Hack in a Box: Portable Docker Labs for Teaching Web Application Security

Katarzyna Mazur[1][0000−0003−3742−3181]

Maria Curie Sklodowska University, Lublin, Poland
katarzyna.mazur@umcs.pl

Abstract. Hands-on cybersecurity training is essential in computer science education as cyber threats continue to grow in sophistication and frequency [25]. Traditional lab environments often require complex infrastructure, expensive cloud resources, or dedicated physical servers, which limit student accessibility. This paper presents an approach in which cybersecurity exercises are prepared as Docker containers, enabling reproducible and portable learning environments that can run on students' laptops. We developed a series of vulnerable web applications packaged as standalone Docker containers, covering various security concepts, including JWT authentication flaws, HTTP header vulnerabilities, remote code execution, client-side code injection (cross-site scripting, XSS), and server-side request forgery (SSRF) attacks. Each exercise is designed as a self-contained environment that requires only simple commands to deploy. We integrated these exercises into a *Cybersecurity and Cryptography* course and evaluated their effectiveness using student performance metrics and survey data ($n = 26$ students). The results demonstrate positive learning outcomes and high levels of student satisfaction. Survey responses indicate that students valued the ability to practice at home and repeat exercises at their own pace. This work contributes a lightweight approach to scaling practical cybersecurity education without infrastructure barriers.

Keywords: web application security · hands-on training · cybersecurity · learning by doing.

1 Introduction

The rapid evolution of cyber threats presents an ongoing challenge for computer science education. As organizations face increasingly sophisticated attacks targeting web applications and cloud infrastructures, there is a critical need to equip students with practical defensive skills that extend beyond theoretical knowledge. Industry reports consistently highlight a global cybersecurity skills gap, with employers seeking graduates who can immediately contribute to securing systems and identifying vulnerabilities in real-world applications [29], [30], [34].

Hands-on laboratory exercises are widely recognized as essential pedagogical tools in cybersecurity education, enabling students to experience authentic attack scenarios, understand vulnerability exploitation mechanisms, and develop

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

defensive strategies through active learning [56], [22]. However, traditional approaches to delivering practical security training face significant barriers.

Virtual machine (VM)-based labs, while comprehensive, require substantial disk space (often 10–20 GB per environment), consume significant memory resources, and present compatibility challenges across different host operating systems [37]. Cloud-based training platforms offer scalability but introduce recurring costs, require stable internet connectivity, and may raise data privacy concerns. Dedicated physical lab infrastructure provides controlled environments but limits accessibility to specific campus locations and scheduled lab hours, thereby restricting opportunities for self-paced learning and practice.

These accessibility barriers are particularly problematic in the context of modern education, where students increasingly expect flexible, anytime-anywhere learning opportunities. The COVID-19 pandemic further highlighted the limitations of location-dependent lab infrastructure, accelerating the need for solutions that students can deploy on personal devices without institutional IT support.

Container technology, particularly Docker, has emerged as a transformative approach for software deployment, offering lightweight virtualization that packages applications with their dependencies into portable, isolated environments [42]. Docker has been widely adopted in industry for DevOps practices and microservices architectures, and its potential for cybersecurity education has been demonstrated through several academic frameworks ([35], [33], [24]).

Docker containers provide several advantages over traditional virtualization for educational purposes. First, containers share the host operating system kernel, resulting in minimal overhead compared to full VMs. A typical Docker container for a web application might be 100–500 MB, compared to multi-gigabyte VM images. Second, Docker's declarative configuration through Dockerfiles enables exercises to be defined as version-controlled code, ensuring perfect reproducibility across different student machines. Third, Docker's cross-platform compatibility (Windows, macOS, Linux) allows students to use their preferred environment without compatibility concerns. Finally, containers start in seconds rather than minutes, enabling rapid iteration and experimentation.

For cybersecurity education specifically, Docker's isolation model provides an ideal sandbox for creating vulnerable applications. Each exercise can be packaged as a self-contained environment with known vulnerabilities, allowing students to safely exploit security flaws without risking host system compromise or interfering with other students' work.

This paper presents and evaluates a Docker-based approach for delivering hands-on cybersecurity exercises focused on web application security vulnerabilities. Our primary research objective is to determine whether lightweight, container-based exercises can provide effective practical training while eliminating the infrastructure barriers associated with traditional lab environments. This work makes the following contributions to the field of computer science education:

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

- A validated educational approach for teaching web application security using Docker containers, demonstrated through deployment in a university *Cybersecurity and Cryptography* course,
- A collection of containerized vulnerable applications and exercises covering critical security concepts, including authentication flaws (JWT), HTTP protocol vulnerabilities, injection attacks (XSS), and server-side vulnerabilities (SSRF, RCE),
- Empirical evidence from student survey data demonstrating the effectiveness and accessibility of the approach,
- A reproducible, open approach that other institutions can adopt without expensive infrastructure investments.

