Verbetering van de Robuustheid van Netwerkintrusiedetectie via Data-Augmentatie: een Onderzoek van CIC-IDS-2017

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*—Naarmate onze wereld steeds meer met elkaar verbonden raakt, worden onze netwerken bedreigd vanuit meer bronnen dan ooit tevoren. Bestaande beveiligingsmaatregelen hebben moeite om gelijke tred te houden met de snel evoluerende dreigingen in de cyberwereld, wat snelle verbeteringen noodzakelijk maakt. Een mogelijke weg naar verbetering is het gebruik van op machine learning gebaseerde netwerk-inbraakdetectiesystemen (ML-NIDS), die geautomatiseerde bescherming aan netwerken bieden. De bruikbaarheid van deze modellen neemt echter snel af naarmate nieuwe vormen van aanvallen worden ontdekt, en hun robuustheid is twijfelachtig bij het omgaan met variaties van aanvallen. In dit artikel verkennen we een nieuwe methode om de robuustheid van NIDS-modellen te vergroten door middel van datasetverrijking met behulp van ConCap.**

I. Introductie

Met de opkomst van het internet zijn fysieke bedreigingen voor onze veiligheid overgeslagen naar de virtuele wereld. Onze privacy wordt voortdurend bedreigd naarmate we er steeds meer van opgeven ten behoeve van het gemak dat de onderling verbonden wereld biedt. Tal van kwaadwillenden zijn maar al te bereid om aan te vallen, te stelen en onze systemen te vernietigen. Om deze dreiging, als een zwaard van Damocles, het hoofd te bieden, is netwerkbeveiliging een belangrijk onderwerp van onderzoek geworden. Hoewel er aanzienlijke vooruitgang is geboekt in netwerkbeveiliging dankzij verbeteringen in berichtversleuteling, de implementatie van firewalls en algemene beveiligingspraktijken, blijven aanvallen op onze netwerken frequent voorkomen. Tijdige detectie en onderbreking van deze aanvallen is van cruciaal belang om de vertrouwelijkheid, integriteit en beschikbaarheid van onze netwerken te waarborgen.

Een Netwerk Intrusie Detectie Systeem (NIDS) dient dit doel door het netwerkverkeer te monitoren en alarm te slaan bij de detectie van kwaadaardig verkeer. Bestaande systemen [16, 24, 28] worden geïmplementeerd op basis van voorgeprogrammeerde regels en kenmerken van het verkeer. De snelle evolutie van cybersecurity maakt deze systemen echter al snel ineffectief, omdat voortdurend nieuwe aanvallen en varianten van bestaande aanvallen worden ontdekt en uitgevoerd. Om deze en andere redenen hebben onderzoekers hun aandacht gericht op op Machine Learning gebaseerde NIDS systemen (ML-NIDS). Zowel klassieke, shallow-learning methoden [10, 17, 3] als deep learning-methoden [15, 23] zijn eerder in de literatuur onderzocht. Deze methoden zijn sterk afhankelijk van de kwaliteit van de data waarop deze modellen getrained worden, en daarom zijn er diverse datasets [19, 1, 13, 22] ontwikkeld en voorgesteld als benchmark voor ML-NIDS-onderzoek.

Helaas is gebleken dat deze datasets fouten bevatten [5, 6], wat hun bruikbaarheid vermindert. Problemen zoals gebrekkige documentatie, foutieve implementaties van aanvallen en onjuiste labeling zijn aan het licht gekomen, met als resultaat enige terughoudendheid van netwerkbeheerders in het implementeren van deze modellen [2].

Om deze en andere knelpunten aan te pakken, is het ConCap-framework [25] ontwikkeld en ingezet. Dit maakt gebruik van de kracht van Kubernetes-clusters om netwerken in-silico op te bouwen, aanvallen erop uit te voeren en verkeer vast te leggen ten behoeve van datasetcreatie. In dit artikel onderzoeken wij het gebruik van deze technologie om de robuustheid van bestaande ML-NIDS-datasets te verbeteren via datasetverrijking, waarbij de CIC-IDS-2017-dataset [19] als casestudy wordt gebruikt.

We doen dit als volgt: in Sectie II beschrijven we ons proces van het reconstrueren van de CIC-IDS-2017-dataset in ConCap-scenario's en de keuzes die we daarbij maken. In Sectie III voeren we experimenten uit met de dataset om aan te tonen dat onze reconstructie betrouwbaar en bruikbaar is. In Sectie IV verkennen we mogelijke verbeteringen in modelrobuustheid via adversarial training door de CIC-IDS-2017-dataset te verrijken met ConCap-verkeer.

