IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/47SE/2026/4714526/
Vol. 2026 (14), Article ID 4714526

Detection latency in container-based Cyber Ranges*

Szymon CHMIELEWSKI

Military University of Technology, Warsaw, Poland,
ORCID: 0009-0001-3030-2515

Correspondence should be addressed to: Szymon CHMIELEWSKI, szymon.chmielewski01@wat.edu.pl

*Presented at the 47th IBIMA International Conference, 29-30 June 2026, Madrid, Spain

Abstract

Container-based cyber ranges are increasingly used as lightweight alternatives to virtual machine–based environments for cybersecurity experimentation and training. However, empirical evaluation of detection-related runtime properties in such environments remains limited. This paper presents an experimental study of detection latency and scenario-level observability in a minimal container-based cyber range based exclusively on application-level telemetry. A multi-step application-layer attack scenario was executed, including reconnaissance, SQL injection-like and XSS-like events, and administrative access attempts. Detection latency and Telemetry Coverage Index (TCI) were used as complementary metrics. The experiments covered a baseline condition, partial observability conditions with selectively disabled telemetry, and load conditions with benign background traffic from 0 to 75 requests per second. The results showed full observability in the baseline and all load conditions (TCI = 1.00), while partial observability reduced TCI to 0.80 and 0.60. Detection latency remained within a narrow range across the evaluated conditions, and increasing benign workload did not substantially affect the upper tail of the latency distribution. The study provides a practical empirical baseline for future comparisons involving more complex monitoring pipelines and alternative cyber range architectures.

Keywords: container-based, cyber ranges, detection latency, application-level telemetry.

Introduction

Cyber ranges are widely used as controlled environments for cybersecurity experimentation, training, and evaluation of attack and defense scenarios. Traditionally, such environments have been implemented using virtual machines, which provide strong isolation and flexible configuration at the cost of higher provisioning overhead and operational complexity. With the growing adoption of cloud-native technologies, container-based cyber ranges have become an attractive alternative due to their lightweight deployment model, reproducibility, and reduced resource requirements.

At the same time, the monitoring of cloud-native systems increasingly relies on application-level telemetry, such as structured logs and event traces, rather than on packet-level inspection alone. This shift raises an important question: to what extent can application-layer telemetry support reliable and timely detection of attack activity in minimal container-based cyber range environments? Although containerized cyber ranges have been discussed in the context of orchestration, automation, and training support, empirical evaluation of their detection-related properties remains limited. In particular, application-layer detection latency is rarely treated as an explicit, measurable system property.

Cite this Article as: Szymon CHMIELEWSKI Vol. 2026 (14) "Detection latency in container-based Cyber Ranges" Communications of International Proceedings, Vol. 2026 (14), Article ID 4714526, https://doi.org/10.5171/2026.4714526

In response to this gap, a minimal container-based cyber range was constructed and used to study detection latency and scenario observability under controlled conditions. Rather than introducing a new detection algorithm, the focus was placed on a simplified monitoring pipeline operating solely on application-level telemetry. This made it possible to examine how quickly selected attack-related events were detected and whether full observability of a multi-step attack scenario could be maintained under telemetry degradation and increasing benign workload.

Related work

Cyber range research has traditionally focused on the design of controlled environments for cybersecurity training and experimentation. Particular attention has been given to architecture, scenario construction, supported functions, and deployment models, as discussed by Yamin, Katt and Gkioulos (2020). Virtualized environments remain an important reference point in this area, and broad reviews, such as that by Ukwandu et al. (2020), confirm their continuing relevance in research and training applications.

Container-based cyber ranges have been investigated as lightweight alternatives, mainly in the context of reproducibility, reduced resource consumption, and deployment efficiency, as shown by Nakata and Otsuka (2020). In this line of work, the main emphasis has usually been placed on scalable and easy-to-deploy platforms rather than on measuring detection-oriented runtime properties. Similar tendencies can also be observed in more recent container-based training platforms, where automation and usability are emphasized over empirical evaluation of detection behavior, as noted by Chouliaras et al. (2023).

