Examining traffic micro-structures to improve model development

Henry Clausen
University of Edinburgh
Edinburgh, UK
henry.clausen@ed.ac.uk

David Aspinall
University of Edinburgh
Alan Turing Institute
Edinburgh, UK
david.aspinall@ed.ac.uk

***Abstract*—We demonstrate how machine-learning-based network intrusion detection models can be validated and developed by probing models using traffic with specifically controlled micro-structures. We show our methodology by probing two published state-of-the-art models to find classification flaws and and understand misbehaviour. These models fail for input traffic with particular characteristics such as retransmissions or overly dispersed flow interarrival times. After we make simple corresponding model corrections, detection rates already improve between $2-4\%$. We believe this shows promise for using tailored data with controllable and labelled characteristics to effectively improve model development in NID, a practice that helped model development significantly in several other areas of machine-learning.**

***Index Terms*—Machine learning, traffic micro-structures, network intrusion detection**

I. Introduction

The model development process of machine-learning (ML) based network intrusion detection (NID) models usually ignores specific traffic characteristics and lacks the ability to extensively explore model failings. The main reason for this is likely the lack of precise datasets with specifically curated characteristics and corresponding information. In this paper, 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 and corresponding descriptive labelling to develop more suitable models. 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 [12]. 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 to traffic datasets in the hope of edging out competitors. 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' by Paxson and Sommer [13], and is seen to be partly responsible for the lack of success machine-learning had in network intrusion detection. This claim has been supported and partly extended by Harang [5] in 2014 and by Liu et al. in 2019 [8].

In this work, we aim demonstrate the usefulness of the control and information on traffic micro-structures for model validation and development. We show the inspection of two state-of-the-art network intrusion detection models with specially generated traffic to identify model flaws, understand model behaviour better, and subsequently boost corresponding results. We hope to find new ways to improve model development in NID and increase the efficiency with which models can learn traffic micro-structures.

A. Outline

The remainder of the paper is organized as follows. Section II discusses the necessity for probing to validate and understand ML-models, and our methodology of using traffic micro-structure control for model probing. In Sections III and IV we demonstrate how to perform model probing and implement corresponding design improvements on two network intrusion detection models. Section V concludes the results and discusses limitations of our work and directions for future work.

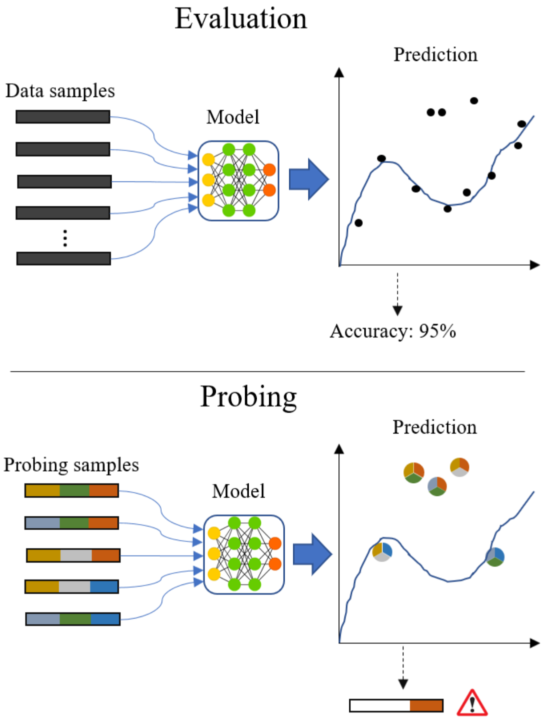

Fig. 1: Comparison between numerical model evaluation and model probing with specifically controlled data characteristics, indicated as colours.

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 **model probing**. Model validation is essential to understand how particular data structures are processed, and enables researchers to develop their models accordingly. The emergence of several perturbative model analysis tools that give insights into which structures are important in the decision-making of deep learning methods underline the importance of proper model validation. Existing tools such as the *What-If tool* [18] are however either only capable of computing perturbations for small data sequences, or rely on domain-knowledge such as *DeepLIFT* [11]. In the case of network traffic models, the lack of ground-truth labels for individual traffic traces prohibits us to associate patterns in a packet sequence with specific traffic shaping characteristics such as activities or congestion.

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-micro-structures**, 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 the communication protocol as well as external effects such as network reliability or available computation resources. An analogue to this is speech, were influence factors such as accents or emotions can introduce structural characteristics to a conveyed message.

