ATHENA CTF: A Modular Framework for Instructional Capture-the-Flag Challenges

Anonymized for submission.

Abstract

Capture The Flag (CTF) challenges are a widely used mechanism for hands-on cybersecurity training, but creating, deploying, and assessing custom challenges remains complex and time-consuming. Existing platforms often rely on static challenge repositories, heavyweight infrastructure, or problems for which solutions have been publicly shared, limiting their suitability for scalable instruction and formal assessment. We present ATHENA CTF, an open source modular framework for the rapid creation, containerization, and deployment of web-based CTF challenges. ATHENA CTF enables challenge authors with modest programming experience to implement fully functional levels in as few as ten lines of code, while providing a framework wide standardized execution environment through containerization. The framework supports per-user and per-team solution parametrization, reducing answer sharing and enabling assessment-ready deployments alongside static configurations for demonstrations and collaborative exercises. ATHENA CTF optionally integrates a centralized database to support learner tracking, LMS-based user provisioning, and automated result export. To assist learners without disclosing full solutions, the framework includes optional, context-constrained LLM-assisted hints generated from limited interaction history and creator-authored solution process. This design balances instructional support with safeguards against solution leakage and misuse. Together, these features allow instructors and organizations to rapidly prototype, distribute, and assess CTF challenges while maintaining control over deployment, security, and learner interaction.

1 Introduction

In recent years, both the volume and sophistication of cyber incidents have increased substantially [11]. In 2024, Mitre, the organization responsible for the CVE reporting framework, published a record 40,009 vulnerabilities, representing a 38% increase over 2023 [14]. These vulnerabilities have contributed to large-scale data breaches and financial harm, including an estimated $638 million in losses to Canadian citizens in 2024 alone [20, 23]. Together, these trends underscore a persistent challenge: as software systems evolve rapidly, secure development practices often struggle to keep pace [19].

Addressing this gap requires developers to understand not only how vulnerabilities are mitigated, but also how they are identified and exploited. A challenge for this form of education is the pace of technological change, including recent advances in AI-assisted development [6], which necessitate continual updates to security knowledge. Capture-the-flag (CTF) exercises are a widely used approach for delivering hands-on security training, allowing learners to engage directly with realistic vulnerability scenarios [3]. CTFs encourage adversarial thinking and exploratory learning, and can be scaled to accommodate a range of skill levels [15]. However, supporting scalable, up-to-date, and assessment-ready CTF deployments remains a practical challenge.

Large language models (LLMs) have recently demonstrated promise in supporting both self-directed and instructor-assisted learning when provided with appropriate context [10]. Prior work has explored their application in cybersecurity, including exploit detection [1] and educational support [2, 17]. At the same time, the use of LLMs in instructional settings raises concerns around solution leakage, over-scaffolding, and misuse if not carefully constrained.

With this context, we present ATHENA CTF, a containerized framework for authoring, deploying, and assessing web-based CTF challenges. ATHENA CTF uses Docker to provide portable and reproducible challenge environments and offers a modular abstraction for level creation that enables rapid prototyping and sharing. Challenges can be deployed locally or online and configured for instructional, collaborative, or assessment-oriented use.

The framework optionally integrates a centralized database to support learner tracking, LMS-based user provisioning, and result export, while allowing fully standalone deployments when persistence is not required. To support learner progress without disclosing full solutions, ATHENA CTF includes op-

tional, context-driven LLM-assisted hints. These hints are generated using limited interaction history and creator-authored solution paths, and are intentionally constrained to reduce the risk of solution leakage and misuse.

ATHENA CTF also supports deterministic per-user or per-team parametrization, enabling personalized solutions that reduce trivial answer sharing in formal assessments. Instructors may disable parametrization or LLM-assisted hints to support collaborative activities or demonstrations. To illustrate the framework's expressiveness, we provide a collection of sample challenges spanning introductory to moderate-difficulty web security tasks, including HTTP manipulation and authentication vulnerabilities.

In summary, this paper makes three contributions: (1) a lightweight, modular framework for authoring and deploying web-based CTF challenges with minimal setup overhead; (2) assessment-oriented mechanisms, including deterministic per-user parametrization and optional centralized tracking, designed to reduce answer sharing and support formal evaluation; and (3) a constrained approach to LLM-assisted hint generation that balances learner support with safeguards against solution leakage.

2 Related Work

This section reviews related work in security education and compares existing solutions with the design goals and affordances of our framework. We group prior work into three categories: deployment frameworks, capture the flag (CTF) learning platforms, and wargame platforms.

2.1 Deployment Frameworks

CTF deployment frameworks generally address two complementary needs: (1) a front-end interface that aggregates challenge links and manages scoring, and (2) a back-end mechanism for deploying interactive levels.