A related area of research concerns security and observability in cloud-native systems. In such environments, application-level logs and metrics have become important security-relevant signals, especially where packet-level inspection is constrained by service abstraction, encryption, or overlay networking, as explained by Theodoropoulos et al. (2023). This creates a natural connection between cloud-native observability research and container-based cyber range design, particularly when attack activity is expected to be detected through application-generated events.

Recent reviews also indicate that cyber range literature still lacks sufficient empirical benchmarking and systematic validation of measurable runtime properties. For example, Lillemets et al. (2025) point out that the literature still contains important gaps in systematic validation. Although surveys of cyber ranges and test-beds, such as the one by Ukwandu et al. (2020), point to the need for deeper characterization and comparison of existing platforms, controlled studies explicitly addressing application-layer detection latency and scenario-level observability remain limited.

Methodology

Experimental Scope

The methodology was limited to a minimal container-based cyber range operating on application-level telemetry. Two complementary properties were considered throughout the study: detection latency and scenario-level observability. The first was analyzed with respect to repeated attack-related events, while the second was evaluated in relation to the visibility of a complete multi-step attack scenario.

Cyber Range Architecture

The experimental cyber range was implemented as a containerized environment on a Windows 11 host using WSL2 (Ubuntu 22.04 LTS) and Docker Desktop. The architecture comprised three main containers: an attacker container, a victim application container, and a SOC component container. All components were connected through an isolated virtual network and executed within a single-host environment. The single-host deployment was intentionally selected to reduce environmental variability and to isolate the timing behavior of the simplified application-level detection loop. This setup made it possible to measure detection latency without additional effects introduced by inter-host communication, distributed log transport, or orchestration overhead. At the same time, this design does not fully reflect production-grade or distributed cyber range environments, where attacker, victim, and monitoring components may be deployed across multiple machines. In such settings, additional latency sources related to network transmission, log shipping, and cross-host time synchronization may influence the observed detection timing and should be considered in future studies.

The attacker container was responsible for generating HTTP requests directed at the victim application. The victim application container hosted a deliberately vulnerable web service instrumented with application-level logging. The SOC component container continuously monitored the generated logs and applied predefined detection rules in order to generate alerts. The SOC component implemented a simplified rule-based detection method operating directly on application-generated logs. Detection was based on matching predefined log patterns associated with the attack steps included in the scenario, such as SQL injection-like payload markers, XSS-like input patterns, and administrative access attempts. Each rule generated an alert when the corresponding log entry satisfied the expected condition. No probabilistic scoring, anomaly detection, or multi-event correlation was applied; the monitoring logic was limited to direct rule matching on single log events. This design was intentionally chosen to keep the detection pipeline transparent and reproducible and to focus the evaluation on latency and observability rather than on detector sophistication.

The architecture was intentionally limited to application-level telemetry. No packet capture, traffic mirroring, or external telemetry pipeline was used. This design made it possible to examine the behavior of a simplified monitoring loop and to evaluate detection latency without introducing additional transport, aggregation or correlation stages. The resulting architecture is presented in Fig. 1.