B. Generating controllable traffic micro-structures

We use DetGen [3], a tool we developed that generates various types of traffic with a fine-grained control over traffic shaping factors. In this work, we look traffic generated from scenarios that include regular HTTP communication, requests to an SQL-server, multi-host file-synchronisation, SQL-injection attacks, botnet traffic, as well as FTP-, SSH-, and SMTP-communication. The tool offers control over 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 [14].

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

c) Transferred data: The amount and content of transferred data influences the overall packet number, rate, and size such as shown by Biernacki [2] for streaming services.

d) 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 and less connections being established [4].

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.

In this work, we focus mainly on influence from the factors *a), b), f)* and partly *c).*

Labels that describe the respective setting for each factor are attached to each traffic sample after generation, thus enabling us to provide ground-truth information about the precise generation setting of individual samples. 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 probing data to make up at least 10% of the particular traffic type in the training set. This is to ensure that the model is able to see and learn the structures in the probing data, 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. Improved traffic separation for a classifier with congestion level information

Our first example looks at how descriptive ground truth information on traffic characteristics can improve a traffic classification model 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. [7] as an example, which aims at distinguishing various types of malicious activity from benign traffic. The model achieved some of the highest detection rates of packet-based classifiers in a recent survey [15]. The model classifies connections on a packet-level using a *Long-short-term memory* (LSTM) network [1], and is claimed to achieve detection and false-positive (FP) rates of **99.7%** and **0.03%** respectively.

We train a model on a set of different HTTP-activities in order to detect SQL-injections. Rather than providing an accurate and realistic detection setting, this example shows how traffic information can be linked to model failures and slumping performance. We use real-world HTTP-traffic from the *CAIDA anonymized traffic traces* [16] as background traffic (85% of connections) and add SQL-injection attack traffic (7.5%) as well as different HTTP-activities for probing (7.5%). In total, we use 50,000 connections for training the model, or slightly less than 2 million packets.

The initially trained model performs relatively well, with an *Area under curve* (AUC)-score[2] of **0.981**, or a detection and false positive rate[3] of **0.96%** and **2.7%**. However, these rates are still far from enabling operational deployment. Now suppose we want to improve these rates to both detect more SQL-injections and retain a lower false-positive rate.

We initially explore which type of connections are misclassified most often. For this, we perform a correlation analysis between the numeric or categoric labels available for the probing data, and the binary response whether the corresponding connection was misclassified. Unsurprisingly, the highest correlation to misclassification was measured for the conducted activity, with a particular attack scenario (19% correlation) and connections with multiple GET-requests (11% correlation) being confused most often. This was followed by the amount of simulated latency (12% correlation), which we are now examining closer.

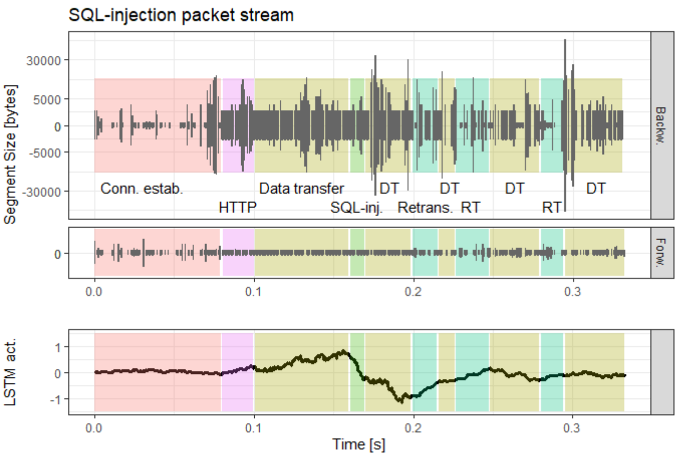

Fig. 2: LSTM-output activation in dependence of connection phases. Depicted are packet segment streams and their respective sizes in the forward and backward direction, with different phases in the connection coloured and labelled. Below is the LSTM-ouput activation while processing the packet streams.

Fig. 3 depicts classification scores of connections in the probing data in dependence of the emulated network latency. The left panel depicts the scores for the initially trained model, which shows 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. Increased latency can both increase variation in observed packet interarrival times (IATs), and lead to packet out-of-order arrivals and corresponding retransmission attempts. Both of these factors can decreases the overall sequential coherence for the model, i.e. that the LSTM-model loses context too quickly either due to increased IAT variation or during retransmission sequences.

