Can Machine Learning Techniques be effectively used in real networks against DDoS attacks?

Jarrod N. Bakker
Department of Internal Affairs
45 Pipitea St, Thorndon, Wellington 6011, New Zealand
email: jarrod.bakker@dia.govt.nz

Bryan Ng and Winston K.G. Seah
School of Engineering & Computer Science
Victoria University of Wellington, New Zealand
{bryan.ng,winston.seah@ecs.vuw.ac.nz}

***Abstract*—The threat of distributed denial of service (DDoS) attacks has worsened recently with the proliferation of unsecured Internet of Things (IoT) devices. Detecting these attacks is often difficult when using a traditional networking paradigm as network information and control are decentralised. We study the effectiveness of using machine learning (ML) to detect DDoS attacks, facilitated by Software-Defined Networking (SDN), a recent paradigm that aims to improve network management by centralising network information and control. In this study, ML algorithms are implemented on *nmeta2*, an SDN-based traffic classification architecture, and evaluated on a physical network testbed to demonstrate their efficacy during a DDoS attack scenario, especially in accurately classifying non-malicious traffic. This is unlike most approaches that aim to identify/classify malicious traffic but also misclassify non-malicious traffic, inadvertently leading to degraded performance for legitimate network traffic. Furthermore, there is potentially considerable data loss during DDoS attacks that can further degrade classification performance. We examine these issues that arise when using ML to detect DDoS attacks in live network scenarios.**

I. Introduction

Distributed denial of service (DDoS) attacks utilise many attacking entities to prevent legitimate use of a resource via consumption [1]. Though motivations for carrying out such attacks differ, the aim is to disrupt the victim or victims. The scale of DDoS attacks often means that the infrastructure used to forward the malicious traffic becomes collaterally damaged.

The debilitating potential of DDoS attacks has increased with the advent of the Internet of things (IoT). IoT has been a disruptive agent within the domain of computer networks as objects, such as, fridges and security cameras now have connectivity to the Internet. The ubiquity of IoT devices has unfortunately attracted the attention of malicious parties. The world has been repeatedly shown that IoT devices are vulnerable to being used as a platform to perform DDoS attacks, e.g. the Mirai botnet on KerbsOnSecurity in 2016 [2] and other recent IoT botnets like Reaper [3].

The ability to gather information from network forwarding elements into a centralised location makes Software-Defined Networking (SDN) an ideal candidate for traffic classification (TC), which is a process whereby network traffic is classified to assist in the management of network resources, network security and quality of service (QoS) provisioning [4]–[6]. By determining the nature of traffic within a network, network operators can better respond to extreme, as well as, subtle changes in traffic behaviour in a timely manner.

Machine learning (ML) approaches have shown promising results in off-line test environments with collected network traffic data. However, it comes with a serious undesirable side effect of misclassifying legitimate traffic as malicious DDoS traffic. Worse, considerable data loss is likely in a live network especially during DDoS attack scenarios leading to the question of whether ML techniques can perform as well under such adverse conditions. This motivates our study with an emphasis on SDN-based network architectures.

The structure of this paper is as follows. Section II presents the related work. Section III describes the process for selecting classifiers to be tested on a physical network testbed. Section IV evaluates the performance of the selected classifiers. In Section V, we discuss key observations from this study and Section VI presents our conclusions and future work.

II. Related Work

The logically centralised control-plane offered by SDN has made it an attractive platform for detecting DDoS attacks and performing TC in general. This shows its versatility within the contexts of network management, security and QoS. Earlier approaches all exploited this feature by performing detection at the controller.

Firstly, Braga *et al*. [7] addressed the difficulties of distinguishing legitimate traffic from DDoS attack traffic by utilising the NOX OpenFlow controller to collect statistics on flow table entries and a Self-Organising Maps (SOM) algorithm. Mehdi *et al.* [8] investigated three anomaly detection approaches (rate-limiting, entropy and NETAD) to detect TCP portscan, TCP SYN-flood and UDP flood. Their results showed that entropy and NETAD methods did not perform as well as the rate-limiting method. Qian *et al*. [5] classified HTTP traffic within a 3G mobile network data-plane. Their solution is unique compared to previous work as they utilise signatures based on HTTP headers instead of just categories based on port number. Giotis *et al.* [9] utilised sFlow to collect flow statistics from switches every 30 seconds, which they claim represents nearly real-time detection, and applied entropy and TRW-CB. When using entropy, the FPR was high ranging from 23% to 39.3%, which is characteristic of anomaly detection approaches. Generalising from specifically detecting

TABLE I
SUMMARY OF RELATED RESEARCH. LOCATION: WHERE THE CLASSIFICATION OF TRAFFIC OCCURS; METHOD: METHOD USED TO CLASSIFY TRAFFIC.