The remainder of this paper is organized as follows. Section 2 reviews related work in cybersecurity education and containerized learning environments. Section 3 details the technical implementation of Docker-based exercises, while section 4 explains the integration of these exercises into our *Cybersecurity and Cryptography* course and our evaluation methodology. Section 5 presents results from student surveys and finally section 6 concludes with future directions for this work.

2 Related Work

Hands-on learning has been recognized as essential for cybersecurity education, with numerous initiatives developed over the past two decades. The SEED project, initiated by Du in 2002, pioneered the development of instructional laboratories for computer security education, creating over 40 labs covering software security, network security, and web security [28], [26]. SEED labs have been adopted by many institutions worldwide and have demonstrated that "learning by doing" significantly improves student engagement and retention compared to purely lecture-based approaches [55]. Du's work emphasized providing free, high-quality labs that minimize non-essential activities and focus on fundamental security principles [27].

Other platforms have emerged to support cybersecurity training, including CTF (Capture The Flag) competitions, DETERLab for network security research [43], and various commercial cyber ranges. However, these approaches often require significant infrastructure investment or recurring costs, which limit accessibility, particularly for smaller institutions or individual students practicing at home.

The emergence of Docker and container technology has created new opportunities for delivering hands-on cybersecurity labs. Irvine et al. developed Labtainers, a Docker-based framework that packages cybersecurity exercises as parameterized containers, allowing instructors to create isolated, reproducible lab environments [35], [36]. Thompson and Irvine demonstrated that containerized labs could address key challenges in cybersecurity education, including individualization of exercises and automated assessment [58].

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

Alsalamah et al. explored combining virtualization and containerization techniques, showing that Docker containers resolve issues with VM configuration, networking performance, and file sizes [20]. Their work highlighted that containers are particularly effective when used alongside VMs, rather than as complete replacements, offering complementary benefits. Chouliaras et al. presented a lightweight cyber range platform based on Docker and OpenStack, achieving maximized performance and scalability while reducing costs through an Infrastructure-as-Code (IaC) methodology [23].

EDURange, developed by Weiss, Mache, and colleagues, provides an authoring framework for instructors to construct Docker-based cybersecurity exercises using Terraform for infrastructure provisioning [60]. Their work emphasized reducing the learning curve for exercise creation and enabling deployment across multiple cloud platforms. O'Connor demonstrated the use of Docker for teaching binary exploitation and vulnerability research at the undergraduate level, showing that containers can support complex security topics previously considered too difficult for educational settings [50].

Recent work has also examined student learning outcomes within container-based environments. Tobarra et al. analyzed student acceptance and learning patterns in virtual laboratories, finding high user satisfaction and effective skill acquisition when proper pedagogical frameworks support the technology [59].

Despite advances in virtual lab technology, significant barriers remain. Traditional VM-based labs require substantial resources and considerable memory allocation, limiting the number of concurrent users and reducing student accessibility [40], [33]. Performance degradation occurs when many VMs run simultaneously on shared infrastructure, and setup complexity creates friction for students, particularly those with limited technical backgrounds [39].

Cloud-based solutions address some scalability concerns but introduce recurring costs and reliance on internet connectivity. Studies have shown that VM-based labs can support approximately 300 concurrent users before experiencing significant performance issues, requiring substantial server infrastructure [40]. More fundamentally, VM-based approaches create distribution challenges. Large VM images are difficult to share, version, or update, and ensuring that all students have identical environments across different host operating systems (Windows, macOS, Linux) remains problematic. This reproducibility challenge affects the validity of assessments and creates inconsistent learning experiences [31].

While existing research highlights the potential of Docker for cybersecurity education, most implementations emphasize complex multi-container architectures, cloud deployment requirements, or advanced orchestration tools. Although powerful, these approaches introduce a level of complexity that may be unnecessary for many educational scenarios and can create barriers similar to those they seek to overcome. Research on simple, single-container exercises that students can run directly on personal laptops - without relying on cloud infrastructure, complex networking configurations, or orchestration frameworks - remains limited. Moreover, empirical evidence regarding student learning outcomes and

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

satisfaction with truly portable, self-contained Docker-based security exercises is scarce in the literature.

3 Solution Overview and Architecture

Our approach centers on delivering cybersecurity exercises as self-contained Docker containers that students can deploy on their personal laptops with minimal technical overhead. Unlike complex multi-container architectures requiring orchestration tools or cloud infrastructure, each exercise is designed for maximum simplicity and accessibility. The typical workflow for students consists of straightforward steps: pull the exercise container(s), run the container(s), access the vulnerable application via web browser or specified port.

