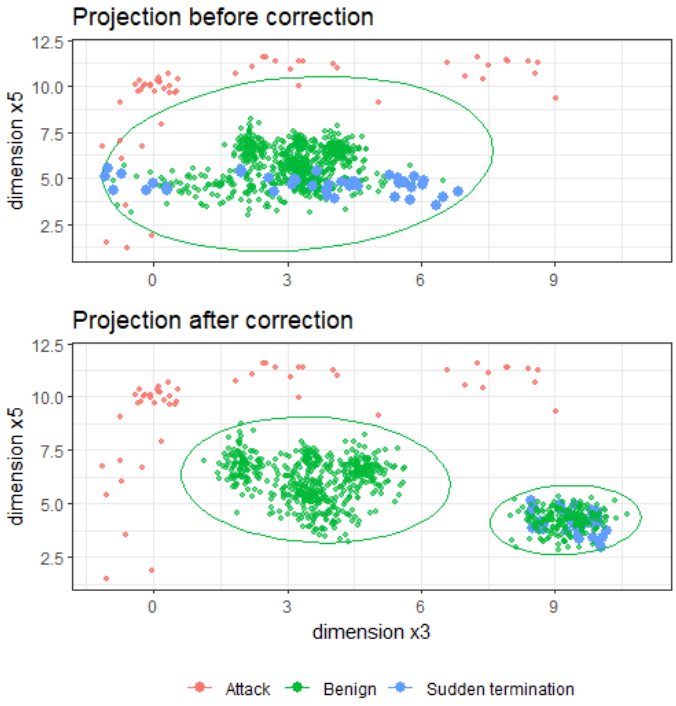

Fig. 4: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold

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

V. Conclusions

In this paper, demonstrated the impact of traffic generation with extensive micro-structure control as well as detailed corresponding documentation on researchers ability to evaluate and understand network intrusion detection models. We presented two use-cases in which we implemented and trained state-of-the art detection models before performing and detailed evaluation of their behaviour when encountering different traffic types.

By using HTTP-traffic with congestion settings, we were quickly able to identify the inability of an LSTM-based classifier to handle traffic with significant retransmission rates, which enabled us to improve the model accordingly and increase detection performance by more than 2%. Similarly, the examination of projection consistency of a subspace-clustering method using traffic with artificially similar characteristics revealed an overly high sensitivity to flow interarrival times, while cluster-coherence could be increased significantly by identifying half-open connections that were dropped because of network failure as the source of overly dispersed traffic projections.

These results should encourage researchers to perform more deep-going examinations of data-driven network intrusion detection models. It also shows that DetGen offers strong insights into traffic micro-structures and their effect on traffic models, and allows researchers to analyse the particular characteristics of events that lead to false-positives or model failure as well as their effect on model training.

A. *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 micro-structures, 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 behaviour 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.

Furthermore, while controlling traffic shaping factors artificially helps at identifying the limits and weak points of a model, it can exaggerate some characteristics in unrealistic ways and thus both affect the training phase of a model as well as tilt the actual detection performance of a model in either direction. Additionally, the artificial randomisation of traffic shaping factors can currently not generate the traffic diversity encountered in real-life traffic and thus only aid at exploring model limits extensively. The lack of realistic traffic heterogeneity however is at the moment significantly more pronounced in commonly used network intrusion datasets such as the CICIDS-17 dataset, where the vast majority of successful FTP-transfers consist of a client downloading a single text file that contains the Wikipedia page for ‘Encryption’ insert DetGen citation.

B. *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.

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.

REFERENCES

[1] 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.

[2] A. Biernacki. Analysis and modelling of traffic produced by adaptive http-based video. *Multimedia Tools and Applications*, 76(10):12347–12368, 2017.

[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] C. Fricker, P. Robert, J. Roberts, and N. Sbihi. Impact of traffic mix on caching performance in a content-centric network. In *2012 Proceedings IEEE INFOCOM Workshops*, pages 310–315. IEEE, 2012.

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

[8] R. Marx, J. Herbots, W. Lamotte, and P. Quax. Same standards, different decisions: A study of quic and http/3 implementation diversity. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pages 14–20, 2020.

[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. Stöber, M. Frank, J. Schmitt, and I. Martinovic. Who do you sync you are? smartphone fingerprinting via application behaviour. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 7–12, 2013.