Enhancing Network Intrusion Detection Robustness via Dataset Augmentation: A CIC-IDS-2017 Case Study

Jozef Jankaj, Miel Verkerken, Laurens D'hooge, Tim Wauters, Bruno Volckaert
IDLab, Department of Information Technology at Ghent University - imec, Ghent, Belgium
{jozef.jankaj, miel.verkerken, laurens.dhooge, tim.wauters, bruno.volckaert}@UGent.be

Abstract—**As our world becomes increasingly interconnected, our networks face threats from more places than ever before. Existing protections have a hard time keeping up with the quickly evolving threats in the cyberworld, calling for quick improvements. One avenue for improvements is the area of machine learning based network intrusion detection systems (ML-NIDS), providing automated protection to the networks. However, the usefulness of these models quickly diminishes as new forms of attacks are discovered, and their robustness is questionable when facing variations of attacks. In this article, we explore a new way of increasing robustness of NIDS models through dataset augmentation using ConCap.**

I. Introduction

With the advent of the internet, physical threats to our security have spread to the virtual world. Our privacy is continuously threatened as we give more and more of it up for the convenience of the interconnected world. Plenty of adversaries are more than happy to attack, steal and destroy our systems. To address this Sword of Damocles, network security defenses have been a hot topic of research. While significant improvements in network security have been made thanks to the improvements in the message encryption, implementations of firewalls and general security practices, attacks against our networks are still prevalent. Timely detection and interruption of these attacks is of paramount importance for maintaining the confidentiality, integrity and availability of our networks. A Network Intrusion Detection System (NIDS) fulfills this purpose by monitoring the traffic and raising an alarm upon detection of malicious traffic. Existing systems [16, 24, 28] are implemented using preprogrammed rules or features of traffic.

However, the fast-paced nature of cybersecurity renders these systems quickly ineffective, as new attacks and variants of existing attacks are discovered and deployed. For this and other reasons, researchers turned their attention to Machine Learning based Network Intrusion Detection Systems (ML-NIDS). Both classical, shallow-learning methods [10, 17, 3] and deep learning methods [15, 23] have been previously explored in the literature. These methods rely heavily on high-quality data, and naturally, many datasets [19, 1, 13, 22] have been constructed and proposed as the benchmark for the ML-NIDS research.

Unfortunately, these datasets have been shown to contain mistakes [5, 6], reducing their usefulness. Issues such as poor documentation, misimplementations of attacks and faulty labeling, have been raised and as a result, the NIDS practitioners are reserved about implementing these models [2].

To address these and other issues, the ConCap framework [25] has been developed and utilized, leveraging the power of Kubernetes clusters to build out networks in-silico, conduct attacks and capture traffic for dataset creation. In this article, we look at using this technology for enhancing robustness of existing ML-NIDS datasets through dataset augmentation and take the CIC-IDS-2017 dataset [19] as a case study.

We do this as follows: In Section II, we describe our process of reconstructing the CIC-IDS-2017 dataset into ConCap scenarios and choices we make along the way. In Section III, we experiment with the dataset to show that our reconstruction is faithful and useful. In Section IV, we explore potential improvements to model robustness through adversarial training by augmenting the CIC-IDS-2017 dataset with ConCap traffic.

II. CIC-IDS-2017 reconstruction

The CIC-IDS-2017 dataset consists of traffic captures over the course of a workweek in July 2017, during which a test network is hit with numerous attack classes: Bruteforce, Denial-of-Service, Heartbleed, Web Attacks, Infiltration, Botnet, Distributed Denial-of-Service and Portscan.

We reconstruct the attacks in the dataset by collecting and writing Docker images for the various attackers and targets. By utilizing these images in ConCap scenarios, we can effectively reconstruct the dataset as PCAP files representing the attacks.

A. Monday

No attacks take place on Monday, only benign traffic, generated by the B-Profile [20]. This benign traffic is present in all days and serves as background activity in the network. We do not specifically reconstruct this traffic, but rather reuse the traffic from this day in our experiments.

B. Tuesday

FTP and SSH bruteforce attacks take place on Tuesday. These attacks are implemented using Patator [11]. It is unclear what wordlists the authors use to conduct these attacks, we therefore opt to use well-known dictionaries from the SecLists [12] repository.

C. Wednesday

Denial-of-Service and Heartbleed attacks take place on Wednesday. Authors perform attacks with the Slowhttptest and Slowloris [21], GoldenEye [18] and HULK [4] tools, each attacking a different part of the HTTP protocol. We reconstruct these attacks with the respective tools, reusing Slowhttptest to implement both Slowhttptest and SlowLoris attacks, as this tool is capable of both.

Heartbleed bug [8] is also exploited. We follow the authors in using the Heartleech [7] tool to conduct the attack against a vulnerable OpenSSH server version 1.0.1f.