For most exercises, a single container suffices, packaging the vulnerable web application with all necessary dependencies. However, some exercises require multi-container configurations deployed using Docker Compose. These scenarios typically involve applications requiring separate database services, microservice architectures where vulnerabilities span multiple components, or exercises demonstrating inter-service attacks. The Docker compose files were included with each exercise, maintaining the same simplicity principle - students need only a single command to deploy complete, multi-service vulnerable environments. This approach preserves accessibility while enabling more realistic application architectures. For example, SSRF exercises benefited from multi-container setups where students could observe attacks targeting internal services only accessible from within the Docker network, simulating real enterprise environments with segmented network architectures. This simplicity is intentional. By reducing deployment friction, students can focus cognitive effort on understanding and exploiting vulnerabilities rather than troubleshooting infrastructure issues. The containers include all necessary dependencies, database configurations, and application code, ensuring identical behavior across diverse student hardware and operating systems.

A distinguishing characteristic of our approach is the emphasis on testing vulnerabilities in real-world applications with documented Common Vulnerabilities and Exposures (CVE) [44] identifiers rather than artificially constructed vulnerable code. While many educational frameworks create deliberately flawed toy applications for demonstration purposes, we selected actual production-grade software packages that contained known security issues discovered and disclosed through responsible vulnerability research. This approach differs fundamentally from purpose-built vulnerable applications like DVWA (Damn Vulnerable Web Application) [61], WebGoat [53] or JuiceShop [38], which are designed exclusively for training with intentionally obvious security flaws. Instead, our students encountered vulnerabilities as they naturally occurred in legitimate software projects - buried within complex codebases, protected by some security measures, and requiring careful analysis to identify and exploit.

This design decision serves several critical pedagogical objectives. First, students gain experience with the realistic process of vulnerability discovery. Rather

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

than being told "this application has SQL injection," students must analyze application behavior, intercept HTTP traffic, identify potential injection points, and systematically test payload - mirroring real penetration testing methodology. Second, working with production-grade applications exposes students to modern web technology stacks including REST APIs, authentication systems, database ORMs, and templating engines. Students must understand these architectural components before identifying how vulnerabilities compromise them. Third, exploiting real CVEs demonstrates that security flaws affect professionally developed software maintained by experienced developers, not just hastily written example code. This realization helps students appreciate that secure development requires systematic practices, code review, and ongoing vigilance rather than simply avoiding obvious mistakes.

Throughout the course, we integrated systematic exposure to CVE identifiers and Common Weakness Enumeration (CWE) [45] classifications. Each exercise was mapped to relevant CWE categories, helping students understand vulnerability taxonomies used in professional security practice. Students researched corresponding CVE entries in the National Vulnerability Database (NVD) [49] for each real-world application vulnerability they exploited. This integration exposed students to vulnerability disclosure practices, severity scoring systems (CVSS), patch timelines, and the importance of tracking security advisories. Students learned to interpret CVE descriptions, understand affected software versions and configurations, recognize attack prerequisites and impact ratings, and appreciate how vulnerabilities are discovered, reported, and remediated in practice.

4 Exercise Categories and Coverage

Our containerized exercise collection spans multiple dimensions of web application security, aligned with the OWASP Top 10 2025 framework [52] - the industry-standard taxonomy of the most critical web application security risks. This alignment ensures students gain exposure to vulnerabilities they will encounter in professional security assessments and that organizations prioritize in their security programs. The OWASP Top 10 provides a consensus-driven ranking based on vulnerability prevalence, exploitability, and business impact, making it an ideal curricular framework for comprehensive security education.

A critical distinction of our approach is the use of real, widely-deployed production applications rather than purpose-built training platforms. Students worked with containerized versions of industry-standard software including Apache HTTP Server, Jenkins automation server, WordPress CMS, Cockpit content platform, LimeSurvey, Sonatype Nexus Repository, and numerous other production systems that power millions of websites worldwide. These applications represent the actual technology stack students will encounter in professional environments, providing authentic context for vulnerability assessment and exploitation.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

Each exercise category corresponds to critical vulnerability classes identified in the OWASP Top 10:2025, ensuring alignment with industry-recognized security standards and preparing students for real-world security challenges. Example exercises from the course can be found at [41].

4.1 Authentication and Authorization Flaws

This category addresses OWASP A01:2025 - Broken Access Control and A07:2025 - Authentication Failures, which remain among the most prevalent security risks in web applications. JSON Web Token (JWT) [21] security exercises formed a comprehensive module covering multiple attack vectors against authentication systems. Students began with foundational exercises understanding JWT structure, encoding/decoding, and signature verification mechanisms using Python's PyJWT library [54]. Progressive exercises introduced authentication bypass techniques including algorithm confusion attacks, weak signing key exploitation through dictionary attacks using Hashcat [32] and John the Ripper [51], and token tampering without signature verification.

