NeDaGen: Towards a Realistic Network-based Intrusion Detection Dataset Generator

Dennis van Wijk
University of Amsterdam
dennis.van.wijk@student.uva.nl

Jeroen van Saane
University of Amsterdam
jeroen.van.saane@student.uva.nl

***Abstract*—The damage an attacker can do to a network is well understood, the prevention less so. The heterogeneity of data and the variety of protocols and services make intrusion detection complex and challenging. Currently, most network intrusion detection systems incorporate artificial intelligence, specifically machine learning and deep learning. Training such systems requires representative and standardized network-based datasets. These datasets are rare, frequently lacking the required volume of traffic combined with an insufficient attack diversity. This study presents NeDaGen (Network traffic Dataset Generator), a tool for fully automated building of an autonomously operating network and producing high-quality network-based intrusion detection system (NIDS) datasets. This study also discusses some of the components encompassing high-quality network-based datasets and exemplifies how it overcomes shortcomings in existing datasets. Finally, NeDaGen is evaluated based upon resource utilization performance, the quality of the resulting dataset and its use cases — the tool as well as the source code is available for public use and further research.**

I. Introduction

Intrusion detection systems (IDS) are intended to protect computer networks from threats by analysing network traffic and detecting attacks and anomalies [1]. Training such systems requires representative and commonly accepted network-based datasets. These datasets are rare, frequently lacking the required volume of traffic combined with an inadequate attack diversity [2]. A fundamental cause for the scarcity of reliable datasets is that they can often not be published due to privacy issues [3]. Therefore, publicly available datasets do not reflect current attack diversity trends or are subject to data anonymization, resulting in missing or incorrect metadata [2]. For that reason, many such systems are evaluated against outdated datasets containing various inherent problems [4].

Since recent research into the taxonomy of IDS datasets concluded that publicly available datasets result in low effectiveness due to the lack of current malware trends [3], this work proposes NeDaGen, a highly flexible and expandable tool for generating labelled datasets intended for the evaluation of network-based intrusion detection systems (NIDS). NeDaGen expands on work of preliminary studies that rely on packet captures as input [5] with a tool capable of building user-defined networks with higher flexibility due to customizable features. User-defined characteristics in the form of protocols and services can be included to simulate different scenarios. Furthermore, this tool can also account for temporal aspects and simulate the behaviour of a host in a network that evolves over time, a frequently neglected aspect yet vital for host profiling research [6]. The used tool is published to contribute towards the open source community [7].

A. Contribution

The contributions of this study were three-fold:

1. provided an overview of the state-of-the-art network-based traffic dataset generators based upon existing literature;
2. released a tool for creating customized and scalable datasets intended for NIDS that is available for public use and further research [7];
3. practically exemplified the tool's use cases through automated adversary simulations.

Moreover, this study also demonstrates how the tool can be utilized to represent a realistic threat model with (automated) operating threat agent(s) inside network traffic.

The remainder of the paper is structured as follows. Related work into the construction and evaluation of NIDS datasets is covered in Section II.. Section III. first discusses various prerequisites for creat-

ing high-quality datasets and the conditions and characteristics thereof. Secondly, it provides a detailed explanation of the proposed tool and exemplifies the architecture and its use cases. Section IV. illustrates the proposed tool's performance as well as its capacity to generate tailored yet high-quality datasets. Subsequently, in Section V., the study's limitations in terms of characteristics are explored, and in Section VI., the study is concluded by providing insights into intended future work.

II. Related Work

Various research into the generation of network-based datasets specific to NIDS has been published over the years. Vasilomanolakis et al. for instance, presented the Intrusion Detection Dataset Toolkit (ID2T) for creating synthetic labelled datasets that contain user-defined attacks [5], an extension of preliminary research conducted by Cordero et al. [4]. They discussed important objectives for generating high-quality datasets, and the problems intrusion detection datasets have with the injection of inadvertent anomalies, such as, but not limited to, the TTL value distribution and the IP Address distribution. This study intended to overcome the TTL value distribution issues by modifying the default TTL values for a limited number of nodes inside the network (to still represent a majority default TTL). The base of the proposed network accounts for IP Address distribution by having user-defined subnet ranges for each of the individual subnetworks. A substantial difference with the proposed toolkit ID2T is that it requires network traffic datasets as input, which it extends by injecting or replicating packets/communication flows. As a result, the injected traffic may be unrealistic, especially in terms of malicious traffic, as it cannot take unexpected reactions into account, for example, a system crash. NeDaGen addresses this issue with a realistic network to account for potential failures. Finally, the toolkit from Vasilomanolakis cannot represent temporal aspects and simulate host behaviour in a network that evolves over time.

In other research by Sharafaldin et al., it was concluded that many datasets are out of date and unreliable to use [2]. For that reason, Sharafaldin et al. developed a new dataset that contains benign and seven common families of attacks. However, they selected a subset of attacks not corresponding to a curated knowledge base/model, such as the MITRE ATT&CK framework. Instead, they relied on the most common attacks based on the 2016 McAfee report [8]. Even though this resembled the current attack trends at the time (now outdated), it did not include full attacker stories but rather independent attacks such as a Denial of Service. It also did not take some of the addressed issues from Vasilomanolakis into account, for example, the TTL value distribution. In addition, the resulting dataset from 2018 is published, but the model to generate the dataset is not. For that reason, it neglects a vital property: reproducibility.