II. CIC-IDS-2017 reconstructie

De CIC-IDS-2017-dataset bestaat uit verkeersopnames gedurende een werkweek in juli 2017, waarin een testnetwerk wordt getroffen door verschillende soorten aanvallen: Brute-force, Denial-of-Service, Heartbleed, Webaanvallen, Infiltratie, Botnet, Distributed Denial-of-Service en Portscan.

Wij reconstrueren de aanvallen in de dataset door Docker-images te verzamelen en te schrijven voor de verschillende aanvallers en doelwitten. Door deze images te gebruiken in ConCap-scenario's kunnen we de dataset effectief reconstrueren als PCAP-bestanden die de aanvallen representeren.

A. Maandag

Op maandag vinden er geen aanvallen plaats; er is uitsluitend onschadelijke verkeer aanwezig dat is gegenereerd door

het B-Profile [20]. Dit onschadelijke verkeer komt op alle dagen voor en fungeert als achtergrondactiviteit binnen het netwerk. Wij reconstrueren dit verkeer niet afzonderlijk, maar hergebruiken het verkeer van deze dag in onze experimenten.

B. Dinsdag

FTP- en SSH-bruteforceaanvallen vinden plaats op dinsdag. Deze aanvallen worden geïmplementeerd met behulp van Patator [11]. Het is onduidelijk welke woordenlijsten de auteurs gebruiken om deze aanvallen uit te voeren; daarom kiezen wij ervoor om bekende woordenlijsten uit de SecLists-repository [12] te gebruiken.

C. Woensdag

Op woensdag vinden Denial-of-Service- en Heartbleed-aanvallen plaats. De auteurs voeren deze aanvallen uit met de tools Slowhttptest en Slowloris [21], GoldenEye [18] en HULK [4], waarbij elk een ander deel van het HTTP-protocol aanvalt. Wij reconstrueren deze aanvallen met de respectieve tools, waarbij we Slowhttptest hergebruiken om zowel de Slowhttptest- als de Slowloris-aanvallen te implementeren, aangezien deze tool beide ondersteunt.

Daarnaast wordt ook misbruik gemaakt van de Heartbleed-bug [8]. Wij volgen de auteurs in het gebruik van de Heartleech-tool [7] om de aanval uit te voeren op een kwetsbare versie 1.0.1f van de OpenSSH-server.

D. Donderdag

Webaanvallen vinden plaats op donderdag en bestaan uit drie subklassen: Bruteforce, SQL-Injection en Cross-Site Scripting. Elk van deze aanvallen wordt uitgevoerd op een onderdeel van de Damn Vulnerable Web App [26], met behulp van het Selenium-framework. Aangezien we geen toegang hebben tot de oorspronkelijke code, schrijven we onze eigen Python-scripts om deze aanvallen uit te voeren.

Op deze dag vindt ook een infiltratieaanval plaats, waarbij het slachtoffer een kwaadaardig programma downloadt en uitvoert. Dit bestand creëert een omgekeerde Meterpreter-shell naar de aanvaller, via welke een Portscan-aanval wordt uitgevoerd. Wij kiezen ervoor om deze aanval niet te reconstrueren, om twee redenen: Ten eerste beperkt het gebrek aan ondersteuning voor meerfasige aanvalsexecutie binnen ConCap onze mogelijkheid om deze aanval getrouw te reproduceren. Ten tweede vindt er weinig interessant netwerkverkeer plaats: het is het specifieke bestand dat een bestandsoverdracht kwaadaardig maakt, niet de handeling van de overdracht zelf.

E. Vrijdag

Op vrijdag worden de resterende aanvalsklassen uitgevoerd. Een Botnet-aanval wordt uitgevoerd met behulp van het Ares-botnet [27]. Wij nemen deze aanval niet op in onze reconstructie vanwege technische beperkingen: Ares biedt geen mogelijkheid om het botnet aan te sturen vanaf Linux-hosts, wat onze mogelijkheden om de aanval vanuit Docker-containers uit te voeren beperkt.

Vervolgens wordt een Distributed Denial-of-Service-aanval uitgevoerd met LOIC [14]. Hoewel LOIC primair via een grafische interface wordt aangestuurd, biedt de auteur een mogelijkheid om het op afstand te aansturen via een IRC-server. Wij gebruiken de multi-target-functionaliteit van ConCap om zowel het doelwit als een IRC-server te starten. Een bot maakt verbinding met deze server en wacht tot LOIC verbindt, waarna LOIC wordt aangestuurd om de aanval uit te voeren door het kanaalonderwerp aan te passen naar het bijbehorende aanvalcommando.