Real-world CVE-based exercises included exploiting Watcharr (CVE-2024-50634) [15], where students performed privilege escalation by forging JWT tokens with weak cryptographic secrets, and Memos (CVE-2023-4696) [11], demonstrating how inadequate JWT verification enables complete authentication system compromise. Students also exploited Go LDAP Admin (CWE-798) [48], cracking hardcoded JWT signing secrets to escalate privileges from regular user to administrator. The golang-jwt (CVE-2025-30204) [49] exercise taught Denial of Service attacks through malicious JWT parsing, demonstrating security implications beyond authentication bypass.

4.2 Injection Attacks

This category of exercises addressed OWASP A05:2025 - Injection. Cross-site scripting (XSS) exercises covered reflected, stored, and DOM-based variants across multiple production applications. The exercise progression built from basic proof-of-concept payloads to complete attack scenarios involving credential theft and account compromise through a three-phase exploitation methodology: first, confirming JavaScript execution; second, exfiltrating stolen credentials (cookies and local storage) to attacker-controlled HTTP servers in separate Docker containers; third, weaponizing stolen credentials via cURL [57] to perform privileged operations such as creating administrator accounts or changing victim passwords. Advanced exercises demonstrated stored XSS through file upload mechanisms - particularly dangerous because malicious payloads persist in application storage and execute repeatedly across victim sessions.

The Microweber (CVE-2022-0963) [5] exercise proved instructive: students discovered the upload module blocked JavaScript file extensions but accepted SVG and XML files without content inspection. Students crafted malicious SVG images embedding JavaScript which, when accessed by administrators, stole session cookies and local storage contents, exfiltrating them to collection servers.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

Using captured tokens, students performed authenticated API requests via cURL to create administrator accounts, achieving complete account takeover. Similar patterns were demonstrated in Memos (CVE-2022-4690) [8], FlatPress (CVE-2022-4605) [7], and LibreNMS (CVE-2023-5060) [12], showing that diverse application types - from blogging platforms to network monitoring systems - contain exploitable file upload XSS vulnerabilities.

4.3 Architectural vulnerabilities

This category addresses OWASP A06:2025 - Insecure Design, which represents fundamental architectural flaws that enable attacks even when implementation is technically correct. Server-Side Request Forgery (SSRF) exercises demonstrated how insufficient validation of user-supplied URLs allows attackers to abuse server functionality to access internal resources, scan internal networks, and interact with services protected by network segmentation.

Real-world exploitation covered diverse application contexts across multiple technology stacks - photo management tools, customer relationship management platforms, documentation systems, and note-taking applications. The exercise progression built from basic URL manipulation to complete attack chains involving internal network reconnaissance, service exploitation, and credential attacks through a multi-phase exploitation methodology: first, confirming SSRF through external resource fetching; second, performing internal network reconnaissance through port scanning and service identification; third, weaponizing discovered services through credential attacks and privilege escalation.

The BookStack (CVE-2023-4624) [10] exercise proved particularly instructive: students exploited webhook functionality in this documentation platform to perform systematic internal port scanning and identifying services listening within Docker's isolated network. After discovering open ports through response timing and status code analysis, students manually connected to discovered services and examined service banners, identifying specific applications running internally. The next step involved performing automated credential attacks on discovered services using Hydra [1] with common username and password wordlists. Using compromised credentials, students escalated access to sensitive internal systems that were completely unreachable from external networks, demonstrating how SSRF serves as an initial foothold for deeper exploitation. The BookStack exercise also included evaluating Denial of Service attack potential, where students systematically sent repeated requests to observe resource exhaustion and service degradation. This taught students that SSRF vulnerabilities threaten system availability beyond traditional confidentiality and integrity concerns. Students analyzed server behavior under overload conditions and discussed appropriate mitigation strategies. This comprehensive exploitation scenario - combining reconnaissance, service discovery, credential compromise, and availability impact - provided realistic experience with how single vulnerabilities enable multi-faceted attacks across different security dimensions.

Similar exploitation patterns were demonstrated in Lychee (CVE-2025-53018) [19], SuiteCRM (CVE-2023-6124) [13] and Memos (CVE-2025-22952), [16].

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

4.4 Software and Data Integrity Failures

This category addresses OWASP A08:2025 - Software and Data Integrity Failures, which encompasses vulnerabilities where code execution or critical data manipulation occurs without adequate integrity verification [52]. Remote Code Execution (RCE) exercises demonstrated the most severe vulnerability class where attackers achieve arbitrary command execution on target systems, representing complete security compromise.

