NetSecBed: A Container-Native Testbed for Reproducible Cybersecurity Experimentation

Leonardo Bitzki[1,2], Diego Kreutz[2], Tiago Heinrich[3], Leandro Bertholdo[1], Silvio Quincozes[2], Douglas Fideles[2], Angelo Nogueira[2]

[1]Universidade Federal do Rio Grande do Sul (UFRGS)

[2]AI Horizon Labs
Programa de Pós-Graduação em Engenharia de Software (PPGES)
Universidade Federal do Pampa (UNIPAMPA)

[3]Max Planck Institute for Informatics (MPI-INF)

{bitzki,bertholdo}@ufrgs.br,theinric@mpi-inf.mpg.de

{kreutz,silvioquincozes}@unipampa.edu.br

{douglasfideles,angelonogueira}.aluno@unipampa.edu.br

Abstract. *Cybersecurity research depends on reproducible evidence, traffic traces, logs, and labeled datasets, but public datasets are often static and difficult to re-execute, especially in heterogeneous multi-protocol environments. This paper presents NetSecBed, a container-native, scenario-oriented testbed for controlled generation of network evidence and execution artifacts in IoT, IIoT, and pervasive settings. NetSecBed integrates 60 attack scenarios, 9 target services, and benign traffic generators as single-purpose containers, with plug-and-play extensibility and traceability through declarative specifications. Its pipeline automates parameterized execution, packet capture, log collection, service probing, feature extraction, and dataset consolidation. The result is a repeatable, auditable, and extensible framework that reduces operational bias and supports continuous dataset renewal.*

1. Introduction

Experimental cybersecurity research increasingly depends on reproducible evidence, such as traffic traces, logs, telemetry, and labeled datasets, to enable rigorous and comparable evaluation of detection, classification, and incident response mechanisms. Yet generating such evidence remains an open technical and methodological challenge. In modern environments, experimental fidelity is tightly coupled with ecosystem heterogeneity, spanning traditional services (e.g., HTTP, SSH, and SMB), IoT protocols (e.g., MQTT and CoAP), and emerging communication frameworks (e.g., Zenoh and XRCE-DDS), where even minor variations in software versions, timing, configuration, and background noise may significantly alter observed artifacts and compromise cross-experiment comparability. This creates a recurring trade-off between realism, operational cost, and reproducibility: higher realism typically increases deployment, instrumentation, and auditing complexity.

This challenge is further aggravated by the growing obsolescence of public cybersecurity datasets. Continuous changes in service behavior, protocol stacks, and adversarial strategies progressively reduce the temporal validity of static collections, limiting

both scientific evaluation and model training. Consequently, the central problem is no longer merely how to obtain data, but how to systematically generate experimental evidence in a repeatable, auditable, and extensible manner, preserving full traceability of origin, parameters, temporal context, and execution conditions.

The state of the art remains fragmented. Existing testbeds often optimize isolated dimensions, such as physical realism, protocol specificity, low-cost deployment, or fidelity in specialized domains (e.g., IoT, IIoT, SCADA, and CPS), while still exhibiting recurring limitations in *(i)* attack coverage, *(ii)* protocol heterogeneity, *(iii)* experimental scalability, and *(iv)* support for systematic re-execution and continuous dataset generation. In practice, many approaches still rely on static collections, domain-coupled scenarios, or partially manual pipelines, which hinder comparability, auditing, and incremental evolution of the experimental environment.

To address these limitations, we propose *NetSecBed*[1], a container-native, scenario-oriented, and natively extensible testbed for reproducible evidence generation in heterogeneous network environments. The current architecture integrates 60 attack scenarios, 9 target services, and parameterizable benign traffic models, all encapsulated as single-purpose containers and described through declarative metadata specifications. Unlike approaches centered on static datasets, *NetSecBed* provides a complete and auditable experimental pipeline, including scenario selection, parametrized execution, packet capture, log collection, service probing, feature extraction, and automatic dataset consolidation.

Our central methodological hypothesis is that explicitly separating declarative scenario specification from orchestrated execution reduces operational variability, improves traceability, and enables incremental scalability of the attack catalog without coupling extensions to the system core. This design allows new attacks, services, and benign profiles to be incorporated through YAML specifications and containerized artifacts, supporting continuous sample renewal and improving the external validity of experimental results. This makes *NetSecBed* particularly relevant for cybersecurity experimentation in ubiquitous and pervasive computing environments, where heterogeneous devices, protocols, and continuously evolving traffic patterns are intrinsic characteristics.

