Observable and Reproducible Sandbox Environment for Cybersecurity Research

Vojdan Kjorveziroski*, Sonja Filiposka*, Anastas Mishev*
*Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Email: {vojdan.kjorveziroski,sonja.filiposka,anastas.mishev}@finki.ukim.mk

***Abstract*—The availability of controlled sandbox environments from which relevant network traces and system logs can be obtained is a prerequisite for studying network attacks. Such data is also essential for the development of novel security tools as well as for evaluating the effectiveness of various incident response strategies. A persistent challenge faced by security researchers is how to set up these environments in a consistent manner, guaranteeing the integrity of the underlying infrastructure while still providing meaningful data. In this paper we present the design of a secure, reproducible, and observable sandbox in which purposefully vulnerable applications are deployed and exploited. Every interaction is monitored in real time using strategically placed network probes, performance monitoring agents, and log shippers. The use of modern configuration management tools eliminates configuration drift and entropy, guaranteeing reproducibility of the deployed scenarios. We describe practical approaches for implementing the sandbox, detailing the tools and methods required for real-world deployment. Using a recent sandbox instantiation as an example, we demonstrate how the acquired network data can be used to identify attack patterns, analyze defense strategies, and assess overall incident response effectiveness. The obtained dataset is open-sourced and can be reused to support further research.**

***Index Terms*—cybersecurity sandbox, incident response, network monitoring, network flow analysis, attack vector analysis**

I. Introduction

Cybersecurity plays an essential role in the connected lives of both individuals and enterprises. To counter modern threats, continuous research is required into novel attack strategies, software vulnerabilities, effectiveness of defensive measures, and quality of incident responses. To support this research, real-world datasets are a prerequisite, consisting of network flows, packet captures, application logs, and performance data [1]. Although this data is part of any production network, researchers often have difficulty obtaining it due to privacy and regulatory concerns. Network operators are wary of sharing any traces obtained from their live networks in order to prevent data leaks or disclosure of any sensitive information about the infrastructure, such as active defense measures in place.

A commonly practiced approach to alleviate privacy concerns is to first anonymize the data before using it in further research [2], although this is not always feasible. Anonymization of network data can reduce its value as concrete threat actors can no longer be identified. Additionally, anonymization of full packet captures or even log traces is an extremely hard challenge, as the data is unstructured and meticulous attention is required to ensure no sensitive data is leaked, making this a prohibitively expensive and time-consuming process.

An alternative approach to acquiring data from live networks to support cybersecurity research is to leverage sandbox environments. These sandbox environments can either be replicas of real-world networks, acting as their digital twins or exist independently on their own [3]. By providing strong isolation and utilizing strategically placed network probes to analyze network traffic along with performance data and log traces, controlled sandbox environments can be a valuable resource to study attack patterns and incident response effectiveness. However, for the sandbox environments to be useful, they must be easily deployable, resource efficient, reproducible, and most importantly, easily observable, so that raw data can be seamlessly obtained or even streamed in real time.

The goal of this work is to present the design of a secure, reproducible, and observable sandbox environment for cybersecurity research where purposefully vulnerable applications and honeypots can be deployed to study attack patterns and effectiveness of applied defensive strategies. Special attention is paid to network monitoring and environment observability, placing sensors along strategic locations in the infrastructure. This enables capturing heterogeneous data, such as network flows, service logs, and performance information of involved machines. The described observability approach is not limited to the presented sandbox design and can be used in any live network as is. The network flows collected in this work have been open-sourced and are freely available for further use [4].

The main contributions of this work can be summarized as:

- Design a secure, reproducible, and observable sandbox environment for running cybersecurity experiments.
- Describe how the sandbox environment can be used in practice, drawing on experience from a recent real-world deployment.
- Demonstrate the utility of the sandbox environment by studying attack patterns, defense strategies, and overall incident response effectiveness through analysis of the acquired network data and log traces.