Front-end competition platforms such as CTFd [5] and RootTheBox [13] provide dashboards listing available challenges and award points when users submit correct flags. RootTheBox uses static flags, whereas CTFd supports dynamic flags, but generating them requires paid hosting. In both cases, the delivery of the challenge itself is external; a separate deployment mechanism is needed whenever challenges are not purely static files.

Back-end deployment frameworks such as kCTF [9] and EDURange [22, 24] automate the provisioning of challenges using Kubernetes. To deploy a level, authors must containerize their challenge logic and provide a fully configured Dockerfile. Although powerful, this introduces substantial complexity for instructors or students who lack systems-engineering experience. In contrast, our single-line deployment framework removes the need for custom Dockerfiles, substantially lowering the barrier to creating and publishing web-based challenges.

Other systems, such as Labtainers [12], provide a custom virtual machine (VM) environment encompassing a broad spectrum of security exercises, including networking and systems challenges. Students complete activities within the VM, and instructors later grade exported artifacts. While Labtainers supports a wide range of challenge types, creating new scenarios or deploying lightweight web-based challenges is comparatively cumbersome. Moreover, because Labtainers does not maintain a database, instructors cannot track live user progress or provide immediate, context-aware feedback; capabilities that our framework is explicitly designed to support.

2.2 Capture The Flag Learning Platforms

Self-directed learning platforms offer persistent CTF-style challenges, typically enhanced with gamification such as points and leaderboards. Unlike ephemeral competitions, user progress is stored over time and challenges remain publicly accessible.

picoCTF [4] is one of the most widely used platforms, providing challenges ranging from web exploitation to cryptography. Its large library supports skill development, but public availability makes it unsuitable for graded coursework: solutions and walkthroughs are easy to find, and users must create external accounts. Its hint system is static and not tailored to learner context, and flags are not personalized, increasing the likelihood of answer-sharing.

The pwn.college DOJO platform [17, 18] offers an interactive in-browser environment for each challenge, including options for a VSCode interface, a full VM, or a shell. This reduces setup friction but introduces reliance on strong network connectivity and can incur slowdowns from VM overhead. The browser-based environment also limits students working on web-security challenges, who benefit from their personal browsers and extensions. DOJO allows users to host their own instance and create custom challenges, but, similar to Labtainers, authors must fully implement each level, which remains complex for simple web challenges. DOJO also includes LLM-powered context-aware hints, though this approach incurs higher computational cost and a greater risk of prompt injection relative to our lightweight hinting mechanism.

SEED Labs [7] provide structured, self-directed security labs accessible through a custom VM. While effective for teaching Linux-based exploitation, the VM requirement makes even introductory web challenges overly complex for beginners and similarly restricts users to a non-native environment.

2.3 Wargames Platforms

Wargame platforms offer collections of static challenges, often binaries, packet captures, or SSH-based exercises, with minimal scaffolding or instructor feedback. In web security, commonly used platforms include bWAPP [21], DVWA [25], and OWASP Juice Shop [8]. These tools expose learners to standard vulnerabilities, but updates are infrequent and public availability results in widespread write-ups, making them unsuitable for assessing student mastery in formal coursework.

BWapp and DVWA lack centralized data storage, making it difficult to track learner progress or manage multi-user classroom deployments. Although Juice Shop supports multi-instance deployment and progress tracking, extending it to include new challenges requires deep integration with the existing code base. Most instructors, therefore, lack the time or expertise needed to create custom levels, limiting its suitability as a flexible teaching tool.

3 ATHENA CTF Framework

In this section, we describe the design of our CTF framework and how it can be leveraged to create modular and distributable levels with context-enabled hints. Additionally, we will explore how the database implementation can be used for assignments and competitions.

3.1 Web-Based Architecture

ATHENA CTF is built as an entirely web-based interaction framework, allowing both challenge creators and participants to deploy and access levels through standard web browsers. Challenges can be distributed either as containerized services or as source repositories deployed directly on bare-metal systems. When containerization is used, challenge images can be shared through standard container registries such as Docker Hub and pulled for local or remote deployment. Alternatively, creators may distribute source repositories, enabling administrators to deploy challenges using provided spawning scripts without requiring container tooling.

The framework is designed to operate across Unix- and Windows-based systems, allowing administrators to select deployment workflows that align with their technical comfort and available infrastructure. In contrast to VM-based platforms, ATHENA CTF does not require learners to install additional software or work within preconfigured virtual environments. Once deployed, challenges may be hosted on a local network or exposed to the internet via a reverse proxy, supporting classroom use, workshops, and remote participation.

