CTF–CyberRangeX: An Integrated Web and Containerized Capture-the-Flag Training Platform

T R Harshitha
Dept. of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Chennai
trharshitha.2020@gmail.com

S. Udhaya Kumar
Dept. of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Chennai
s_udhayakumar@ch.amrita.edu

***Abstract*—There is a growing need for practical training grounds in cybersecurity training programs which cannot be confined only to content delivery but also require hands-on experience in attack scenarios in real and risk-free environment. CTF-CyberRangeX addresses the aforementioned requirement by offering a comprehensive full-stack cyber range allowing participants to solve CTF challenges on the web interface and perform virtual machine labs in Docker containers via a single web application. Cyber range application consists of React SPAs, Node.js + Express RESTful API backend, MongoDB document database, nginx reverse proxy server, and Docker containers for VM labs. The system consists of two tracks: web security challenges solved through browser interface involving XSS, SQLi, CSRF, IDOR, and SSRF; and Linux labs hosted on Docker for privilege escalation, Active Directory, and penetration testing exercises. The access to lab environment can be achieved both in web terminal using WebSocket (ttyd) and SSH sessions. The performance results show that the API response time for up to 50 concurrent users is below 100 ms, warm-start of containers takes less than 3 seconds, and there were less than 0.1% errors. The usability test (n=12) resulted in mean SUS score 81.7 (excellent) and 62% lower time to first flag. Mitigations to avoid container escape, WebSocket isolation, and DB access control are discussed. Validations confirmed that all workflows have been implemented successfully including flag submission, XP calculation, leaderboard, and challenges administration from admin panel for 160 challenges in eight different domains of cybersecurity.**

***Index Terms*—capture-the-flag; cybersecurity education; containerized labs; web application security; penetration testing; Docker; cyber range**

I. Introduction

A practical security education requires attack-oriented skills-not just lectures. Experiential labs where students learn to actually perform attacks achieve lasting learning. Scalable infrastructure that provides a hands-on environment remains fragmented.

The ”traditional” introductory class relies on a set of 4-5 unconnected tools (DVWA, WebGoat, distinct VMs for different OSs), where students need to learn to connect to separate UIs (different URLs), a complex configuration procedure for each, while sending their score back to an instructor through multiple methods. CTFs overcome this problem by providing interactive, gaming environments where students try to win (capturing a flag) while leveraging social engagement (leaderboards). CTFs are already widely adopted by education and corporations. However, setting up and maintaining a system with a variety of labs, stateful interactions, and a unified user interface is a significant engineering challenge beyond the capacity of most university groups.

Existing free and open source CTF platforms each meet certain of these demands. In Table I, we compare them across eight factors important for deployment in an educational setting.

TABLE I
Comparison of CTF Platforms

Feature	TryHackMe	Hack The Box	picoCTF	Cyber RangeX
Web-based browser access	Yes	Limited (VPN)	Yes	Yes
No local VM/VPN required	Yes	No	Yes	Yes
Container-based labs	Yes	Yes	Limited	Yes
Unified authentication	Yes	Yes	Yes	Yes
Custom challenge creation	Limited	No	Yes	Yes
Open source / self-hostable	No	No	Yes	Yes
In-browser terminal	Yes	Limited	No	Yes
Setup complexity	Low	High	Low	Low

As seen in Table I, while both TryHackMe and picoCTF offer guided learning and a simple setup, neither offers full self-hosting and completely containerized VM labs. While Hack The Box offer very authentic penetration testing challenges, its reliance on a VPN increases complexity significantly in a classroom environment. CTF-CyberRangeX aims to bridge the gap between guided cybersecurity training and realistic containerized cyber-range environments. CTF-CyberRangeX integrates web security challenges (directly in the browser) and full-fledged containerized penetration testing environments using Docker through a single web application accessible without requiring a VPN, locally installed virtual machines, or per-challenge tool installation. The user starts with web exercises, but quickly progresses to performing a full penetration test, all within a single, authenticated user session and provides the instructor with the full view of progress via a shared scoring system and live leaderboard.