This study is supported by the NATO Science for Peace and Security Programme as part of SENTAI MYP (grant number G7593) and the Faculty of Computer Science and Engineering at the Ss. Cyril and Methodius University in Skopje, under the FORT project.

979-8-3195-4340-0/26/$31.00

The rest of the paper is organized as follows: Section II reviews existing sandbox environments for cybersecurity research and approaches for collecting network and log data. Section III presents the design of a secure, observable, and reproducible sandbox environment. Section IV describes a real-world deployment and demonstrates the usefulness of the collected data in identifying attack and defense strategies. Finally, Section V concludes the paper and outlines future work directions.

II. Related Work

The availability of high quality raw data is a common requirement in cybersecurity research, whether concrete malware samples are studied, network attack patterns are identified, or incident response strategies are evaluated. The concept of using a dedicated, isolated environment to study a malicious software's behavior is a common tactic in malware research. Such software based sandbox environments have been extensively described in literature and have become even more relevant with recent advancements in artificial intelligence (AI) and machine learning (ML). Guven et al. [5] have developed a software sandbox for dynamic analysis of malware samples, where all network traffic is recorded during their execution. They implement highly accurate classification pipelines for the captured network traffic, based on distinct extracted network features. Kachare et al. [6] implement a similar sandbox environment focused on internet of things (IoT) malware with capabilities for static, dynamic, and network analysis. However, in both cases, the focus is on host-level malware executed on a single device, without analyzing network vulnerabilities, attack patterns, or incident response strategies.

At the network level, a number of real-world datasets have been described in literature, such as CIC-IDS-2017, CSE-CIC-IDS-2018 [7], UNSW-NB15 [8], and ISCXIDS2012 [9]. However, these datasets are now relatively outdated and do not capture modern attack techniques. Additionally, they are often imbalanced, with benign traffic outweighing malicious traffic and certain attack types being underrepresented. This reduces the usefulness of the dataset for development of new security solutions, as well as hindering effective training of AI models.

To support research of network-level threats while considering the limited availability of recent and balanced real-world datasets, researchers have turned to generating synthetic datasets. Such synthetic dataset generation is made significantly easier with the use of generative AI tools. Partovian et al. [10] introduce LogGenST, a synthetic log generation framework utilizing multiple large language models (LLMs) in an adversarial setup. The authors of [11] extend this idea of using LLMs for synthetic dataset generation and apply it to a social media context. They fine-tune an existing LLM to support generating security traces and indicators of compromise (IoC) related to malicious URLs, emails, and file hashes, replicating the style of a real curated dataset. Synthetic dataset generation is not limited only to log-based or message-based datasets. Jiang et al. [12] generate high-fidelity network traces that resemble real-world traffic from a text-based description.

A commonly explored alternative to synthetic dataset generation is the use of honeypots or network traps. The authors of [13] and [14] explore the use of software-based honeypots to acquire real-world, relevant data for cybersecurity research.

A number of strategies have been explored in state-of-the-art literature to overcome the problem of insufficient raw data for cybersecurity research. However, no single work combines real-world data collection with a comprehensive description of the underlying infrastructure architecture enabling heterogeneous dataset capture. We bridge this gap by offering both a realistic dataset and an observable and reproducible sandbox environment that can be implemented as-is, aiming to improve attack detection and incident response quality.

III. Sandbox Environment Design

The primary design goal of the sandbox environment is to allow safe deployment of purposefully vulnerable software and use of offensive tools to exploit it, without jeopardizing any production networks or risking data leakage. Extensive monitoring ensures traces from all offensive and defensive actions are recorded in real-time and are available for analysis. To ensure that the obtained traces are suitable for use as research datasets, while minimizing the effort required for multiple deployments, the sandbox should also be reproducible. Strong extensibility via additional scenarios introduces a dynamic aspect to the environment, allowing it to be continuously enhanced to accommodate emerging threats and attack patterns.