Author	Year	Traffic Type	Location	Method	Environment
Braga *et al*. [7]	2010	DDoS	Controller	SOM	Virtualised
Mehdi *et al*. [8]	2011	Anomaly	Controller	Rate-limiting, Entropy, NETAD	Virtualised
Qian *et al*. [5]	2013	HTTP	Controller	BLINC	Not stated
Ng *et al.* [4]	2014	Any	Controller	Static (primarily)	Virtualised
Giotis *et al.* [9]	2014	DDoS	Controller	Entropy and TRW-CB	Physical
Wang *et al*. [10]	2015	DDoS	OpenFlow edge switch	Entropy	Virtualised
Lin *et al*. [11]	2015	SYN Flood & Web attacks	Switch and NFV	Packet header analysis	Virtualised
Hayes *et al*. [12]	2016	Any	DPAE	Static (primarily)	Virtualised

anomalous traffic, especially DDoS, to a generic TC platform is *nmeta* by Ng *et al.* [4], that allows the network operator to define their own classifiers. Traffic can be classified using static, identity or ML techniques.

Controller-based approaches do not scale especially under adverse DDoS attack scenarios. Switch-based solutions were the obvious alternatives. Wang *et al*. [10] implemented an entropy-based DDoS attack detection mechanism by modifying the Open vSwitch software switch to count the number of packets received within a predefined time period. Their entropy-based detection algorithm suffered the same drawback of high FPR, at 25%. To detect SYN flooding and web application attacks, Lin *et al*. [11] used a two-tier SDN TC architecture. At tier one, a classification module on a switch inspects TCP/IP and application headers. Failing a classification at tier one, traffic is sent to tier two where it is subject to DPI on a network function virtualisation (NFV) module. This reduced the amount of traffic sent from the data-plane to the control-plane by 99.95% when classifying HTTP packets with a layer 7 load balancer but they did not consider the accuracy of the distributed classification mechanism.

Moving classification/detection away from both controller and switch to address the performance and scalability concerns that had been identified [4], Hayes *et al.* [12] further developed *nmeta* into *nmeta2*. Switches forward traffic to a separate host running the Data Plane Auxiliary Engine (DPAE) to perform classification as required and results are sent to the controller via a dedicated connection over the control-plane. As well as supporting the same classification techniques as *nmeta*, *nmeta2* also supports payload inspection.

The related work explored above demonstrates variety in several ways. First, SDN/OpenFlow has been combined with classification techniques to classify network in various scenarios. Although the TC use-case in this study concerns DDoS attacks, it is important to note that SDN TC can be used for other traffic patterns, both existing as well as those from new IoT devices.

The use of anomaly detection techniques such as information entropy is popular within DDoS attack detection. Mechanisms that used entropy have reported a high FPR. False positives are near impossible to avoid but they cannot be ignored within a networking context. The law of truly large numbers suggests that a FPR of 5%, which may be considered to be small in domains outside of TC, can result in large amounts of traffic being misclassified as the total number of flows only increases with time.

Classification mechanisms traditionally operate on the controller. Lin *et al*. [11] suggest that the processing overheads imposed by classifiers and detection algorithms on a controller can be reliably removed by moving the function to another device. This move is well justified as it has been shown to be scalable and necessary [12].

III. CLASSIFIER SELECTION

This section provides an overview of the process for selecting classifiers to detect DDoS attacks on a physical network testbed; details can be found in [13]. The investigation started with the exploration of various statistical classification methods. The methods were combined with network traffic features to form classifiers and tested against DDoS attack scenarios in two off-line experiments to determine their effectiveness. Each scenario is contained in a Packet Capture (PCAP) file from the dataset provided by the Information Security Centre of Excellence (ISCX) at the University of New Brunswick (UNB) [14].