The main contributions of this work are: *(i)* an observable, container-native architecture for reproducible cybersecurity experimentation; *(ii)* a structured and extensible catalog of 60 attacks with support for multiple protocols and services; *(iii)* an automated pipeline for continuous artifact and dataset generation; and *(iv)* a methodological framework that reduces operational bias and improves comparability across heterogeneous experiments.

2. Related Work

Recent cybersecurity testbeds span a broad spectrum of objectives, ranging from IoT vulnerability assessment and traffic generation to cyber-physical attack impact analysis. However, as summarized in Table 1, the current state of the art remains fragmented across four critical methodological dimensions: scalability, protocol heterogeneity, attack coverage, and reproducibility.

Recent survey studies have reinforced the role of cybersecurity testbeds as scien-

[1]*NetSecBed* is available at `https://github.com/GT-IoTEdu/sbcup2026-ataques`

Table 1. State-of-the-art comparison of cybersecurity testbeds.

Testbed	Scientific Contributions	Scalability	Protocols	#Atk	Target Services	Main Limitations
Gotham	Reproducible IoT testbed; multi-protocol support; real dataset generation	∼140 nodes / machine	MQTT, CoAP, RTSP	18–25	MQTT broker, CoAP, RTSP	Restricted service scope; limited device diversity; no MITRE ATT&CK mapping
RESLab	Security experimentation and dataset collection	∼50 substations	DNP3	4	DNP3 masters / outstations	Very limited attack diversity; single-protocol scope; no MITRE mapping
IoT Security Testbed	Open-source automated vulnerability assessment platform for IoT devices	Low (1 device / run)	HTTP(S), UDP, FTP, SSH, Telnet	13	Camera, smart bulb, IoT web services	Few attacks; minimal scalability; no native IoT protocols or MITRE mapping
CPS Microgrid	Real-time co-simulation for CPS attack impact analysis	Very low	Modbus, IEC 61850, DNP3	3	IEDs, RTUs, PMUs	Low scalability; protocol limitations; proprietary hardware dependence
QR-GridEx	HIL-based large-scale coordinated attack simulation for smart grids	Medium	DNP3, IEC 61850, C37.118	10	SCADA/WAMS, substations	Limited attack catalog; low reproducibility; energy-specific scope
MENTORED	Reproducible x86 and IoT testbed; real dataset generation	Medium	HTTP(S), TCP, SSH	3	Web services, SSH	Limited attack coverage; no native IoT protocols or MITRE mapping
NetSecBed (ours)	Reproducible attack pipeline; structured attack catalog; feature extraction; observability and extensibility by design	High (container-native)	Web, MySQL, SSH, Telnet, SMB, MQTT, CoAP, Zenoh, XRCE-DDS	60	Web, DB, SSH, SMB, IoT/IIoT brokers	External threat focus; no firmware/CVE-specific or industrial protocol attacks

tific infrastructures for reproducible experimentation, dataset generation, and systematic benchmarking. In IoT and IIoT domains, recent reviews consistently highlight persistent challenges in scalability, protocol heterogeneity, reproducibility, and experimental realism [de Santana et al. 2024, Akinremi et al. 2025]. Similar limitations have also been reported in evaluation frameworks for anomaly detection in built environments, further emphasizing the need for large-scale, repeatable, and well-instrumented infrastructures for trustworthy security research as [Alosaimi et al. 2025], as well as [Jr. et al. 2021] and [Santana et al. 2025], which emphasizes large-scale DDoS/IoT experimentation over multi-cluster infrastructure.

Within this landscape, representative testbeds typically optimize specific methodological dimensions. In the IoT domain, the *Gotham Testbed* [Cámara et al. 2023] provides a reproducible architecture with multi-protocol support (MQTT, CoAP, and RTSP) and moderate scalability, reaching approximately 140 nodes per machine. However, it remains restricted to a narrow set of IoT services, limited device diversity, and lacks explicit alignment with standardized adversarial taxonomies such as MITRE ATT&CK. Similarly, the *IoT Security Testbed* [Siboni et al. 2019] advances automated vulnerability assessment for real devices, but offers only 13 attacks, low scalability, and no support for native IoT messaging protocols such as MQTT or CoAP, limiting its representativeness in modern IoT/IIoT scenarios.

A second line of work focuses on industrial control systems and SCADA experimentation. *RESLab* [Wlazlo et al. 2021] and the *Lightweight SCADA Testbed* [Khan et al. 2020] provide relevant infrastructures for traffic generation and dataset construction, yet their methodological breadth is constrained by single-protocol designs, very limited attack catalogs, and reduced cross-domain applicability. Likewise, cyber-physical platforms such as the *CPS Microgrid Testbed* [Gupta et al. 2021], *QR-*

GridEx [Ravikumar et al. 2020], and the *LoRaWAN MitM Testbed* [Pospisil et al. 2021] improve realism through co-simulation, HIL, and hardware coupling, but remain strongly domain-specific, infrastructure-dependent, and difficult to generalize.