Taking into consideration these characteristics, four main functional requirements can be identified which need to be fulfilled by the sandbox environment: i) Security; ii) Extensibility; iii) Reproducibility; iv) Observability. We describe each requirement in more detail below and discuss how it is fulfilled in practice, referencing concrete tools and methods. Fig. 1 provides a visual overview of the discussed design, following the established requirements.

A. Security

A core feature of any sandbox environment is its effectiveness in providing strict isolation from and to the rest of the infrastructure. This is realized at two different levels: network isolation and runtime isolation.

The network isolation of the sandbox environment follows a multi-layered approach. All sandbox components are executed in an isolated network segment segregated from production using virtual local area networks (VLANs). A dedicated OPNsense network firewall appliance controls all inbound and outbound traffic, with strict isolation rules in place, depending on the active scenario within the sandbox. Remote access to the environment is provided using secure virtual private networks (VPNs) [15], where the firewall also acts as a VPN server. Each user is assigned a personalized OpenVPN profile where only relevant network traffic is passed through the tunnel, employing a split tunnel topology. To avoid providing any hints during reconnaissance phases in terms of the utilized subnet ranges, a summary route covering utilized private IP ranges is pushed to the clients.

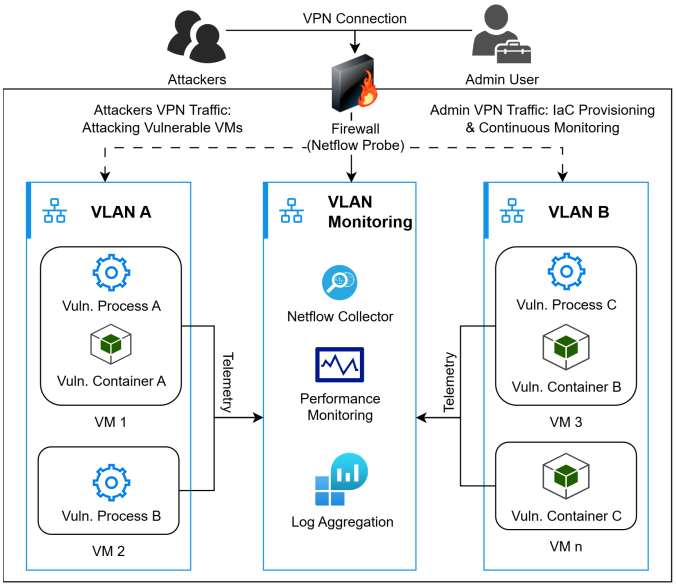

Fig. 1. Sandbox Environment Design

The described network setup where only access to required resources within the sandbox is whitelisted ensures that the underlying integrity of the infrastructure is protected. Additionally, such strict isolation can also be leveraged to introduce multitenancy aspects to the sandbox, where multiple independent scenarios can be executed in parallel.

Strong runtime isolation guarantees that workloads within the sandbox cannot escape and access the memory of other applications running on the same hardware. This is ensured by running all scenario components within virtual machines. To support a wider variety of possible scenarios, applications can also be run in containers. Taking into account the weaker isolation that containers provide by themselves compared to virtual machines, all container hosts run within a virtual machine in the sandbox environment.

B. Extensibility

For a sandbox environment to be useful, it must have a dynamic component, allowing it to adapt to new and emerging threats and defense strategies. The proposed sandbox environment is highly extensible, allowing additional scenarios to be defined at any point in time. Each scenario is described using an infrastructure as code (IaC) approach [16], utilizing the SecGen framework [17] and using vulnerable containers from the Vulhub project [18]. Additional dynamic customization, such as creating user accounts for each team member or the necessary secrets to encrypt the monitoring data in transit is conducted using Ansible playbooks, acting as a configuration management tool. This approach enables flexible composition of runtime environments, allowing some components to be deployed natively within the VM while others run as containers.

C. Reproducibility

The use of the IaC paradigm ensures all sandbox scenarios are reproducible and reusable. Once the initial deployment of the environment is completed using SecGen, additional customization can be applied using Ansible or other configuration management tools. By decoupling the initial deployment from the subsequent customization, the same base image can be reused for multiple scenarios, conserving time and resources.