Students began with foundational CWE-434 [47] arbitrary file upload exercises where vulnerable HTTP servers accepted user-uploaded files without proper validation. Initial exercises involved uploading PHP web shell scripts accepting command parameters and executing them via server-side functions. The exercise progression introduced increasingly sophisticated defensive mechanisms requiring bypass techniques. Students learned to manipulate HTTP Content-Type headers to bypass MIME type validation, exploit extension filtering weaknesses through double extensions or case sensitivity issues, and craft payloads that appeared benign while containing executable code. This progression taught students that effective file upload security requires multiple layers of validation including magic byte inspection, sandboxed execution environments, and strict allowlisting rather than denylisting approaches. Real-world CVE-based exercises demonstrated RCE across diverse contexts and attack vectors. Apache HTTP Server (CVE-2021-42013), [4] demonstrated path traversal leading to RCE through CGI-bin exploitation, where students crafted HTTP requests combining directory traversal with access to system binaries, enabling command execution. Students used multiple tools and approaches - manual HTTP request construction for protocol understanding and scripted exploitation for automated attack sequences. The exercise included a reconnaissance component where students researched vulnerability prevalence using internet device search engines, discovering thousands of unpatched Apache installations worldwide - emphasizing real-world impact while ethically restricting actual exploitation attempts. This demonstrated that even widely-deployed, security-conscious projects like Apache occasionally release versions with critical vulnerabilities affecting millions of servers. WordPress (CVE-2022-1103) [6] demonstrated authenticated RCE through upload plugin vulnerability. Students deployed complete WordPress installations using Docker Compose with MySQL database, installed vulnerable plugin and exploited insufficient file type validation to upload PHP web shells despite authenticated context. This exercise taught that plugins and extensions frequently introduce vulnerabilities even in hardened core applications, and that authentication does not eliminate RCE risk - low-privilege authenticated users can achieve complete server compromise when vulnerable components exist. The widespread use of WordPress across corporate websites, government portals, and e-commerce platforms makes such vulnerabilities particularly concerning from an organizational security perspective. The Jenkins (CVE-2019-1003000), [3] exercise introduced DevOps infrastructure attacks through Groovy script sandbox bypass. Students established reverse shell connections by pasting Groovy payloads into Jenkins's Script Console that spawned bash shells con-

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

necting back to attacker-controlled listeners. This technique proved particularly valuable because reverse shells succeed even in restricted network environments where firewalls block inbound connections. Students also analyzed the organizational impact of Jenkins compromise. These systems typically have access to source code repositories, cloud deployment credentials, artifact repositories, production infrastructure access, and sensitive environment variables. Compromising Jenkins often provides attackers with capabilities to inject malicious code into software builds (supply chain attacks), steal intellectual property, or establish persistent backdoors across entire development pipelines. This contextualized RCE vulnerabilities within broader software supply chain security concerns, demonstrating why DevOps infrastructure represents high-value targets for sophisticated attackers. The Cockpit (CVE-2023-4195) [9] exercise demonstrated multi-container attack scenario, where initial RCE enabled lateral movement to additional systems. Students uploaded PHP web shell and executed reconnaissance commands to discover internal network topology. Next, they harvested SSH keys and successfully pivoted to SSH service running in separate container. This realistic scenario taught students that RCE vulnerabilities serve as initial footholds enabling deeper infrastructure penetration through credential harvesting, network reconnaissance, and lateral movement techniques commonly observed in advanced persistent threat (APT) campaigns.

Path traversal exercises demonstrated directory traversal attacks exploiting insufficient input sanitization. Students began with simple exercises reading sensitive files, understanding how relative path sequences allow attackers to escape directory restrictions and access arbitrary filesystem locations. Progressive exercises introduced payload encoding, filter bypassing, and multi-stage exploitation across diverse application types: file managers (TinyFileManager, CWE-22 [46]), frontend build tools (Vite, CVE-2025-30208 [17]), photo management systems (Lychee, CVE-2025-50202 [18]), content platforms (Cockpit, CVE-2018-15540 [2]), artifact repositories (Sonatype Nexus Repository, CVE-2024-4956 [14]) and web servers (Apache HTTP Server, CVE-2021-42013 [4]). Students learned that path traversal impact extends beyond reading configuration files to authentication credentials, source code disclosure, cryptographic materials (i.e. SSH private keys) and log files. The Sonatype Nexus (CVE-2024-4956 [14]) exercise demonstrated complete attack chain progression. Students used automated tools with path traversal wordlists to systematically identify working payloads, extracted system files to enumerate accounts, then pivoted to harvesting cryptographic credentials from service account directories. They authenticated to service running in separate Docker container, simulating lateral movement in enterprise environments. This multi-stage sequence provided authentic experience with chaining vulnerabilities for deeper system penetration.