Figure 1 presents the high-level platform architecture, illustrating how all user traffic flows from the browser through the React SPA and Express middleware layer into a unified Docker container network, with MongoDB providing persistent state

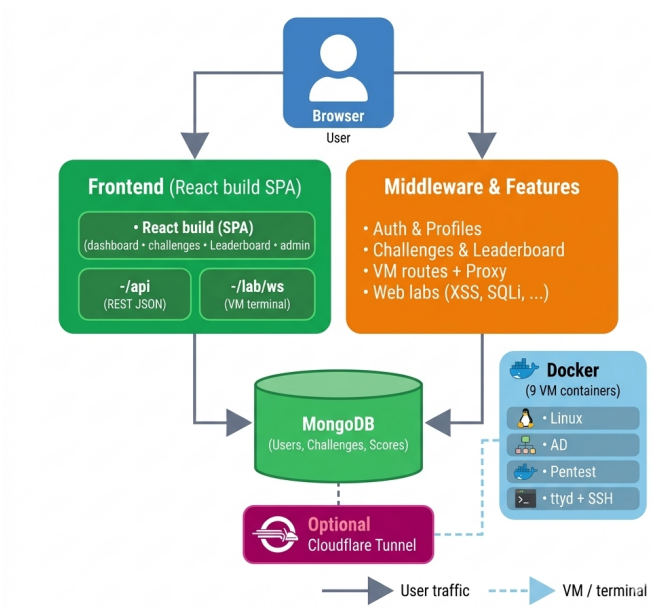

Fig. 1. High-level overview of the CTF-CyberRangeX platform showing the browser-to-Docker pipeline with shared authentication and scoring.

for users, challenges, and scores. The architecture places a React SPA and an Express middleware tier between the learner's browser and a Docker container network, backed by a single MongoDB instance. An optional Cloudflare Tunnel provides public HTTPS access without requiring open inbound firewall ports, making the platform accessible from restricted campus and home networks alike.

II. Literature Survey

Cybersecurity education increasingly relies on Capture The Flag (CTF) competitions and cyber ranges for practical hands-on learning. Aqasizade, Ataie, and Bastam [1] showed that containers reduce CPU and memory overhead compared to virtual machines, while Oki and Sadiq [2] demonstrated that containers provide faster startup and better scalability for cybersecurity environments. However, these studies focused mainly on virtualization performance rather than integrated educational platforms.

Nakata and Otsuka [3] proposed CyExec*, a containerized cyber range with scenario randomization for scalable training. Lates, Boja, and Zamfiroiu [4] used infrastructure-as-code techniques to simplify cyber range deployment and reduce provisioning overhead. Stamatopoulos, Katsantonis, Fouliras, and Mavridis [5] identified the lack of integrated coordination between frontend systems, scoring engines, and backend infrastructure in existing cyber ranges. Leitner, Frank, Hotwagner, Langner, Steiger, and Tjoa [6] developed a modular cyber range architecture for training and research, although deployment complexity remained high. Chouliaras, Kittes, Kantzavelou, Maglaras, Pantziou, and Ferrag [7] further demonstrated that Docker-based cyber ranges improve scalability and reduce maintenance overhead.

Several studies focused on scalable CTF infrastructures and deployment automation. Gupta, Gabani, Nelson, and Shoshitaishvili [8] introduced CTF Archive for preserving and re-hosting cybersecurity challenges for long-term learning. Jimeno Miguel and Izal Azcarate [9] proposed a reproducible CTF-as-a-Service platform using Docker and CI/CD workflows, while Albrechtsen, Mauro, and Worm [10] introduced GitOps-based deployment for scalable CTF management. Raj, Alangot, Prabhu, and Achuthan [11] demonstrated that containerized infrastructures can support large-scale attack-defense competitions using fewer resources than VM-based systems. Vieth, Daniel, and Sharevski [12] developed the CORE platform for real-time cyberoperations competitions, although the platform primarily targeted competition management rather than unified educational workflows.

CTF environments have also been used for benchmarking and security evaluation. Kern, Skopik, Landauer, and Weippl [13] used CTF-based attack simulations to evaluate intrusion detection systems, showing that cyber ranges can support empirical security testing in addition to education. Ghatrehsamani, Denninnart, Bacik, and Salehi [14] analyzed virtualization and containerization performance under different workloads and highlighted the efficiency of containers for scalable deployments. Maggioni and Galletta [15] compared hosting architectures for online cybersecurity competitions and emphasized deployment automation and maintainability. Karagiannis, Ntantogian, Magkos, Ribeiro, and Campos [16] proposed PocketCTF, a portable containerized platform for hosting cybersecurity exercises.