Utilizing the IaC paradigm also enables easy sharing of scenarios with the wider community, facilitating collaboration. It also greatly reduces infrastructure entropy, guaranteeing stable versioning of deployed components and tools.

D. Observability

One of the main aspects of the sandbox environment is to support creating relevant cybersecurity datasets which can subsequently be used in research activities. To accomplish this, monitoring agents and probes are placed at strategic locations within the environment to capture all actions in real-time. The data that can be acquired in such a fashion include: network flows, application logs, and performance monitoring of the VMs and containers where the applications are executed.

As the network monitoring tool landscape is diverse, the proposed design is compatible with a variety of tools, and below we present one implementation option. Regardless of the selected software suite, it is advised that it supports open formats, so that the resulting datasets are freely reusable.

The recommended approach to obtain network flow information is to leverage the dedicated sandbox firewall as a Netflow probe, sending information to a Netflow collector, such as nTop. The nTop collector then persists the data into a ClickHouse database and offers a dedicated web interface from where the obtained flows can be visualized and analyzed. Depending on the network topology, dedicated Netflow probes such as nProbe can be placed in each VM that is part of the scenario, ensuring that layer 2 traffic between the sandbox VMs is analyzed as well. Alternatively, a single nProbe instance can extract flow information from multiple sources, provided that port mirroring is in place at the switch level.

Application logs are acquired using Grafana Promtail as a log shipping agent deployed on each VM in the sandbox and are sent to a VictoriaLogs instance acting as a dedicated log collector for analysis and safekeeping. Each log message can be further enriched to include additional metadata information, including the source application, host, and optionally container details. VictoriaLogs includes a web interface for easy browsing, filtering, and log analysis.

Similarly to application logs, performance metrics are obtained from each VM in the sandbox using dedicated Zabbix monitoring agents which forward the data to a central server. Such performance metrics are not only useful for later analysis, but can also be leveraged as a continuous healthcheck of the infrastructure, notifying of any failures or excessive resource usage, thus also safeguarding the availability and integrity of the underlying infrastructure.

In scenarios where communication with other systems is required, such as for forwarding network flows or logs to a central server, strict firewall rules following the least privilege principle are configured to whitelist the traffic in question.

IV. Real World Case Study

The described sandbox environment has been implemented as part of a cybersecurity challenge aimed at generating a dataset for the future training and evaluation of cybersecurity AI models. A total of 119 participants were divided into groups of up to 5 members, forming 24 teams in total. The challenge comprised two phases, a blue phase and a red phase.

During the blue phase, participants were asked to survey their own VMs, identify vulnerabilities, and install monitoring and security software of their own, to better protect their environment. No attacks were permitted during this phase, and the network infrastructure was configured to only allow access to team VMs to the respective team members.

The subsequent red phase focused on real-time defense and exploitation. Each team was allowed to attack the VMs of other teams, as the network isolation between the VMs in the sandbox environment was lifted. This two-phased approach enables the sandbox environment to more closely resemble a production setting, where standard defense mechanisms are in place. Team members act as dedicated operators monitoring the infrastructure in real-time, able to respond to ongoing threats, thus making the resulting dataset more dynamic and reminiscent of a production network under attack.

The scenario was conducted over a period of 2 weeks. To ensure the security of the underlying infrastructure, a total of four separate VLANs were utilized, with the 24 team VMs distributed among them. The use of multiple VLANs not only provides isolation from the rest of the network but also encourages participants to do a more thorough reconnaissance of the sandbox environment. Participants accessed the environment via personalized OpenVPN profiles.

The implementation of the sandbox environment adheres to the 4 design principles described earlier: Security, Extensibility, Reproducibility, and Observability. In total 14 different vulnerable applications were deployed, with both virtual machines and containers being utilized to host them. Table I provides further details on the scenarios and their runtime environments. Each VM contains a randomly selected combination of up to 5 vulnerabilities, with a weak user password being present in all cases, without exception.