To examine the exact effect of retransmission sequences on the model output, we generate two similar connections, where 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. 2 depicts the evolution the LSTM-output layer activation in dependence of difference connection phases. 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

[1] a deep learning design for sequential data

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

[3] tuned for the geometric mean

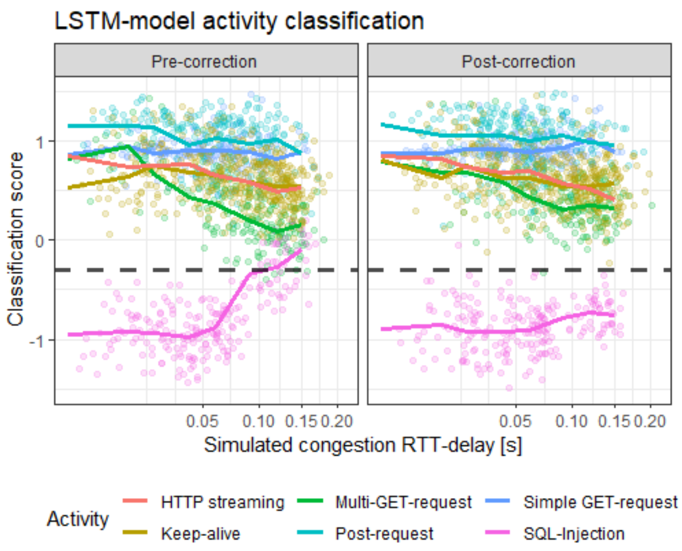

Fig. 3: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold.

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, both during training and classification. This leads to significantly better classification results during network latency, as visible in the right panel of Fig. 3. 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

Next, we show how ground-truth traffic information can help produce more coherent clusters and thus refine the benign traffic model in anomaly-detection. In particular, we will examine a simplified version of *Kitsune* [9], a recent deep learning anomaly-detection model based on stacked autoencoders. *Kitsune's* AUC-scores surpassed those of other state-of-the-art methods for a variety of attacks, including various types of Botnet traffic and *man-in-the-middle* attacks.

The model takes connection packet streams as input, which are pushed through an artificial information bottleneck before reconstruction, which forces the model to learn and compress reoccurring traffic structures. The compressed connection representation is essentially a positional projection into a lower-dimensional vector space, where spatial boundaries around benign traffic can be drawn. For demonstration purposes, we use a widely-used clustering approach for anomaly-detection rather than *Kitsune's* more complex ensemble method. Here, anomalous outliers are detected using the Mahalanobis-distance of a projected connection from identified cluster centers. Benign traffic should ideally be distributed evenly around the cluster centres to allow a tight borders and good separation from actual abnormal behaviour.

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

TABLE I: Outline of the traffic settings for examining projection consistency. The numbers below each setting describe the measured Mahalanobis-distances (blue:average, red:maximal) for the corresponding projections.

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 [10] intrusion detection dataset (80%). Again, we add 20% probing traffic consists of HTTP, FTP, SSH, and SMTP communication, 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 or Brute-Forcing, as well as Mirai botnet traffic. We train the model with in total 150,000 connections.

A. *Projection coherency evaluation*

Like many approaches that generate representations of benign traffic for anomaly detection, *Kitsune* projects 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 [6].

To verify the models projection consistency, we generate traffic from near-identical conditions to provide certainty on the expected traffic similarities. We generate a small dataset that consists of HTTP-requests, file-synchronisation, and Botnet communication. For each of the three traffic types we fix four settings that vary in the performed activity and network latency, with the traffic shaping described in Section II-B being held constant within each setting except for small variations in the transmitted message or file. Table I summarises the traffic for 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. 4. The first thing to notice is that the model projects samples from each group within the same cluster, thus confirming the capture of a 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, especially when exposed to more latency. We also find that the corresponding dimension, x_3, with the most projected dispersion seems to be the same for each of the four settings. This suggests that the cause for the dispersion is the same for the different traffic types.

We now focus on the influence of input features on the projected positions exclusively in the x_3-direction. Here, we can again perform a simple correlation analysis between different the input feature values and the corresponding x_3-value. We observe that the arrival time of packet bears the most correlation (5.4%) for the selected settings. We also see that this influence is concentrated primarily on connections that are opened shortly after a previous connection, with the temporal separation between these two connections apparently being the primary cause for the spread on the x_3-axis. The connection interarrival times are naturally an important feature for *Kitsune* to detect attacks such as *Man-in-the-Middle*, which could explain the weight this feature plays in the projection process.

B. Investigating individual cluster incoherences

When examining false-positive and corresponding anomaly scores, we noticed that 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. 5. A significant portion of traffic in that cluster seems to be spread significantly 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 probing data, we see that HTTP-traffic with the label "Sudden termination" are distributed across the cluster axis in a similar fashion, also depicted in Fig. 5, 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 attached to the end of a packet sequence 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 coherence. The detection rate on Brute-Force attack traffic could thus be improved from **89.7%** to **94.1%**.