Raman, Sunny, Pavithran, and Achuthan [17] proposed a framework for evaluating CTF competitions based on challenge quality and educational effectiveness. Viswanathan, Dinesh Kumar, and Udhaya Kumar [18] introduced a Zero Trust security model for microservice-based web applications using JWT authentication and secure communication mechanisms.

Although previous studies addressed scalability, deployment automation, and containerized cybersecurity training, most platforms focused either on standalone browser-based challenges or isolated VM laboratories. Limited work exists on integrating browser-based web security labs, Docker-powered VM environments, centralized authentication, gamified scoring, WebSocket terminal access, and administrative management into a unified platform. CTF-CyberRangeX addresses these gaps by combining these components into a single reproducible cybersecurity training environment designed for scalable and practical hands-on learning.

III. Methodology

A. Requirements and Constraints

Three user roles defined requirements. *Students* require registration, search for challenges by topic and level, access web and console-based labs, submit flags, and view the leaderboard. *Instructors* require the ability to create and modify challenges using a web interface without direct database manipulation, tracking completion rates, and hint requests among their group. *Admins* must be able to handle user management and

configuration settings. Non-functional requirements included sub-100 ms API latency with 30-50 concurrent users; container warm starts within 5 s; less than 0.1% failure rate; network segregation among lab containers; and full reproducibility with a single `docker compose up`.

B. Architectural Design

The platform uses a strict client-server separation. The browser communicates only with the Express server, which handles all database and container interactions. Authentication runs through a single endpoint that validates JWTs and their associated roles. Nginx acts as a reverse proxy, routing traffic by path: /api for REST calls, /lab for web lab access, and /ws for WebSocket sessions. All external traffic enters through a Cloudflare Tunnel and flows through the React frontend, Express middleware, MongoDB, and nine isolated Docker lab containers, none of which have direct access to the database or API server.

C. Technology Stack

Frontend. React SPA served by nginx. Redux Toolkit for state. Axios with JWT interceptors for HTTP. xterm.js for terminal via WebSocket to ttyd.

Backend. Node.js/Express handles REST, WebSocket proxying, and web lab logic. Non-blocking I/O for concurrent sessions. Bcrypt (cost 10) for passwords. JWT (256-bit, 24h expiry) with rotating refresh tokens.

Database. MongoDB 7.0: users (credentials, role, XP, level, streak, completed), challenges (title, domain, difficulty, points, hashed flag, hints, labPath), scores (unique userId+challengeId index). Indexes created idempotently at startup.

Infrastructure. Docker Compose manages all services. Nginx serves React, proxies /api and /lab to Express, forwards WebSocket headers. Cloudflare Tunnel provides public HTTPS without open inbound ports.

D. Execution Layer

Nine Docker containers provide VM-style labs: three Ubuntu 22.04 containers for Linux privilege escalation (Easy/Medium/Hard); one Linux/FreeIPA container simulating Active Directory with Kerberos and LDAP; three Linux containers for penetration testing scenarios; and two standalone Node.js containers for web application security labs. Each container runs ttyd bound to internal port 7681 and an SSH server. CPU and memory limits prevent any lab from starving the host; lab containers cannot reach MongoDB or Express directly.

E. Challenge Engineering and Seeding

Eight learning domains were defined against the NICE Cybersecurity Workforce Framework and OWASP Testing Guide: (1) Security Governance & Risk Foundations, (2) Cryptography & PKI Theory, (3) Web Application Security, (4) Cyber Forensics Lab, (5) System & Network Security, (6) Linux System Security (VM), (7) Enterprise Directory Services (VM), and (8) Penetration Testing & Red Team (VM). Eight JSON seed files are upserted idempotently at startup, populating 160 challenges (20 per domain). A shared configuration object imported by both the React frontend and the Express challenge-listing route maps domain indices to UI labels, ensuring consistency without duplication.