A. Method Selection

The method selection started with a set of seven well-known and established classification methods that have been used to solve classification problems inside and outside the domain of networking. The purpose of this study is to apply existing methods in an SDN environment to demonstrate their feasibility, and not develop new classification methods. The seven selected methods are: Linear Discriminant Analysis (LDA) [15], [16], Quadratic Discriminant Analysis (QDA) [16], Support Vector Machine (SVM) [16], k-nearest neighbours (KNN) [17], Naive Bayes [17], Decision Tree [17] and Random Forest [18].

B. Feature Selection

We next identified features that describe network traffic flows, as listed in Table II. These features describe two statistical properties of traffic: (i) amount of information sent in one direction and (ii) duration of a connection. These properties can be used to describe the standard behaviour of a host, with deviations from standard behaviour suggesting anomalous behaviour such as a DDoS attack. A DDoS flooding attack for instance would result in a large volume of traffic being sent over a shorter period of time than a normal session.

TABLE II
FEATURES USED TO DESCRIBE NETWORK TRAFFIC.

Feature		Details
totalSourceBytes	tSB	Number of bytes sent by *source*.
log(totalSource-Bytes)	ltSB	Base10 logarithm of totalSourceBytes.
totalDestinationBytes	tDB	Number of bytes sent by *destination*.
totalSource-Packets	tSP	Number of packets sent by *source*.
FlowDuration	FD	Duration of a bidirectional flow (secs).
log(Flow-Duration)	lFD	Base10 logarithm of FlowDuration.
SourceBytes-per-Packet	SBpP	Number of bytes sent from *source* divided by number of packets sent by *source*.
DestinationBytes-per-Packet	DBpP	Number of bytes sent from *destination* divided by number of packets sent by *destination*.

source – host/client that sent the first packet within the flow; *destination* – recipient/server of the first packet.

Each feature set combined the different statistical properties of network traffic to form a different set of predictors to be used by the classification methods.

C. Experiment Setup for Offline Training

We use the *Scikit-learn* open source ML library/package which contains implementations of various algorithms for performing classification, regression and clustering tasks. The ISCX data was split into training and testing sets using the k-fold cross validation technique. This technique has advantages over hold-out validation where the dataset is simply split into two parts [19], which is known to result in over-fitting. We set k to 30 and used the folds in an unorthodox fashion in order to accommodate classifiers utilising the SVM method. k-fold cross validation traditionally uses $k-1$ folds to train the classifier with one fold being used for testing. Due to the size of the dataset (571698 samples), training an SVM-based classifier with 29 folds (each with roughly 19056 flows) results in an impractically long training time as the SVM training algorithm is a quadratic optimisation problem. As a result, the training and testing sets were swapped for all experiments, i.e. one fold was used for training and 29 folds were used for testing. Each experiment was repeated ten times to increase the number of results for evaluation. Scikit-learn supports this by providing methods to randomly generate k folds; we used the *StratifiedKFold* class. Taking into consideration the cross validation process and the repeated experiments, each classifier was subjected to 300 classification trials.

D. Prediction Metrics

The predictions made by classifiers can be collected to form a suite of performance statistics [20], of which the following are relevant to our study. The metrics are calculated from the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

The *true positive rate* (or recall) is the ratio of successful predictions made to cases of class A, referring to an arbitrary class of positive predictions (e.g. malicious traffic.) This is referred to as the *detection rate* (DR) and defined as follows:

$$DR = \frac{TP}{TP + FN}. \quad (1)$$

TABLE III
SELECTED CLASSIFIERS

Method	Features	f-measure
Random Forest	tSB, tSP, tDB, tDP and FD	0.954795172
k-Nearest Neighbours	tSB, tDB and FD	0.948387955
Support Vector Machine	ltSB, tSP and FD	0.931058739

The *false positive rate* (FPR) is the ratio of unsuccessful predictions made to cases of class A', the inverse of A that refers to negative predictions (non-malicious traffic) and defined as follows:

$$FPR = \frac{FP}{TN + FP}. \quad (2)$$

Accuracy is the ratio of successful predictions made to both classes, defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (3)$$

F-measure (or F1 score) considers both the DR and precision of a classifier to measure its quality, defined as:

$$F\text{-}measure = 2 \times \frac{DR \times Precision}{DR + Precision}. \quad (4)$$

where *Precision* (or positive predictive value) is the ratio of correct predictions made for class A, and defined as:

$$Precision = \frac{TP}{TP + FP}. \quad (5)$$

The prediction results for each flow must be recorded and collected during the replay of the PCAP file through the classifiers. Once collected, the number of TP, TN, FP and FN were determined, and the mean values of DR, FPR and F-measure calculated for each classifier. These metrics were then used to select the most suitable classifiers [13].

