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, 6 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 feed-forward and recurrent neural 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 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. In-depth analyses regarding which traffic characteristics lead to inaccurate predictions or cause a model to misbehave, 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.

A significant cause 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 in the way that is common practice in other areas. With the exception of very specific applications, we are not aware of any testbeds or datasets that aim to accurately document the generation and traffic shaping process. For this reason, 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 documenting different traffic shaping factors such as the exact corresponding computational activities, failure modes, or data input that impact traffic on a packet and flow level.

In this work, we aim demonstrate the usefulness of the control and information on traffic micro-structures that such a tool can provide. 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.

Advantages of traffic

- *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

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

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 substantial set of scripted *scenarios* setting such as HTTP server-client with various server implementations, file-synchronisation between several hosts, or malicious settings such as an SQL-injection attack with SQL-backend server. Each scenario consists of several sub-activities such as specific application actions (GET, POST, etc.) and input modes (content caching, wrong password, connection termination, etc.) as well as external influences (network congestion, host load, etc.) to strictly control major traffic shaping factors. 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

Extensive ground-truth labels on different traffic influences as well as the ability to generate traffic with specific characteristics are arguably the most important contribution of the DetGen framework. We demonstrate their benefits for model-understanding and evaluation on two examples using a recent traffic classification model by Hwang et al. [6] and a highly regarded anomaly detection model by Casas et al. [3].

2.1 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.

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

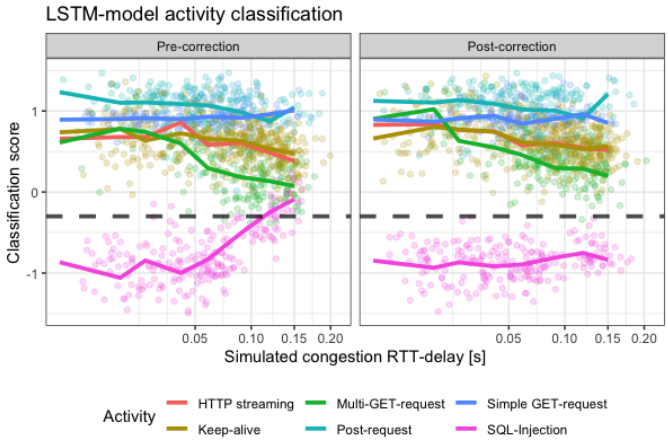

Figure 1: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold

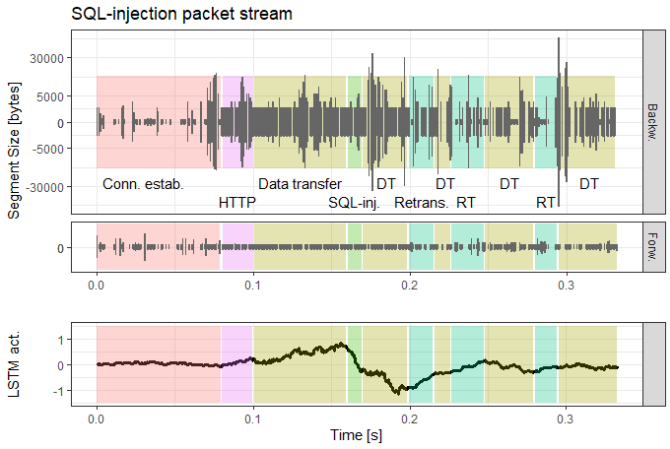

Figure 2: LSTM-output activation in dependence of connection phases. Depicted are packet segment streams and their respective sizes in the forward and backward direction, with different phases in the connection coloured and labelled. Below is the LSTM-ouput activation while processing the packet streams.

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 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.

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. 2 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 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%**.

2.2 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.

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

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.

2.2.1 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

[2] a measure describing the overall class separation of the model
[3] tuned for the geometric mean

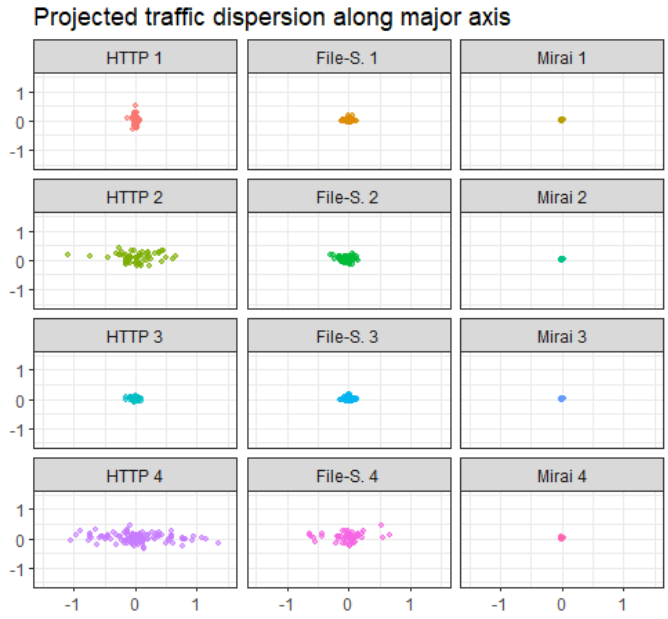

Figure 3: Dispersion of projected traffic samples from each setting, plotted along the two most dispersed axes.

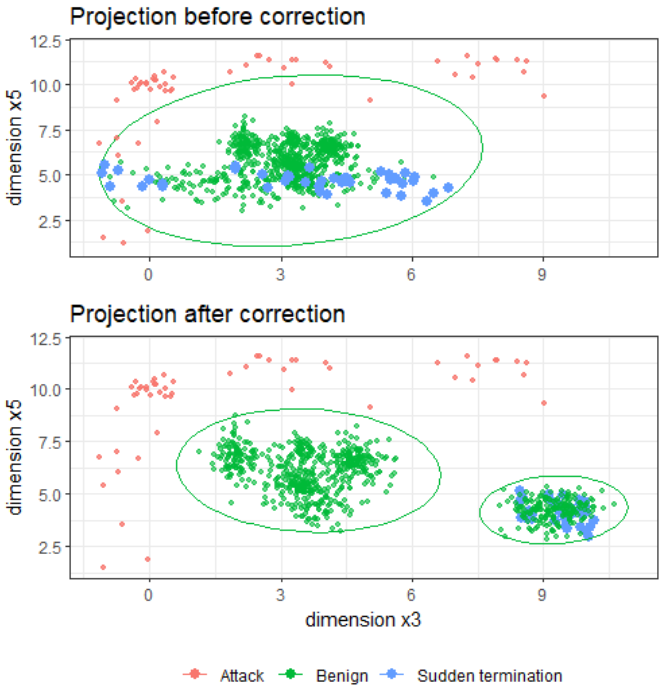

Figure 4: Scores for the LSTM-traffic classification model in dependence of simulated network congestion, along with the classification threshold

constant. Within each setting, we only allow for minor randomised effects such as packet reordering or slight variations in the activity input. Table 1 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 1 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 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.

2.2.2 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.

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

3 BACKGROUND AND MOTIVATION

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.

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.

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 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 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.

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.