Overall, prior testbeds trade breadth and reproducibility for realism, protocol specificity, or lightweight deployment. NetSecBed addresses the complementary gap: a container-native, catalog-driven framework with 60 attacks, heterogeneous traditional and IoT/IIoT protocols, pcap/log collection, feature extraction, and observability for repeatable dataset generation.

3. NetSecBed Architecture

NetSecBed adopts an architecture-first design for reproducible cybersecurity experiments. Attack execution is treated as a controlled pipeline that generates traceable pcaps, logs, telemetry, metrics, datasets, and reports. The design follows four principles: *(ii)* component isolation; *(ii)* standardized execution; *(iii)* declarative extensibility; and *(iv)* end-to-end traceability, to reduce operational bias and support comparable re-execution. *NetSecBed* assumes a container-native environment in which attackers, targets, and benign clients are controlled by the testbed. It focuses on network-centric attacks and service telemetry, not full production fidelity; physical host compromise and persistence beyond containers are outside the threat model.

Architecturally, NetSecBed comprises four subsystems: *(i)* a containerized artifact repository, comprising attackers, target services, and benign traffic generators; *(ii)* an experiment operation and configuration layer; *(iii)* an orchestration and instrumentation layer, responsible for transforming attack execution into a complete experimental cycle that produces verifiable and comparable artifacts; and *(iv)*. As shown in Figure 1, the operator defines parameters, the orchestrator instantiates containers, and execution artifacts are captured, processed, and consolidated into datasets, reports, and evidence traces. The decoupling of specification, orchestration, and artifact generation enables catalog expansion and dataset regeneration as services and tools evolve.

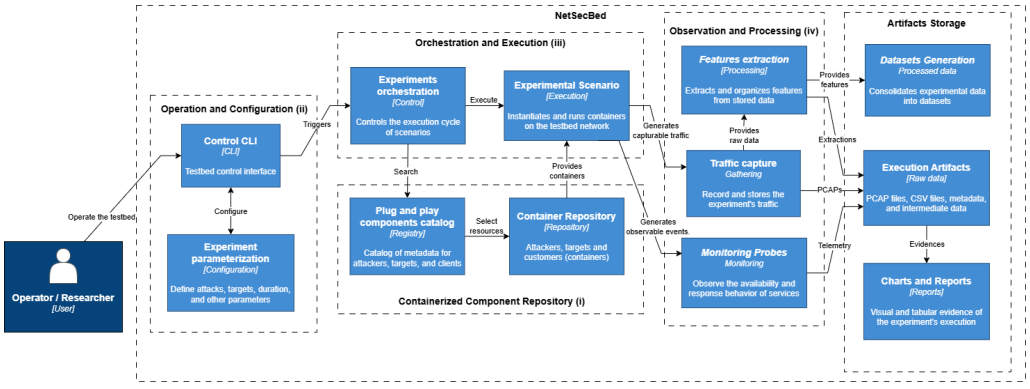

Figure 1. Overview of the *NetSecBed* architecture.

The key architectural innovation is the explicit decoupling between scenario specification, execution orchestration, and artifact generation. This separation transforms the testbed from a simple attack execution platform into a reproducible scientific framework for evidence generation, enabling scalable expansion of attack catalogs and systematic regeneration of up-to-date datasets as services, tools, and network conditions evolve.

3.1. Containerized Artifact Repository

A central methodological contribution of *NetSecBed* is its containerized artifact repository, which formalizes each attack scenario as an isolated, single-purpose experimental component. In this design, every attack is encapsulated as an independent container, enforcing strict execution isolation, reducing ambiguity, and improving experimental traceability. Rather than relying on loosely coupled scripts or operator-dependent procedures, each scenario is modeled as a controlled composition of an attacker, one or more target services, optional benign clients, and the execution parameters that define the experimental conditions. This architectural decision is fundamental to reproducibility: by enforcing one objective per container, each experiment becomes fully auditable, allowing precise identification of what was executed, for how long, against which target, and under which parameters, while minimizing coupling and operational variability across repeated runs.

Beyond attacker containers, *NetSecBed* includes a heterogeneous repository of target service containers, including web servers, MySQL, SSH, Telnet, SMB, MQTT, CoAP, Zenoh, and XRCE-DDS services, as well as parameterizable benign traffic generators capable of producing deterministic background traffic (e.g., service type, access frequency, inter-arrival interval, and total duration). This heterogeneity is a key scientific differentiator, enabling systematic experimentation across both traditional and IoT/IIoT communication ecosystems.