5 Student Feedback and Evaluation

To evaluate the effectiveness of our Docker-based approach, we conducted a comprehensive survey of students who completed the *Cybersecurity and Cryp-*

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

tography course. The survey assessed multiple dimensions including learning outcomes, practical applicability, technical accessibility, and overall satisfaction with the containerized exercise approach. Survey responses were collected using a 5-point Likert scale ranging from "Strongly Disagree" to "Strongly Agree" for quantitative assessment, supplemented by open-ended questions for qualitative feedback.

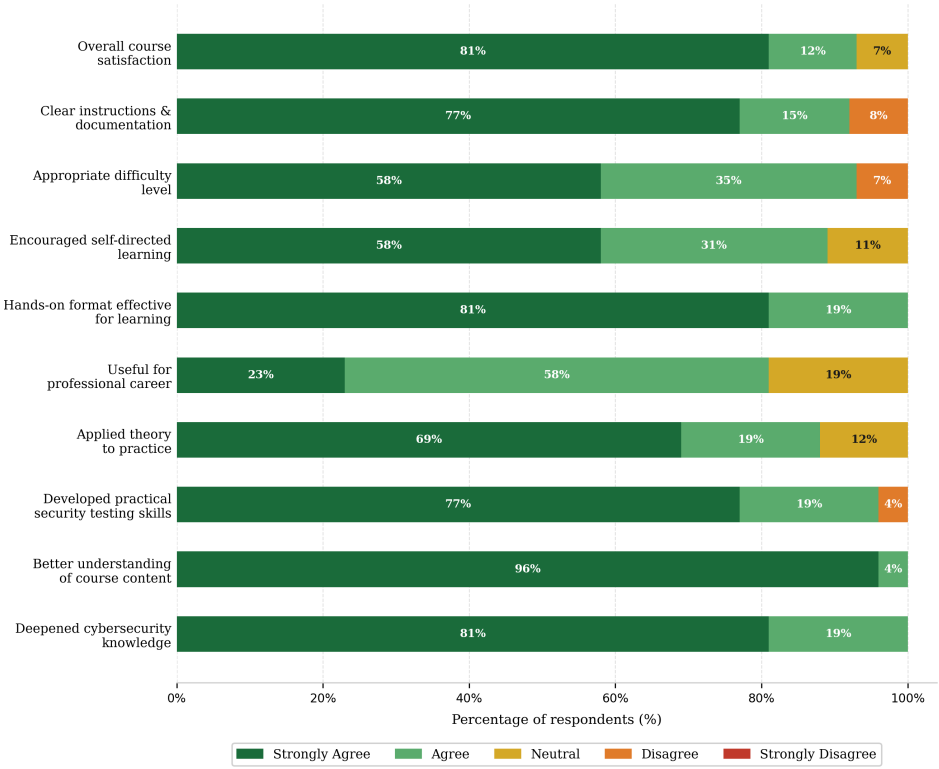

Fig. 1. Survey results

Student responses indicated exceptionally strong positive outcomes regarding skill acquisition and practical learning. All students (100%) agreed that Docker-based exercises helped them deepen their knowledge in cybersecurity, with 81% strongly agreeing and 19% agreeing. This unanimous positive response suggests that containerized exercises effectively conveyed security concepts through hands-on exploitation.

The exercises' contribution to better understanding of course content received perfect agreement, with 100% of students responding positively (96% strongly agree, 4% agree). This demonstrates strong alignment between theoretical lectures and practical laboratory work. Similarly, 96% of students agreed

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

that exercises met their expectations for practical application security learning (92% strongly agree, 4% agree), with only 4% disagreement, validating the pedagogical approach.

Regarding practical skill development, 96% of students agreed that exercises developed their practical security testing skills (77% strongly agree, 19% agree). Students reported that exercises enabled application of theoretical knowledge to practice, with 88% positive responses (69% strongly agree, 19% agree, 12% neutral). This high agreement rate confirms successful integration of theory and practice.

When asked about career relevance, 81% of students agreed that acquired knowledge would be useful in their professional careers (23% strongly agree, 58% agree), with 19% neutral. This perception of professional applicability validates our approach of using real CVEs affecting production applications rather than synthetic training environments.

The quality and design of exercises received overwhelmingly positive feedback. Students rated the overall quality of materials and laboratory exercises highly, with 96% positive responses (77% strongly agree, 19% agree). The practical laboratory format using Docker and attack scenarios was considered interesting by 93% of students (81% strongly agree, 12% agree), validating the containerized approach.

