AttackZoo: A Reproducible Testbed for Attack Execution and Network Traffic Dataset Generation

Leonardo Bitzki[1,2], Diego Kreutz[2], Leandro Bertholdo[1], Cristhian Kapelinski[2]

[1]Universidade Federal do Rio Grande do Sul (UFRGS)
[2]AI Horizon Labs, PPGES, Universidade Federal do Pampa (UNIPAMPA)
{bitzki,bertholdo}@ufrgs.br, diegokreutz@unipampa.edu.br,
cristhianavila.aluno@unipampa.edu.br

Abstract. *The recurring generation of network traffic datasets for cybersecurity research is constrained by fragmented experimental workflows, limited reproducibility, and restricted coverage of attack vectors. We present AttackZoo, a reproducible testbed that integrates a containerized repository of 60 attacks categorized according to the MITRE ATT&CK framework and an orchestration layer that centers on a command-line interface (CLI) and automates the full cycle of execution, data collection, and consolidation. The pipeline incorporates traffic capture, log inspection, and feature extraction using multiple tools, producing structured and traceable datasets. In the released campaign, AttackZoo executed all 60 attacks across four execution levels and five repetitions, producing 1200 packet captures and 220.4 GB of derived datasets from 672.1 GB of raw traffic. Overall, the catalog covers 35 MITRE ATT&CK technique/subtechnique entries and 9 tactics, while the orchestration layer reduces the operational overhead of experiments and enables the systematic generation of comparable datasets, with all artifacts openly available.*

1. Introduction

Networked cybersecurity has become an economic and infrastructural problem on a global scale. Industry estimates project worldwide security spending to reach USD 377 billion by 2028[1] and 21.1 billion connected Internet of Things (IoT) devices in 2025, growing to 39 billion by 2030[2]. These figures indicate that traffic-based security research must handle traffic at a scale and diversity that are difficult to reproduce with isolated experiments. This challenge is reflected in the detection literature: a systematic review of 119 top-cited anomaly-based intrusion detection studies underscores the central role of evaluation datasets and metrics [Yang et al. 2022], motivating reproducible dataset generation.

Within this context, network security and cybersecurity research increasingly depends on reproducible methodologies for generating and analyzing traffic under controlled conditions, especially in service-oriented, IoT, and heterogeneous environments [Fideles et al. 2025, Guo et al. 2024, Croft et al. 2023, Whang et al. 2023, Soares et al. 2021]. The closest prior testbeds illustrate the remaining gap. Gotham [Sáez-de Cámara et al. 2024] provides reproducible IoT topologies and scenario generation, but its workflow is organized around complete IoT scenarios and

[1]IDC, *Worldwide Security Spending to Increase by 12.2% in 2025 as Global Cyberthreats Rise*: https://my.idc.com/getdoc.jsp?containerId=prEUR253264525.
[2]IoT Analytics: https://iot-analytics.com/number-connected-iot-devices/.

its demonstrated attack coverage is limited to 19 attacks in our comparison. HackInSDN [Brito et al. 2025] focuses on orchestration, monitoring, detection, threat intelligence, and mitigation for SDN (Software-Defined Networking) environments. Neither integrates a broad catalog of attack units, benign traffic generation, packet capture, feature extraction, and dataset consolidation into a single reproducible pipeline. This gap hinders comparable experiments and keeps operational effort high.

Additionally, generating realistic and up-to-date datasets remains a critical challenge [Ahmad et al. 2023, Kraust et al. 2025, Bragança et al. 2026]. Detection and analysis techniques are sensitive to service evolution, communication patterns, and adversarial strategies, which can impose a short validity period on datasets (e.g., 6 months) [Ahmad et al. 2023, Bragança et al. 2026]. Without systematic and reproducible processes, comparability between studies and result validity are both compromised [Kraust et al. 2025, Bragança et al. 2026, Ahmad et al. 2023].