Ring et al. conducted a literature survey of network-based intrusion detection datasets. The paper provides a comprehensive overview of existing datasets and suggested "providing network-based datasets in standard packet-based or flow-based formats as they are captured in real network environments" to overcome issues with requiring standard input data formats [9].

III. NeDaGen

The first part of this section discusses various requirements for creating high-quality datasets and the conditions and characteristics thereof. Based upon these requirements, the remainder of this section describes the architectural design of the proposed tool and exemplifies its use cases through automated adversary simulations.

A. Requirements

High-quality datasets are critical for validating any IDS approach, since these measure the success of the proposed model in terms of detecting anomalies and intrusive behaviour. There are various required conditions for generating realistic network datasets in order to be relevant for NIDS evaluation. This subsection discusses some of those criteria, and also discusses some of the desired conditions. It is worth mentioning that the tool and corresponding dataset become more generic as more requirements are met. For that reason, an important property in the realm of automated generated datasets is to discuss the required characteristics. Research indicates many inherent problems in existing datasets. For instance, the DARPA / KDD Cup99 datasets are out-of-date and do not reflect recent malware attacks. The dataset CAIDA is limited to network traffic traces from Distributed Denial-of-Service attacks and neglects attack diversity as well as missing important features of regular network traffic, making it unsuitable for anomaly detection evaluation [3].

To improve upon the existing body of work, the network architecture must adhere to various characteristics. Vasilomanolakis et al., discuss four important 'non-functional' requirements that influence the included properties and resulting overall quality of the dataset: availability, reproducibility, interoperability and flexibility [5]. In summary, the tool must be publicly available to support reproducibility. Generating the datasets should support scalability in terms of arbitrary network sizes and output packet captures. The tool must also provide a means for the user to ex-

pand included properties or alter current ones, such as templates or an API.

Ring et al. distinguish datasets based upon packet-based network traffic containing the payloads, and flow-based network traffic that only contain meta-information about network connections [9]. Vasilomanolakis et al., emphasized the importance of payload availability, necessary by intrusion detection algorithms that wish to detect intrusions that are only present in the payload, for example, but not limited to, exploits, SQL Injections and phishing attacks [5]. For that reason, another requirement for a high-quality dataset is to support packet-based network traffic. Furthermore, Ring et al. defined detailed dataset properties for a more qualitative approach to compare network-based IDS datasets inherently. These properties are described below.

1) General Information: the year of creation describes the year in which the underlying network traffic was captured. Sharafaldin et al. concluded that many datasets are out of date. Therefore, the year of creation is an important property to validate the relevancy of the dataset. Subsequently, the dataset should be publicly available so that its quality can be further evaluated by third parties. Another component to measure the quality of is whether the dataset encompasses regular traffic as well as malicious traffic. Ring et al. stated that "the presence of normal user behaviour is indispensable for evaluating an IDS". It is worth mentioning that a lack of regular traffic does not render a data collection useless; rather, it signals that it must be combined with other datasets.

2) Nature of the data: Successfully interpreting communication flows inside the dataset is problematic for third parties. For that reason, Ring et al. emphasized the importance of including metadata to provide further information about the network architecture and attack scenarios. They also distinguish datasets upon two main formats: packet-based network traffic (pcap)—containing the network traffic as well as the payload and flow-based network traffic - only containing meta-information about the communication flows. Data anonymization is also part of this category. Research by Khraisat et al. concluded that privacy issues are a fundamental cause for the scarcity of datasets [3]. Ring et al. further discuss that this affects qualitative attributes such as metadata anonymization (IP addresses) and payload anonymization in the case of packet-based network traffic.

3) Data Volume: To address limitations in existing work with regard to the count and traffic distribution, several elements were implemented. First, a script was created that captures the network traffic on all networks in the background of bridge and router interfaces, which can be configured with a dynamically set recording time (specified inside the main configuration of the project)—this enables a user-defined count or duration. Secondly, to adhere to the traffic distribution as emphasized by Maciá-Fernández et al. [10], task scheduling activities were configured inside each of the created containers. These tasks can be dynamically set. For example, in the default architecture, traffic generation scripts on the clients only run on workdays during specific times. The times are based upon random numbers to simulate a varying start-time and end-time (as well as any potential breaks) for each of the clients.

4) Recording Environment: Ring et al. categorize the origins of traffic into three types: real, emulated, and synthetic. Where real indicates network traffic being captured within a 'production' network environment (albeit the definition of productive being absent). Emulated entails the traffic being captured inside an emulated network environment. Synthetic includes network traffic that was created synthetically; not captured by a real (or virtual) network device. Furthermore, Ring et al. discussed a clear distinction between variant types of networks. For example, due to the environments being fundamentally different for small vs medium-sized companies. In addition to the type of network, Sharafaldin et al. further defined the completeness of the network by distinguishing datasets with a single host for the generation of traffic (honeypot-based approaches) and those containing several hosts and routers, this property was adopted by Ring et al.

5) Evaluation: Ring et al. define that the quality of various IDS is hard to compare if the same subsets are used for both, the training and evaluation. Finally, they evaluate datasets based upon the balance with respect to their class labels, an important property for anomaly detection and the further processing of the dataset.

Section IV. further discusses the relation between the requirements, the properties as defined by Vasilomanolakis et al. and Ring et al., and the proposed tool.

B. Architecture

The network of NeDaGen was built upon the open source Containerlab project [11] and was designed with extensibility and scalability in mind. This was realized by applying Infrastructure as Code (IaC) principles and relying on service orchestration and configuration management solutions that support

machine-readable definition files. This architecture enabled a scalable network since it can be expanded based upon user-defined variables. For instance, the total amount of Local Area Network (LAN) and Wide Area Network (WAN)-simulating clients, the number of departments such as administration, operations and developers, and the percentage of clients within those departments. Furthermore, the tool supports probability traffic distribution for each of the departments.