IV. EVALUATION

From the selection process described in the previous section, we identified three classifiers (Table III) to be integrated with *nmeta2* and evaluated on a physical network testbed. The suitability of each statistical classifier when being deployed within a SDN/OpenFlow environment was assessed. This evaluation assessed the prediction performance and the execution performance provided by each classifier. The experiments performed on the testbed environment are described as on-line experiments, and this is in contrast to the off-line experiments mentioned in Section III.

A. Test Environment

Fig. 1 illustrates the network testbed topology and provides a high-level overview of the traffic sent between each device. The network consists of two parts: (i) the first for network traffic sent over the data-plane and (ii) the second for the control-plane and test automation. The data-plane consisted of three hosts and a switch:

- Traffic Source: replays network traffic
- Sink: receives the traffic
- DPAE: runs *nmeta2* DPAE
- AT-x930-28GSTX: an OpenFlow 1.3 compliant switch

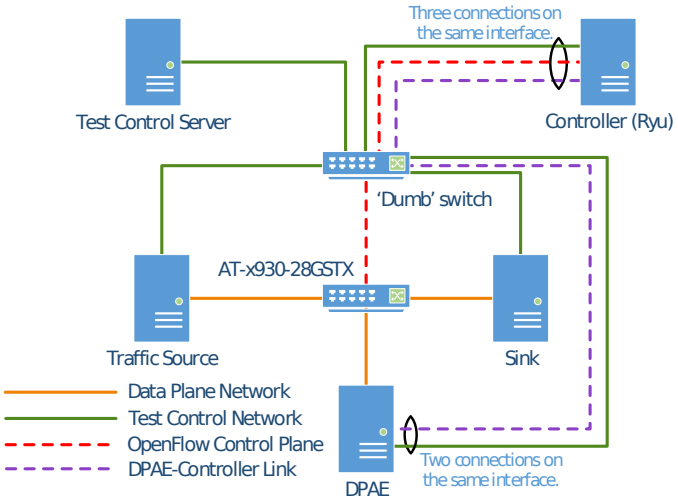

Fig. 1. Physical network testbed topology used for the on-line experiments.

TABLE IV
MEAN CLASSIFIER INITIALISATION TIMES

Classifier	DPAE	Mean Proc. Time (s)	Mean Wall Time (s)
Random Forest	Active	5.6848638	6.258123016
KNN	Active	5.448394	6.004147053
SVM	Active	443.2090188	443.9620652
Random Forest	Passive	5.7031904	6.177779818
KNN	Passive	5.4808608	6.127974558
SVM	Passive	443.1051298	443.8416832

TABLE V
MEAN PACKET PROCESSING TIMES

Classifier	DPAE	Mean Proc. Time (s)	Mean Wall Time (s)
No classifier	Active	0.000512707	0.000974115
Random Forest	Active	0.004153738	0.004366805
KNN	Active	0.001187487	0.001447127
SVM	Active	0.000788909	0.001097702
No classifier	Passive	0.000480256	0.001844048
Random Forest	Passive	0.004201095	0.004400715
KNN	Passive	0.001138495	0.002002197
SVM	Passive	0.000740066	0.002011115

The remaining hosts were:

- Test Control Server: executes Ansible playbooks to orchestrate experiments
- Controller (Ryu): runs the *nmeta2* controller application

DPAE runs in two modes: active and passive with flow-level packet sampling. In active mode, packets that require traffic classification beyond the capabilities of the switch are forwarded by the switch to the DPAE to be processed whereas in passive mode, packets are cloned by the switch and sent to the DPAE. The three classifiers and a control experiment (i.e. *No classifier*) were tested using the two DPAE modes.