We address these limitations by integrating a repository of containerized attacks with an orchestration tool for the experimental cycle[3]. AttackZoo unifies attack execution, benign traffic generation, data capture, feature extraction, and dataset generation in a standardized pipeline. New attacks are incorporated through container definitions and catalog metadata without changes to the orchestration pipeline, supporting incremental extension and maintenance.

This paper makes three concrete contributions: (i) the AttackZoo testbed, including CLI-based orchestration of the full experimental cycle (Sections 3 and 3.2); (ii) a containerized and extensible attack repository that initially contains 60 attacks in 7 categories, mapped to 35 MITRE ATT&CK[4] technique/sub-technique entries and 9 tactics (Section 3.1); and (iii) an experimental campaign with 1200 packet captures and 220.4 GB of public derived datasets, reported in Section 4 and made available as described in Section 5.

2. Related Work

Recent cybersecurity testbed literature highlights the need for controlled and realistic environments for attack evaluation, behavior observation, and countermeasure validation [Fideles et al. 2025, Guo et al. 2024, Croft et al. 2023, Whang et al. 2023, Soares et al. 2021]. Table 1 positions the main testbeds by interaction with real devices, emulation support, and protocol scope ([1]–[4]); replicability and artifact availability ([5]–[6]); and MITRE ATT&CK coverage. Prior works typically cover restricted subsets of the attack cycle, with a median of four attacks, whereas AttackZoo covers 60 attacks across 35 technique/sub-technique entries and 9 tactics.

In general, works based on real devices [Katuri et al. 2024] prioritize high fidelity by capturing timing and operational noise, which is relevant in cyber-physical scenarios. Emulation, virtualization, and containerization prioritize scalability, lower cost, and flexibility, favoring repeated and reproducible experiments. These studies indicate the need to balance realism and experimental control.

[3]Code & Data: `https://github.com/GT-IoTEdu/attackzoo-sbseg26`

[4]The complete mapping of techniques and tactics is available in the MITRE ATT&CK mapping document in the project repository.

Table 1. Characteristics and coverage of related testbeds

Work	**Testbed Characteristics**						**MITRE ATT&CK Coverage**		
Reference	[1]	[2]	[3]	[4]	[5]	[6]	**Attacks**	**Tech./Sub-tech.**	**Tactics**
[Kouril et al. 2014]	✗	✓	✗	✓	✓	✓	1	3	2
[Bernieri et al. 2017]	✗	✓	✓	✓	✓	✗	4	2	4
[Siboni et al. 2019]	✗	✓	✓	✓	✓	✗	5∗	9	7
[Dhanapal and Nithyanandam 2021]	✗	✓	✗	✓	✗	✓	1	1	1
[Wlazlo et al. 2021]	✗	✓	✓	✗	✓	✗	3	4	4
[Prates Jr. et al. 2021]	✗	✓	✓	✓	✓	✓	4	3	3
[Sáez-de Cámara et al. 2024]	✗	✓	✓	✓	✓	✓	19∗	12	7
[Freitas et al. 2024]	✗	✓	✗	✓	✓	✓	4	4	4
[Katuri et al. 2024]	✓	✓	✓	✓	✓	✗	2	1	1
[Huang et al. 2024]	✗	✓	✓	✗	✓	✗	8∗	4	7
[Brito et al. 2025]	✗	✓	✓	✓	✓	✓	5∗	8	6
This work	✓	✓	✓	✓	✓	✓	**60**	**35**	**9**

Legend: [1] *Can interact with real or externally reachable devices/services*, [2] *Support for emulated / virtualized devices*, [3] *IoT protocols*, [4] *General Ethernet application protocols*, [5] *Experiment replication instructions*, [6] *Accessible artifact repository available for public use*.
∗ The authors do not explicitly report the number of attacks; this value was inferred from the demonstrations described in the paper.