D. Thursday

Web Attacks take place on Thursday, consisting of three subclasses: Bruteforce, SQL-Injection and Cross-Site Scripting. Each of these is conducted against a part of the Damn Vulnerable Web App [26] using Selenium framework. As we do not have access to the original code, we write our own Python scripts to conduct these attacks.

Infiltration attack also takes place on this day, by having the victim download and run a malicious executable, which creates a reverse Meterpreter shell to the attacker, through which a Portscan attack is executed. We opt not to reconstruct this attack for two reasons: First, a lack of support for multi-stage attack execution by ConCap limits our ability to reproduce this attack faithfully. Second, there is little interesting traffic happening on the network: it is the specific file that makes a file transfer malicious, not the act of transferring a file.

E. Friday

On Friday, the remaining attack classes are executed. A Botnet attack is launched using the Ares [27] botnet. We do not include this attack in our reconstruction due to technical limitations: Ares does not provide a way of controlling the botnet from Linux hosts, limiting our ability to conduct an attack from inside Docker containers.

Next, Distributed Denial-of-Service attack is performed using LOIC [14]. Though LOIC is primarily controlled through a GUI, the author does provide a way to control it remotely using an IRC server. We use the multi-target functionality of ConCap to launch both the target and an IRC server. A bot connects to this server and waits for LOIC to make the connection, before directing LOIC to perform the attack by changing the channel topic to the corresponding attack command.

Finally, a Portscan is executed. After traffic analysis, we crafted a container with open ports 21, 22, 80, 139 and 445, pointing to respectively an FTP server, SSH server, HTTP server and an SMB service. The portscan attack is executed against this container. Due to difficulties with setting up SMB, we have decided to conduct SMB portscans separately, merging the traffic with the other portscans during preprocessing.

III. Experimental verification

We first want to verify that the traffic generated by ConCap has similar features to the physical traffic in CIC-IDS-2017. To this end, we preprocess the generated PCAP files into NetFlows using the CICFlowmeter [9]. CICFlowmeter extracts more than eighty features out of the traffic capture, on which we can train ML models. We employ two-way methodology in our verification: Train a model on ConCap traffic and measure its performance on CIC-IDS-2017 traffic, and vice-versa.

To guarantee soundness of our experiments, we utilize the fixed version of CIC-IDS-2017 provided by Engelen et al. [5] as a stand-in for the actual dataset provided by Sharafaldin et al., where numerous labeling mistakes are fixed, chief among which incorrectly implemented attacks.

For model training, we use a Decision Tree with a single root node, utilizing a single feature to make the decision on, and Gini criterion. After preprocessing the datasets, we add equal amount of benign NetFlows to the attacks NetFlows to achieve class balance. We take these benign NetFlows from CIC-IDS-2017 Monday. Afterwards, we train the model on each feature and measure its accuracy, precision and recall, as well as ROC AUC score.

We are looking for a set of features that show high predictive power in both ways. While preferable, we do not expect the set of best performing features to be the same both ways, due to the underlying networks being fundamentally different: the ConCap network is a theoretically perfect network implemented fully in-silico. Though ConCap does provide a way to configure network artifacts (packet drops, corruption, delays, reordering...), we lack this information about the CIC-IDS-2017 network, making it impossible to reproduce.

These experiments show the presence of these common features with high predictive power as measured by the ROC AUC score. Naturally, different attacks require different features for detection. In Table I, we present the top three features for each attack class found according to their average ROC AUC Score for the two ways of experimenting.

Overall, we find plenty of features that perform well for both ways of training, regularly scoring above .9 on the ROC AUC score and none going below .8. It must be noted that due to low amount of samples of some classes in CIC-IDS-2017 (Heartbleed and Web attacks), models of these attack classes show signs of overfitting. Nevertheless, we have shown that the ConCap framework can be reliably used to replicate datasets and potentially extend them, as the models based on either the dataset or the ConCap traffic can reliably predict the other.

IV. Dataset Augmentation

Model robustness refers to a model's ability to maintain performance over unseen samples or over variations of previously seen samples. Variants of existing attacks can be easily generated through ConCap as additional scenarios. After verifying ConCap's usefulness as a stand-in for the CIC-IDS-2017 dataset in the previous section, we perform additional experiments to see the effect of ConCap on the robustness of datasets. More specifically, we are interested in the effect of dataset augmentation through adversarial training.