Each scenario is formally described through structured metadata, including identifier, description, image reference, supported parameters, operational and security notes, and explicit mappings to MITRE ATT&CK tactics and techniques. Importantly, these metadata act as an *experimental contract*, i.e., the minimum declarative specification required to reproduce the scenario and correctly interpret the generated artifacts, which is one of the main mechanisms through which *NetSecBed* ensures traceability and long-term reproducibility. Furthermore, artifacts are semantically organized according to MITRE ATT&CK [Roy et al. 2023], linking each implementation to both the adversarial objective (*why*) and its operational mechanism (*how*), which improves consistency, extensibility, and comparability across scenarios.

Adding new attacks: Another important contribution is the declarative extensibility model. New attacks are incorporated through three lightweight steps: *(i)* creating a dedicated directory within `docker/`, where a `Dockerfile` defines the base image, dependencies, scripts, and execution environment; *(ii)* encapsulating the attack logic within an `entrypoint.sh` hook; and *(iii)* registering the artifact in `attack.yaml`, which connects it to the orchestration layer. This design allows the attack catalog to scale incrementally without modifying the system core, directly addressing a key limitation of prior testbeds: poor extensibility and high maintenance overhead.

3.2. Experiment Operation and Orchestration Engine

A key scientific contribution of *NetSecBed* is its orchestration engine, which transforms an abstract experiment specification into an executable, observable, and fully repeatable workflow. Rather than executing attacks as isolated scripts, this layer formalizes experimentation as a controlled lifecycle, reducing operator-dependent variability and enabling systematic comparison across repeated runs. In practice, the engine receives parameters such as target service, attack set, intensity levels, repetitions, capture interface, traffic fil-

ters, and phase duration, and converts them into a structured execution matrix organized by service, attack, level, and repetition, supporting controlled sensitivity and comparative analyses.

Operationally, orchestration follows an explicit *warmup–attack–cooldown* lifecycle, separating service stabilization, adversarial interference, and recovery phases. In parallel, the engine continuously performs availability and latency probes over instrumented services, labeling each sample according to its temporal phase. This transforms each execution into a contextualized time series suitable for impact analysis, temporal robustness studies, and reproducible metric generation, as illustrated in Figure 2. To preserve extensibility, attack execution is delegated to parameterized external *hooks*, decoupling offensive logic from the system core and allowing new triggering mechanisms without redesigning the pipeline. Each run also produces a dedicated directory containing raw traffic captures, probe records, and metadata describing contextual parameters and execution timestamps, ensuring traceability and exact replay.

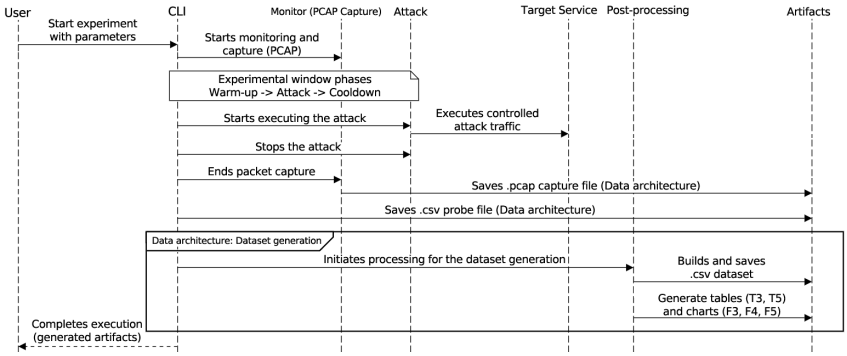

Figure 2. Execution pipeline of *NetSecBed* for artifact generation.

3.3. Data Processing and Artifact Generation

The data processing layer converts raw execution signals into analytically usable evidence. Beyond packet captures, it incorporates service probing time series and execution metadata, preserving the full context of each run. Feature extraction is performed through multiple independent mechanisms, including TShark, Scapy, and NTLFlowLyzer [Shafi et al. 2025], while maintaining separate extraction tracks prior to consolidation. This is a deliberate methodological choice: different extractors operate at distinct abstraction levels and protocol coverage, introducing different biases and sensitivities.

By preserving parallel extraction pipelines, *NetSecBed* enables consistency verification, information-loss analysis, and robustness assessment of feature representations. The extracted outputs are then consolidated into standardized tabular datasets explicitly linked to execution metadata, enabling re-analysis, auditing, reproducibility, downstream detection tasks, and parameter sensitivity studies. In addition, the framework generates synthetic reports and visual analytics, including summary tables, latency metrics, failure-aware observations, temporal evolution, and cumulative distributions. This layer therefore transforms raw observations into traceable and continuously renewable evidence artifacts, directly supporting the generation of up-to-date cybersecurity datasets.