With respect to scope, the literature is divided between vertical and horizontal proposals. The former target domains such as Smart Grid and ICS (Industrial Control Systems) [Kraust et al. 2025, Wlazlo et al. 2021, Katuri et al. 2024, Bernieri et al. 2017]. The latter focus on general-purpose IP networks, exploring denial-of-service (DoS), man-in-the-middle (MitM), scanning, and service exploitation. Cloud-oriented examples include a cloud-based cyberattack simulation testbed [Kouril et al. 2014] and an OpenStack framework for HTTP flooding [Dhanapal and Nithyanandam 2021]; both are reproducible, but narrow in attack or service scope. Domestic and industrial IoT scenarios [Sáez-de Cámara et al. 2024, Prates Jr. et al. 2021, Siboni et al. 2019, Huang et al. 2024] and MQTT-VET [Freitas et al. 2024] add protocol-specific evidence while leaving broader multi-protocol orchestration outside their scope.

Despite these advances, important limitations remain. Half of the surveyed testbeds do not release a public artifact repository ([6] in Table 1) and, although most document their experiments ([5]), no existing proposal simultaneously supports real devices, IoT protocols, general-purpose TCP/IP applications, and open artifact availability. The experimental process is often fragmented across multiple tools, and its scope is typically limited to a small set of attacks or protocols, hindering comparative analyses and compromising reproducibility.

Beyond the feature-level comparison in Table 1, the closest proposals differ from AttackZoo mainly in architectural abstraction. Gotham [Sáez-de Cámara et al. 2024] is a reproducible IoT testbed implemented as middleware over the GNS3 network emulator, emphasizing realistic IoT topologies, emulated devices, and scenario generation for dataset construction. Its strength is topology fidelity at IoT scale. However, its workflow is organized around complete IoT scenarios, whereas AttackZoo organizes experimentation around a broad catalog of self-contained attack containers, each described by metadata, parameters, target mappings, runtime limits, and MITRE ATT&CK references.

HackInSDN [Brito et al. 2025] addresses cybersecurity experimentation from a different perspective: programmable network infrastructures, SDN-based orchestration, monitoring, detection, threat intelligence, and mitigation. Its architecture is suitable for teaching and defensive workflows. In contrast, AttackZoo focuses on attack execution and evidence generation as reusable experimental units. Rather than centering the architecture on mitigation modules, it standardizes the dataset-generation path from attack selection to packet capture, feature extraction, dataset construction, telemetry, reports, and execution metadata.

Therefore, the architectural distinction of AttackZoo is fourfold: (i) attacks are packaged as isolated and replaceable Docker units; (ii) the catalog is declarative and dynamically loaded from `attack.yaml` files, avoiding hardcoded experiment logic; (iii) executions follow a uniform warmup/attack/cooldown pipeline with optional probes, telemetry, and packet capture; and (iv) post-processing is integrated into the same workflow through multiple feature extractors and dataset consolidation. This design turns the testbed into an end-to-end operational pipeline rather than only a topology, a set of scripts, or a domain-specific validation environment. This broader coverage is enabled by the catalog architecture itself: attacks are reusable execution units and the pipeline remains unchanged as new techniques, services, and protocol families are added.

3. Architecture

AttackZoo is composed of a containerized attack repository and a CLI-centered orchestration tool for managing the experimental cycle. Together, they address the two key limitations identified in the literature: fragmented experimental processes and limited attack scope. The repository provides extensible coverage, while the orchestration layer integrates execution, data collection, and consolidation into a standardized pipeline. Figure 1 presents the testbed architecture and core functionalities.

The main architectural decision is to treat each attack as a first-class experimental unit. In AttackZoo, an attack is a containerized component associated with structured metadata, parameter schemas, target-service mappings, runtime controls, safety notes, and MITRE ATT&CK classification. This metadata-driven design decouples the catalog from the orchestration layer: new attacks require only a Docker definition and an `attack.yaml` file, without changing the CLI or experiment pipeline. The same logic can execute attacks against containerized targets, host-exposed services, or reachable real devices intentionally included in the laboratory network.