A web-based architecture was selected to maximize accessibility and realism for web-security tasks. Learners interact with challenges using their native browsers and tooling, enabling direct engagement with HTTP-level vulnerabilities and client-side behaviours without the overhead of managing virtual machines. This design choice also improves compatibility with constrained or low-resource environments, such as high-school or university computer labs, where installing custom software or virtualisation platforms may be impractical.

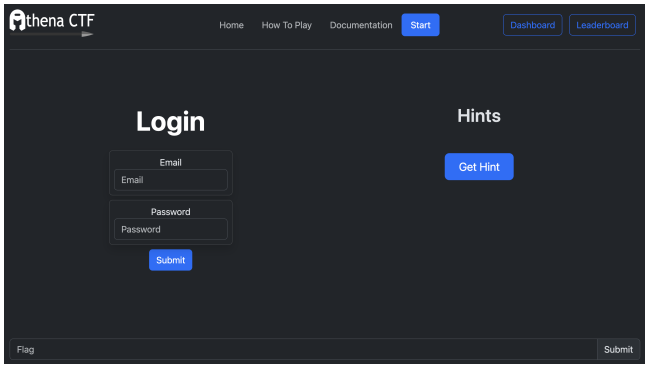

Figure 1: Example ATHENA CTF challenge requiring a participant to bypass a vulnerable login form using native browser interaction. The challenge illustrates the framework's web-based design, which allows learners to engage directly with HTTP-level and client-side vulnerabilities without requiring virtual machines or specialized tooling.

The web-based design further enables flexible configuration of user experiences at deployment time. Administrators may expose different interfaces or hinting mechanisms for the same challenge, such as enabling static hints for collaborative demonstrations or context-driven LLM-assisted hints for individual learners. These configurations can be isolated per user or deployment instance without modifying challenge logic, allowing instructors to adapt challenges to different instructional and assessment contexts. Figure 1 shows an example challenge in which users must bypass a vulnerable login form using browser-based interaction.

3.2 Centralized Database

ATHENA CTF optionally supports a centralized database to enable learner tracking, assessment workflows, and context-aware assistance. The database is not required for challenge execution and can be disabled entirely for standalone or demonstration deployments. When enabled, it provides a lightweight mechanism for recording user interaction data across challenges while minimizing storage overhead and configuration complexity.

We implement this functionality using a document-oriented NoSQL data model (MongoDB [16] JSON format), as each user's interaction history can be naturally represented as a single, compact record. This design simplifies data access for features such as progress tracking, hint generation, and result export, while avoiding the schema rigidity of relational databases. Administrators enable database support by provid-

ing credentials to an existing local or cloud-hosted instance and specifying a target collection; the framework automatically initializes required fields and indexes.

In practice, the storage requirements are minimal. In a demonstration deployment with 15 challenges, the average user record occupied approximately 115 bytes, and the index size was roughly 1 MB across approximately 11,000 users. This footprint allows the framework to scale to large classes or workshops using commodity infrastructure, including free-tier cloud database services.

Database support is required for advanced features such as LLM-assisted hints and automated assessment export, but remains optional to preserve flexibility and ease of deployment. By decoupling challenge execution from persistent storage, ATHENA CTF allows instructors to select configurations that best match their instructional, privacy, and infrastructure constraints.

3.3 AI-Assisted Hints

ATHENA CTF includes optional, context-aware LLM-assisted hints designed to support learner progress while minimizing the risk of solution disclosure and misuse. Rather than providing an open-ended conversational agent, the framework generates one-shot plain text hints in response to explicit user requests. This design intentionally constrains interaction with the LLM, reducing exposure to prompt injection attacks, uncontrolled dialogue, and unintended leakage of full solutions.

Hint generation is grounded in two sources of controlled context: the user's recent interaction history and a creator-authored solution path. User interaction data, including prior requests and failed attempts, is retrieved from the centralized database when enabled. The solution path, provided by the challenge author as either a structured write-up or executable script, is stored locally on the host system and loaded into memory at run time. This approach ensures that hints are aligned with the intended solving strategy while preventing the model from inventing alternative or unintended solution paths.

By combining limited historical context with a bounded, creator-defined reference, the LLM is guided to produce targeted guidance related to the user's current progress rather than complete answers. The non-conversational, stateless nature of hint requests further constrains the output space and enables predictable inference costs, making the approach suitable for classroom-scale deployments.

LLM-assisted hints are an optional feature and can be replaced with static hints or disabled entirely through configuration, allowing administrators to select appropriate levels of assistance for demonstrations, collaborative exercises, or graded assessments. The framework supports both commercial LLM APIs and locally hosted models, enabling deployments that align with available infrastructure, cost constraints, and data governance requirements.