Using the obtained network flow information and log messages, we analyzed the behavior of the participating teams after the event concluded. This analysis aims to characterize the resulting dataset by providing insight into the interactions, attack patterns, and defensive behaviors captured during the challenge. In addition, we assess the effectiveness of the identified attack and defense strategies, highlighting the dataset's applicability for training and evaluating cybersecurity models.

When it comes to the network flows, the analysis is focused on connections either originating or terminating from the VPN tunnel subnet or one of the VLAN subnets where the team VMs are placed. These are the connections of interest, as they might be related to malicious behavior and exploitation of software vulnerabilities. In total, 47,071,281 such flow records have been identified during the course of the activity. Focusing

TABLE I
Vulnerable Application Scenarios

Name	Description	Runtime[1]
SSH authentication	Weak user password	VM
MySQL database	Weak password & vulns.[2]	VM
vsftpd FTP	Backdoor	VM
ProFTPD FTP	Backdoor	VM
Netcat shell	Backdoor	VM
DVWA web app	OWASP Top 10	VM
Juiceshop web app	OWASP Top 10	C
PhpMyAdmin	File inclusion	C
Grafana	File inclusion	C
Wordpress 2	Many vulnerabilities	VM
Wordpress 3	Many vulnerabilities	VM
Joomla	Remote code execution	C
Gitlist	Remote code execution	VM
Adminer	Arbitrary file read	C

[1] VM – Virtual Machine; C – Container.
[2] Vulnerabilities.

solely on traffic aimed at the vulnerable services, Fig. 2 shows a breakdown of how many network flows have been recorded for each vulnerable application. SSH and HTTP have recorded the most traffic, which is expected, as the vulnerabilities associated with these services are generally more accessible and widely targeted. It is worth pointing out that under "HTTP" traffic we classify traffic to the following vulnerable web applications: Wordpress 2, Wordpress 3, Gitlist, DVWA.

Focusing on traffic to and from each vulnerable VM, we see different strategies being employed by the teams. Fig. 3 classifies teams by posture, with most teams choosing a defense-heavy strategy, and only two showing aggressive or highly aggressive behavior. The classification is done depending on the ratio between outgoing and incoming flows from the VM to the other VLAN subnets or from the relevant VLAN subnets to the VM. Interestingly, the most aggressive team has an attack/defense ratio of 19.59, with the second most aggressive team having a more modest value of 1.55.

To better gauge the most active times of day with the highest network activity, Fig. 4 presents a heatmap of flows per hour