1) Network Architecture: A heterogeneous network is created by default through different devices, protocols and services, resulting in a set of interconnected nodes and links (Figure 1). External communication is achieved by using both internal, and external nameservers. Characteristics of realistic (enterprise) networks were simulated by creating three individual, but interconnected, heterogeneous subsets. These are hereafter considered the internal, external and demilitarized networks. Each of these subsets operates inside one or multiple unique subnets. As shown in Figure 1 (upper left), the LAN consists of various departments such as administration with IP address range 192.168.10.0/24, operations 192.168.20.0/24 and development 192.168.30.0/24, which are all subnets from the 192.168.0.0/16 private IPv4 address range. Besides the local network, the network architecture also includes a demilitarized zone (DMZ) (upper right). The DMZ is a logical subnetwork that contains the external-facing services to an untrusted, usually larger, network such as the Internet [12]. The reason for implementing a DMZ is that external-facing services generally expose a greater threat to the local network. Isolating these services enables for better detection of security breaches before they reach the internal network. Furthermore, the network architecture is designed to be able to represent a multitude of devices and services. Including, but not limited to, routers, firewalls, web servers and routing protocols. The preconfigured network topology in Figure 1 shows, for example, the use of Nginx web servers and FRRouting with Open Shortest Path First and Border Gateway Protocol to redistribute their routes. This is not only done to acquire heterogeneity, but also to support the additional vulnerabilities that can be represented. Finally, the flexibility of containers provides support for further addition of devices and/or services.

2) Benign Traffic Generators: To have realistic benign network traffic in the resulting dataset, various commonly used protocols were implemented inside the network architecture. Each of the LAN and WAN clients contains user-specified scripts for the generation of the corresponding network traffic. There are currently eight protocols for which the clients can issue autonomous network traffic with a set time interval: HTTP/HTTPS, SMB, FTP, ICMP, SSH, SMTP and IMAP. To include HTTP(S) and DNS packet flows, LAN clients query the locally hosted domains and randomly picked domains of the million most popular domains as of January 31, 2022, retrieved from Tranco, a research-oriented top sites ranking hardened against manipulation [13]. WAN clients are limited to issuing HTTP(S) requests

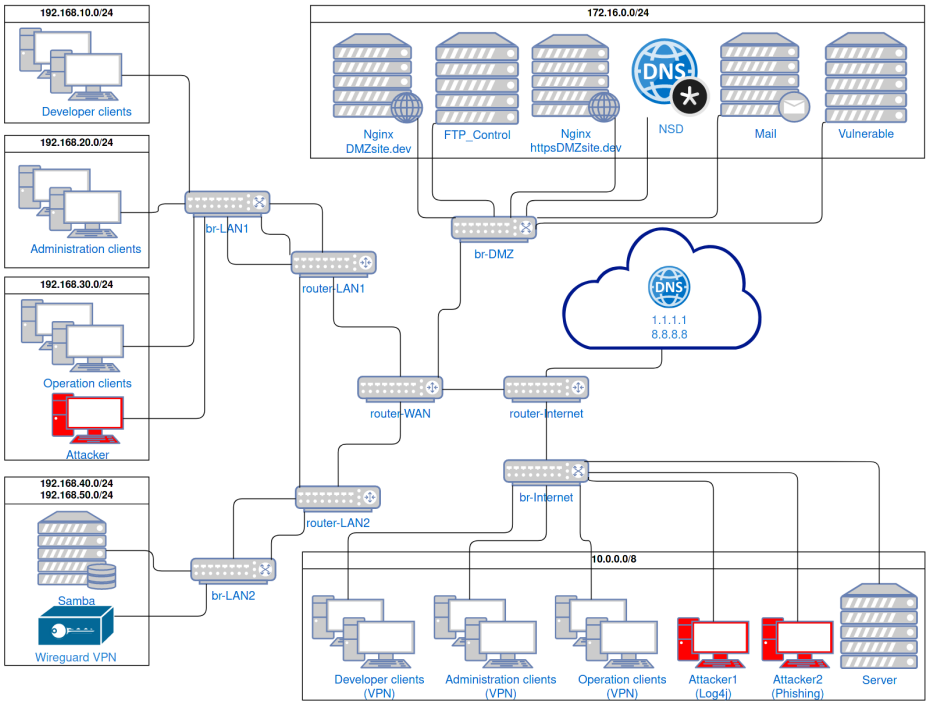

Figure 1: Architecture of the simulated network

towards locally hosted domains, since only this traffic is included inside the network traffic capture, as it would not be realistic to capture egress traffic for these clients. To support storage-related network traffic, SMB and FTP servers are implemented. Both LAN, as well as WAN clients, read and write files to both servers with a user-defined time interval. WAN clients, however, will perform the same operations to the SMB server through the Wireguard VPN (Figure 1). Another traffic generator running on the FTP_Control server is a script that utilizes the Ping utility to issue ICMP requests to validate the health status for each of the servers. Furthermore, LAN clients issue SSH connections to this server, browse through the file system, and write a set of random data which is deleted shortly after. This simulates and represents benign SSH and ICMP network traffic inside the dataset. In addition, a mail server was implemented inside the proposed network architecture. As a result, several protocols were supported such as Simple Mail Transfer Protocol (SMTP) and Internet Message Access Protocol (IMAP). The generation of traffic for these protocols was based upon WAN clients sending mails to a shared mailbox on the mail server, based upon a user-defined time interval, and LAN clients collecting these mails.