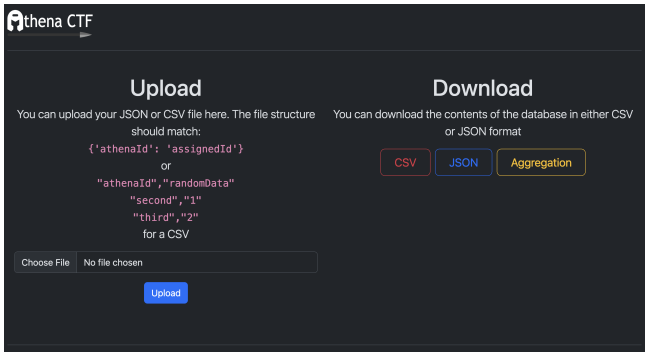

Figure 2: Administration interface used for assessment-oriented workflows in ATHENA CTF. Instructors can provision users via LMS-exported identifiers and export challenge results without storing personally identifying information, supporting separation between learner interaction data and institutional records.

3.4 Ease of Assessment

ATHENA CTF provides a dedicated administration interface, deployed as a separate container, that enables instructors to manage users and assess participant performance without modifying challenge logic. This separation allows assessment workflows to be configured independently of challenge deployment, supporting both instructional and evaluative use cases.

By default, users are assigned an anonymous identifier upon first access, which is stored locally as a browser cookie and used to track challenge progress when database support is enabled. For formal assessments, administrators may instead provision users explicitly by uploading a CSV or JSON file exported from a learning management system (LMS). User creation is driven solely by a designated identifier field (e.g., ATHENAID); no additional data from the import file is stored or persisted by the framework. This design ensures that assessment can be conducted without collecting or retaining personally identifying information.

The same identifier may be assigned to multiple participants to support team-based challenges, allowing progress and submissions to be recorded collectively. Throughout an assessment period, the framework records limited interaction metadata, including challenge completion status, submitted requests, and hint usage. At the conclusion of an assessment, administrators can export the collected data directly from the administration interface. These records can be joined with the original LMS export to support grading and review while maintaining separation between platform data and institutional records. Figure 2 shows the administration interface used for user provisioning and result export.

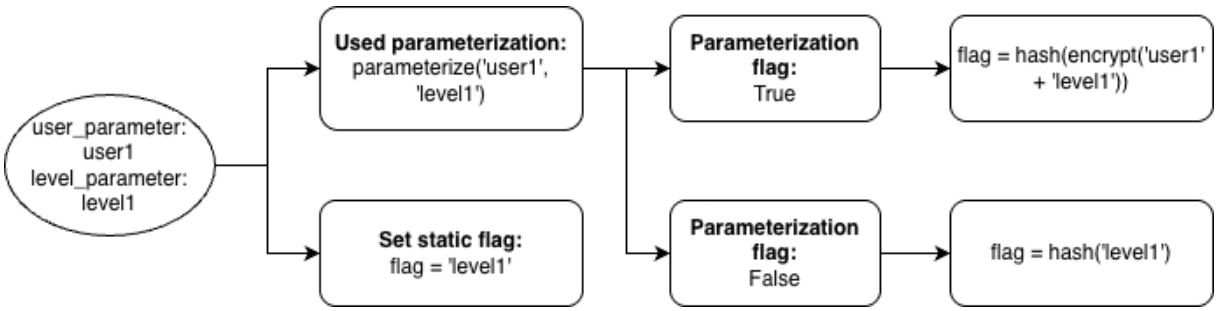

Figure 3: Verification flow for static and parametrized challenge solutions. When parametrization is enabled, user-specific identifiers are combined with a level-specific secret to deterministically generate unique solutions per participant, reducing trivial answer sharing while preserving reproducibility.

3.5 Parametrization

ATHENA CTF includes a parametrization mechanism that enables the generation of deterministic, user-specific solutions on a per-level basis. This design mitigates trivial answer sharing and supports fair assessment by ensuring that each participant must independently solve a challenge, even when working on identical challenge logic.

At level creation time, the framework generates a level-specific secret value. During verification, this secret is combined with a user-specific identifier—derived either from an automatically assigned browser cookie or an administrator-provisioned account —to compute a unique expected solution for that user. As a result, repeated attempts by the same user yield a consistent solution, while solutions shared across users or teams are invalid. This approach preserves reproducibility for individual learners while preventing straightforward collusion.

Parametrization is configurable on a per-deployment and per-level basis. Administrators may disable parametrization to use static solutions for in-class demonstrations or collaborative exercises, or selectively enable it for advanced challenges intended for individual assessment. To support this flexibility without increasing implementation complexity or introducing divergent code paths, the parametrization function is invoked during every flag verification request, regardless of whether parametrization is enabled. When disabled, the function deterministically produces a static, level-wide solution.