B. Classifier Execution Performance

Three sets of measurements are used to assess the classifiers' performance: initialisation time, packet processing time and number of packets processed. Tables IV and V show two time keeping methods: the elapsed time (i.e. *Wall Time*) and the time taken for the code to run on the processor (i.e. *Proc. Time*). These were taken by calling Python's `time.time()` and `time.clock()` functions respectively. The former was chosen as it captures the effects of receiving traffic, especially DDoS attack traffic, on ML-based statistical classifiers.

1) Classifier Initialisation Time: This is defined as the time taken for a classifier to be in a state where it can make predictions. This includes the time taken for training data to be loaded from file as well as the time taken for the classifier to train. These results are displayed in Table IV.

Both instances of the KNN classifier were the quickest to initialise within their respective DPAE mode scenarios and time keeping methods. This result is unsurprising as the KNN method is a lazy learner [21]. Each KNN classifier was followed closely by the Random Forest classifiers. The SVM classifier took the longest to initialise, taking roughly 7 minutes 23 seconds.

The difference between the mean processor and wall times for each classifier-DPAE scenario is fairly consistent. This suggests that the DPAE host was able to consistently concentrate on the initialisation process without being interrupted by other processes on the host. It is well understood that modern operating systems (OS) share processor time among numerous processes. A significant difference between each time keeping method time might indicate that modern OSs are not suitable for performing this kind of classification. An alternative option might involve the use of dedicated hardware running a unikernel; however, this would cost more to develop in a temporal and monetary sense.

2) Packet Processing Time: This is defined as the time taken for a classifier to gather the required information to make a prediction and then make the prediction itself. These results are displayed in Table V.

The results confirm that the mean packet processing for the control experiment is smaller compared to when a classifier is used. They also show that the SVM-based classifiers have smallest packet processing times on average compared to the other classifiers. It would be expected that KNN-based classifiers would have the highest mean packet processing time since they are considered to be lazy learners [21]; however, this was not the case.

The mean processor time for each classifier scenario was higher when the DPAE was in active mode, except for the Random Forest classifier. This trend fits the assumption that an active DPAE is put under more stress as it must process all traffic being forwarded to its attached switch. The same trend is not observed when using mean wall time. This is counter-intuitive as it suggests that an active DPAE increases the processor time instead of the elapsed time from the perspective of the OS.

3) Prediction Accuracy: The prediction results presented in Table VI show very low DRs. This shows that the classifiers were ineffective in detecting the malicious flows in the DDoS attack. The accuracy measurements show promising results

TABLE VI
PREDICTION OF SELECTED CLASSIFIERS - RANKED BY MEAN ACCURACY.

Classifier	DPAE	Mean DR	Mean FPR	Mean Accuracy
SVM	Active	0.144378318	0.002040893	0.93468575
KNN	Active	0.000273643	0.000092519	0.92575702
KNN	Passive	0.000700116	0.000119214	0.92521195
Random Forest	Passive	0.000053412	0.002272015	0.92318754
Random Forest	Active	0.003937928	0.002657403	0.92252053
SVM	Passive	0.000749199	0.002534000	0.92159593

TABLE VII
NO. OF PREDICTIONS (PRED.) & PACKETS RECEIVED BY SINK (S. PKT.).

Classifier	DPAE	Mean Pred.	Mean S. Pkt.
No classifier	Active	22913939.4	24326964.6
Random Forest	Active	13182770.6	16099466.8
KNN	Active	21781835.4	23817711.8
SVM	Active	22798603	24116786.6
No classifier	Passive	22648513.2	20003418.0
Random Forest	Passive	13361685.6	20212835.4
KNN	Passive	24862596.8	19517048.2
SVM	Passive	23788578.2	19477950.4

however as each classifier was roughly 92% to 93% accurate.

Eqn. (3) showed how the calculated value is governed by the TP and TN terms in the numerator which explains the high accuracy despite the low DR. The high accuracy was attributed to the low mean FPR. The low mean DR for each classifier informs us that the number of TP predictions must have been smaller than the number of FN predictions. Similarly, the low mean FPR informs us that the number of TN predictions must have been larger than the number of FP predictions. Using these two pieces of information, one can infer that the high accuracy was due to a large number of TN predictions. This tells us that the classifiers were successful in correctly identifying non-malicious flows of traffic.