3) Traffic Distribution: Each of the traffic generator scripts supports traffic profiles. Through these profiles, the traffic distribution can be specified, but it can also be specified at what time these scripts are starting through task scheduling functionality. For instance, each of the client containers starts generating traffic at a random time between 08:45 and 09:15 and stop generating traffic at a random time between 16:45 and 17:15 to further simulate realistic benign traffic. Through these profiles, the individual containers were grouped as part of departments (user-defined percentile relation with total LAN clients), such as developers, administration, and operations. Naturally, these departments are different in nature and the corresponding network traffic should represent that. For this reason, the traffic generation scripts that are by default configured for each of the departments correspond to the activities for that department and their corresponding traffic distribution. More specifically, Administration is more likely to primarily use HTTP/HTTP(S) and Mail-related protocols, whilst developers are more likely to also use SMB and SSH. Furthermore, through these profiles, the distribution of traffic unique to those containers inside each of the departments can be configured. For example, by assigning weights to each of the protocols, the user defines with what intensity the traffic is generated. These weights are unique to the departments, and the start/end for this traffic generation is unique to each of the containers.

4) Virtualization: OS-virtualization was utilized in the form of containers, in contrast to a hypervisor-based network virtualization solution such as VMware ESXi. The use of containers makes the network architecture flexible and uniform since the required storage, CPU and memory that would have been required for full virtualization is significantly reduced. This improves portability since the entire network architecture can be moved, with the added benefit of being idempotent. The proposed tool utilizes the dockerd runtime environment, but the flexible architecture also enables support for podman (beta support), containerd (experimental support) and ignite [14]. Docker was selected due to its widespread use and currently being fully supported by Containerlab. This rise in popularity of containerization indicates increased support for the development of images that can be used to test malicious activities with and against, or to add further network services inside the topology. In addition, containers are highly scalable because they can be created on-demand, more so than virtual machines which require longer boot times and more extensive support in terms of storage solutions and hardware performance.

5) Extensibility: One of the requirements for this project was for the model to be extensible. For this reason, the networks that can be generated are based upon machine-readable definition files that interprets a single configuration file, which is provided by the end user. With the use of IaC, more specifically, Ansible, it is possible to perform application development and configuration management. The configuration management is done with support through the Jinja extensible templating engine. This enables input configuration files to be dynamically generated upon the end user's request. For instance, the end user can tailor the total size, characteristics and output packet captures.

6) Idempotency and Mutable Infrastructure: The reason for using a configuration management system is that it supports idempotency - ensuring that the same result is acquired upon each (re-)configuration. This automatic provisioning and deployment of infrastructure results in a mutable infrastructure where updates can more easily be applied on a regular basis while preserving state - a beneficial component for host profiling research where a host evolves over time.

7) Configuration-based Attack Generation: The network architecture supports flexibility and extensibility in terms of supported attacks due to the use of atomic-operator. Atomic-operator is a testing framework that launches techniques based upon

MITRE ATT&CK [15]. Through atomic-operator, user-defined machine-readable definition files can be used to simulate a malicious threat agent. This threat agent can model a multitude of attacks, against a multitude of hosts. In addition, the parameters of this agent can be adjusted. For instance, by selecting random IP addresses, or modifying TTL values [5] in order to represent multiple adversaries inside captured packets. Besides modifying the parameters of this threat agent, the adversary can also be arbitrarily placed inside any of the networks, which enables insider attacks to be simulated (typically an internal user with authorized system access).

8) Labelling: Research by Ring et al. stated the importance of labelled traffic inside the dataset. This labelling entails distinguishing benign traffic from malicious traffic. The absence of labelling affects the usability of the resulting dataset, since training and evaluating NIDS often requires labelled data to define anomalies. Various techniques can be utilized to distinguish the malicious traffic. For example, by filtering for the adversaries IP address. However, this causes issues in terms of detecting activity performed as a result of lateral movement and pivoting techniques where other hosts are used to execute the activities. An idea to address this issue is by interconnecting the activity from an adversary to that of anomalous network traffic created by any given instance. Another method to distinguish malicious traffic is by using timestamps when adversaries are actively operating inside the network and comparing these with arrival times or epoch fields within the captured frames. A combination of all three aforementioned approaches improves the successful mapping of malicious network traffic and subsequent labelling of the data. The proposed tool supports full logging of adversary activity (timestamps or adversary IP addresses) whilst collecting all network traffic to potentially match malicious traffic based upon anomalous behaviour.

C. Adversary Simulation

The representation of malicious traffic is a vital element towards generating high-quality datasets to train and evaluate a NIDS. Through two individual and automated adversary simulations, the versatility and use cases of NeDaGen are practically demonstrated. Both adversary simulations consist of a sequence of components, the combination of these components represents a threat agent following the tactics from MITRE, specifically, the ATT&CK Matrix for Enterprises [16]. Both stories contain references to the relevant technique ID's.