Figure 3 illustrates the verification flow used to compute and validate challenge solutions under both parametrized and static configurations.

3.6 Modular Design and Ease of Creation

ATHENA CTF adopts a modular design that enables rapid challenge creation while enforcing uniform execution and verification semantics. Each challenge level is defined as a self-contained page object, allowing authors to implement fully functional levels in as little as ten lines of code. Page objects are automatically registered with the framework at import time, enabling the application to maintain a consistent ordering and navigation structure across challenges.

Each page object exposes a small, well-defined interface that encapsulates all level-specific behaviour, including an instruction endpoint and a verification endpoint. To implement a basic level, authors provide instructional text and a verification function that returns a Boolean indicating completion. Verification functions may optionally return structured error codes and messages, allowing authors to control the verbosity of feedback without modifying client-side logic. This uniform interface ensures that all challenges follow the same execution and validation paths, reducing implementation errors and unintended bypasses.

To minimize front-end complexity, ATHENA CTF provides a library of server-side helper components for rendering common interface elements such as tables and accordions. These components generate HTML that is injected into a small set of shared templates using the Jinja2 templating engine. By default, all injected content is sanitized to prevent unsafe rendering; authors may explicitly disable sanitization only when providing pre-validated content. This safe-by-default design supports rapid prototyping while maintaining consistent styling and reducing the risk of unintended client-side vulnerabilities.

Because all levels rely on a shared page abstraction and a limited number of templates, the framework supports straightforward customization and white-label deployment without requiring changes to challenge logic. Figure 4 presents a high-level UML overview of the framework structure and its modular components.

3.7 Containerization

ATHENA CTF supports containerized challenge deployment using Docker to provide reproducible and portable execution environments. Containerization ensures that challenges behave consistently across heterogeneous host systems, reducing configuration-induced variability during deployment

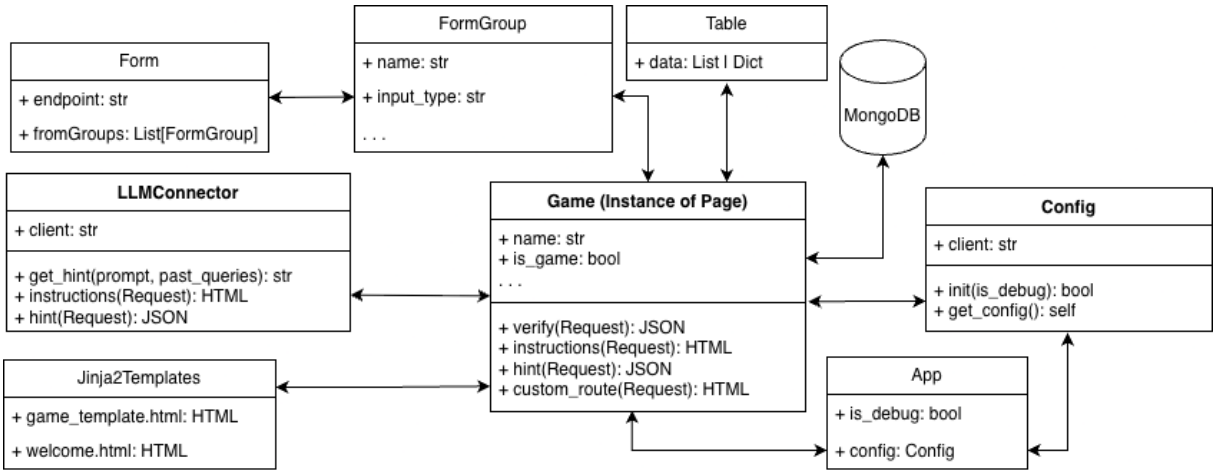

Figure 4: High-level architectural overview of ATHENA CTF, illustrating the modular page abstraction, shared application components, optional database integration, and constrained LLM connector. The design enforces uniform verification paths while allowing optional persistence and assistance features to be enabled per deployment.

and assessment and allowing challenge authors to share levels without requiring recipients to recreate dependencies manually.

Compared to virtual machine–based approaches, containerized challenges impose lower resource overhead while still supporting realistic web-security workflows. This design choice enables instructors to deploy challenges using commodity infrastructure and supports controlled execution without constraining learners to preconfigured environments.

Containerization also provides a foundation for future extensions. While the current framework supports shared-instance deployments, container orchestration platforms such as Kubernetes could be used to enable per-user or per-team isolation, supporting a broader class of challenges and stronger isolation guarantees in high-stakes assessment settings.