3.1. Attack Repository

The attack repository is organized into three Docker container categories: attackers, target services, and clients. The attacker catalog comprises 60 attacks, each implemented in a single-purpose container using open-source tools. Individual containers improve isolation, reduce cross-execution interference, and simplify maintenance and future expansion of the catalog. The target services are provided by 9 dedicated containers covering an intentional set of protocols and technologies: a web server with PHP and a database, SSH, Telnet, SMB, an MQTT broker, a CoAP (Constrained Application Protocol) server, an XRCE-DDS agent, a Zenoh router, and a web server with a deliberately outdated SSL/TLS configuration. This diversity distinguishes our approach from works centered on a single protocol or domain.

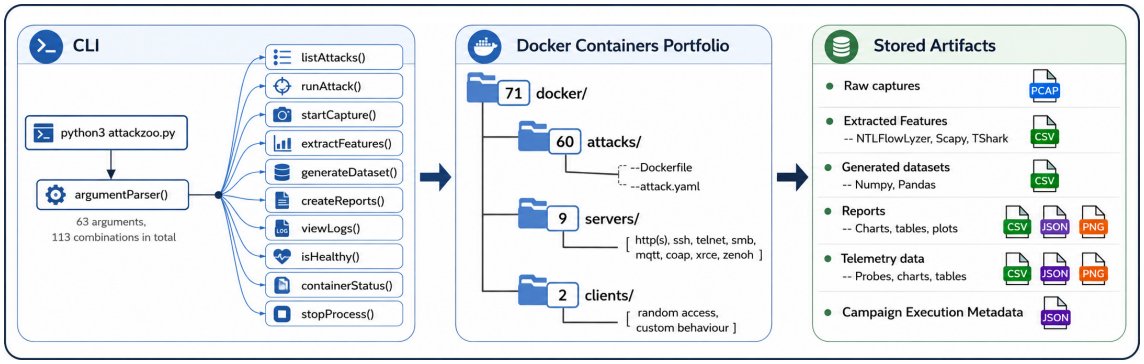


Figure 1. Overview of the testbed architecture and core functionalities.

To interact with these services and compose realistic scenarios, benign traffic can be generated by up to 10 parameterizable client instances, configured by target service, request count, access interval, and execution time. This supports controlled baselines and future use of realistic profiles derived from public traces or benign traffic datasets.[5]

Each attacker is described by metadata covering identification, image, parameters, safety notes, MITRE ATT&CK classification, and contextualized warnings. Catalog entries were curated through build checks, short authorized executions against the intended target, and confirmation of expected packet- or service-level effects from logs, captures, parameters, and MITRE mappings before inclusion; the same metadata drives the tool interfaces and extension workflow.

3.2. Orchestration and Control CLI Tool

The primary operation layer of AttackZoo is the `attackzoo.py` command-line interface, designed to reduce fragmentation across independent scripts. The CLI exposes subcommands for catalog inspection, direct attack execution, automated experiments, capture listing, feature extraction, dataset generation, report generation, container inspection, log retrieval, and controlled shutdown. In particular, `list` discovers the attack catalog, `run` executes an attack by identifier, `experiment` coordinates repeated warmup/attack/cooldown runs with probes and packet capture, `features` extracts traffic attributes, `dataset` consolidates derived data, and `report` regenerates summaries. This modular CLI makes the workflow explicit, reproducible, and scriptable.

Scenario execution follows a controlled pipeline. The CLI validates parameters, instantiates containers in isolated Docker environments, starts packet capture when requested, executes attack or experiment hooks, and stores artifacts for later analysis. Automated experiments apply the same warmup, attack, cooldown phases, and artifact layout to every run, optionally collecting probes, telemetry, captures, features, datasets, and reports. Execution metadata records the attack identifier, level, run number, parameters, and artifact paths.

3.3. Implementation

The repository comprises 71 container definitions: 60 attackers, 9 target servers, and 2 benign client models supporting multiple instances. All containers are based on public

[5]For example, public datasets of internal datacenter traffic [Polák et al. 2024] describe temporal stability and daily application traffic patterns in enterprise environments.

