

— CYS401 · PROJECT REPORT

Mini Capture-The-Flag Security Lab Platform

Prince Sultan University · College of Computer and Information Sciences ·
CYS401: Fundamentals of Cybersecurity

PREPARED BY

Aljawhara Mohmmad Nasser Alruzuq · Areej Aadel Salem Almari · Leen
Hussamladeen Adel Nasser · Shoug Fawaz Abdullah Alomran

INSTRUCTOR

Dr. Basmah Alkanjr

SEMESTER · SUBMISSION

261 · November 4, 2026

01

— SECTION 1

Introduction

Cybersecurity education entails the use of practical exercises to reinforce theoretical concepts. The CTF events are great tools in helping the learners go through realistic situations, find weaknesses, and solve security problems in order to collect flags which develop important critical thinking and technical skills.

02

SECTION 2

Security Background and Analysis

What needs protection in the CTF platform, the risks that threaten it, the flaws that could be exploited, and the attacks most relevant to its functionality.

2.1 ASSETS

7

2.2 THREATS

6

2.3 VULNERABILITIES

6

2.4 ATTACKS

4

Four attacks most relevant to the platform

The identified vulnerabilities could be exploited through several attacks against the CTF Security Lab. These attacks may compromise the confidentiality, integrity, or availability of platform assets. The following attacks are particularly relevant to the functionality of the proposed system.

CIA PRINCIPLE AT RISK			
Attack	C	I	A
Brute-force login	●	●	○
Injection	●	●	○
Broken access control	●	●	○
Denial-of-service	○	○	●

Brute-Force LoginC · I

TARGET
Authentication System

Repeated automated login attempts leading to unauthorized account access and user impersonation.

Injection AttackC · I

TARGET
Input Fields & Database

Submitting malicious input via form fields to manipulate data, bypass application logic, or leak stored data.

Broken Access ControlC · I

TARGET
Progress & Score Data

Exploiting authorization flaws to access restricted resources, modify user scores, or alter challenge content.

Denial-of-Service (DoS)A

TARGET
App & Hosting Infrastructure

Flooding the server with excessive requests to consume resources and cause platform slowness or downtime.

How the platforms compare

FEATURE	PICOCTF	CTFD	OVERTHEWIRE	OUR PLATFORM
Web Challenges	● Yes (Web Exploitation)	● Depends on organizer	● Yes (Natas)	● Yes
Cryptography	● Yes	● Depends on organizer	● Yes (Krypton)	● Yes
Awareness	● No dedicated category	● Depends on organizer	● No	● Yes
Ethics	● No	● Depends on organizer	● No	● Yes
Flag Validation	● Automated	● Automated (static and regex flags)	● No flag submission; each level's password unlocks the next level	● Automated, using hashed flags
Hints	● Yes	● Yes (free or cost points)	● No built-in hints	● Yes, free
Progress Tracking	● Yes (picoGym)	● Yes	● No	● Yes
Scoreboard	● Yes (during competitions)	● Yes	● No	● Yes

Existing platforms like CTFd, OverTheWire, and picoCTF focus exclusively on technical skills, requiring custom content, command-line setup, or lacking hints (CTFd LLC, n.d.; OverTheWire, n.d.; Carnegie Mellon University, n.d.). Our browser-based lab addresses this gap by offering non-punitive hints, post-solution explanations, and dedicated Security Awareness and Ethics tracks (Švábenský et al., 2021).

Four domains, ten challenges

4

CHALLENGES

Web Security

Identify common web application vulnerabilities.

Source code inspection

SQL injection

Browser storage

Access control

2

CHALLENGES

Cryptography

Recognize weak encryption and decoding techniques.

Classical ciphers

Encoding/decoding

Hash recognition

2

CHALLENGES

Security Awareness

Recognize online security risks and prevent safe cyber hygiene failures.

Phishing email indicators

Password strength

Suspicious links

2

CHALLENGES

Cybersecurity Ethics

Understand responsible disclosure and ethical security testing boundaries.

Authorized scope

Privacy

Ethics

Professional responsibility

The stack: static pages, serverless backend

FRONTEND

HTML5, Tailwind CSS 3 (Play CDN), vanilla JavaScript (ES modules)

All 20 pages are static HTML styled with Tailwind utility classes; no framework and no build step. Shared header/footer, the challenge cards, and the guided tour are plain JS modules (`assets/js/`).