Tot slot wordt een Portscan uitgevoerd. Na analyse van het netwerkverkeer hebben wij een container samengesteld met open poorten 21, 22, 80, 139 en 445, respectievelijk wijzend naar een FTP-server, SSH-server, HTTP-server en een SMB-dienst. De portscan-aanval wordt op deze container uitgevoerd. Vanwege moeilijkheden bij het opzetten van SMB hebben wij ervoor gekozen de SMB-portscans afzonderlijk uit te voeren en het verkeer tijdens de preprocessingfase samen te voegen met de overige portscans.

III. Experimentele verificatie

We willen eerst verifiëren of het verkeer dat door ConCap wordt gegenereerd vergelijkbare kenmerken vertoont met het fysieke verkeer in CIC-IDS-2017. Daartoe preprocessen we de gegenereerde PCAP-bestanden naar NetFlows met behulp van CICFlowmeter [9]. CICFlowmeter extraheert meer dan tachtig kenmerken uit de verkeersopname, waarop we ML-modellen kunnen trainen. We hanteren een tweezijdige methodologie voor onze verificatie: we trainen een model op ConCap-verkeer en meten de prestaties op CIC-IDS-2017-verkeer, en omgekeerd.

Om de betrouwbaarheid van onze experimenten te waarborgen, maken wij gebruik van de gecorrigeerde versie van CIC-IDS-2017, aangeboden door Engelen et al. [5], als vervanging voor de oorspronkelijke dataset van Sharafaldin et al., waarin diverse labelingsfouten zijn hersteld, met name verkeerd geïmplementeerde aanvallen.

Voor het trainen van het model gebruiken we een beslissingsboom met een enkele root, die een enkele feature gebruikt om de beslissing op te baseren, met het Gini-criterium. Na het preprocessen van de datasets voegen we een gelijke hoeveelheid goedaardige NetFlows toe aan de aanval-NetFlows om klassebalans te bereiken. Deze goedaardige NetFlows nemen we van het verkeer van maandag in de CIC-IDS-2017, zoals hierboven al aangehaald. Vervolgens trainen we het model op elke feature afzonderlijk en meten we de nauwkeurigheid, precisie en recall, evenals de ROC AUC-score.

We zijn op zoek naar een set features met een hoge voorspellende waarde in beide richtingen. Hoewel het wenselijk is, verwachten we niet dat de best presterende features aan beide zijden identiek zijn, omdat de onderliggende netwerken fundamenteel verschillend zijn: het ConCap-netwerk is een theoretisch perfect netwerk dat volledig in-silico is geïmplementeerd. Hoewel ConCap de mogelijkheid biedt om netwerkartefacten te configureren (pakketverlies, corruptie, vertragingen, herschikkingen...), ontbreekt deze informatie voor het CIC-IDS-2017-netwerk, waardoor reproductie onmogelijk is.

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

Deze experimenten tonen de aanwezigheid van deze gemeenschappelijke kenmerken met een hoge voorspellende waarde, gemeten aan de hand van de ROC AUC-score. Uiteraard vereisen verschillende aanvallen verschillende kenmerken voor detectie. In Tabel I presenteren we de drie belangrijkste kenmerken voor elke aanvalsklasse, bepaald op basis van hun gemiddelde ROC AUC-score voor de twee experimenten.

Over het algemeen vinden we veel kenmerken die goed presteren bij beide trainingsmethoden, met regelmatig scores boven de 0,9 op de ROC AUC-score en geen enkele onder de 0,8. Het moet worden opgemerkt dat, door het geringe aantal voorbeelden van sommige klassen in CIC-IDS-2017 (Heartbleed- en Web-aanvallen), modellen van deze aanvalsklassen tekenen van overfitting vertonen. Niettemin hebben we aangetoond dat het ConCap-framework betrouwbaar kan worden gebruikt om datasets te repliceren en mogelijk uit te breiden, aangezien de modellen die zijn gebaseerd op ofwel de originele dataset ofwel het ConCap-verkeer, de andere betrouwbaar kunnen voorspellen.