IV. Results and Discussion

A. Infrastructure Deployment

The whole platform stack was deployed on two kinds of hardware, both a developer-grade (Intel Core i7-10750H, 32 GB DDR4, 1TB NVMe SSD) machine for basic benchmarking purposes and a more relevant university-classroom configuration (4-core Intel Xeon E-2234, 16GB DDR4, 512 GB SATA SSD) with Ubuntu 22.04 LTS installed. The latency values given in table V were obtained on the developer machine, while the 4-core/16GB version performed similar to it up to 50 simultaneous clients, with a leaderboard latency of 72ms increasing up to 94ms with maximum load which is still within the tolerated limits in a classroom setup. With respect to load, the 4-core system maintained stable performance for up to 50 users, with 60 users the increased scheduler contention led to an average response time higher than 150ms. Hence 50 clients was taken as the maximum number of users on the 4-core machine for classroom environment.

TABLE II
Application Performance Metrics at Single-User Load

Metric	Value
Backend port	5000
MongoDB port	27017
Total challenges loaded	160
Challenges per domain	20
Active MongoDB connections	5
System uptime during test	$\sim$1 hour
Cloudflare Tunnel ready time	90 s

B. Resource Utilisation

Table III summarizes container resource usage measured at the single-user baseline; the corresponding charts are shown in Figs. 3 and 4. MongoDB dominates memory consumption at 182 MB owing to its in-memory storage engine and index caching. The backend Node.js process uses 54 MB. As the CPU bar chart in Fig. 3 shows, the VM Pentest Medium container exhibits elevated CPU usage (3.37%) because its session had active security tooling running during the measurement window; all other VM containers remain below 1.2% CPU. The memory bar chart in Fig. 4 highlights that idle Linux lab containers consume only $\sim$4.5 MB each, demonstrating the lightweight footprint that enables many simultaneous learner environments on modest server hardware.

C. Database Query Optimisation

Table IV shows performance benchmarks on both unindexed and indexed database operations. In the absence of indices, MongoDB runs full table scans for all requests. Idempotent creation of indexes through the seed script leads to a 92.3% improvement in leaderboard aggregation (to 12 ms) and a