We augment the existing dataset with scenarios that try out different options of the existing tools. More specifically:

TABLE I
VERIFICATION RESULTS

Attack class	Feature	ROC-AUC Score
FTP Bruteforce	Bwd RST Flags	0.978288
	Packet Length Mean	0.976290
	Average Packet Size	0.976290
SSH Bruteforce	Fwd Seg Size Min	0.922024
	Fwd IAT Min	0.854907
	Bwd Segment Size Avg	0.817511
DoS Slowloris	Total TCP Flow Time	0.910773
	Bwd Packet Length Max	0.868125
	Total Length of Bwd Packet	0.868125
DoS Slowhttptest	Total TCP Flow Time	0.905697
	Fwd IAT Min	0.875804
	Fwd IAT Total	0.842901
DoS GoldenEye	Bwd Packet Length Std	0.939286
	Packet Length Variance	0.927954
	Packet Length Std	0.927954
DoS HULK	Bwd Packet Length Std	0.977942
	Fwd RST Flags	0.954837
	Subflow Bwd Bytes	0.951484
Heartbleed	Bwd Packet Length Std	1.000000
	Flow Bytes/s	0.931818
	Packet Length Std	0.905702
Web Attack Bruteforce	Fwd Seg Size Min	0.934932
	Fwd IAT Min	0.891781
	FIN Flag Count	0.828767
Web SQL Injection	Fwd Seg Size Min	0.884615
	FIN Flag Count	0.865385
	Bwd IAT Min	0.816719
Web XSS	Fwd Seg Size Min	0.930556
	Bwd Packet Length Std	0.869792
	Packet Length Max	0.845486
DDoS LOIC	Fwd Seg Size Min	0.934932
	Fwd IAT Min	0.860959
	FIN Flag Count	0.821918
Portscan	Fwd Packet Length Max	0.949676
	Total Length of Fwd Packet	0.948696
	Fwd Segment Size Avg	0.947596

- **FTP Bruteforce**: Turn persistence off, forcing one attempt per connection
- **DoS GoldenEye**: Attack with the POST HTTP verb
- **DDoS LOIC**: Attack over UDP

For each of these extensions, we generate the adversarial traffic and train a model using the CIC-IDS-2017 traffic on the best feature for the respective class as found above. Afterwards, we measure the model's performance on the generated adversarial traffic using ROC AUC score, forming our baseline. Next, we prepare a mix of CIC-IDS-2017 traffic and adversarial traffic, retrain the model and measure its performance. Just like in the experiments in previous section, we balance the training and testing sets with random samples of benign traffic from Monday traffic capture of CIC-IDS-2017.

In Table II, we report the average ROC AUC Scores for the different extensions across ten runs. All models perform poorly at first, having less than 50% probability of predicting a random malicious sample as malicious. This probability increases substantially after adversarial training. Our results show that ConCap can be effectively used for robustness enhancements of ML-NIDS models.

TABLE II
AVERAGE ROC AUC SCORES FOR DIFFERENT ATTACKS

Attack class	Baseline	Adversarial	Difference
FTP no persistence	0.4982	0.59236	0.09416
GoldenEye POST	0.48923	0.84631	0.35708
LOIC UDP	0.30098	0.78868	0.4877

TABLE III
DATASET AUGMENTATION: BASELINE + ADVERSARIAL ROC AUC SCORES ON CIC-IDS-2017

Attack class	Baseline	Adversarial	Difference
FTP No Persistence	0.99577	0.99615	0.00038
GoldenEye POST	0.98522	0.84716	-0.13807
LOIC UDP	0.78831	0.78879	0.00048

Lastly, we do a sanity check experiment for the model performance on original samples after adversarial training. The adversarial training would be of little value if the model performance on the original dataset degrades significantly. This is done by using the test split of CIC-IDS-2017 traffic and having the adversarial model predict it. Table III presents our findings.

We see no significant drop in performance for the FTP No Persistence and LOIC UDP models, but we do notice a degradation for the GoldenEye POST model. We posit that this is caused by this adversarial change being more significant than the other two classes. Nevertheless, the models maintains its predictive power with ROC AUC Score above 0.8, from which we conclude that the model remains usable.

V. CONCLUSION

This article presents a case study of a new method of robustness enhancement of ML-NIDS systems through dataset augmentation.

After an analysis of CIC-IDS-2017, we use ConCap to reconstruct the attacks in this dataset. Using Decision Trees, we have experimentally verified that the produced traffic can be used as a substitute for the dataset, with models achieving high levels of accuracy.

Finally, we have constructed three variations of existing attacks and shown that the models can be adversarially trained to recognize these new attacks.

VI. FUTURE WORK

Building on this article, further research can look into supporting more elaborate network architectures and scenarios, as ConCap is quite limited in this regard. Furthermore, ConCap can be utilized as a simulation environment, providing an excellent environment for blue team training and practice, as well as new attack modelling for red teams. Finally, real-time extensions to the ML-NIDS systems can prove useful to not only construct Intrusion Detection Systems, but also Intrusion *Prevention* Systems, stopping attacks as they are happening.

References

[1] *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018) - Registry of Open Data on AWS — registry.opendata.aws*. https://registry.opendata.aws/cse-cic-ids2018. [Accessed 03-05-2025].