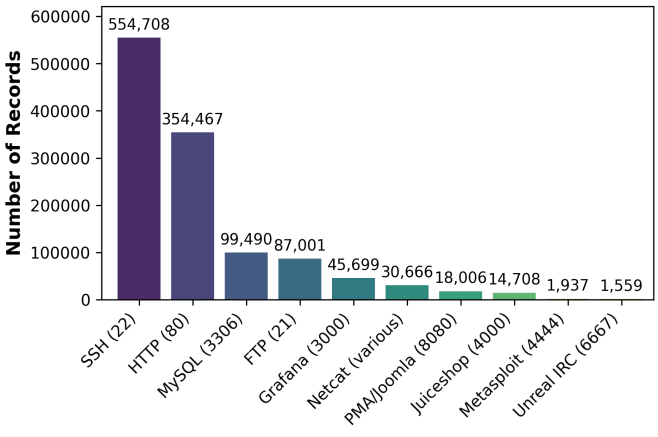

Fig. 2. Network flow distribution across vulnerable services

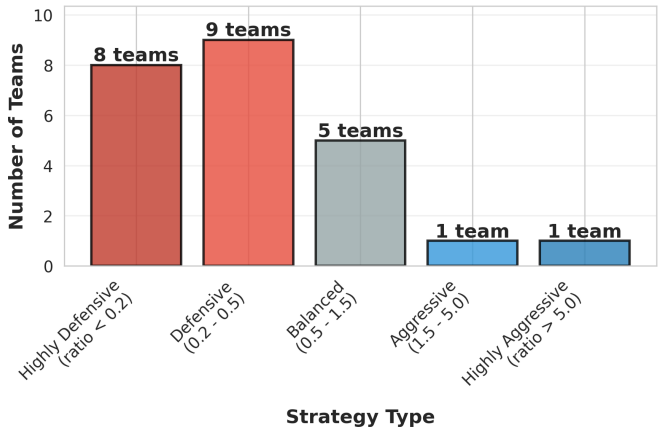

Fig. 3. Team Strategy Distribution: Attack vs Defense Focus

and weekday, covering the challenge duration. Saturday is the most active day of the week in terms of recorded activity. Interestingly, Monday and Tuesday afternoon along with the early morning hours have also been periods of high activity. In general, across all weekdays, afternoon hours are more active, with Saturday, 5:00 A.M. being a single outlier. This extreme value is due to multiple very aggressive port scans being initiated from one of the VMs, accounting for 98 per cent of all generated traffic within the hour in question.

To get a more detailed picture of the events that have occurred during the competition, as well as to evaluate the effectiveness of the employed attack and defense strategies, we have also analyzed the log messages collected from each VM. In total, 16,924,089 log messages have been recorded. Interestingly, more than half of all recorded messages, 8,454,568 have been recorded by a single VM. This same team has been classified as employing a highly defensive strategy, with

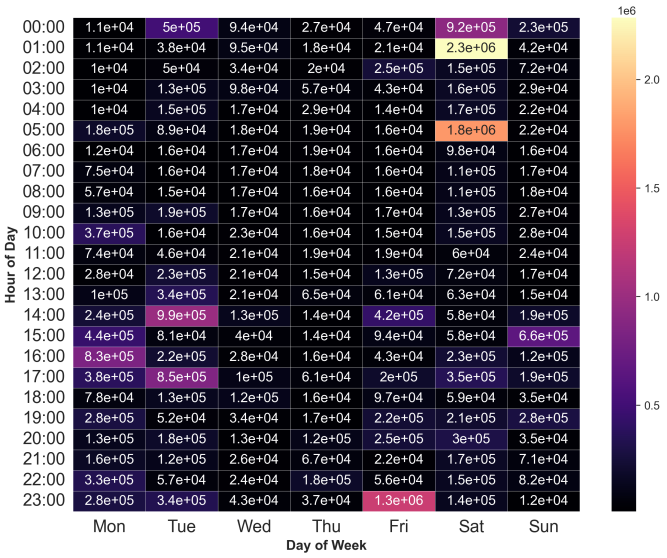

Fig. 4. Number of flows per hour and weekday

an outgoing/incoming connection ratio of less than 0.2, as depicted in Fig. 3. This explains the extreme number of log messages, as this team has been under a constant attack, a fact further corroborated by the network flow analysis.

Focusing on concrete exploitation strategies, SSH dictionary attacks have been most common. As by default the OpenSSH server daemon logs the usernames of failed login attempts, this can be a good indicator whether custom or ready-made dictionaries have been used during the attacks. Most attacks have used custom lists for the usernames, as the majority of failed attempts follow the correct username pattern for the preprovisioned accounts. For each team member a dedicated user account on the team VM had been created ahead of the challenge. All of these accounts use the same username pattern - "user$id_number" where $id_number is a unique identifier for the participant. The majority of the login attempts have been made with usernames conforming to this naming scheme, leading to the conclusion that participants have given a thought to the most optimal strategy for conducting the attack. Generic usernames found in popular dictionary lists have also been attempted, but to a lesser extent.

As OpenSSH also logs the successful login attempts, we can also analyze which VMs have had the most logins. Although this is not a perfect metric by itself since both legitimate and illegitimate logins are counted, combined with the available information from the network flows, clear conclusions can be drawn. The top 5 VMs based on the number of successful logins attribute for more than 50 per cent of all successful logins (2,379 in total). Analyzing the posture of the teams in question, they are all characterized with low attack/defense ratios (2 highly defensive; 2 defensive; 1 balanced).