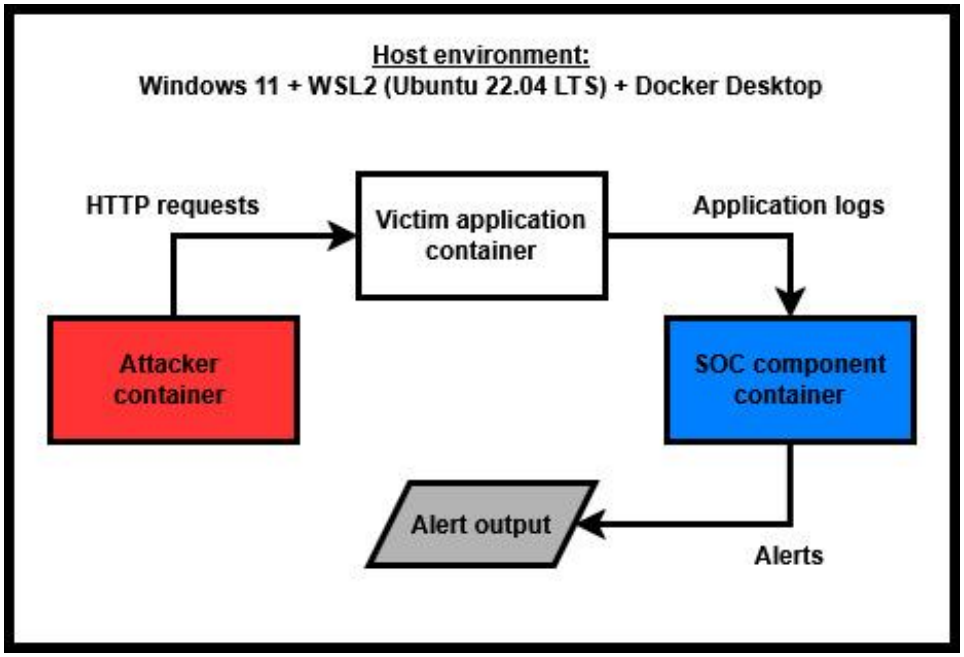

Fig. 1. Architecture of the container-based cyber range.

Data Collection

Data collection was based exclusively on artifacts produced within the application layer. Two main data sources were used throughout the experiments: logs generated by the victim application and alerts generated by the SOC component. The victim-side logs captured attack-related application events, while the SOC-side output recorded detection events produced by the monitoring loop.

For latency estimation, timestamps associated with attack-related events in the victim application logs were matched with timestamps of the corresponding alerts generated by the SOC component. Because all containers were executed on the same physical host, timestamp comparison was not affected by inter-host clock drift. The latency estimates therefore relied on a shared local time base provided by the host environment. This simplification would not necessarily hold in a distributed multi-host deployment, where reliable time synchronization mechanisms would be required to preserve the validity of event-to-alert latency measurements. This made it possible to reconstruct the temporal relation between event occurrence and alert generation without relying on packet-level traces or external telemetry systems. The collected data were therefore sufficient to evaluate the timing behavior of the simplified detection pipeline under controlled conditions.

To associate latency values with specific detections, each SQL injection-like event recorded in the victim-side log was matched to the corresponding SOC alert generated from the same request instance. The matching procedure was based on event order and rule-specific event type consistency across the preserved run artifacts.

Each experimental run included one execution of the complete attack scenario, followed by 50 additional repetitions of the SQL injection-like step. Since the complete scenario also contained a single SQL injection-like event, each run produced 51 latency observations for the measured attack type. This convention was preserved consistently across all experimental conditions and runs.

The collected artifacts were stored in separate files for subsequent analysis. In particular, the victim-side access logs, the alert output, and the derived latency records were preserved for each run, making it possible to aggregate results across repeated executions and compare baseline, partial observability, and load conditions.

Attack scenario

The experimental attack scenario was defined as a multi-step application-layer sequence combining reconnaissance, exploitation-oriented activity, and administrative access attempts. The scenario was designed to reflect a simplified but internally coherent chain of attacker actions observable through application-level telemetry.

The sequence began with a benign search request representing reconnaissance activity. This step was followed by a SQL injection-like request and an XSS-like input, both intended to emulate application-layer exploitation attempts. In the final part of the scenario, access to a protected administrative endpoint was requested first without valid authorization and then with a valid API key. This made it possible to include both unsuccessful and successful administrative access events within a single scenario. The implemented attack chain is shown in Fig. 2.