1) Email Phishing: A common technique widely used by adversaries to gain sensitive information and/or access to victim systems. Phishing consists of multiple variants, albeit all a form of electronically delivered social engineering. In targeted phishing, also known as spear phishing, specific individuals or companies are targeted. These attacks predominantly encompass tricking the user into clicking a malicious link, subsequently performing unwanted actions such as downloading malware. The implemented automated phishing attack is depicted in Figure 2. Phishing represents step one of this adversary simulation—*initial access* (ID: T1566.002). The second stage is the MITRE tactic *execution*. This simulation focused on user execution of a malicious link (ID: T1204.001) and file (ID: T1204.002) for command and scripting interpreters to execute commands, more specifically, abusing Unix shells (ID: T1059.004). After execution, an adversary may attempt to hide artefacts in order to evade detection. This story focused on storing artefacts inside hidden files and directories (ID: T1564.001). Such traces are only shown upon explicit request by the user. After obtaining initial access, adversaries desire more permissions on the victims' device. In this scenario, the adversary got *credential access* by brute-forcing the root user's password, more specifically, password guessing (ID: T1110.001). With the obtained password, the adversary succeeded the *privilege escalation* by having credentials, thereby login access, of a valid local account with more privileges on the system (ID: T1078.003). Subsequently, the adversary attempts to achieve persistence by abusing task scheduling functionality for the recurring execution of malware, in this case, the Cron utility was utilized (ID: T1053.003). The next element of this adversary simulation is focused on *discovery*, where an adversary performs reconnaissance and scanning techniques to discover other remote systems in the network that can be used to gather more data, or become a new entry point to a system. This story makes use of the utilities Ping and Netcat to see which hosts are reachable in the computer network (ID: T1018) and to scan for open ports/ services (ID: T1046). After gaining knowledge of the network structure, the adversary performs *lateral movement*: a tactic where adversaries attempt to pivot through multiple systems and accounts to achieve their primary objective. The adversary in this story abused weak password management principles by the root user, namely, that secrets were disclosed inside the bash command history. More specifically, an SSH authentication attempt was logged that included the credentials to connect to the external server (ID: T1021.004).

Having established a reverse connection with another server, the next goal of an adversary is typically to obtain desired data, also known as the

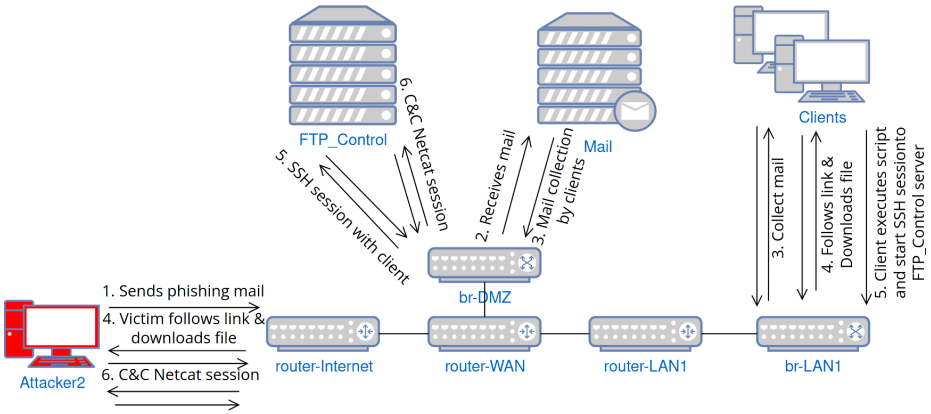

Figure 2: Email Phishing: Exploitation Overview

collection tactic. By making the SSH connection, the adversary collected data from the server which was locally stored at that system (ID: T1005). On the same system, the adversary started a Netcat session with TCP, on a port assigned by the Internet Assigned Numbers Authority, to its device outside the victim's network. Establishing a reverse shell connection to leverage the *command and control* tactic with the non-application layer protocol technique (ID: T1095). Having collected valuable information and having an open session, adversaries attempted to perform techniques for data *exfiltration*. In this scenario, the adversary exfiltrated the data over the command and control channel (ID: T1041). As a final step, adversaries usually attempt to disrupt availability or compromise integrity by manipulating data or processes, known as the *impact* tactic. In this scenario, the adversary manipulated the contents of data in a file after the collection and exfiltration (ID: T1565.001).

2) LOG4J: In December 2021, a critical remote code execution (RCE) vulnerability inside the widely-used logging library *Log4j* was released. This vulnerability matches closely with an initial access technique by MITRE (ID: T1190) where "adversaries may attempt to take advantage of a weakness in an Internet-facing computer or program using software, data, or commands in order to cause unintended or unanticipated behaviour" [16]. To practically demonstrate the opportunities of NeDaGen, a container with a vulnerable Log4j library was configured inside the DMZ of the proposed network architecture. The vulnerable container represents an integral element of this adversary simulation—*initial access*. In this case, an adversary was actively gathering information about the DMZ through active scanning approaches (ID: T1595.002). More specifically, with the commonly used network scanner *Nmap*. As a result, the adversary obtains the information that an outdated web server is exposed. In light of recent attacks, the adversary configured infrastructure to exploit the outdated Log4j library. Figure 2 depicts a summary of the Log4j exploitation process. The re-