BACKEND

Firebase (serverless): Firebase Authentication + Firestore Security Rules v2

There is no custom server. Email/password accounts run on Firebase Auth, and `firestore.rules` does the server-side work: it SHA-256 hashes each submitted flag and compares it against the stored hash, enforces exact point awards, one solve per challenge, the 5-second anti-brute-force throttle, and user/admin roles.

DATABASE

Cloud Firestore (NoSQL)

Users, roles, solves, scores, attempt log, and the hidden `flagHashes` collection that no client can read. Firebase JS SDK 10.14.1, loaded from `gstatic`.

HOSTING

GitHub Pages, deployed by GitHub Actions; Firebase hosts auth/database

The workflow checks internal links, assembles the site, and publishes to Pages on every push to `main`. Served at `ctf-lab.shoug-tech.com` over HTTPS.

TOOLING

Visual Studio Code · Git + GitHub

IDE plus Prettier formatting on save, and the Firebase CLI emulators for local testing. Version control on `github.com/Shoug-Alomran/cys401`; `.gitignore` keeps the private flag and answer files out of the repo.

From submission to score

01

Flag storage

Challenge (CTF) flags will be stored securely on the server as cryptographic hashes to prevent participants from viewing the original flags in the website's source code.

02

Submission processing

When a participant submits a flag, the server identifies the challenge, normalizes the input, hashes the submitted flag, and compares it with the stored hash.

03

Correct & incorrect submission

If the flag is correct, the system records the completion and awards 100 points. If it is incorrect, the system displays an error message and awards no points.

04

Successful submission

After a correct submission, the system displays a success message and updates the participant's score.

05

Duplicate submissions

The system checks whether the participant has already completed the challenge. If so, it confirms the existing completion without awarding additional points. A database uniqueness constraint prevents duplicate completion records.

— LIVE DEMONSTRATION

From login to the scoreboard

01

Log in

Sign in as a participant with Firebase Authentication.

[Open ↗](#)

02

Select a challenge

Open a Web Security challenge and read its scenario.

[Open ↗](#)

03

Solve and submit

Find the flag, submit it, and see it validated against the stored hash.

[Open ↗](#)

04

Watch the scoreboard

The points and ranking update after the solve.

[Open ↗](#)

Confidentiality, integrity, availability

C

Confidentiality

Flags, passwords, and each user's private records (profile, solves, attempts) must stay hidden from other users.

CONTROLS

Flags are stored only as SHA-256 hashes in a database collection that no user can read. Passwords are handled by Firebase Authentication and are not stored in our database. Each user can read only their own profile, solves, and attempts; admins can read all.

I

Integrity

Scores, solves, and challenge content must change only through legitimate actions.

CONTROLS

A score can increase only by the points of a challenge solved in the same transaction. Each challenge can be scored once per user. Solve and attempt records cannot be edited or deleted. Only admins can change challenges and flag hashes. Timestamps are set by the server, not the browser.

A

Availability

Students must be able to reach challenges and submit flags during lab sessions.

CONTROLS

The website is static and hosted on GitHub Pages, and data is stored on Firebase's managed infrastructure. Each user can submit at most one flag every 5 seconds, which limits flooding through the submission path.

Every threat, mapped to a control

The controls below map the threats and vulnerabilities from Sections 2.2 to 2.4 to the controls implemented in the platform. Control types follow the course classification (preventive and detective).

Brute-force login (2.4) / Weak authentication (2.3) Preventive Firestore Authentication blocks sign-in temporarily after repeated failures (“too many requests”). Passwords must be at least 8 characters.	Flag guessing by repeated submissions Preventive A server rule allows one submission per user every 5 seconds.	Flag disclosure (2.2) / Insecure flag storage (2.3) Preventive Flags are stored only as SHA-256 hashes in a collection no user can read. Pages do not contain plain flags.
Injection / Malicious input (2.2, 2.4) Preventive The database is Firestore (NoSQL), accessed through its SDK, so no SQL statements are built from user input. Server rules check input type and length (flags up to 200 characters, usernames by pattern). Usernames are HTML-escaped before display to prevent cross-site scripting.	Broken access control / Unauthorized access (2.2, 2.4) Preventive Server rules let users read and change only their own records. Role is fixed to “user” at sign-up. Only admins can write challenges and flags.	Data manipulation of scores and progress (2.2) Preventive A score update is accepted only with a new, valid solve in the same transaction. Each challenge scores once per user.
Session misuse on shared lab computers Preventive Automatic logout after 30 minutes of inactivity; current password required for sensitive account changes.	Undetected misuse Detective Attempts and hints are logged with server timestamps and cannot be edited or deleted. Admins can review them.	Security misconfiguration (2.3) Detective Automated tests check the security rules (for example, “nobody can read flag hashes” and “a user cannot promote themselves to admin”).