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

IV. DATASETAUGMENTATIE

Modelrobustheid verwijst naar het vermogen van een model om prestaties te behouden bij ongeziene voorbeelden of variaties van de eerder geziene voorbeelden. ConCap kan eenvoudig worden ingezet om de modelrobustheid te vergroten door verkeer te genereren van variaties op aanvallen.

Na het verifiëren van het nut van ConCap als vervanger voor de CIC-IDS-2017 dataset in de vorige sectie, voeren we aanvullende experimenten uit om het effect van ConCap op de robuustheid van datasets te onderzoeken. We zijn met name geïnteresseerd in het effect van datasetaugmentatie door middel van adversariële training.

We breiden de bestaande dataset uit met scenario's die verschillende opties van de bestaande tools testen. Meer specifiek:

- **FTP Bruteforce**: Persistence uitzetten, waardoor er per verbinding slechts één poging wordt gedaan
- **DoS GoldenEye**: Aanval met de POST HTTP-verb
- **DDoS LOIC**: Aanval via UDP

Voor elk van deze uitbreidingen genereren we het adversariële verkeer en trainen we een model op het beste kenmerk voor de betreffende klasse, zoals hierboven vastgesteld, gebruikmakend van het CIC-IDS-2017-verkeer. Vervolgens meten we de prestatie van het model op het gegenereerde adversariële verkeer met behulp van de ROC AUC-score, wat onze basislijn vormt. Daarna stellen we een combinatie samen van CIC-IDS-2017-verkeer en adversariële verkeer, hertrainen we het model en meten we opnieuw de prestaties. Net als in de experimenten in de vorige sectie, balanceren we de trainings- en testsets met willekeurige samples van goedaardig verkeer uit de maandagse verkeerstrace van CIC-IDS-2017.

In Tabel II rapporteren we de gemiddelde ROC AUC-scores voor de verschillende extensies over tien runs. Alle modellen presteren aanvankelijk slecht, met minder dan 50% kans om een willekeurig kwaadaardig sample correct als kwaadaardig te classificeren. Deze kans neemt aanzienlijk toe na adversariële training. Onze resultaten tonen aan dat ConCap effectief kan worden ingezet voor het vergroten van de robuustheid van ML-NIDS-modellen.

Tot slot voeren we een sanity check-experiment uit om de modelprestaties op originele samples na adversariële training te evalueren. De adversariële training zou van weinig meerwaarde zijn als de modelprestaties op de originele dataset erdoor significant verslechteren. Dit wordt getest door gebruik

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

te maken van de test-split van het CIC-IDS-2017-verkeer, waarbij het adversariële model deze samples voorspelt. Tabel III toont onze bevindingen.

We zien geen significante prestatievermindering bij de modellen voor FTP No Persistence en LOIC UDP, maar we merken wel een degradatie op bij het GoldenEye POST-model. Wij vermoeden dat dit komt doordat de adversariële wijziging bij deze klasse ingrijpender is dan bij de andere twee. Desondanks behoudt het model zijn voorspellende kracht met een ROC AUC-score boven de 0,8, waaruit we concluderen dat het model bruikbaar blijft.

V. CONCLUSIE

Dit artikel presenteert een casestudy van een nieuwe methode voor het verbeteren van de robuustheid van ML-NIDS-systemen door middel van datasetaugmentatie.

Na een analyse van CIC-IDS-2017 hebben we ConCap gebruikt om de aanvallen in deze dataset te reconstrueren. Met behulp van beslissingsbomen hebben we experimenteel aangetoond dat het gegenereerde verkeer als vervanging van de dataset kan dienen, waarbij modellen hoge nauwkeurigheid bereiken.

Tot slot hebben we drie variaties van bestaande aanvallen geconstrueerd en aangetoond dat de modellen adversariëel getraind kunnen worden om deze nieuwe aanvallen te herkennen.

VI. TOEKOMSTIG ONDERZOEK

Voortbouwend op dit artikel kan vervolgonderzoek zich richten op het ondersteunen van uitgebreidere netwerkarchitecturen en scenario's, aangezien ConCap op dit vlak nog vrij beperkt is. Daarnaast kan ConCap worden ingezet als simulatieomgeving, wat een uitstekende basis biedt voor training en oefening van blue teams, evenals voor het modelleren van nieuwe aanvallen door red teams. Tot slot kunnen real-time uitbreidingen van ML-NIDS-systemen van waarde zijn, niet alleen voor het opbouwen van Intrusie Detectie Systemen, maar ook van Intrusie *Preventie* Systemen, die aanvallen in real-time kunnen stoppen.

REFERENTIES

[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 Biflow 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.