Analyzing the source IP address of the failed login attempts, most participants have opted to run the brute force software on their own hardware, instead of relying on the team VM. Most failed SSH login attempts originate from the VPN tunnel subnet rather than the dedicated VLAN subnet ranges for the team VMs. The timeframes for the SSH dictionary attacks also support the conclusions drawn from Fig. 4, with the most intense attacks occurring on Monday, Friday, and Saturday, matching periods of increased network activity.

Leveraging the fact that multiple other log files are also available, apart from SSH login data, a plethora of other information can be extracted. The precreated user accounts given to each participant are sudoer accounts and all commands executed with root privileges have also been logged. Through the analysis of the most commonly executed commands, many of the applied defense strategies by the various teams become clear. All of the top commands relate to reading segments of particular log files, adding IPTables deny rules targeting a single (offending) IP address, or sending alerts via external channels (most commonly Discord messages, via appropriate webhooks). Judging by the log files which have been opened, many teams have opted to use Suricata in an intrusion detection system (IDS) mode, Tripwire, and Inotify. Tripwire can detect changes to local files by periodically computing their hash value, while Inotify listens for file system events (open,

modify, delete) related to particular watched files. A number of teams have also opted to use UFW as their firewall of choice instead of relying directly on IPTables.

Analyzing the most commonly executed commands via cron paints a similar picture to the commands executed as a root user. All of the top results are custom scripts for either banning repeated offending users or sending notifications.

As Fail2Ban has been identified as a popular defensive tool deployed on the VMs, we can compare the number of issued bans to the total number of recorded failed SSH login attempts. There have been 126,747 failed SSH login attempts, while only 739 Fail2Ban bans in total. This suggests that many of the teams that have installed Fail2Ban did not complete its configuration or misconfigured it, reducing its effectiveness.

The analysis shows that most teams used well-known intrusion prevention and host firewall software and have adopted a defense-heavy strategy. An interesting phenomenon is the heavy use of customized scripts to send personalized alerts when anomalies are detected on the team VMs. When attacking, focus is given to brute force, with exploitation of more complicated software vulnerabilities being less prevalent.

V. Conclusion

A persistent challenge in cybersecurity research is the absence of high-quality datasets. To mitigate this issue, we designed and implemented an observable, reproducible sandbox environment, where purposefully vulnerable software can be run and exploited. By monitoring the network and infrastructure at strategic points, high-quality datasets comprised of network flow data and log messages can be created.

We have demonstrated the effectiveness of the sandbox environment in practice using a cybersecurity challenge scenario with 119 participants. To highlight the usefulness of the generated dataset, we analyzed the attack and defense strategies employed during the event by each team, drawing conclusions about their effectiveness. Using the obtained data, we identified that most teams primarily focused on defensive measures, while offensive activities were present but less pronounced. Analysis of the collected log messages further indicates that participants have shown critical thinking and have adapted to the environment at hand, modifying dictionary brute force lists to match the identified username patterns.

The collected dataset is not explicitly labeled due to the inherently dynamic nature of the experimental setup. Accurate labeling would require a highly controlled environment with predefined, static, attack scenarios and complete visibility into all actions, which would reduce the realism and diversity of interactions observed in the challenge. Nevertheless, the dataset remains valuable as it captures realistic traffic patterns and adversarial behavior. It is particularly well suited for unsupervised and semi-supervised approaches, such as anomaly detection, as well as for developing and evaluating statistical methods that rely on deviations from baseline behavior.

In the future we plan to develop an AI model leveraging the generated datasets from the sandbox environment for automatic identification of malicious network patterns.

References

[1] H. Ahmetoglu and R. Das, "A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions," *Internet of Things*, vol. 20, p. 100615, Nov. 2022.

[2] G. Longo, F. Lupia, A. Merlo, F. Pagano, and E. Russo, "A data anonymization methodology for security operations centers: Balancing data protection and security in industrial systems," *Information Sciences*, vol. 690, p. 121534, Feb. 2025.