4) Number of Predictions: Table VII shows the mean number of predictions made (by DPAE) and the mean number of packets received by the sink for each classifier-DPAE scenario. Comparing these quantities against the number of packets in the PCAP trace (i.e. 34983042) allows us to quantify packet loss. Note that the DPAE will never make predictions for all packets in the PCAP trace as DNS, DHCP, LLDP and ARP packets are not classified. The largest mean number of packets received by the sink (i.e. 24326964.6) was roughly 10 million fewer packets than the 34983042 contained in the PCAP file. Another way of detecting packet loss involves counting the packets that are received by the sink. This could be done as all of the packets sent on the data-plane should be forwarded to the sink, regardless of their type.

The order of both columns in Table VII for the active DPAE scenario matches the order of both packet processing time columns in Table V for the same scenario. The same observation cannot be made for the passive scenario. By putting the DPAE into passive mode and giving the switch the responsibility to sample packets, the expected results have changed. The cause of this cannot be concluded given there are two contributing factors. However, the results suggest that switch-based packet sampling is not desirable. All traffic should ideally be forwarded to the DPAE for classification where sampling decisions can be made away from the data-plane.

The average number of predictions made by the Random Forest classifier is still less than the average number of packets received by the sink within the passive DPAE scenario. This is in contrast to the other set of experiments within the same DPAE scenario. This is to be expected as the mean packet processing times for the Random Forest method were higher than the other classifiers. Therefore, it is unsurprising that fewer predictions were made.

V. KEY OBSERVATIONS

The classifier execution results show that classifiers with shorter initialisation periods may not be desirable. Despite having the longest initialisation time, the SVM classifier had the least significant impact on the average packet processing time. The packet processing results for the Random Forest classifier indicate that it is the least suitable for deployment. Compared to the control experiment, the mean processor time increased by a factor of 10 (or 1000 μsecs.) SVM only increased the mean processor times by 200 μsecs. The longer packet processing time for the Random Forest classifier could be attributed to two things:

1) The selected Random Forest classifier utilised a feature set that contained five features whereas the other two classifiers utilised three. A larger feature set increases the time spent on comparing the values of features.
2) The Random Forest method relies on a search through a tree-like data structure.

An SVM classifier combined with the DPAE in active mode provided the highest F-measure and accuracy. Using the DPAE in a situation where no packet sampling was being performed proved to be advantageous. A greater volume of network traffic information typically resulted in more attacks being detected. The experiments also showed that the SVM classifier had the shortest packet processing time on average. This was reflected in the deficit between the amount of traffic received from a switch to the DPAE and the amount of traffic sent back. This information is important to consider when deploying ML techniques in networks, where realtime response is needed.

Anomaly detection approaches utilised in previous research demonstrated a tendency to misclassify non-malicious traffic despite having a high DR [9]. Ignoring the low mean DRs of the classifiers that were evaluated on the physical testbed, which can be attributed to packet loss during the DDoS attack itself, the mean FPRs were all smaller than 0.3%. The highest FPR experienced during our classifier selection experiment was no larger than 3%. These results suggest that statistical classification approaches can be used to reduce the number of misclassified non-malicious flows.

The use of SDN potentially enables us to discern if performance is caused by the control and/or data planes. The packet loss experienced by the DPAE during the DDoS attack suggests that further improvements to the data-plane are necessary. By better handling traffic during a DDoS attack,

more information can be gathered thus improving the chances of determining the offending flows. Making such conclusions using a traditional networking paradigm is difficult as the control and data planes have a tighter coupling.

VI. Conclusions and Future Work

This paper has shown how statistical classification can be deployed using SDN to detect DDoS attacks. Three classifiers were selected in an off-line environment to be integrated with *nmeta2*. These were then evaluated on a physical network testbed by replaying a DDoS attack scenario.

While statistical classification can be deployed using SDN to classify traffic, careful consideration must be made to pick classifiers that result in the smallest possible packet processing overhead. Although the classifiers did not demonstrate a high DR, results did suggest that particular statistical classification methods can classify network traffic under normal conditions using the *nmeta2* architecture. Under a DDoS attack scenario however, nothing is safe.