4 Formative Expert Feedback

In this section, we describe two expert feedback sessions conducted with teaching assistants and professors experienced in cybersecurity education. The goal was early design validation rather than coverage. These sessions were designed as a formative, informative evaluation to validate core design assumptions, assess the clarity and usability of the framework from a challenge-author perspective, and identify limitations or missing functionality that could inform future development. The goal of this process was not to measure learning outcomes or effectiveness, but to gather early feedback on the framework's practicality for classroom and instructional use.

4.1 Working Session & Tasks

Each expert feedback session consisted of three stages: (1) a brief presentation outlining the purpose, scope, and structure of the framework, (2) a line-by-line walkthrough of an existing challenge level, and (3) a guided pair-programming session in which the expert implemented and deployed a new level, asking questions as needed.

The selected tasks were designed to exercise a range of framework features. In particular, experts interacted with HTML and JavaScript templates, required functions such as `verify` and `instructions`, and several provided extensions, including tables and login forms. The initial presentation covered many of the framework's available features, allowing participants to identify areas of ambiguity, sources of confusion, and opportunities for additional functionality.

By the end of each pair-programming session, the expert had successfully completed and deployed a functional challenge level. This process enabled feedback from both a level-creation perspective, focused on API clarity and development workflow, and a solver's perspective, including observations on interface consistency and overall intuitiveness.

4.2 Observations & Design Implications

Overall, experts reported that the framework's core design was easy to understand and that its abstractions enabled rapid creation of simple challenge levels. Participants noted that the naming conventions and modular, widget-based design reduced boilerplate code and lowered the barrier to implementing new levels. The web-based interface was also described as straightforward to navigate.

Several limitations were identified that directly inform future work. A recurring point of feedback was the lack of

inline documentation within the framework. While external documentation was available through a GitHub wiki, experts indicated that embedded docstrings, particularly for extension functions such as table and form generation, would improve discoverability and ease of use during development. Addressing this gap is planned for future releases.

Another concern was the need to use components of the underlying FastAPI library directly for certain tasks, such as defining a `/login` endpoint to handle POST requests. While this flexibility enables advanced use cases, it introduces additional complexity for novice programmers. Future work will explore higher-level extensions that encapsulate common patterns, such as authentication workflows, to further abstract away low-level framework details.

These observations are intentionally formative. They do not constitute an evaluation of learner performance or pedagogical effectiveness; instead, they highlight design trade-offs and opportunities for refinement within a systems framework intended to support secure, assessment-ready deployment.

5 Discussion

In this section, we discuss limitations and design trade-offs of ATHENA CTF, along with their implications for secure deployment, extensibility, and evaluation. We do not claim that ATHENA CTF provides complete isolation or adversarial security guarantees in its current configuration; rather, it is designed to support controlled, assessment-oriented deployments under explicitly defined assumptions. These considerations help contextualize the framework's current contributions and motivate directions for future refinement and study.

Shared State and Isolation Trade-offs. ATHENA CTF currently deploys all users of a given challenge within a shared machine state. This design choice prioritizes deployment simplicity and low resource overhead, making the framework accessible in instructional environments with limited infrastructure. However, shared state introduces an important limitation in adversarial settings: if a challenge is misconfigured or insufficiently sandboxed, actions performed by one participant may affect the environment observed by others. As a result, certain classes of challenges, particularly those involving persistent state or cross-user interference, such as stored cross-site scripting, are difficult to support safely under the current deployment model.

This limitation reflects an intentional trade-off between ease of deployment and isolation strength. A natural extension of the framework is support for per-user or per-team instance isolation using container orchestration platforms such as Kubernetes. While such an approach would enable a broader range of vulnerability scenarios and stronger isolation guarantees, it introduces additional complexity, including the need for container lifecycle management, authentication for privileged operations, and increased resource consumption. Exploring these trade-offs is an important direction for future work, particularly for high-stakes assessment or competition settings.

Abstraction Level and Challenge Expressiveness. The framework intentionally exposes a small set of well-defined abstractions to reduce implementation complexity and minimize unintended behaviour. While this design supports rapid creation of common web-security challenges, it also limits expressiveness for certain advanced scenarios. Feedback from expert sessions highlighted that some tasks, such as authentication workflows or persistent data handling, require direct interaction with underlying web framework components, increasing complexity for novice challenge authors.

Future extensions may include higher-level primitives for common patterns, such as authentication handlers, intentionally vulnerable data stores, or instrumentation for monitoring exploitation events. Providing such abstractions would allow authors to implement more expressive and security-relevant challenges while preserving the framework's safe-by-default design philosophy.