[3] M. Attaran and B. G. Celik, "Digital Twin: Benefits, use cases, challenges, and opportunities," *Decision Analytics Journal*, vol. 6, p. 100165, Mar. 2023.

[4] V. Kjorveziroski, S. Filiposka, and A. Mishev, "An unlabeled netflow cybersecurity sandbox dataset with attack and benign traffic," Apr. 2026. [Online]. Available: https://doi.org/10.5281/zenodo.19452465

[5] M. Guven, "Dynamic Malware Analysis Using a Sandbox Environment, Network Traffic Logs, and Artificial Intelligence." *International Journal of Computational and Experimental Science and Engineering*, vol. 10, no. 3, Sep. 2024.

[6] G. P. Kachare, G. Choudhary, S. K. Shandilya, and V. Sihag, "Sandbox Environment for Real Time Malware Analysis of IoT Devices," in *Computing Science, Communication and Security*, N. Chaubey, S. M. Thampi, and N. Z. Jhanjhi, Eds. Cham: Springer International Publishing, 2022, pp. 169–183.

[7] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *4th International Conference on Information Systems Security and Privacy*, Mar. 2026, pp. 108–116.

[8] N. Moustafa and J. 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)*, Nov. 2015, pp. 1–6.

[9] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, May 2012.

[10] S. Partovian, F. Flammini, A. Bucaioni, and J. Thornadtsson, "LogGenST: A Framework for Synthetic Log Generation Using LLMs for Smart-Troubleshooting," in *Software, System, and Service Engineering*, G. Kardas, I. Luković, B. Milašinović, A. Popović, Ł. Radliński, M. Staroń, J. Swacha, and A. Przybyłek, Eds. Cham: Springer Nature Switzerland, 2025, pp. 64–83.

[11] A. Almorjan, M. Basheri, and M. Almasre, "Large Language Models for Synthetic Dataset Generation of Cybersecurity Indicators of Compromise," *Sensors*, vol. 25, no. 9, Apr. 2025.

[12] X. Jiang, S. Liu, A. Gember-Jacobson, P. Schmitt, F. Bronzino, and N. Feamster, "Generative, High-Fidelity Network Traces," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, ser. HotNets '23. New York, NY, USA: Association for Computing Machinery, Nov. 2023, pp. 131–138.

[13] S. C. Sethuraman, T. G. Jadapalli, D. P. V. Sudhakaran, and S. P. Mohanty, "Flow based containerized honeypot approach for network traffic analysis: An empirical study," *Computer Science Review*, vol. 50, p. 100600, Nov. 2023.

[14] X. Yang, J. Yuan, H. Yang, Y. Kong, H. Zhang, and J. Zhao, "A Highly Interactive Honeypot-Based Approach to Network Threat Management," *Future Internet*, vol. 15, no. 4, Mar. 2023.

[15] V. Kjorveziroski, C. Bernad, K. Gilly, and S. Filiposka, "Full-mesh VPN performance evaluation for a secure edge-cloud continuum," *Software: Practice and Experience*, vol. 54, no. 8, pp. 1543–1564, 2024.

[16] E. Krajchevska and V. Kjorveziroski, "VulnerSec: A Flexible, Automated and Open-Source Cybersecurity Framework," in *22nd International Conference on Informatics and Information Technologies*, Strumica, North Macedonia, Apr. 2025.

[17] Z. C. Schreuders, T. Shaw, M. Shan-A-Khuda, G. Ravichandran, J. Keighley, and M. Ordean, "Security Scenario Generator (SecGen): A Framework for Generating Randomly Vulnerable Rich-scenario VMs for Learning Computer Security and Hosting {CTF} Events," in *2017 {USENIX} Workshop on Advances in Security Education ({ASE} 17)*, 2017.

[18] "Vulhub - Open-Source Vulnerable Docker Environments." [Online]. Available: https://vulhub.org/