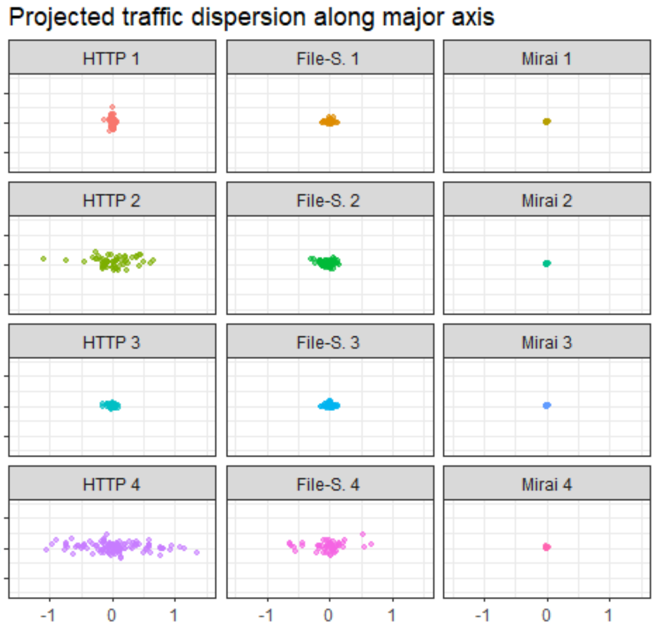

Fig. 4: Dispersion of projected traffic samples from each setting, plotted along the two most dispersed axes.

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 implemented and trained two state-of-the art detection models before extensively probing their behaviour and limitations 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 have encouraged us to perform more deep-going probing of data-driven network intrusion detection models. We believe that in combination with strong NID-dataset,

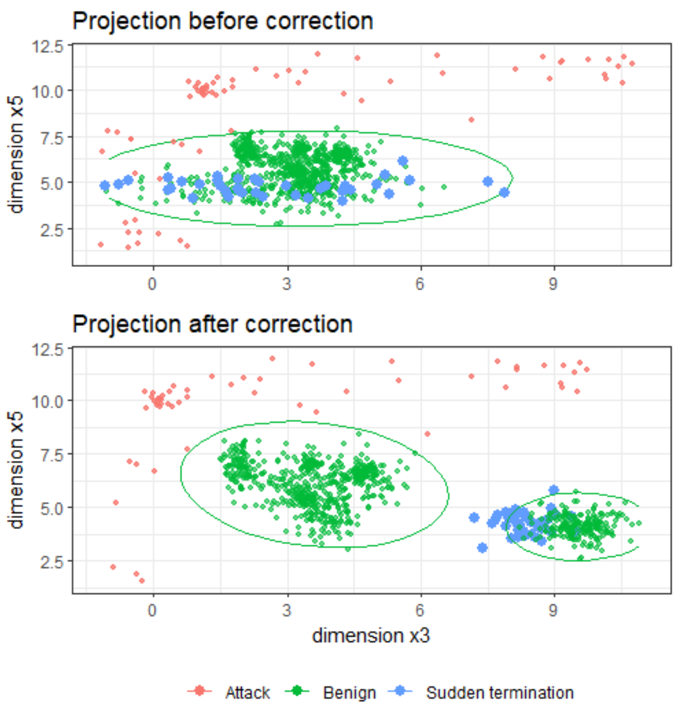

Fig. 5: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold

extensive model validation and corresponding development with targeted traffic samples might hold the key to reduce false positives of detection models to an acceptable rate, as well as help models replicate detection rates in practical settings.

DetGen is openly accessible on GitHub.

A. *Models suitable for probing*

Refining the ability of classifiers to identify traffic characteristics related to particular activities can lead to privacy infringements or even discrimination against users or traffic types. It is therefore important to also investigate how applications can implement privacy-enhancing measures that conceal traffic characteristics related to particular content or sensitive applications, such as proposed by Wang et al. [17].

B. *Difficulties and limitations*

Controlling traffic shaping factors artificially can exaggerate some traffic 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’.

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] H. Clausen, R. Flood, and D. Aspinall. Traffic generation using containerization for machine learning. In *DYnamic and Novel Advances in Machine Learning and Intelligent Cyber Security (DYNAMICS) Workshop*, 2019.

[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] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141. IEEE, 2017.

[7] 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.

[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] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[11] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

[12] 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.

[13] 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.

[14] 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.

[15] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar. The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications*, 154:102538, 2020.

[16] C. Walsworth, E. Aben, K. Claffy, and D. Andersen. The CAIDA UCSD anonymized internet traces 2018,”, 2018.

[17] T. Wang and I. Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1375–1390, 2017.

[18] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.