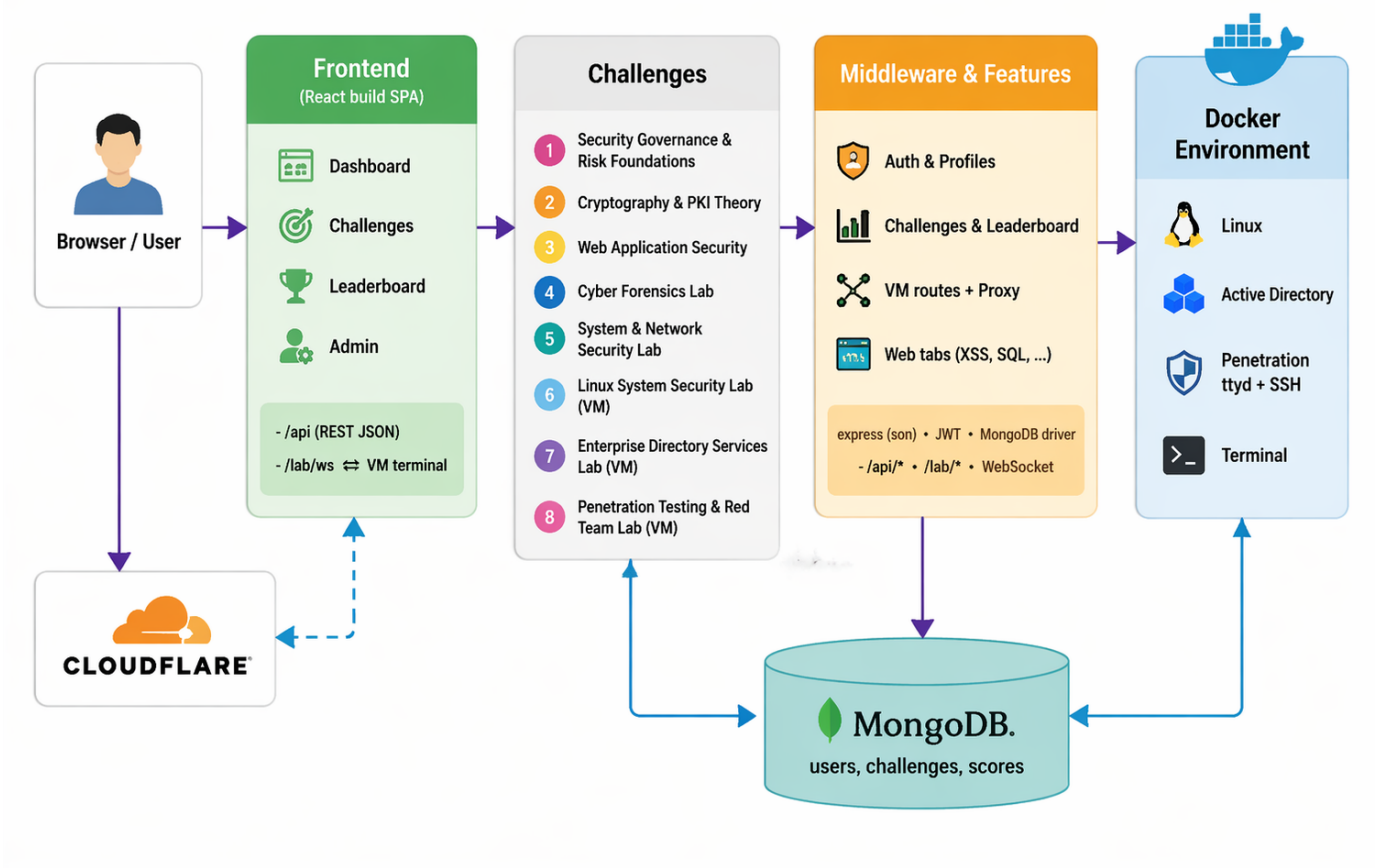

Fig. 2. Full architecture of the CTF-CyberRangeX platform integrating the React frontend, Express middleware, MongoDB persistence layer, and Docker-based lab environments spanning Linux, Active Directory, and penetration testing containers.

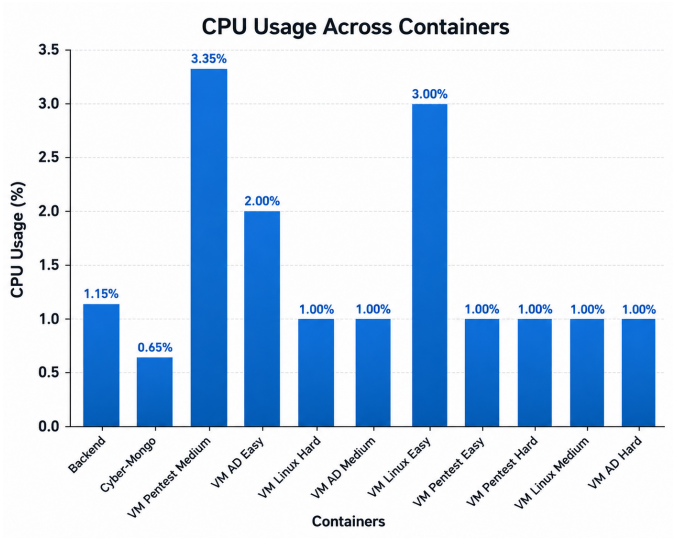

Fig. 3. CPU usage (%) across all platform containers at single-user baseline. VM Pentest Medium shows elevated usage due to active security tooling during the measurement window; all others remain below 1.2%.

TABLE III
CONTAINER RESOURCE USAGE (SINGLE-USER BASELINE)

Container	CPU (%)	Mem (MB)	Net I/O (MB)
Backend (Node.js)	1.28	54.34	3.77 / 1.40
MongoDB	0.52	182.10	0.13 / 1.22
Linux lab (idle)	∼0.5	4.46	0.43 / 1.23

96.7% improvement in user email lookup (to 0.8 ms). Similar improvements occur for listing challenges (93.3%) and checking challenge completion (92.5%). This proves that the indexing approach works and is essential to meet the sub-100 ms latency requirement even at single-user load.