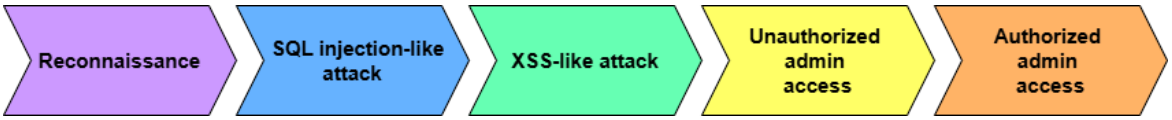

Fig. 2. Implemented application-layer attack chain.

Each step of the scenario was expected to generate a distinct telemetry artifact. As a result, the complete scenario could be used to assess whether all relevant attacker actions remained visible under different experimental conditions. The mapping between attack steps and telemetry sources is presented in Table 1.

Table 1: Mapping of attack steps to application-level telemetry.

Step	Attack action	Scenario role	Telemetry Source
1	Reconnaissance (search request)	Reconnaissance	Application access log
2	SQL injection-like payload	Exploitation attempt	Application access log
3	XSS-like payload	Exploitation attempt	Application access log
4	Unauthorized admin access	Administrative access attempt	Security event log
5	Authorized admin access	Administrative access attempt	Security event log

Experimental conditions

The experiments were organized into three groups of conditions: a baseline condition, partial observability conditions, and load conditions. The baseline condition represented complete telemetry and no additional benign background traffic. It was used as the reference point for both latency and observability measurements.

The partial observability conditions were introduced by selectively disabling chosen telemetry sources while keeping the remaining monitoring pipeline unchanged. Two such variants were considered: one with XSS-related telemetry removed and one with telemetry related to administrative access removed. These conditions were used to examine how incomplete instrumentation affected visibility of the attack scenario and whether telemetry loss was accompanied by changes in detection timing.

The load conditions were defined by introducing benign background traffic at increasing rates while preserving complete telemetry. Five levels were considered: 0, 10, 25, 50, and 75 requests per second. These conditions were used to assess whether increasing benign workload influenced detection latency and whether full scenario observability could still be maintained. The selected load levels were intended to provide a controlled moderate workload range for comparative evaluation within the minimal experimental setup. They were not designed to reproduce the full background traffic characteristics of production-scale cyber range environments.

Each experimental condition was repeated three times. Within each run, the complete attack scenario was executed once in order to evaluate observability, followed by additional repetitions of the SQL injection-like step used for latency estimation. The complete set of experimental conditions is summarized in Table 2.

Table 2: Summary of experimental conditions.

Experiment ID	Experiment	Telemetry modification	Benign load [rps]
A	Baseline	None	0
B1	POST: XSS logs off	XSS disabled	0
B2	POST: admin logs off	Admin-access disabled	0
C1	Load: 0 rps	None	0
C2	Load: 10 rps	None	10
C3	Load: 25 rps	None	25
C4	Load: 50 rps	None	50
C5	Load: 75 rps	None	75

Metrics

Detection latency

Detection latency was used to characterize the temporal responsiveness of the monitoring loop. It was defined as the time difference between the timestamp of an attack-related event recorded in the victim-side application logs and the timestamp of the corresponding alert generated by the SOC component. Formally, detection latency was calculated as:

$$L = t_{alert} - t_{event}$$

where t_{event} denotes the timestamp of the application-layer event and t_{alert} denotes the timestamp of the corresponding detection alert.

In the reported experiments, latency estimation was based on SQL injection-like events. Aggregated statistics included mean latency, median latency (p50), the 95th percentile (p95), and the minimum and maximum observed values.

Telemetry Coverage Index

Scenario-level observability was quantified using the Telemetry Coverage Index (TCI). This metric expressed the proportion of expected attack-related events that were successfully observed through the available telemetry during execution of the complete scenario. The metric was defined as:

$$TCI = \frac{N_{observed}}{N_{defined}}$$

where $N_{observed}$ denotes the number of scenario steps for which the expected telemetry artifact was observed and $N_{defined}$ denotes the total number of steps defined in the attack scenario.