[2] Giovanni Apruzzese, Pavel Laskov, and Johannes Schneider. "SoK: Pragmatic assessment of machine Learning for Network Intrusion Detection". In: (2023). eprint: 2305.00550 (cs.CR).

[3] Xiaohui Bao, Tianqi Xu, and Hui Hou. "Network intrusion detection based on support vector machine". In: *2009 International Conference on Management and Service Science*. Beijing, China: IEEE, Sept. 2009.

[4] Sumalya Chatterjee. *R3DHULK / HULK*. 2024. URL: https://github.com/R3DHULK/HULK.

[5] Gints Engelen, Vera Rimmer, and Wouter Joosen. "Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study". In: *2021 IEEE Security and Privacy Workshops (SPW)*. 2021, pp. 7–12. DOI: 10.1109/SPW53761.2021.00009.

[6] Robert Flood et al. "Bad Design Smells in Benchmark NIDS Datasets". In: *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*. 2024, pp. 658–675. DOI: 10.1109/EuroSP60621.2024.00042.

[7] Robert David Graham. *robertdavidgraham / heartleech*. 2014. URL: https://github.com/robertdavidgraham/heartleech.

[8] *Heartbleed*. URL: https://heartbleed.com/.

[9] Arash Habibi Lashkari. *CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection. https://github.com/ISCX/CICFlowMeter*. 2018.

[10] Yihua Liao and V.Rao Vemuri. "Use of K-Nearest Neighbor classifier for intrusion detection11An earlier version of this paper is to appear in the Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, August 2002". In: *Computers & Security* 21.5 (2002), pp. 439–448. ISSN: 0167-4048. DOI: https://doi.org/10.1016/S0167-4048(02)00514-X. URL: https://www.sciencedirect.com/science/article/pii/S016740480200514X.

[11] Sebastien Macke. https://github.com/lanjelot/patator.

[12] Daniel Miessler. *danielmiessler / SecLists*. 2025. URL: https://github.com/danielmiessler/SecLists.

[13] Nour Moustafa and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *2015 military communications and information systems conference (MilCIS)*. IEEE. 2015, pp. 1–6.

[14] Jorge Oliveira. *NewEraCracker / LOIC*. 2022. URL: https://github.com/NewEraCracker/LOIC.

[15] Duc Minh Pham et al. "Network Intrusion Detection with CNNs: A Comparative Study of Deep Learning and Machine Learning Models". In: *2024 2nd International Conference on Computer, Vision and Intelligent Technology (ICCVIT)*. 2024, pp. 1–6. DOI: 10.1109/ICCVIT63928.2024.10872423.

[16] Martin Roesch. "Snort - Lightweight Intrusion Detection for Networks". In: *Proceedings of the 13th USENIX Conference on System Administration*. LISA '99. Seattle, Washington: USENIX Association, 1999, 229–238.

[17] Shailendra Sahu and Babu M Mehtre. "Network intrusion detection system using J48 Decision Tree". In: *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2015, pp. 2023–2026.

[18] Jan Seidl. *jseidl / goldeneye*. 2020. URL: https://github.com/jseidl/GoldenEye.

[19] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*. INSTICC. SciTePress, 2018, pp. 108–116. ISBN: 978-989-758-282-0. DOI: 10.5220/0006639801080116.

[20] Iman Sharafaldin et al. "Towards a Reliable Intrusion Detection Benchmark Dataset". In: *Software Networking* 2017 (Jan. 2017), pp. 177–200. DOI: 10.13052/jsn2445-9739.2017.009.

[21] Sergey Shekyan. *shekyan / Slowhttptest*. 2024. URL: https://github.com/shekyan/slowhttptest.

[22] Ali Shiravi et al. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection". en. In: *Comput. Secur.* 31.3 (May 2012), pp. 357–374.

[23] Youngrok Song, Sangwon Hyun, and Yun-Gyung Cheong. "Analysis of Autoencoders for Network Intrusion Detection". In: *Sensors* 21.13 (2021). ISSN: 1424-8220. DOI: 10.3390/s21134294. URL: https://www.mdpi.com/1424-8220/21/13/4294.

[24] *Suricata*. 2025. URL: https://suricata.io/.

[25] Geert-Jan (ugent)01802387 Van Nieuwenhove. *Genereren van DDoS aanvallen voor het creëren van een dataset*. und. 2023. URL: http://lib.ugent.be/catalog/rug01:003150617.

[26] Robin Wood. *digininja / DVWA*. 2025. URL: https://github.com/digininja/DVWA.

[27] sweetsoftware. *sweetsoftware / Ares*. 2017. URL: https://github.com/sweetsoftware/Ares.

[28] zeek. *zeek / zeek*. 2025. URL: https://github.com/zeek/zeek.