All controls are technical.

What we tested and what happened

ID	FEATURE	TEST	EXPECTED	ACTUAL	STATUS
T01	Login	Log in with a valid email and password.	The user logs in.	The website displayed the user’s account after login.	PASS
T02	Challenge	Open a Web Security challenge.	The challenge instructions are displayed correctly.	The lab opened and displayed the challenge instructions and input fields.	PASS
T03	Flag validation	Submit the correct flag for WEB-01.	The flag is accepted and 50 points are awarded.	The flag was accepted, WEB-01 was marked as solved, and 50 points were awarded.	PASS
T04	Invalid flag	Submit an incorrect flag for WEB-02.	The flag is rejected and no points are awarded.	An incorrect-flag message was shown, and WEB-02 stayed unsolved.	PASS
T05	Dashboard	Complete a challenge and check the user dashboard.	The dashboard updates the number of solved challenges and earned points.	The dashboard displayed 1 out of 12 solved challenges and 50 out of 1200 points. WEB-01 was also listed as completed.	PASS
T06	Scoreboard	View the scoreboard after completing a challenge.	The scoreboard updates to show the new points and ranking.	The scoreboard reflected the updated score and ranking for the user.	PASS

The CTF platform functions properly for the primary features, according to the testing. The user received 50 points after the system recognized the correct flag. Additionally, it rejected an incorrect flag, leaving the challenge unresolved. The dashboard displayed one of the twelve challenges that had been completed and updated the user’s progress. The correct points and user ranking were also shown on the scoreboard. During these tested features, no issues were discovered.

Four ways to harden the platform

R1

Require email verification before submitting flags

PROBLEM ADDRESSED

The 5-second submission limit applies per account. A user could create several accounts with fake emails and guess flags in parallel, or add fake accounts to the scoreboard.

RECOMMENDATION

Accept flag submissions only from accounts with a verified email address. Firestore security rules can check the verified status of the signed-in user (Firebase, n.d.-c).

EXPECTED BENEFIT

Creating many accounts becomes slower, and every account on the scoreboard is linked to a real inbox.

R2

Store hashes instead of plain text in the attempts log

PROBLEM ADDRESSED

The attempts log stores the exact value each user submitted, including correct flags. Anyone with admin access, or anyone who takes over an admin account, could read the real flags from the log.

RECOMMENDATION

Store only the hash of each submitted value, or only whether it was correct.

EXPECTED BENEFIT

The log still supports auditing, but it no longer contains the secrets the platform is built to protect.

R3

Enable multi-factor authentication for admins

PROBLEM ADDRESSED

Admin accounts can change roles, edit challenges, and replace flag hashes. A stolen admin password would compromise the integrity of the whole platform.

RECOMMENDATION

Enable multi-factor authentication in Firebase Authentication (Firebase, n.d.-a) and require it for every admin account.

EXPECTED BENEFIT

A leaked or phished password alone is no longer enough to take over an admin account. This adds a second authentication factor (something you have), as covered in the course.

R4

Enable Firebase App Check

PROBLEM ADDRESSED

The database can be called directly by scripts outside our website, which allows automated flooding (the DoS attack in Section 2.4).

RECOMMENDATION

Enable Firebase App Check so the database accepts requests only from our real website (Firebase, n.d.-b).

EXPECTED BENEFIT

Bots and scripts have a harder time abusing the database, which protects availability.

09

SECTION 9

Conclusion

Mini CTF Security Lab Platform succeeds in connecting cybersecurity theory and practice by providing a proper hands-on learning platform. By filling the educational gap in the non-technical security field, the platform combines the main technical aspects (Web Security, Cryptography) with relevant human-centered ones (Security Awareness, Cybersecurity Ethics).

LEARNING AT THEIR OWN PACE

Automated flag verification

Non-punitive hints

Post-solve explanations

Progress monitoring

Live scoreboard

The key features of the system give participants an opportunity to learn at their own pace and get immediate feedback from the system.

SECURITY BUILT IN

Cryptographic hashing

Input sanitization

Access control

The system incorporates basic CYS401 knowledge in terms of application layer security mechanisms to protect itself from injection attacks, broken access control, and data leakage.

— THANK YOU

Questions?

Try the challenge tracks yourself, or see how the project is scoped and graded.

Explore Challenges

Project Overview