A value of TCI = 1.00 indicated full observability of the scenario, whereas lower values reflected partial loss of visibility caused by disabled telemetry sources.

Results

Partial Observability

The results obtained for the baseline and partial observability experiments are presented in Fig. 3 and summarized in Table 3. In the baseline experiment (A), full scenario observability was achieved, resulting in TCI = 1.00. The mean detection latency was 0.0771 s, with p95 = 0.1439 s.

Under partial observability, experiment B1, in which XSS-related telemetry was disabled, reduced the Telemetry Coverage Index to 0.80. The mean latency in this case was 0.0787 s, while p95 reached 0.1431 s. In experiment B2, where telemetry related to administrative access was disabled, observability decreased further to TCI = 0.60, whereas the mean latency was 0.0755 s and p95 = 0.1399 s.

These results show that telemetry degradation consistently reduced observability, but its influence on latency was not uniform. Although both partial observability conditions lowered TCI, the timing characteristics of SQL injection-like detection remained within a relatively narrow range.

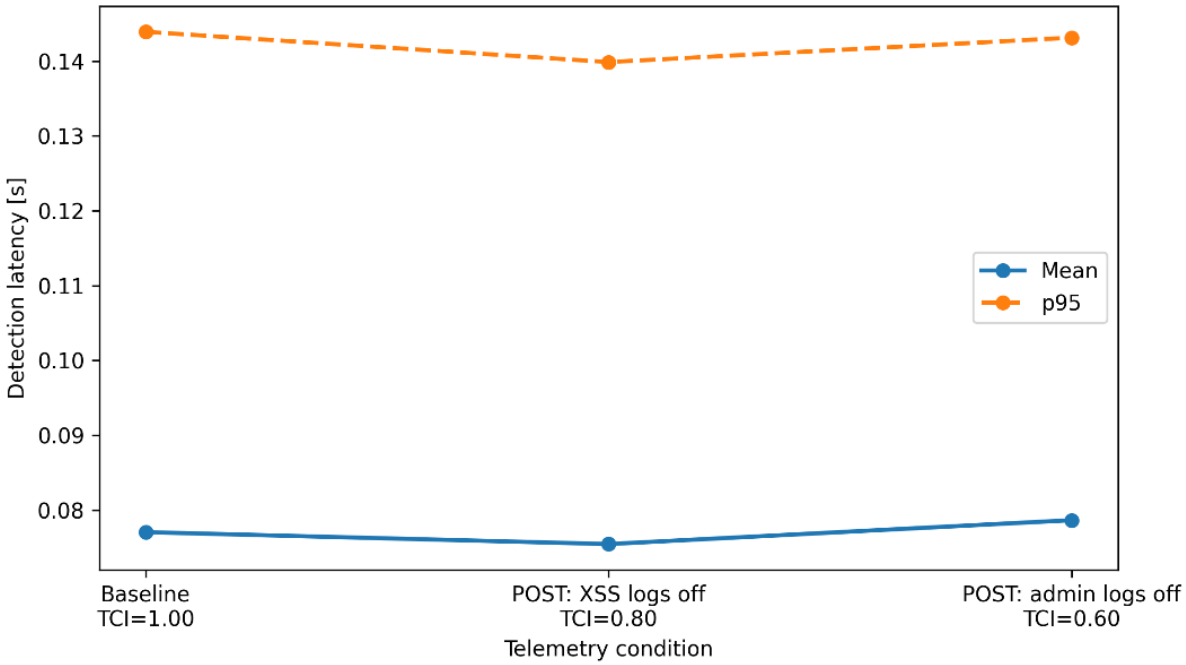

Fig. 3. Detection latency under telemetry degradation conditions.