Evaluation Scope and Research Trajectory. The formative expert feedback described earlier was designed to validate core design assumptions and identify areas for refinement, rather than to measure learning outcomes or security effectiveness. As such, we do not claim that the current evaluation establishes pedagogical impact or robustness under adversarial conditions. Instead, it informs ongoing development and motivates more comprehensive evaluations.

Future studies will target distinct stakeholder groups, including challenge authors and challenge solvers, using structured protocols and larger participant pools. These evaluations will examine factors such as development effort, usability of abstraction, interface clarity, and perceived usefulness of context-driven, LLM-assisted hints. Together, these directions support a broader research trajectory focused on scalable, assessment-ready deployment of hands-on security challenges.

6 Conclusion

ATHENA CTF presents a modular, containerized framework for creating, deploying, and assessing web-based capture-the-flag challenges with a focus on reproducibility, deployment control, and assessment integrity. By combining a web-native architecture with deterministic per-user parametrization, optional centralized tracking, and constrained LLM-assisted hinting, the framework enables instructors and organizations to deploy hands-on security challenges that scale from informal learning to structured assessment.

The framework lowers the barrier to challenge creation by providing uniform abstractions for instruction, verification, and interface generation, allowing authors with modest programming experience to implement fully functional levels while preserving consistent execution semantics. Deployment flexibility enables challenges to be hosted locally or online, with configuration options that support transitions between instructional and evaluative modes without modifying challenge logic. Administrative tooling further supports assessment workflows by separating learner interaction from identity management and result export.

From a solver's perspective, ATHENA CTF provides a consistent interaction model across challenges while supporting targeted assistance through context-driven, non-conversational LLM-assisted hints. These hints are intentionally constrained and grounded in creator-authored solution paths, balancing learner support with safeguards against solution leakage and misuse. The shared interface and modular design further enable the development of cohesive challenge collections with predictable behaviour across levels.

Together, these design choices position ATHENA CTF as a flexible systems framework for delivering hands-on cybersecurity challenges under controlled conditions. By prioritizing modularity, reproducibility, and security-aware assistance, the framework provides a foundation for future work exploring stronger isolation guarantees, richer challenge primitives, and more comprehensive evaluations of secure, assessment-ready CTF deployments.

Acknowledgements

Anonymized for submission.

Ethical Considerations

ATHENA CTF is designed as an educational framework for teaching cybersecurity concepts through hands-on interaction with intentionally vulnerable systems. As with any system that exposes learners to real-world exploitation techniques, careful consideration must be given to how vulnerabilities are presented, deployed, and shared. To mitigate potential misuse, all publicly released sample challenges are deliberately scoped to well-known, already documented classes of vulnerabilities and do not include any undisclosed, zero-day, or pending exploits discovered during development. This aligns with responsible disclosure practices and ensures that the platform does not contribute to the dissemination of harmful or unreported security flaws.

The framework incorporates mechanisms to reduce academic misconduct and unauthorized sharing of solutions is particularly relevant in formal educational settings. Per-user and per-team flag parametrization ensures that solutions cannot be trivially shared across participants, supporting fair assessment while preserving the exploratory nature of CTF-style learning. Instructors may additionally disable automated hints or use static flags when appropriate for collaborative or demonstration-based activities.

ATHENA CTF includes optional LLM-assisted hints to support learner progress. These hints are intentionally designed to be non-conversational, context-constrained, and grounded in creator-authored solution paths. This design choice reduces the risk of solution leakage, hallucinated guidance, or unintended disclosure of complete answers, while also limiting susceptibility to prompt injection attacks. Administrators retain full control over whether LLM-assisted hints are enabled, which model is used, and how much contextual information is provided.

Regarding data privacy, the platform minimizes the collection and storage of personal information. When the centralized database is enabled, user records consist solely of anonymized identifiers, challenge interaction metadata, and limited historical request data required to generate contextual hints or support assessment. No demographic or personally identifying information is required or stored by default.

The formative expert feedback activities described in this work fall under quality assurance and program evaluation as defined by TCPS 2 (2022), Article 2.5, and REB SOP 203 §4.1.3, and therefore do not constitute research requiring REB review. The purpose of these sessions was limited to early design validation of the ATHENA CTF framework, focusing on usability, clarity of abstractions, and practicality of instructional deployment. The activities were conducted to inform iterative system refinement rather than to generate generalizable knowledge or evaluate pedagogical effectiveness.

Participation was limited to a small number ($N = 2$) of subject-matter experts acting in a professional capacity. No students were involved, and no personal, sensitive, or identifying data were collected. Feedback was anonymized and focused on system design and development workflow, meeting the criteria for exemption from REB review under institutional and TCPS guidelines.