As a concrete downstream application, the consolidated CSV datasets produced by NetSecBed can be directly ingested by machine learning pipelines for intrusion detec-

tion model training and benchmarking. In the evaluated scenario, the generated dataset comprises labeled flows with 80+ features extracted by NTLFlowLyzer, stratified by experimental phase (warmup, attack, cooldown), enabling supervised classification experiments with ground-truth labels derived directly from the execution metadata.

3.4. Development and Experimental Environment

NetSecBed is designed for lightweight deployment in both bare-metal and virtualized environments, prioritizing low replication cost, automated installation, and open-source components to improve reproducibility and facilitate adoption in research and teaching settings. The framework was developed and validated on commodity hardware in two representative environments: *(i)* Ubuntu 24.04 under WSL on a Windows 10 host with an Intel Core i7-7700HQ, 16 GB RAM, GTX 1060, and SSD storage, and *(ii)* Kubuntu 24 LTS in bare-metal mode on an AMD Ryzen 5 5600X, 32 GB RAM, RTX 3070 Ti, and NVMe storage, demonstrating that the proposed architecture can be reliably reproduced without specialized infrastructure.

4. Methodology

Methodologically, *NetSecBed* is formalized as a controlled evidence generation framework rather than a simple attack execution platform. Under this perspective, the focus extends beyond attack triggering to the experimental design itself, explicitly defining *(i)* environmental control variables, *(ii)* independent variables such as attack and benign traffic parameters, *(iii)* dependent variables expressed as observable metrics, *(iv)* the repetition plan, and *(v)* acceptance criteria used to verify experimental consistency and artifact validity.

Each experiment is modeled as a tuple $\mathbf{E} = (\mathbf{C}, \mathbf{P}, \mathbf{B}, \mathbf{I}, \mathbf{R})$, where $\mathbf{C}$ represents the scenario (e.g., attack, target, and minimal topology), $\mathbf{P}$ defines the parameter space (e.g., rate, duration, port, and target), $\mathbf{B}$ describes the benign baseline (e.g., traffic profile, number of clients, and intervals), $\mathbf{I}$ specifies the instrumentation plan (e.g., *pcap*, logs, and artifact generation), and $\mathbf{R}$ determines the repetition plan (e.g., number of runs and warmup/cooldown windows). This formalization improves traceability between configuration and results while enabling consistent comparison across scenarios and studies.

Operationally, experiments are executed in multiple rounds and explicitly segmented into *warmup*, *attack*, and *cooldown* phases. The instrumentation plan includes availability probes, latency, error rate, relative time, packet capture, optional feature extraction, and automatic generation of analytical artifacts such as time series, latency CDFs, and failure-rate curves. This design enables methodological scalability, as distinct scenarios can be compared through homogeneous and reproducible metrics. Among the evaluated metrics, latency percentiles p50, p95, and p99 are explicitly considered. While p50 captures the typical system response, p95 and p99 characterize tail behavior, exposing degradation in the worst 5% and 1% of requests, respectively. This choice is particularly suitable for cybersecurity experiments, as availability attacks tend to disproportionately affect distribution tails before significantly shifting the median [Yang et al. 2025].

In practice, acceptance criteria R include conditions such as: (i) probe success rate above 95% during the warmup phase, confirming service stability before adversarial injection; (ii) absence of packet loss at the capture interface, ensuring trace completeness

and (iii) full service recovery (success rate $\geq 95\%$) during the cooldown phase, confirming temporal localization of the impact. Runs that violate these conditions are flagged and excluded from aggregation, preserving experimental integrity.

A key methodological principle is the explicit separation between execution and interpretation. The pipeline preserves both raw artifacts (*pcaps* and logs) and derived artifacts (extracted features, consolidated datasets, tables, and figures), maintaining a complete evidence trail that supports reproducibility and auditing. Scientific analyses, including detection, classification, and metric computation, remain fully re-executable from the raw evidence, reducing dependence on *ad hoc* transformations and minimizing the risk of conclusions based on irreproducible intermediate steps.

Finally, threats to validity are explicitly considered, including uncontrolled variables related to hardware noise, CPU/IO contention, host-level network interference, and sufficiency of the repetition plan, all of which may affect measurement stability and inference robustness.

5. Evaluation

To demonstrate the evidence-generation capabilities of *NetSecBed*, we selected the `DoS SYN Flood` attack from the catalog of 60 implemented scenarios. This attack was chosen because it produces an immediate and measurable impact on target service availability, making it well suited to illustrate the relationship between attack intensity, duration, and disturbance onset. This design aligns with prior evaluation strategies [Bernieri et al. 2017, Siboni et al. 2019, Dharini et al. 2026], in which attack parameters and execution windows are systematically varied to analyze changes in system behavior, including network utilization, attack duration, and timing effects.