Load Conditions

The load-related results are shown in Fig. 4 and summarized in Table 3. Across all load conditions (C1 - C5), full observability of the attack scenario was preserved, resulting in TCI = 1.00 for every tested load level.

Mean detection latency remained within a narrow range from 0.0703 s to 0.0770 s, while p95 varied only slightly between 0.1397 s and 0.1448 s. The lowest mean latency was observed at 25 rps (C3), whereas the highest mean latency occurred at 75 rps (C5). Despite these differences, the upper tail of the latency distribution remained relatively stable across the tested load levels.

Overall, the load experiments indicate that increasing benign background traffic did not reduce scenario observability and had only a moderate influence on the timing characteristics of the detection pipeline within the tested interval. However, this conclusion is limited to the evaluated workload range up to 75 rps and should not be interpreted as evidence of stability under substantially higher or production-scale traffic conditions.

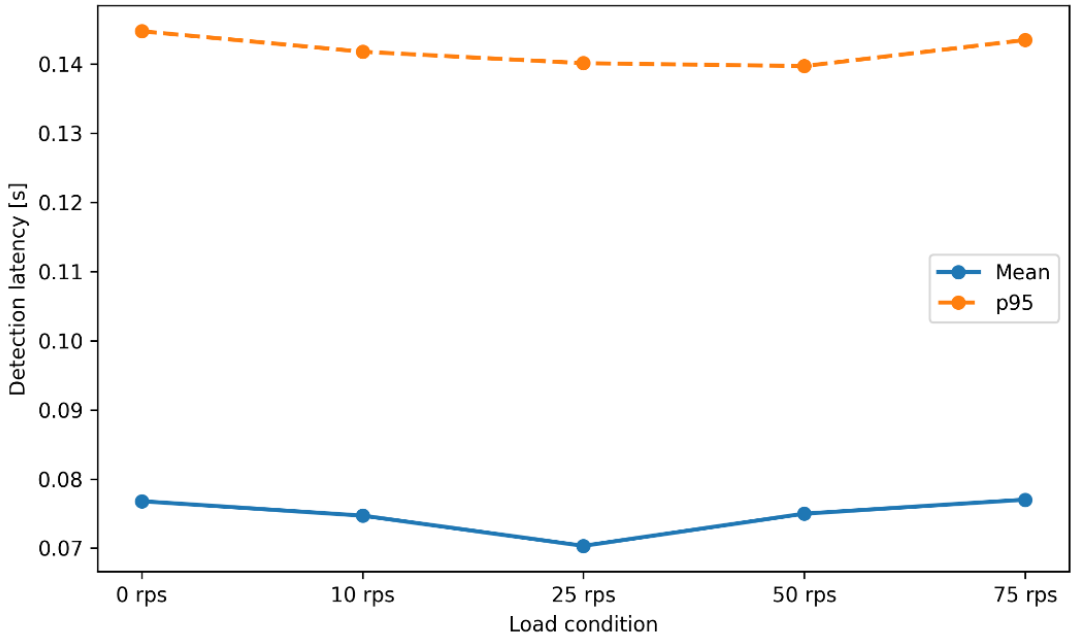

Fig. 4. Detection latency under increasing benign workload.

Summary of aggregated results

The complete set of aggregated results across all experiments is presented in Table 3. The table combines observability and latency statistics for the baseline, partial observability, and load conditions, and serves as the main quantitative reference for further discussion.

Table 3: Detection latency and observability across all experimental conditions.