Regarding content interest, 96% of students found exercises interesting from a substantive perspective (77% strongly agree, 19% agree). The exercises increased student engagement in the subject for 96% of respondents (73% strongly agree, 23% agree). Furthermore, 89% of students reported that exercises encouraged independent knowledge deepening and tool exploration (58% strongly agree, 31% agree, 12% neutral), indicating the exercises successfully promoted self-directed learning beyond classroom requirements.

The hands-on format (labs, exploits, vulnerability analysis) was considered effective for learning by 100% of students (81% strongly agree, 19% agree). This perfect positive response rate demonstrates that the practical, exploitation-focused approach resonated strongly with students' learning preferences.

Regarding appropriate challenge level, 93% agreed that exercise difficulty was appropriate for their skill levels (58% strongly agree, 35% agree). The instructions and documentation were clear and understandable for 92% of students (77% strongly agree, 15% agree). These results suggest successful differentiated instruction through progressive hints and scaffolding.

Overall assessment was unanimously positive, with 100% of students providing favorable ratings (88% "definitely positive", 12% "rather positive"). This perfect satisfaction rate, combined with 93% satisfaction with course realization and proposed exercises (81% strongly agree, 12% agree, 7% neutral), demonstrates exceptional student approval of the Docker-based approach.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

6 Conclusions

This paper presented and evaluated a Docker-based approach for delivering hands-on cybersecurity education focused on web application security vulnerabilities. Our deployment in a university *Cybersecurity and Cryptography* course demonstrates that container-based exercises successfully provide effective practical training while eliminating infrastructure barriers associated with traditional lab environments.

The Docker-based methodology offers significant advantages: lightweight containers enable rapid deployment, perfect reproducibility across diverse student hardware ensures consistent learning experiences, and portable exercises enable anytime, anywhere learning without institutional network requirements. Our emphasis on real-world applications with documented CVEs rather than purpose-built vulnerable applications proved pedagogically valuable. Students demonstrated high engagement when exploiting vulnerabilities affecting production systems and reported increased motivation from developing directly applicable professional skills. Survey results validate the approach's effectiveness: 100% of students provided positive overall assessment (88% "definitely positive", 12% "rather positive"), 100% agreed that exercises deepened their cybersecurity knowledge, and 100% agreed that exercises contributed to better understanding of course content.

The approach addresses critical accessibility challenges in cybersecurity education. By eliminating expensive cloud infrastructure, dedicated lab servers, and complex networking configurations, the methodology makes hands-on security training accessible to institutions with limited resources. The version-controlled nature of Docker images enables straightforward distribution and updates through container registries.

References

1. Thc-hydra password cracker. https://github.com/vanhauser-thc/thc-hydra
2. CVE-2018-15540: Path traversal in cockpit cms. https://nvd.nist.gov/vuln/detail/CVE-2018-15540
3. CVE-2019-1003000: Jenkins sandbox bypass. https://nvd.nist.gov/vuln/detail/CVE-2019-1003000
4. CVE-2021-42013: Path traversal and RCE in apache http server. https://nvd.nist.gov/vuln/detail/CVE-2021-42013
5. CVE-2022-0963: XML upload leads to XSS. https://nvd.nist.gov/vuln/detail/CVE-2022-0963
6. CVE-2022-1103: File upload leads to RCE. https://nvd.nist.gov/vuln/detail/CVE-2022-1103
7. CVE-2022-4605: Stored XSS in flatpress. https://nvd.nist.gov/vuln/detail/CVE-2022-4605
8. CVE-2022-4690: Stored XSS in memos. https://nvd.nist.gov/vuln/detail/CVE-2022-4690
9. CVE-2023-4195: File upload leads to rfi in cockpit cms. https://nvd.nist.gov/vuln/detail/CVE-2023-4195