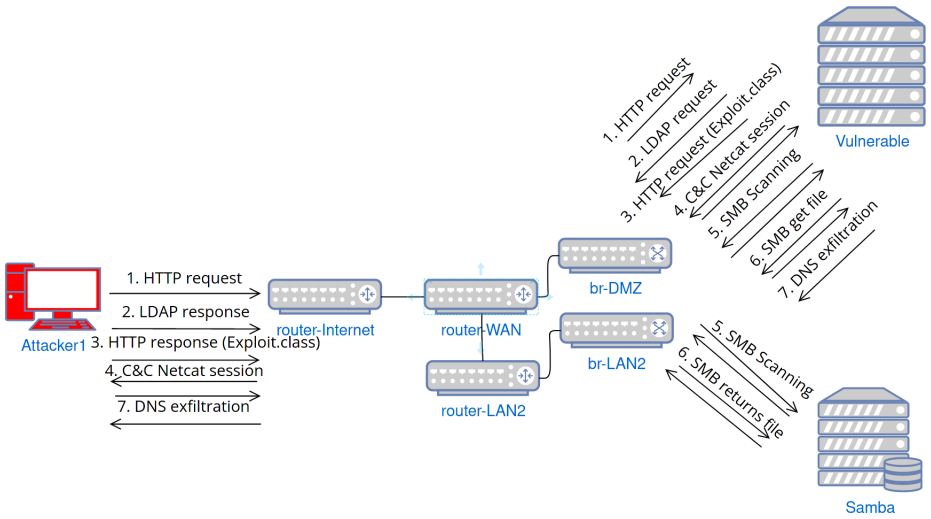

Figure 3: Log4Shell: Exploitation Overview

quired infrastructure by the adversary to successfully obtain a shell session on the vulnerable machine as a result of abusing the RCE vulnerability is as follows (note that these must all be reachable by the victim and in control of the adversary): a web server, an LDAP server and a (Netcat) listener. The exploitation process is depicted in Figure 3 (1-4). The Java Naming and Directory Interface (JNDI) can be abused to contain format strings that, when logged by Log4j, allow information to be retrieved remotely. In combination with the Lightweight Directory Access Protocol (LDAP), an adversary can construct JNDI references inside HTTP(S) requests to point the victim to an adversary-controlled LDAP server. Such a server typically hosts malicious Java classes that are executed upon retrieval (ID: T1059.004). In this scenario, the adversary issued the HTTP request containing the JNDI payload which in turn pointed the vulnerable server to issue an HTTP request to retrieve the malicious Java class (Exploit.class). At this stage, RCE is obtained. In this simulation, the malicious Java class instructed the vulnerable server to connect to the Netcat listener. Subsequently, and as a result of a successful compromise, an adversary typically attempts to move laterally in the network through the new access point. For that reason, this scenario simulated an adversary performing active SMB scanning approaches to retrieve sensitive files (ID: T1135). While pivoting through the vulnerable server (a technique to route traffic from a compromised instance towards elements inside the network not initially accessible by the adversary), sensitive files could be retrieved from an authorized SMB share. This instance was located inside the LAN, as depicted in Figure 1. After retrieval, an adversary typically pursues data exfiltration techniques whilst avoiding detection. In this simulation, the adversary configured a local DNS server and transmitted the sensitive files over DNS (ID: T1048.003) rather than using the existing command and control channel.

IV. Evaluation

The proposed tool is evaluated based upon two main properties. First, the tool is examined against the requirements and conditions for generating high-quality datasets. Afterwards, the quality of the generated dataset is compared to the attributes and characteristics of existing datasets.

A. Performance Evaluation

As discussed in Section III., a vital component for producing high-quality datasets is scalability in terms of the network size (devices, services, servers and protocols) but also the amount of generated traffic. To support scalability whilst also simplifying reproducibility, OS-virtualization was utilized in the form of containers. As a result, it was found that the resulting network architecture requires minimal CPU and Memory usage, as displayed in Figure 4 and Figure 5.

Each line in the graph represents a single node. The trend in the curves indicate a spike in resources whilst the network is being configured; an expected consequence. More importantly, after the network has been configured, the utilized resources are minimal (indicated by the flattening of the curve). Furthermore, the figures depict that the clients, forming a predominant part of the network architecture (represented by the bottom lines), utilize nearly no resources, even whilst running traffic generation scripts.

B. Non-Functional Requirements

As discussed in Section II., various research emphasizes the importance of the following properties: availability/reproducibility, interoperability/flexibility, scalability and quality, to assess the overall quality of the resulting testbed. The relation between those commonly accepted properties in respect to the proposed tool is discussed here. Firstly, as the proposed tool was published open source, it adheres to both availability and reproducibility since the tool can generate datasets on demand. Secondly, a method is provided to the user for creating and/or configuring new attacks or architectural features through user-defined configuration files. The user-defined configuration files also provide the user with a method to specify the size of output packet captures to comply with scalability, according to Vasilomanolakis et al. Finally, the qualitative properties in terms of high-quality datasets are further discussed in the following section.

C. Qualitative dataset Requirements

The four principles of availability, reproducibility, interoperability and flexibility remain to be the common properties and serve as a commonly accepted evaluation basis, as discussed in Section B. However, the proposed tool was also evaluated against the more detailed categories and properties as defined by Ring et al.

1) General Information: To comply with the general information category and its properties, the tool to generate the datasets is publicly available, enabling a flexible year of creation, meanwhile adhering to the property of public availability. In addition, the resulting dataset encompasses regular as well as malicious traffic.

2) Nature of the data: The resulting dataset of NeDaGen is in packet-based network traffic format, more specifically, either in the form of packet captures or JavaScript Object Notation (JSON).