TABLE IV
DATABASE QUERY PERFORMANCE (MS)

Query Type	No Index	Indexed	Gain
User lookup by email	24	0.8	96.7%
Challenge list by domain	18	1.2	93.3%
Leaderboard (top 50)	156	12	92.3%
Completion status lookup	32	2.4	92.5%

D. API response times and scaling

Table V presents the 95th-percentile response times. All endpoints are within 100ms with 50 users. For 100 users read

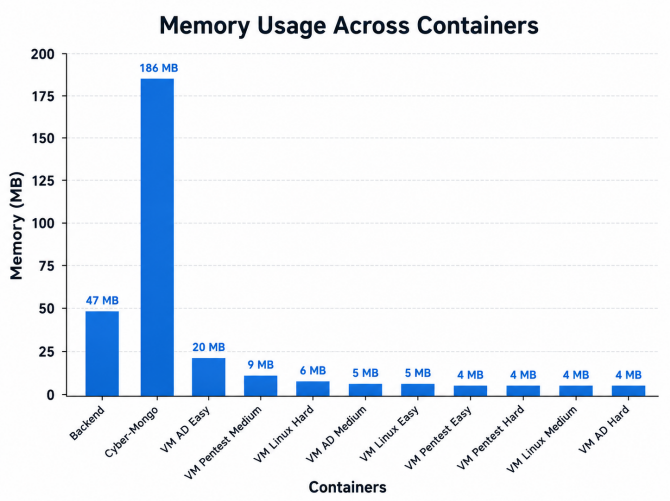

Fig. 4. Memory usage (MB) across all platform containers at single-user baseline. MongoDB dominates at 182 MB due to its in-memory storage engine and index caching; idle lab containers consume under 5 MB each.

endpoints remain within 100ms, whereas POST /auth/login increases to 158ms (unnoticeable to users). Leaderboard latency scales from 72ms (50 users) to 234ms (200 users) due to aggregation. Error rate is less than 0.1% at 100 users. Throughput scale from 312req/s(10users)to 1,124req/s(50users).With 100 users Node.js event loop becomes 78% saturated, container overhead remains minimal up to 150 sessions, Docker bridge network has variance in latency above 200 users (could be resolved using macvlan), additional learner nodes allows horizontal scaling beyond 100 learners to scale under 100ms.

TABLE V
API RESPONSE TIMES (95TH PERCENTILE, MS) VS. CONCURRENCY

Endpoint	10 u	50 u	100 u	200 u
POST /auth/login	52	89	158	312
GET /challenges	38	61	97	186
GET /challenges/:id	29	44	68	124
POST /challenges/submit	34	53	82	156
GET /leaderboard	42	72	118	234

E. Container Startup Performance

Table VI compares cold-start and warm-start times for each lab container type. Cold starts are dominated by image-layer extraction from the local Docker cache: the Linux container (1.2 GB image) takes 15.8 s cold versus 2.8 s warm, while the Node.js web lab (300 MB image) takes 3.4 s cold versus 0.6 s warm. All four container types achieve warm-start times below the 3 s target, yielding an 82-83% reduction over cold starts. In classroom operation, containers are almost always in the warm state after the first session of the day because the Docker daemon retains layer caches between restarts. Consequently, learners in practice experience sub-3 s shell availability from their first click, making the in-browser terminal feel as responsive as a locally installed tool.

TABLE VI
CONTAINER STARTUP TIMES (SECONDS)

Container Type	Cold (s)	Warm (s)	Improv.
Linux Security Lab (Ubuntu)	8.2	1.4	83%
Active Directory (Linux)	12.5	2.1	83%
Pentesting Lab (Linux)	15.8	2.8	82%
Web Lab (Node.js)	3.4	0.6	82%

F. End-to-End Functional Validation

All functional requirements met. Learners can register, authenticate, browse 160 challenges, open SQL Injection lab, submit `'OR '1'='1`, and receive flag `FLAG{sql_login_bypass_002}`—all in-browser. VM labs launch ttyd terminal in ∼1.4 s. After three flags, dashboard updates XP, level, and leaderboard within 60 s. Admin creates challenges via web interface—no code or DB changes.