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

10. CVE-2023-4624: SSRF in bookstack. https://nvd.nist.gov/vuln/detail/CVE-2023-4624
11. CVE-2023-4696: JWT forgery leads to improper access control. https://nvd.nist.gov/vuln/detail/CVE-2023-4696
12. CVE-2023-5060: Dom XSS in librenms. https://nvd.nist.gov/vuln/detail/CVE-2023-5060
13. CVE-2023-6124: SSRF in suitecrm. https://nvd.nist.gov/vuln/detail/CVE-2023-6124
14. CVE-2024-4956: Path traversal in nexus repository. https://nvd.nist.gov/vuln/detail/CVE-2024-4956
15. CVE-2024-50634: Weak JWT allows privilege escalation. https://nvd.nist.gov/vuln/detail/CVE-2024-50634
16. CVE-2025-22952: SSRF in memos. https://nvd.nist.gov/vuln/detail/CVE-2025-22952
17. CVE-2025-30208: Arbitrary file read in vite. https://nvd.nist.gov/vuln/detail/CVE-2025-30208
18. CVE-2025-50202: Path traversal in lychee. https://nvd.nist.gov/vuln/detail/CVE-2025-50202
19. CVE-2025-53018: SSRF in lychee. https://nvd.nist.gov/vuln/detail/CVE-2025-53018
20. AlSalamah, A.K.e.a.: Virtualization in cybersecurity education. In: ISECON (2018)
21. Auth0: Introduction to json web tokens. https://jwt.io/introduction
22. Burton, S.L.: Framework for cybersecurity labs. Laboratories
23. Chouliaras, N.e.a.: Autonomous container-based training platform. PLOS ONE (2023)
24. Copeland, M.e.a.: Pwn lessons with docker. In: SIGCSE (2024)
25. CrowdStrike: 2025 global threat report. https://www.crowdstrike.com/en-us/global-threat-report/ (2025)
26. Du, W.: Computer Security: A Hands-on Approach (2017)
27. Du, W.: Seed labs 2.0 (2021)
28. Du, W.e.a.: Seed instructional labs. In: SIGCSE (2007)
29. EdTech Magazine: Cyber ranges for cybersecurity careers. https://edtechmagazine.com/higher/article/2025/08/cyber-ranges-prepare-students-cybersecurity-careers (2025)
30. Fortinet: 2025 cybersecurity skills gap report. https://www.fortinet.com/content/dam/fortinet/assets/reports/2025-cybersecurity-skills-gap-report.pdf (2025)
31. Garfinkel, T., Rosenblum, M.: Security challenges in virtual machines (2005)
32. hashcat: Hashcat password cracker. https://github.com/hashcat/hashcat
33. Hassan, I.: Secure online labs with docker and lxd (2022)
34. IBM: Cost of a data breach report 2024. https://www.ibm.com/think/insights/cybersecurity-skills-gap-contributed-increase-average-breach-costs (2024)
35. Irvine, C.E.e.a.: Labtainers: Docker-based labs. In: USENIX ASE (2017)
36. Irvine, C.E.e.a.: Parameterized cybersecurity labs. In: National Security Summit (2017)
37. Karagiannis, S.e.a.: Sandboxing cyberspace for education (2020)
38. Kimminich, B.e.a.: Owasp juice shop. https://owasp.org/www-project-juice-shop/
39. Li, P.e.a.: Virtual lab automation in it education (2011)
40. Li, P.e.a.: Comparison of virtual cybersecurity labs. Tech. rep. (2012)

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37

41. Mazur, K.: iccs_2026. https://github.com/katarzynamazur/iccs_2026 (2026), accompanying exercises for a paper submitted to the 26th International Conference on Computational Science (ICCS 2026), Hamburg, Germany, 29 June – 1 July 2026
42. Merkel, D.: Docker: Lightweight containers. Linux Journal
43. Mirkovic, J., Benzel, T.: Teaching cybersecurity with deterlab. IEEE Security & Privacy (2012)
44. MITRE: Common vulnerabilities and exposures (CVE). https://cve.mitre.org/
45. MITRE: Common weakness enumeration (CWE). https://cwe.mitre.org/
46. MITRE: CWE-22: Path traversal vulnerability. https://cwe.mitre.org/data/definitions/22.html
47. MITRE: CWE-434: Unrestricted file upload vulnerability. https://cwe.mitre.org/data/definitions/434.html
48. MITRE: CWE-798: Hard-coded credentials vulnerability. https://cwe.mitre.org/data/definitions/798.html
49. NIST: National vulnerability database. https://nvd.nist.gov/
50. O'Connor, T.J.: Pwn lessons made easy with docker: Toward an undergraduate vulnerability research cybersecurity class. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education, V. 1 (2024)
51. Openwall: John the ripper. https://github.com/openwall/john
52. OWASP Foundation: Owasp top 10:2025. Tech. rep.
53. OWASP WebGoat Project: Webgoat. https://owasp.org/www-project-webgoat/
54. Padilla, J.: PyJWT Documentation, https://pyjwt.readthedocs.io/
55. SEED Labs Project: Seed security labs. https://seedsecuritylabs.org/ (2024)
56. Srivatanakul, T., Annansingh, F.: Active learning in web security courses. Journal of Computers in Education (2022)
57. Stenberg, D.: curl. https://curl.se/
58. Thompson, M.F., Irvine, C.E.: Individualizing cybersecurity labs. IEEE Security & Privacy
59. Tobarra, L.e.a.: Container-based virtual labs analysis. In: TEEM. pp. 245–251 (2019)
60. Weiss, R.e.a.: Cybersecurity education in edurange. IEEE Security & Privacy (2017)
61. Wood, R.: Damn vulnerable web application (dvwa). https://github.com/digininja/DVWA

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_37