Docker Hub images[6] and are defined through `Dockerfiles`, allowing the testbed to be rebuilt in environments that meet the minimum requirements.

The integration between the repository and the tool adopts a modular architecture that favors extensibility and maintainability. To incorporate a new attack, it is sufficient to create the corresponding container directory under `docker/attackers/` and declare its metadata in `attack.yaml`, including its identifier, category, image, parameters, target mapping, runtime limits, usage notes, and MITRE ATT&CK classification. At startup, the catalog is discovered dynamically from `docker/attackers/*/attack.yaml` and converted into the internal attack specifications used by the CLI. Therefore, a new scenario becomes available for listing, parameter inspection, and Docker-based execution without additional manual registration.

Packet capture is performed with `tcpdump`, using the configured interface and optional BPF (Berkeley Packet Filter). If `tcpdump` is unavailable, capture is skipped with a warning. Feature extraction from `.pcap` captures can use `TShark`, the Python `Scapy` library, and NTLFlowLyzer [Shafi et al. 2025]. These extractors are selectable through the CLI and invoked sequentially when combined. Their outputs are normalized and consolidated with `pandas`, combining packet-level and flow-level attributes into derived datasets.

4. Results

This section evaluates AttackZoo as an integrated testbed at catalog scale. The campaign assesses whether the architecture can execute the catalog under a uniform policy, persist traceable artifacts, produce heterogeneous traffic, and support repeated executions comparable across attacks, categories, and levels. It does not evaluate production realism or downstream IDS/ML utility, which require dedicated benign workloads and application-level studies.

The campaign executed all 60 attacks in the catalog across the seven categories described in the metadata. Each attack was exercised with four execution levels (L0–L3), where L0 represents the attack-free control condition and the remaining levels represent attack conditions, with five repetitions per level. Each run used a 60 s warmup, 120 s attack window, and 60 s cooldown, with packet capture in `pcap` format, service probes every 0.5 s with a 2 s timeout, host and target-container telemetry, feature extraction, dataset construction, and report generation. No benign client instances were launched in this campaign; L0 traffic therefore consisted of the target services, probes, telemetry, and host/container traffic induced by the experiment workflow. Packet capture used `tcpdump` on the host `Docker bridge` interface, with snap length 0 and unbuffered writes. BPF filters were generated per attack from the target or probe service ports when applicable, while local-link and network-scan scenarios used an empty BPF filter to capture all traffic visible to that interface. Feature extraction was enabled with all three supported extractors (NTLFlowLyzer, TShark, and Scapy) and the derived datasets reported below were built from their normalized outputs. The campaign was executed on a single host with an AMD Ryzen 5 5600X 6-core processor, 16 GB of DDR4 RAM and basic SSD storage.

This single host is the reference for hardware-dependent values; in high-load attacks such as DoS, CPU/RAM/I/O contention may still affect timing and packet loss.

[6]Docker Inc., Docker Hub, 2026.

Because the campaign ran unattended, guardrails monitored host load and data growth under the campaign output directory. Storage was capped at 1024 GB and the Unix 1-minute load threshold was 12; either limit would stop the active attack container and pause scheduling. Neither guardrail was reached, so no reported capture was truncated or excluded by these limits.

Table 2 summarizes the released campaign. The configuration produced 1200 packet captures and 220.4 GB of derived datasets from 672.1 GB of raw data, compressed into a 16.9 GB public package. These values show that the testbed executes the full catalog and preserves the evidence in structured form, while illustrating the cost of broad repeated campaigns.