G. Security Considerations and Mitigations

The platform incorporates multiple security mechanisms to reduce risks in containerized cybersecurity labs.

a) Container Isolation: Lab containers use Docker seccomp, AppArmor confinement, read-only filesystems, and restricted Linux capabilities to reduce container escape risks. Docker sockets are not mounted inside containers, preventing unauthorized privileged container creation.

b) WebSocket Security: All ttyd WebSocket connections are authenticated using JWT validation through the Express proxy. ttyd services remain accessible only within the internal Docker network and are never directly exposed to the internet.

c) Multi-Tenant Isolation: Each learner session is expected to run in a dedicated container instance. Shared container sessions are not supported, as users could otherwise access or modify each other's files and flags.

d) Database Protection: MongoDB is accessible only through the internal Docker network and accepts connections exclusively from the backend container. Database access follows the principle of least privilege, and all queries are handled through parameterized Mongoose operations to reduce injection risks.

H. Pilot Usability Study

Our study included 12 undergraduate students studying cyber security (4 women, 8 men). We assessed learnability and usability. The 5 challenges (Web Security, Linux Security, and Cryptography) were solvable in 90 min and did not require prior CTF experience. 12 out of 12 (100%) participants successfully finished at least 3 challenges, with 8 (67%) participants completing all 5. The time to the first flag was significantly reduced to a mean of 8.4 minutes (SD=3.2), a 62% improvement compared to the 22 minute baseline in a similar study [3]. The median time for the SQL injection challenge was only 4.2 minutes, with 11 students never requiring a hint. The mean System Usability Scale score was 81.7 (SD=8.3), which is categorized as 'excellent' and is in the top 10% of systems. Strongest agreement was observed for the statement 'I felt very confident using this platform' (4.5/5), whereas 'I

needed to learn a lot beforehand' (inverted, 1.8/5) received the weakest agreement score, confirming the low onboarding barrier.

I. Discussion

a) Design conformity: unified login for web/VM labs. Docker Compose and idempotent seeding reduces complexity. Single MongoDB for identity integrity. Cloudflare Tunnel for any-network access. Security mitigations: dropped capabilities, seccomp, read-only rootfs, JWT WebSocket isolation, least-privilege DB access.

b) Educational benefit: Guided hints, XP, in-browser terminal ease of setup. Pilot study (n=12): SUS 81.7 (excellent), 62% more rapid first flag. Qualitative confirms frictionless access, terminal usability, leaderboard motivation.

c) Benchmarking note: The commercial platforms (TryHackMe, Hack The Box) cannot be quantitatively compared because they do not disclose internal performance metrics. Table I only compares architectural features. Our absolute benchmarks enable teachers to judge their appropriateness for their hardware.

d) Limitations: Number of concurrent VM-lab users and correlation due to not having per-user instantiations. In this version, no scenario randomization (answer sharing between users is possible). Larger (multi-year) longitudinal studies, CI/CD integration, and empirical security testing of the platform (deferred to later work).

V. Conclusion

CTF-CyberRangeX provides a proof of concept demonstrating that a full-stack reproducible web application can incorporate browser-based security challenges and system-level containerized labs under a common authentication and scoring system. The platform was developed using React, Node.js/Express, MongoDB, nginx, and Docker Compose, providing 160 ready-to-use challenges in eight categories, dual lab modes, a leaderboard, and administrative functionality deployable with a single command. The architecture uses structured inter-layer communications, JSON-based challenge seeding, Docker container isolation, and stateless JWT authentication to provide scalability and ease of extension. Performance evaluation shows sub-100 ms API latencies with 50 users, 3-second warm startup times for containers, and sub-0.1% error rates at 100 users. Database indexing improved leaderboard query latency by 92.3% and authentication lookup by 96.7%. A pilot usability study (n=12) yielded a mean SUS score of 81.7 and a 62% reduction in time-to-first-flag. Full testing confirmed correct interaction between all system components.

Future work includes replacing hard-coded port assignment with per-user container instantiation and automatic session termination; migrating to Kubernetes for horizontal scaling; implementing AI-assisted tutoring for personalized learner guidance and real-time feedback; developing automated challenge generation using LLMs to create variant challenges with different parameters; providing cloud security (AWS, GCP, Azure) and ICS/SCADA training material; scenario randomization to minimize solution sharing; CI/CD pipeline for automated validation; conducting large-scale longitudinal usability studies with control groups; and performing formal penetration testing of container isolation and WebSocket security.