For the experimental analysis, we varied the *rate* parameter controlling attack intensity together with the *warmup*, *attack*, and *cooldown* windows that define the temporal structure of the experiment. We focus on levels L0 and L3, representing the lower and upper extremes of the adopted intensity scale, corresponding to up to 100 pps at L0 and an unrestricted rate at L3. This choice was designed to explicitly capture the contrast between a mild perturbation and a severe availability degradation scenario, while intermediate levels follow the same parametric logic and are omitted for space reasons.

Figure 3. Experimental topology for the DoS SYN Flood case study.

Figure 3 illustrates the experimental topology used in the DoS SYN Flood evaluation. The setup comprises three containerized components deployed on a single host: an attacker container executing the SYN Flood logic, an HTTP target service container acting as the victim, and a monitoring component responsible for packet capture and availability probing. A benign client generates background traffic during all phases, enabling phase-aware comparison between normal and adversarial conditions.

Table 2 summarizes the results. Under L0, the HTTP service remains stable across all phases, sustaining 100% success rate and low latency percentiles, which confirms the

baseline robustness of the target service under low-intensity conditions. In contrast, L3 produces a substantial degradation during the attack phase, reducing service success to 55.6% and increasing latency from single-digit milliseconds to p50 = 1275.74 ms and p95/p99 = 2002.2 ms. Importantly, both warmup and cooldown phases preserve near-baseline behavior, which evidences that the observed degradation is temporally localized and directly attributable to the injected attack. This result demonstrates the ability of *NetSecBed* to produce controlled, phase-aware, and quantitatively reproducible evidence of service degradation under adversarial conditions.

Table 2. L0 vs. L3: HTTP success rate and censored latency.

Level	Phase	Samples	Success	Failure	p50(ms)	p95(ms)	p99(ms)
L0	warmup	48	100%	0%	2.96	3.1	7.6
L0	attack	49	100%	0%	3.22	3.8	6
L0	cooldown	49	100%	0%	3.18	3.62	6.88
L3	warmup	49	100%	0%	3	3.8	12.25
L3	attack	49	55.6%	44.4%	1275.74	2002.2	2002.2
L3	cooldown	46	100%	0%	3.25	4.1	14.58

Table 2 further details the aggregated behavior of the HTTP service under L3 across the *warmup*, *attack*, and *cooldown* phases, considering success rate, failure rate, and censored latency. During *warmup*, the service remains stable, sustaining 100% success and low latency (p50 = 3 ms and p95 = 3.8 ms), which defines the nominal operating baseline. During attack phase, degradation becomes severe: the success rate drops to 56%, while median latency rises to 1275.74 ms and both p95 and p99 reach approximately 2002 ms, consistent with the censorship threshold adopted in the experiment. This indicates that the attack compromises not only request availability but also the responsiveness of successfully completed requests, which operate close to the timeout limit.

After the attack interruption, the *cooldown* phase shows full recovery, restoring 100% success and near-baseline latency (p95 = 4.1 ms). This behavior suggests that the observed degradation is strongly localized within the attack window, with no relevant persistence effects after interruption. Taken together, these results demonstrate that, at the most severe level, the `DoS SYN Flood` simultaneously compromises both service availability and responsiveness, whereas L0 preserves near-nominal behavior. The same trend is visually confirmed in Figure 4.

In the illustrative results, we adopt a 5 s *warmup*, 10 s continuous attack window, and 5 s *cooldown*. From the same experimental run, the framework also derives latency-versus-success metrics before, during, and after the attack, clearly capturing the degradation and temporary disruption of the HTTP service during the attack window. Requests with latency above 2000 ms are treated as unsuccessful.

Figure 5 further details the temporal evolution of three host-level resource metrics collected from the target server container during the 20 s experiment. Figure 5(a) shows CPU utilization, which exhibits moderate oscillation during *warmup* and a pronounced increase immediately after attack onset at 5 s, reaching an early peak and a second elevation near the end of the attack window. Figure 5(b) presents system load averages (*load1*, *load5*, and *load15*); as expected for a short experiment, *load1* is the most sensitive and increases sharply during the attack, remaining elevated into the initial *cooldown*, whereas

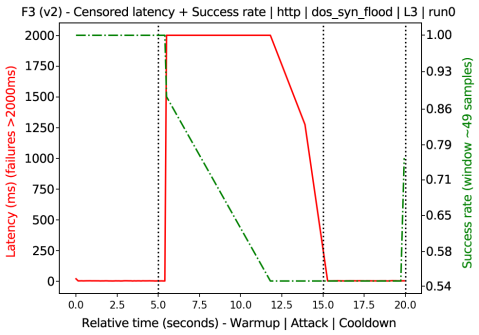

Figure 4. Censored latency versus success rate under HTTP SYN Flood DoS.