Any future studies involving learners will be conducted under separate, REB-approved protocols and are outside the scope of the formative expert feedback reported here.

Overall, ATHENA CTF is designed to balance the pedagogical benefits of realistic security training with safeguards that promote responsible use, learner privacy, and ethical deployment in educational contexts.

Open Science

All anonomized framework code including documentation for level creation is available at:
https://anonymous.4open.science/r/thesis_project_framework-E86E/README.md
Version history and demonstration levels are available upon request.

References

[1] Jacob Klein Alex Moix, Ken Lebedev. Treat Iintelligence Rreport 2025. https://www-cdn.anthropic.com/b2a76c6f6992465c09a6f2fce282f6c0cea8c200.pdf, 2025. [Accessed 13-11-2025].

[2] Hany F Atlam. Llms in cyber security: Bridging practice and education. *Big Data and Cognitive Computing*, 9(7):184, 2025.

[3] Martin Carlisle, Michael Chiaramonte, and David Caswell. Using {CTFs} for an undergraduate cyber education. In *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, 2015.

[4] Peter Chapman, Jonathan Burket, and David Brumley. {PicoCTF}: A {Game-Based} computer security competition for high school students. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.

[5] Kevin Chung. Live lesson: Lowering the barriers to capture the flag administration and participation. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*, 2017.

[6] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys*, 57(7):1–36, 2025.

[7] Wenliang Du. Seed labs: Using hands-on lab exercises for computer security education. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 704–704, 2015.

[8] OWASP Foundation. Juice Shop. http://www.itsecgames.com/index.htm, 2025. [Accessed 06-12-2025].

[9] Google. kCTF. https://google.github.io/kctf/, 2025. [Accessed 17-12-2025].

[10] Sven Grübel. Teaching cybersecurity to high-school students. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 2*, ITiCSE 2024, page 848–849, New York, NY, USA, 2024. Association for Computing Machinery.

[11] Charles Harry, Ido Sivan-Sevilla, and Mark McDermott. Measuring the size and severity of the integrated cyber attack surface across us county governments. *Journal of Cybersecurity*, 11(1):tyae032, 01 2025.

[12] Cynthia E Irvine, Michael F Thompson, Michael McCarrin, and Jean Khosalim. Labtainers: a docker-based framework for cybersecurity labs. In *Proc. 2017 USENIX Workshop on Advances in Security Education*, 2017.

[13] Joe. RootTheBox. https://github.com/moloch--/RootTheBox, 2025. [Accessed 17-12-2025].

[14] Kaaviya. Over 40,000 CVEs Published in 2024, Marking a 38% Increase from 2023 — cyberpress.org. https://cyberpress.org/over-40000-cves-published-in-2024/, 2025. [Accessed 06-11-2025].

[15] Chen Lik Ken, Julia Juremi, Shahab Alizadeh, and Salasiah Sulaiman. Enhancing cybersecurity education through gamified learning and capture the flag (ctf) platform. In *International Workshop on Learning Technology for Education Challenges*, pages 69–82. Springer, 2025.

[16] MongoDB. MongoDB Documentation. https://www.mongodb.com/, 2025. [Accessed 17-12-2025].

[17] Connor Nelson, Adam Doupé, and Yan Shoshitaishvili. Sensai: Large language models as applied cybersecurity tutors. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 833–839, 2025.

[18] Connor Nelson and Yan Shoshitaishvili. Dojo: applied cybersecurity education in the browser. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 930–936, 2024.

[19] Natasha Nelson and Stuart Madnick. Studying the tension between digital innovation and cybersecurity. Association for Information Systems, 2017.

[20] Government of Canada. Fraud Prevention Month 2025. https://antifraudcentre-centreantifraude.ca/features-vedette/2025/02/month-prevention-mois-eng.htm, 2025. [Accessed 13-11-2025].

[21] MME Sec. Bwapp. http://www.itsecgames.com/index.htm, 2013. [Accessed 06-12-2025].

[22] Ryder Selikow, Jens Mache, Jack Cook, Joseph Granville, Richard Weiss, and Hsiao-An Wang. Edurange cloud: On demand cybersecurity sandboxes through kubernetes. In *International Conference on Availability, Reliability and Security*, pages 96–112. Springer, 2025.

[23] James H Senanu. Assessing cyber fraud in the financial sector of canada. 2024.

[24] Richard Weiss, Franklyn Turbak, Jens Mache, and Michael E Locasto. Cybersecurity education and assessment in edurange. *IEEE Security & Privacy*, 15(03):90–95, 2017.

[25] Robin Wood. Dvwa. https://github.com/digininja/DVWA, 2025. [Accessed 06-12-2025].