Table 2. Summary of the experimental campaign

Experiment configuration

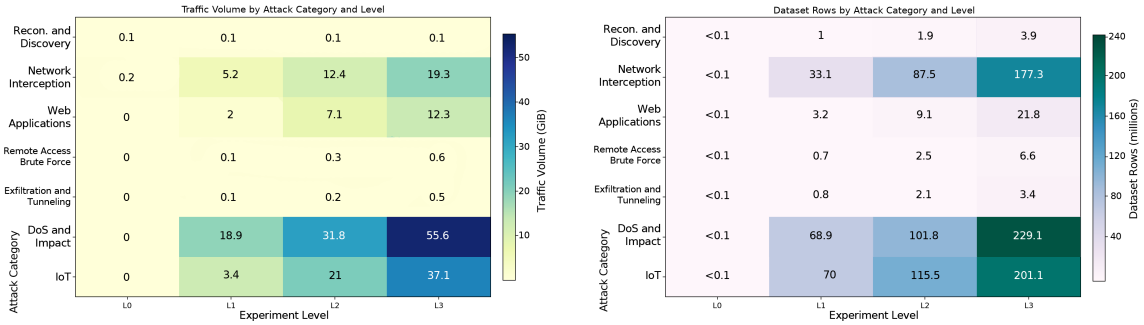
Metric	Value
Attacks executed	60
Categories covered	7
Execution levels	4
Runs per level	5
Campaign duration	approx. 82 hours

Generated data

Metric	Value
`.pcap` captures	1200
Total data volume	672.1 GB
Resulting datasets	220.4 GB
Compressed file*	16.9 GB

∗ Single compressed archive with all datasets at: *https://doi.org/10.6084/m9.figshare.33023765*

Figure 2 details artifact distribution across attack categories and execution levels. Raw capture volume (a) and dataset rows (b) increase in attack conditions, especially for DoS/Impact, IoT, and Network Interception attacks, which include floods, protocol abuse, scanning, and packet manipulation. Remote Access Brute Force and Exfiltration/Tunneling produce smaller volumes, showing that the campaign is not dominated by a single traffic pattern.



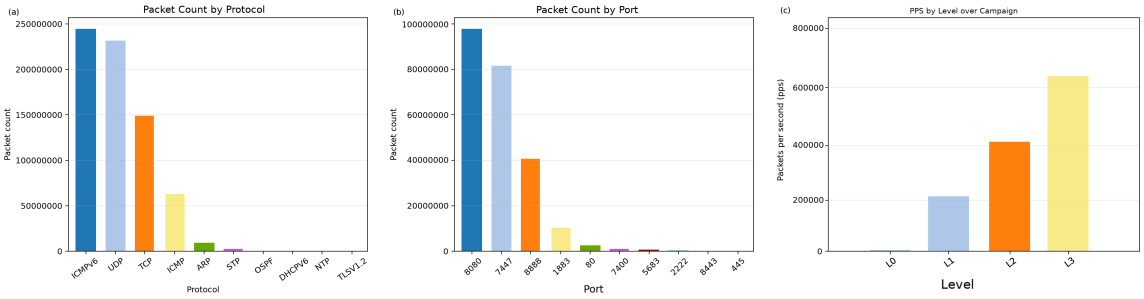
(a) Capture volume by category and level (b) Dataset rows by category and level

Figure 2. Artifact volume during the experimental campaign

The same data illustrates the value of a uniform execution policy: capture size, row count, and protocol distribution can be compared across categories without changing the collection procedure. Across attack/level pairs with five complete post-processing records, the median coefficient of variation was 0.59% for capture size (interquartile range: 0.12–1.56%) and 0.60% for dataset rows (interquartile range: 0.15–1.87%). These values quantify internal repeatability under a fixed setup, not similarity to production traffic.

Architecturally, each execution becomes an observable and reusable experimental unit. Persisted artifacts include raw captures, extracted features, datasets, reports, telemetry, and metadata associated with the attack identifier, level, run number, target service, capture configuration, and post-processing outputs.

Figure 3 characterizes the captured traffic across protocols, ports, and execution levels. Appendix A provides the detailed interpretation and points to the per-attack reports and artifacts available in the repository and dataset package.