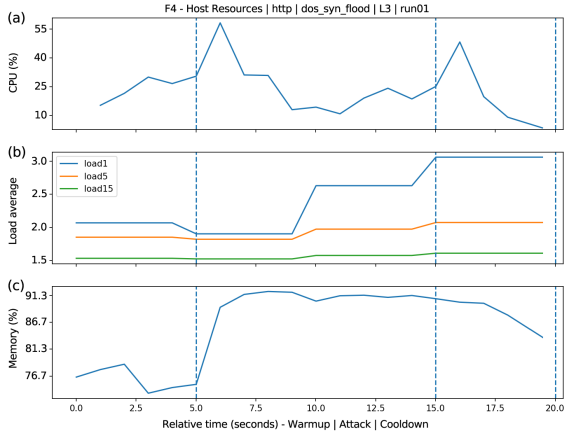

Figure 5. Target server resource usage.

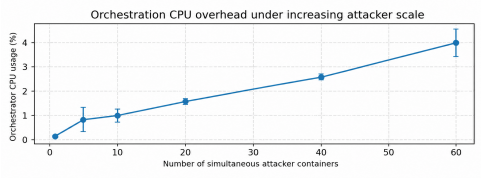

Figure 6. CPU overhead.

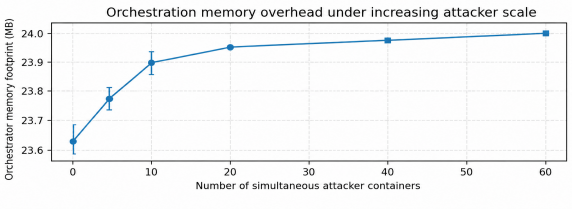

Figure 7. Memory overhead.

load5 and *load15* vary more smoothly due to their longer temporal windows. Figure 5(c) shows memory usage, which increases from 75–77% before the attack to 89–91% during the attack phase, with only partial recovery by the end of the observation window.

Taken together, these results show that the attack affects not only service-level availability and latency, but also imposes measurable internal pressure on the target server. CPU reacts immediately, system load remains elevated for longer, and memory usage does not instantly return to its pre-attack baseline. These host-level metrics therefore complement the evidence obtained from application probes and traffic captures, providing a clearer interpretation of both degradation mechanisms and recovery dynamics.

5.1. Orchestration Scalability

To evaluate the scalability of the NetSecBed control plane, we executed a microbenchmark in which the number of simultaneously instantiated attackers containers was increased from 1 to 60. This experiment does not measure attack throughput, instead, it quantifies the overhead introduced by orchestration, probing, monitoring, and lifecycle management. For each scale level, we executed five repetitions and collected one sample per second of orchestrator CPU usage, orchestrator memory footprint, host CPU usage, host memory usage, container readiness, and probe success rate Table 3.

The results provide initial evidence that the NetSecBed control plane scales predictably up to 60 simultaneous attackers containers in the evaluated host environment. CPU usage increases with the number of containers (Figure 6), while memory consumption remains bounded and does not exhibit abrupt growth (Figure 7). These results support the claim of container-level orchestration scalability, while not implying traffic-scale volumetric attack capacity.

Table 3. Orchestration overhead under increasing numbers of attackers.

Attackers	CPU mean (%)	CPU p95 (%)	RAM memory mean (MB)	Startup (s)
1	0,16	1,00	23,63	1,95
10	0,99	1,95	23,90	6,70
20	1,56	1,94	23,96	11,48
60	4,07	4,98	24,01	76,99

Table 4. Re-execution stability for the HTTP SYN Flood scenario over 20 runs.

Metric	Mean	Std. Dev.	Min.	Max.
Dataset rows	1671.0	2.11	1664.0	1674.0
Execution time (s)	23.861	0.077	23.729	24.018
Attack p99 latency, successful probes only (ms)	4.060	0.926	3.445	7.518
PCAP size (MB)	1.56	0.025	1.55	1.59
Attack stage success rate (%)	57.0	1.0	56.0	58.0

Table 4 summarizes the stability of the HTTP SYN Flood experiment across 20 repeated executions, providing empirical evidence that NetSecBed can re-execute the same scenario under controlled conditions while preserving consistent measurement behavior and artifact generation. Therefore, the results support the reproducibility claim not only at the architectural level, but also through observable stability across independent experimental runs.