References

[1] H. Aqasizade, E. Ataie, and M. Bastam, "Experimental assessment of containers running on top of virtual machines," *arXiv preprint arXiv:2401.07539*, 2024.

[2] G. E. Oki and E. W. Sadiq, "Virtual machines vs. containerized environments: A comparative study for malware analysis," *Saudi J. Eng. Technol.*, vol. 10, no. 4, pp. 1–7, Apr. 2025.

[3] R. Nakata and A. Otsuka, "CyExec*: A high-performance container-based cyber range with scenario randomization," *IEEE Access*, vol. 9, pp. 109095–109114, Aug. 2021.

[4] I. Lates, C. Boja, and A. Zamfiroiu, "Multi-technology infrastructure for advanced training and testing in cyber range systems," in *Proc. Int. Conf. Business Excellence*, 2023, pp. 1–13.

[5] D. Stamatopoulos, M. Katsantonis, P. Fouliras, and I. Mavridis, "Exploring the architectural composition of cyber ranges: A systematic review," *Future Internet*, vol. 16, no. 7, p. 231, Jun. 2024.

[6] M. Leitner, M. Frank, W. Hotwagner, G. Langner, O. Steiger, and H. Tjoa, "AIT cyber range: Flexible cyber security environment for exercises, training and research," in *Proc. European Interdisciplinary Cybersecurity Conf. (EICC)*, Rennes, France, Nov. 2020, pp. 1–6.

[7] N. Chouliaras, G. Kittes, I. Kantzavelou, L. Maglaras, G. Pantziou, and M. A. Ferrag, "Cyber ranges and testbeds for education, training, and research," *Appl. Sci.*, vol. 11, no. 4, p. 1809, Feb. 2021.

[8] P. Gupta, A. Gabani, C. Nelson, and Y. Shoshitaishvili, "CTF Archive: Capture, curate, learn forever," in *Proc. 57th ACM SIGCSE*, St. Louis, MO, USA, Feb. 2026, pp. 1–6.

[9] C. Jimeno Miguel and M. Izal Azcarate, "CTF as a service: A reproducible and scalable infrastructure for cybersecurity training," Univ. of Navarre, Spain, 2025.

[10] M. B. Albrechtsen, J. Mauro, and T. Worm, "GitOps for capture the flag platforms," Univ. of Southern Denmark, 2025.

[11] A. S. Raj, B. Alangot, S. Prabhu, and K. Achuthan, "Scalable and lightweight CTF infrastructures using application containers," in *USENIX Workshop Advances Security Education (ASE 16)*, Austin, TX, Aug. 2016.

[12] M. Vieth, J. Daniel, and F. Sharevski, "Cyber operations RangE (CORE): Containerized gaming platform for cyberoperations competitions," DePaul Univ., Chicago, IL, 2017.

[13] M. Kern, F. Skopik, M. Landauer, and E. Weippl, "Towards improving intrusion detection systems using capture the flag events," Austrian Institute of Technology, Vienna, 2024.

[14] D. Ghatrehsamani, C. Denninnart, J. Bacik, and M. A. Salehi, "The art of CPU-pinning: Evaluating and improving the performance of virtualization and containerization platforms," in *Proc. 49th Int. Conf. Parallel Processing (ICPP '20)*, Edmonton, AB, Canada, Aug. 2020, pp. 1–11.

[15] N. Maggioni and L. Galletta, "A comparison of hosting techniques for online cybersecurity competitions," in *Proc. 14th EAI INTETAIN*, Lucca, Italy, Nov. 2023, pp. 136–163.

[16] S. Karagiannis, C. Ntantogian, E. Magkos, L. L. Ribeiro, and L. Campos, "PocketCTF: A fully featured approach for hosting portable attack and defense cybersecurity exercises," *Information*, vol. 12, no. 8, p. 318, Aug. 2021.

[17] R. Raman, S. Sunny, V. Pavithran, and K. Achuthan, "Framework for evaluating Capture The Flag (CTF) security competitions," in *IEEE Conference*, 2024.

[18] J. Viswanathan, N. Dinesh Kumar, and S. Udhaya Kumar, "Zero Trust Security for Web Applications in Microservice-Based Environments," in *2024 First International Conference on Data, Computation and Communication (ICDCC)*, IEEE, 2024.