(a) Packets by protocol (b) Packets per port (c) Packets per level (pps)

Figure 3. Traffic characterization during the experimental campaign

5. Availability and Demonstration Plan

The AttackZoo source code is available in the project repository[3], with installation instructions, technical documentation, dataset links, and a demonstration video.[7] The testbed runs on bare metal or in a virtual machine and is based on free and open-source software; resource requirements and demonstration steps are summarized in Appendix A.

6. Conclusion and Future Work

We presented AttackZoo, a reproducible and extensible testbed that unifies cybersecurity experimentation from scenario definition to dataset generation. The platform integrates 60 containerized attacks spanning 35 MITRE ATT&CK technique/sub-technique entries and 9 tactics with orchestration for execution, traffic capture, feature extraction, and data consolidation, producing artifacts linked to execution metadata for consistent cross-scenario comparison.

The evaluation demonstrates operational capability at catalog scale, but its results should be interpreted within clear validity boundaries: containerized environments, synthetic traffic profiles, attack parameterization, and concept drift can limit ecological validity and direct transfer to production networks. Future work will add benign-client campaigns, cross-host reproducibility, traffic-fidelity and IDS/ML utility analyses, pipeline-overhead metrics, feature refinements, catalog growth, external-contributor extension studies, and support community validation in public versioning.

A. Demonstration and Supplementary Details

Execution requires Ubuntu 24.04 LTS, at least 4 CPU cores (x86/AMD64 architecture), 8 GB of RAM, 30 GB of available storage, Internet access, and administrative privileges

[7]Demonstration video: `https://www.youtube.com/watch?v=Jl6E-RwcYDA`.

(e.g., `sudo`). Under these conditions, the full installation typically takes between 15–40 minutes, depending on computational resources and network conditions. The repository also describes a reduced evaluation version of the testbed that takes 2–5 minutes to install. The demonstration can be performed on a standard computer provided by the authors and covers installation, attack execution, traffic capture, and the post-processing pipeline, highlighting the integrated orchestration of the complete experimental cycle.

In Figure 3, the protocol distribution is dominated by ICMPv6, UDP, TCP, and ICMP, with lower-frequency ARP, STP, OSPF, DHCPv6, NTP, and TLS traffic. This mix reflects network-layer attacks, general TCP/IP services, and IoT-oriented protocols included in the catalog. The port distribution shows service heterogeneity, with traffic concentrated on Web and IoT-related ports such as 8080, 7447, 8888, 1883, 80, 5683, 2222, 8443, and 445. The packet-rate view separates baseline and attack conditions, preserving the level information needed to analyze traffic intensity; per-attack reports and individual artifact statistics are available in the project repository under `contrib/docs/campaign` folder and in the compressed dataset package.

Acknowledgments. This work was partially supported by RNP, CAPES (Código de Financiamento 001), CNPq (Grant No. 409743/2025-9), FAPERGS (Grant Nos. 24/2551-0001368-7, 24/2551-0000726-1). The authors also acknowledge the use of Generative AI as a supporting tool for reviewing and improving the code and manuscript.

References

Ahmad, R., Alsmadi, I., Alhamdani, W., and Tawalbeh, L. (2023). Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review*, 56(10).

Bernieri, G., Etchevés Miciolino, E., Pascucci, F., and Setola, R. (2017). Monitoring system reaction in cyber-physical testbed under cyber-attacks. *Computers & Electrical Engineering*, 59:86–98.

Bragança, H., Kreutz, D., Rocha, V., Assolin, J., and Feitosa, E. (2026). MH-1M: A 1.34 million-sample multi-feature Android malware dataset with rich metadata. *Scientific Data*, 13(1):153.

Brito, I., Pinheiro, T., Santos, M., Santos, R., Gomes, G., Sampaio, H., Freitas, A., and Sampaio, L. (2025). HackInSDN: Uma arquitetura flexível, incremental e portável para experimentação em cibersegurança. In *XLIII SBRC*, pages 882–895.

Croft, R., Babar, M. A., and Kholoosi, M. M. (2023). Data quality for software vulnerability datasets. In *ICSE*, pages 121–133.