Experiment ID	Experiment	TCI	Benign load [rps]	Mean	p50	p95	Min	Max	Std
A	Baseline	1.0	0	0.0771	0.0786	0.1439	0.0008	0.1484	0.0445
B1	POST: XSS logs off	0.8	0	0.0787	0.0783	0.1431	0.0026	0.1486	0.0442
B2	POST: admin logs off	0.6	0	0.0755	0.0790	0.1399	0.0049	0.1476	0.0440
C1	Load: 0 rps	1.0	0	0.0768	0.0782	0.1448	0.0022	0.1500	0.0452
C2	Load: 10 rps	1.0	10	0.0747	0.0733	0.1418	0.0009	0.1480	0.0432
C3	Load: 25 rps	1.0	25	0.0703	0.0678	0.1401	0.0035	0.1457	0.0439
C4	Load: 50 rps	1.0	50	0.0750	0.0746	0.1397	0.0017	0.1455	0.0438

C5	Load: 75 rps	1.0	75	0.0770	0.0731	0.1435	0.0012	0.2980	0.0585	

Conclusion

This paper presented an empirical study of detection latency and scenario-level observability in a minimal container-based cyber range. The implemented environment was based on application-level telemetry and a simplified rule-based monitoring loop, making it possible to evaluate detection behavior under controlled and reproducible conditions.

The baseline experiment demonstrated full scenario observability with TCI = 1.00 and sub-second application-layer detection latency. Under partial observability, selective removal of telemetry reduced TCI to 0.80 and 0.60, showing that incomplete instrumentation directly affected visibility of the attack scenario. At the same time, the influence of telemetry degradation on latency was not uniform, indicating that observability loss and detection timing should be treated as related but distinct properties.

The load experiments showed that increasing benign background traffic from 0 to 75 rps preserved full observability across all tested conditions. Mean latency varied only within a narrow range, while p95 remained relatively stable, suggesting that the simplified monitoring pipeline was resilient to moderate workload growth in the evaluated interval.

The study was limited to a single-host environment, a simplified rule-based detection mechanism, and latency estimation for one measured attack type under a moderate benign workload range. As a result, the findings should be interpreted as a controlled empirical baseline rather than as a direct representation of distributed or production-scale cyber range deployments. Future work should extend the setup to multi-host environments, incorporate explicit time synchronization assumptions, evaluate higher and more realistic background traffic levels, and examine failure-oriented thresholds in which delayed detection becomes operationally unacceptable.

Acknowledgment

This work was financed/co-financed by Military University of Technology under research project UGB 90/2026.

References

- Chouliaras, N., Kantzavelou, I., Maglaras, L., Pantziou, G. and Ferrag, M. A. (2023), 'A novel autonomous container-based platform for cybersecurity training and research,' PeerJ Computer Science, 9, e1574, DOI:10.7717/peerj-cs.1574.
- Lillemets, P., Bashir Jawad, N., Kashi, J., Sabah, A. and Dragoni, N. (2025), 'A Systematic Review of Cyber Range Taxonomies: Trends, Gaps, and a Proposed Taxonomy,' Future Internet, 17 (6), 259, DOI:10.3390/fi17060259.
- Nakata, R. and Otsuka, A. (2020), 'Evaluation of vulnerability reproducibility in container-based cyber range,' arXiv preprint arXiv:2010.16024v2, DOI:10.48550/arXiv:2010.16024.
- Theodoropoulos, T., Rosa, L., Benzaid, C., Gray, P., Marin, E., Makris, A., Cordeiro, L., Diego, F., Sorokin, P., Girolamo, M., Barone, P., Taleb, T. and Tserpes, K. (2023), 'Security in cloud-native services: A survey,' Journal of Cybersecurity and Privacy, 3 (4), 758-793, DOI:10.3390/jcp3040034.
- Ukwandu, E., Farah, M. A. B., Hindy, H., Brosset, D., Kavallieros, D., Atkinson, R., Tachtatzis, C., Bures, M., Andonovic, I. and Bellekens, X. (2020), 'A Review of Cyber-Ranges and Test-Beds: Current and Future Trends,' Sensors, 20 (24), 7148, DOI:10.3390/s20247148.
- Yamin, M. M., Katt, B. and Gkioulos, V. (2020), 'Cyber ranges and security testbeds: Scenarios, functions, tools and architecture,' Computers & Security, 88, 101636, DOI:10.1016/j.cose.2019.101636.