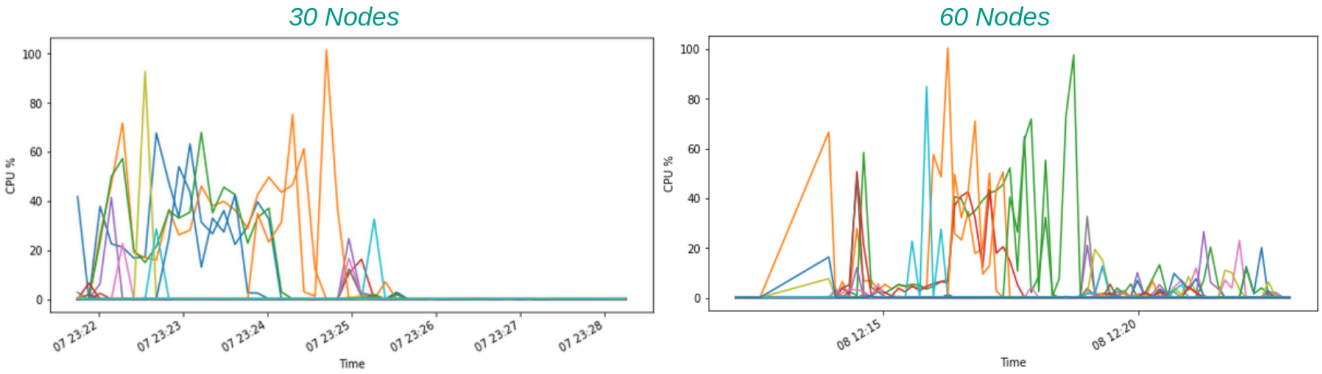

Figure 4: CPU Usage Comparison: 30 nodes vs 60 nodes

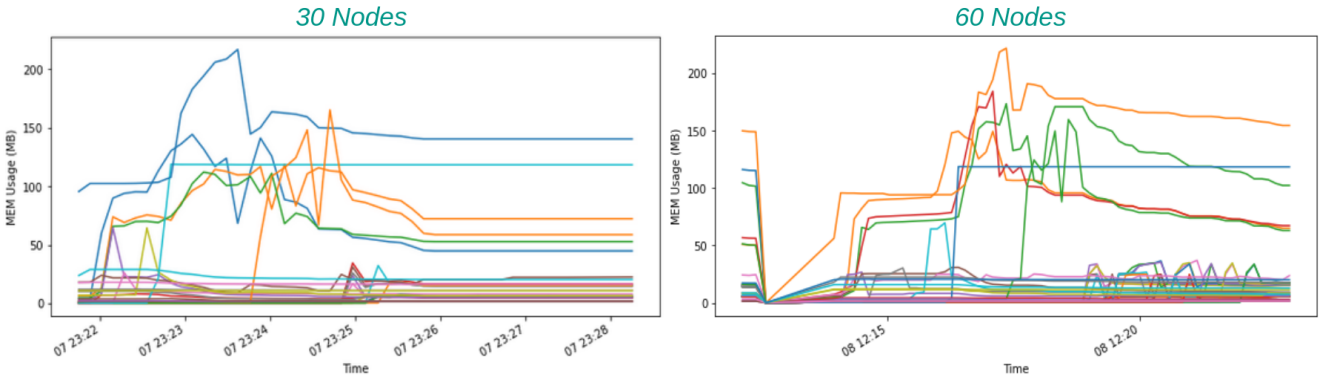

Figure 5: Memory Usage Comparison: 30 nodes vs 60 nodes

Furthermore, since the network architecture and its features can be customized, privacy issues are only a concern in terms of the resulting dataset, which is inherently dependent on the included characteristics inside the network - as defined by the user. Data anonymization is not applicable due to the end-user being in control of the resulting dataset.

3) Data Volume: Sharafaldin et al. stated that high-quality NIDS datasets are rare, frequently due to the lack of the required volume of traffic. Therefore, an important property is the dataset's size (either expressed in the number of contained packets or its physical size). Furthermore, the dataset should represent network traffic recorded over a longer period of time to take periodical effects into account. This is further emphasized by Maciá-Fernández et al., that expressed the importance of traffic distribution in terms of daytime versus nighttime as well as workdays versus weekend days [10].

4) Recording Environment: Due to the exact definition of a productive network environment being absent in the work by Ring et al., it is yet unclear whether the resulting dataset of this proposed architecture is classified as real, or emulated. However, an assumption can be made that the traffic is real since the tool is not merely focused on the reproduction of data, but rather the creation thereof.

5) Evaluation: Since the network architecture is customizable, various subsets may be included to distinguish training from evaluation. However, the proposed network architecture does not support this by default. Finally, due to the traffic distribution as a result of the traffic generation scripts being user-defined, the balance of the dataset in respect to their class is a consequence of the definitions as defined by the user.

V. Discussion

The performance of the proposed tool is especially promising in terms of resource usage with a substantial client representation. The utilized resources are predominantly occupied by the servers, whilst clients occupy a relatively minimal amount of resources. For that reason, the tool is capable of building a network representing over a thousand nodes whilst occupying relatively minimal resources. In addition, the examination of the properties that assess the overall quality, and the inherent quality of the dataset, indicate that the resulting dataset complies with most properties, suggesting the capability to build high-quality datasets. Furthermore, the flexibility and extensibility of the architectural design demonstrated promising results in adversary simulation, specifically in terms of the simplicity through which adversary simulations can be included as a result of supporting a widely used container platform. Finally,

the support for configuration-based attack generation through atomic-operator implies the tool not being limited to prevailing attacks.