5.2. Fidelity limitations

NetSecBed prioritizes controllability, traceability, and low-cost re-execution over physical-network fidelity. Container-native scenarios do not reproduce wireless interference, hardware queues, switch buffering, WAN jitter, ISP bottlenecks, or Internet-scale amplification. Therefore, volumetric attacks should be read as controlled service-degradation scenarios; physical, emulated, or federated experiments remain necessary when link-layer or large-scale traffic realism is the objective.

6. Conclusion

This work presented *NetSecBed*, a container-native testbed and methodological framework for reproducible cybersecurity experimentation that standardizes the full experimental lifecycle, enabling controlled scenario orchestration, traceable evidence generation (e.g., *pcap*, logs, and extracted features), and systematic comparison across runs and scenarios, thereby contributing evidence toward scalable, reproducible, and continuous dataset generation for network security research. Experimental results further demonstrate its effectiveness in capturing and quantifying availability attacks on instrumented services: in the evaluated HTTP SYN Flood scenario, the service maintained 100% success rate with p95 latency below 4 ms during both *warmup* and *cooldown*, whereas the maximum intensity level (L3) reduced success to 55.6% and increased p50 latency to over 1200 ms during the attack window, followed by full recovery after interruption. These results confirm *NetSecBed* as a robust, reproducible, and extensible evidence-generation framework, particularly well suited for controlled experimentation in IoT, IIoT, cyber-physical, and ubiquitous computing environments.

Acknowledgments

This work was partially supported by RNP[2] (GT IoTEdu[3]), CNPq (Grant 409743/2025-9), and FAPERGS (Grants 24/2551-0001368-7 and 24/2551-0000726-1).

References

Akinremi, T. P., Appiah, J. K., Asadi, A. R., Ibitoye, O., Jayathilake, H. M., and Said, H. (2025). Systematic literature review of cybersecurity testbeds for industrial internet of things. In *2025 1st ICSIoT, (SATC)*, pages 1–7. IEEE.

Alosaimi, M., Rana, O., and Perera, C. (2025). Testbeds and evaluation frameworks for anomaly detection within built environments: A systematic review. *ACM Computing Surveys*, 57(9):1–36.

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.

Cámara, X., Flores, J. L., Arellano, C., Urbieta, A., and Zurutuza, U. (2023). Gotham testbed: a reproducible iot testbed for security experiments and dataset generation. *IEEE Transactions on Dependable and Secure Computing*, 21(1):186–203.

de Santana, K. G. Q., Schwarz, M., and Wangham, M. S. (2024). Cybersecurity testbeds for iot: A systematic literature review and taxonomy. *Journal of Internet Services and Applications*, 15(1):450–473.

Dharini, N., Janani, V., and Katiravan, J. (2026). Efficient detection of intrusions in ton-iot dataset using hybrid feature selection approach. *Scientific Reports*.

Gupta, K., Sahoo, S., Panigrahi, B. K., Blaabjerg, F., and Popovski, P. (2021). On the assessment of cyber risks and attack surfaces in a real-time co-simulation cybersecurity testbed for inverter-based microgrids. *Energies*, 14(16).

Jr., N. P., Andrade, A., Mello, E., Wangham, M., 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, Porto Alegre, RS, Brasil. SBC.

Khan, M., Rehman, O., Rahman, I. M., and Ali, S. (2020). Lightweight testbed for cybersecurity experiments in scada-based systems. In *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, pages 1–5. IEEE.

Pospisil, O., Fujdiak, R., Mikhaylov, K., Ruotsalainen, H., and Misurec, J. (2021). Testbed for lorawan security: Design and validation through man-in-the-middle attacks study. *Applied Sciences*, 11(16).

Ravikumar, G., Hyder, B., and Govindarasu, M. (2020). Next-generation cps testbed-based grid exercise - synthetic grid, attack, and defense modeling. pages 92–98.

Roy, S., Panaousis, E., Noakes, C., Laszka, A., Panda, S., and Loukas, G. (2023). Sok: The mitre att&ck framework in research and practice. *arXiv preprint arXiv:2304.07411*.

[2]https://www.rnp.br/
[3]https://gt-iotedu.github.io/

Santana, K., Meyer, B., Gemmer, D., Schwarz, M., and Wangham, M. (2025). Estendendo o mentored testbed para a execução de experimentos de cibersegurança multi-cluster e iot. In *Anais do XLIII SBRC*, pages 868–881, Porto Alegre, RS, Brasil. SBC.

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*, 148:104160.

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.

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*, 6(3):164–177.

Yang, C. C. S., Keat, L. C., Phing, N. Y., and Chi, P. G. (2025). Simulation analysis of syn flood and http flood attacks on cloud infrastructure integrity. *International Journal of Research and Innovation in Social Science (IJRISS)*, 9(10).