Dhanapal, A. and Nithyanandam, P. (2021). An OpenStack based cloud testbed framework for evaluating HTTP flooding attacks. *Wireless Networks*, 27(8):5491–5501.

Fideles, D. R., Lautert, D. P., Kreutz, D., and Quincozes, S. E. (2025). VulnSyncAI: PLN e LLMs para construção e atualização contínua de datasets de vulnerabilidades. In *XLIII SBRC*.

Freitas, A. V., Araújo Júnior, S. R. d., and Silva, H. (2024). MQTT-VET: Exploring MQTT protocol vulnerabilities. In *LADC (Anais Estendidos)*, page 111–113.

Guo, Y., Bettaieb, S., and Casino, F. (2024). A comprehensive analysis on software vulnerability detection datasets: trends, challenges, and road ahead. *International Journal of Information Security*, 23(5):3311–3327.

Huang, H., Wlazlo, P., Sahu, A., Walker, A., Goulart, A. E., Davis, K. R., Swiler, L., Tarman, T. D., and Vugrin, E. (2024). Validating an emulation-based cybersecurity model with a physical testbed. *IEEE Transactions on Dependable and Secure Computing*.

Katuri, K., Park, S.-Y., Zuo, S., Miao, F., and Zhao, J. (2024). Demonstration of DoS attacks on Modbus protocol using an experimental CPS testbed. In *NAPS*, pages 1–6.

Kouril, D., Rebok, T., Jirsik, T., Cegan, J., Drasar, M., Vizvary, M., and Vykopal, J. (2014). Cloud-based testbed for simulation of cyber attacks. In *NOMS*, pages 1–6.

Kraust, S., Heller, P., and Mottok, J. (2025). Concept for designing an ICS testbed from a penetration testing perspective. In *EuroS&PW*, pages 561–568.

Polák, M., Sedlák, D., Fesl, J., and Tvrdík, P. (2024). Real data center network traffic dataset and analysis. In *CloudNet*, pages 1–5.

Prates Jr., N. G., Andrade, A. M., Mello, E. R. d., Wangham, M. S., and Nogueira, M. (2021). Um ambiente de experimentação em cibersegurança para Internet das Coisas. In *Anais do VI Workshop do testbed FIBRE*, pages 68–79.

Shafi, M., Lashkari, A. H., and Roudsari, A. H. (2025). NTLFlowLyzer: Towards generating an intrusion detection dataset and intruders behavior profiling through network and transport layers traffic analysis and pattern extraction. *Computers & Security*.

Siboni, S., Sachidananda, V., Meidan, Y., Bohadana, M., Mathov, Y., Bhairav, S., Shabtai, A., and Elovici, Y. (2019). Security testbed for Internet-of-Things devices. *IEEE Transactions on Reliability*, 68(1):23–44.

Soares, T., Mello, J., Barcellos, L., Sayyed, R., Siqueira, G., Casola, K., Costa, E., Gustavo, N., Feitosa, E., and Kreutz, D. (2021). Detecção de malwares Android: Levantamento empírico da disponibilidade e da atualização das fontes de dados. In *ERRC*.

Sáez-de Cámara, X., Flores, J. L., Arellano, C., Urbieta, A., and Zurutuza, U. (2024). Gotham testbed: A reproducible IoT testbed for security experiments and dataset generation. *IEEE Transactions on Dependable and Secure Computing*, 21(1):186–203.

Whang, S. E., Roh, Y., Song, H., and Lee, J.-G. (2023). Data collection and quality challenges in deep learning: a data-centric AI perspective. *The VLDB Journal*, 32(4).

Wlazlo, P., Sahu, A., Mao, Z., Huang, H., Goulart, A., Davis, K., and Zonouz, S. (2021). Man-in-the-middle attacks and defence in a power system cyber-physical testbed. *IET Cyber-Physical Systems: Theory & Applications*.

Yang, Z., Liu, X., Li, T., Wu, D., Wang, J., Zhao, Y., and Han, H. (2022). A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers & Security*, 116:102675.