A. Limitations

Despite the promising use cases, a multitude of challenges remain. These are discussed below:

1) Microsoft Windows: A major limitation of the tool is the absence of Microsoft Windows devices, services and servers as well as the corresponding protocols, and subsequently, the corresponding telemetry network traffic. The absence hereof indicates the resulting collected traffic being unrealistic because of their popularity inside (enterprise) networks. It is worth mentioning, however, that none of the existing literature indicated the lack of Windows systems as a measure for the quality of the testbed and associated datasets. It merely suggests that improving upon the inclusion and representation of Microsoft Windows characteristics could be done by merging the resulting dataset with one that does contain this telemetry network traffic, or by implementing a multi-node set up through which one part of the architecture is hosted on a Microsoft Windows operating system.

2) Labelling: Another limitation of this study is the missing support for labelling of the malicious traffic in the dataset. A substantial amount of the malicious traffic is distinguished from the benign network traffic by filtering on the source IP address of the adversary. However, both adversary simulations include lateral movement techniques, pivoting from server to server across multiple regions inside the network. Despite this representing realistic adversary traffic (according to MITRE ATT&CK), it causally affects the ability through which benign traffic can be distinguished from malicious traffic since the source and destination IP addresses are modified. Although the proposed tool supports the logging of timestamps and actions of the adversaries, solely relying on such superficial metrics may result in the inaccurate mapping of adversary traffic. For instance, because the timestamp of the logged adversary actions may not fully match with the timestamp included in the packet data, albeit close. Subsequently, false positives can be caused by benign traffic matching the traffic created by the adversaries during a comparable time period. A potential solution to overcome this limitation is to more concretely mark the traffic generated by the adversary or by combining endpoint log files and matching adversary activity, albeit complex to realise.

B. Future Work

Future work primarily encompasses enriching the heterogeneity of the proposed architecture as well as the interoperability in terms of adversary simulations. In addition, a promising idea is to base the preset of user activity profiles in terms of generated traffic, on existing literature into host profiling, resulting in more realistic traffic distribution. Furthermore, the capability of replacing private IP addresses that were used as part of the DMZ, interior router, Internet router network and the 'virtual' Internet into public IP addresses, could better represent a production network environment.

VI. Conclusion

In this paper, an overview of the state-of-the-art network-based traffic dataset generators was provided, and their limitations were addressed. For example, the lack of attack diversity and features, as well as outdated attacks and data anonymization. Various required properties for the generation of high-quality datasets were described, analysed and implemented. Such properties include the relation in respect to the nature of the data: communication flows and packet-based network traffic, in addition to the relation in respect to the data volume: count, duration and traffic distribution. The resulting architectural design, and the design considerations to overcome existing limitations/adhere to properties, were discussed and depicted. The use cases as a result of this architecture were practically demonstrated through the development of two automated adversary simulations. The proposed tool was evaluated based upon the performance, CPU/Memory Utilization, compliance with respect to the non-functional requirements and qualitative dataset requirements. Finally, the said limitations are being addressed and IPv6 support is in development and will be made publicly available upon completion. We hope that our contribution will help researchers and security practitioners, to develop and test novel intrusion detection algorithms.

NeDaGen with its source code is available for public use and further research [7].

Acknowledgements

We would like to express our sincere gratitude to Irina Chiscop and Federico Falconieri for their continuous support, knowledge and constructive feedback throughout the execution of this research.

References

[1] K. Scarfone and P. Mell, *Guide to intrusion detection and prevention systems (idps)*, en, Feb. 2007. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=50951.

[2] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018. DOI: 10.5220/0006639801080116.

[3] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," vol. 2, 2019. DOI: 10.1186/s42400-019-0038-7. [Online]. Available: https://doi.org/10.1186/s42400-019-0038-7.

[4] C. G. Cordero, E. Vasilomanolakis, N. Milanov, C. Koch, D. Hausheer, and M. Mühlhäuser, "Id2t: A diy dataset creation toolkit for intrusion detection systems," in *2015 IEEE Conference on Communications and Network Security (CNS)*, Sep. 2015, pp. 739–740. DOI: 10.1109/CNS.2015.7346912.

[5] E. Vasilomanolakis, C. G. Cordero, N. Milanov, and M. Mühlhäuser, "Towards the creation of synthetic, yet realistic, intrusion detection datasets," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 1209–1214. DOI: 10.1109/NOMS.2016.7502989.

[6] T. Jirsik and P. Velan, "Host behavior in computer network: One-year study," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 822–838, 2021. DOI: 10.1109/TNSM.2020.3036528.

[7] D. van Wijk and J. van Saane, *Nedagen*. [Online]. Available: https://gitlab.com/cossas/nedagen.

[8] *Mcafee labs threats report*. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-dec-2016.pdf, (accessed: 10.01.2022).

[9] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.06.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740481930118X.

[10] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "Ugr'16: A new dataset for the evaluation of cyclostationarity-based network idss," *Computers & Security*, vol. 73, pp. 411–424, 2018, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2017.11.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817302353.

[11] *Containerlab*. [Online]. Available: https://github.com/srl-labs/containerlab.

[12] C.
bibinitperiod I. S. Agency, *Control system security dmz*. [Online]. Available: https://www.cisa.gov/uscert/ics/Control_System_Security_DMZ-Definition.html, (accessed: 09.02.2022).

[13] V. le Pochat, T. van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, ser. NDSS 2019, Feb. 2019. DOI: 10.14722/ndss.2019.23386.

[14] *Containerlab*. [Online]. Available: https://containerlab.srlinux.dev/cmd/deploy/#runtime, (accessed: 01.02.2022).

[15] J. Rickard, *Atomic-operator*. [Online]. Available: https://github.com/swimlane/atomic-operator.

[16] *Att&ck matrix for enterprise*. [Online]. Available: https://attack.mitre.org/.