Future work will explore how the DPAE can be used as part of a network intrusion prevention system. Bakker *et al.* [22] presented a network-wide firewall using SDN/OpenFlow. Integrating their solution with a statistical classifier would facilitate the detection and filtering of malicious traffic in a network-wide manner.

Acknowledgment

Jarrod N. Bakker completed this work as part of the M.E. degree in Victoria University of Wellington (VUW) supported by InternetNZ Grant IR-201502. Bryan Ng and Winston K.G. Seah were supported in part by VUW's Huawei NZ Research Program, Software-Defined Green Internet of Things Project E2881.

References

[1] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, Apr. 2004.

[2] SDxCentral. (2016) IoT Botnet To Blame for Big DDoS Attack. https://www.sdxcentral.com/articles/news/iot-botnet-blame-big-ddos-attack/2016/10/ (Accessed on 02/11/2016).

[3] L. Urquhart and D. McAuley, "Avoiding the internet of insecure industrial things," *Computer Law and Security Review*, 17 Jan 2018.

[4] B. Ng, M. Hayes, and W. K. G. Seah, "Developing a Traffic Classification Platform for Enterprise Networks with SDN: Experiences & Lessons Learned." in *Proceedings of the IFIP Networking Conference*, Toulouse, France, May 2015, pp. 1–9.

[5] L. Qian, B. Wu, R. Zhang, W. Zhang, and M. Luo, "Characterization of 3G Data-Plane Traffic and Application towards Centralized Control and Management for Software Defined Networking," in *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, Santa Clara, CA, USA, Jun. 2013, pp. 278–285.

[6] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust Network Traffic Classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, Aug. 2015.

[7] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow," in *Proceedings of the IEEE 35th Conference on Local Computer Networks (LCN)*. Denver, CO, USA: IEEE, Oct. 2010, pp. 408–415.

[8] S. A. Mehdi, J. Khalid, and S. Khayam, "Revisiting Traffic Anomaly Detection Using Software Defined Networking," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, R. Sommer, D. Balzarotti, and G. Maier, Eds. Springer Berlin Heidelberg, 2011, vol. 6961, pp. 161–180.

[9] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments," *Computer Networks*, vol. 62, pp. 122–136, Apr. 2014.

[10] R. Wang, Z. Jia, and L. Ju, "An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking," in *Proceedings of IEEE Trustcom/BigDataSE/ISPA*, Aug. 2015, pp. 310–317.

[11] Y.-D. Lin, P.-C. Lin, C.-H. Yeh, Y.-C. Wang, and Y.-C. Lai, "An Extended SDN Architecture for Network Function Virtualization with a Case Study on Intrusion Prevention," *IEEE Network*, vol. 29, no. 3, pp. 48–53, May 2015.

[12] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, "Scalable architecture for sdn traffic classification," *IEEE Systems Journal*, pp. 1–12, 18 April 2017.

[13] J. N. Bakker, "Intelligent Traffic Classification for Detecting DDoS Attacks using SDN/OpenFlow," Master's thesis, Victoria University of Wellington, 2017. [Online]. Available: http://researcharchive.vuw.ac.nz/xmlui/handle/10063/6645

[14] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, pp. 357 – 374, 2012.

[15] D. Michie, D. Spiegelhalter, and C. Taylor, *Machine Learning, Neural and Statistical Classification*, D. Michie, D. Spiegelhalter, and C. Taylor, Eds. Ellis Horwood, 1994.

[16] A. J. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer, New York, 2008.

[17] M. Bramer, *Principles of Data Mining*, 2nd ed., ser. Undergraduate Topics in Computer Science. Springer London, 2013.

[18] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[19] S. Yadav and S. Shukla, "Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification," in *Proceedings of the IEEE 6th International Conference on Advanced Computing (IACC)*, Feb 2016, pp. 78–83.

[20] B. Kolo, *Binary and Multiclass Classification*, 1st ed. Weatherford Press, 2011.

[21] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of Improving K-Nearest-Neighbor for Classification," in *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Haikou, Hainan, China, Aug 2007, pp. 679–683.

[22] J. N. Bakker, I. Welch, and W. K. G. Seah, "Network-wide Virtual Firewall using SDN/OpenFlow," in *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, Palo Alto, CA, USA, 2016, pp. 62–68.