CyberSecureHub: An Educational Platform for Cybersecurity Learning

Germán Rodríguez Franceschini
Master in Computer Science
Advisor: Alfredo Cruz, Ph.D.
Polytechnic University of Puerto Rico
Graduate Project EXPO, May 2026

***Abstract* —** *Cybersecurity education is fragmented across separate websites, technical blogs, certification pages, and practice platforms. Entry level professionals must move between resources to learn penetration testing, security operations, and career pathways, which slows learning and disconnects theory from practice. The CyberSecureHub web application was developed using Python and the Flet framework to centralize foundational cybersecurity content into a single browser accessible environment. The platform integrates Red Team and Blue Team material, certification guidance, three tiers of Capture the Flag (CTF) challenges, and an Artificial Intelligence (AI) chatbot that provides support. The system was implemented and tested end-to-end. Results confirm that modular architecture can present offensive and defensive concepts together and that a contextual chatbot can support learning without replacing it.*

***Keywords* —** *AI Chatbot, Capture the Flag, Cybersecurity Education, Penetration Testing.*

INTRODUCTION

Cybersecurity has become one of the most important areas of modern technology, as organizations rely on digital systems, online services, cloud platforms, and interconnected networks for their daily operations. Companies must defend critical information, sustain service continuity, and respond to incidents. This demand has produced a need for educational tools that present offensive and defensive security in an organized way [1].

Learners are often introduced with offensive concepts such as reconnaissance, scanning, exploitation, and privilege escalation independently of defensive concepts such as monitoring, incident response, digital forensics, and vulnerability management. In addition, certification information is also separated from technical material, and practical exercises are typically introduced on different platforms with little connection to the learning content. Even simple concepts can require students to navigate several resources before they take shape.

Another challenge is the lack of immediate support when learners encounter an unfamiliar concept or technical term. Although learners may understand a topic on a general level, they often struggle to connect it to real practice or to other areas of cybersecurity, and many spend more time searching for information than learning. As a result, CyberSecureHub was developed as a centralized web-based educational platform that combines Red Team content, Blue Team content, certification guidance, Capture the Flag (CTF) exercises, and an AI chatbot for real-time support within a single interactive environment [2].

SYSTEM DESIGN

CyberSecureHub was implemented as a web-based educational tool developed in Python using the Flet framework in web mode, allowing the application to run in a standard browser without dedicated installation. The system architecture follows a presentation layer, an application logic layer, and a data layer to support organization, and long-term maintainability.

The presentation layer renders all user-facing views, including the Home Page, navigation, topic cards, dialogs, chatbot, assessments, and CTF screens. The application logic layer handles routing, content delivery, challenge submissions, assessments, and chatbot interactions. The data layer is hybrid in which static educational material is

stored in structured JSON files, while the database persists user accounts, assessment attempts, and CTF progress.

Figure 1 illustrates the three-layer architecture consisting of a user client (web browser), the CyberSecureHub application written in Python with Flet, and the data layer containing the JSON files that drive Red Team, Blue Team, certifications, and CTF content.

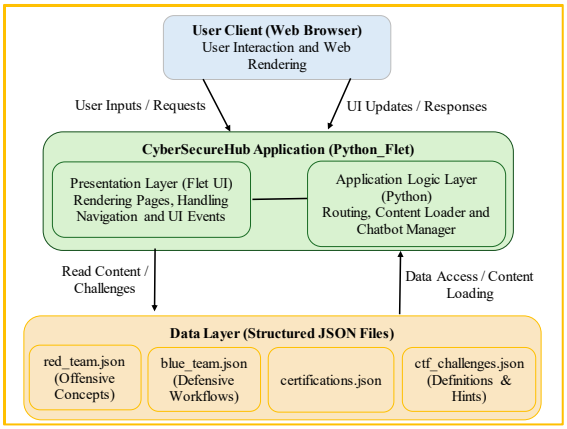

Figure 1
CyberSecureHub High-Level Architecture

The hybrid data design separates static educational content (JSON files) from dynamic user-related data (database tables for users, assessments, and CTF progress), as illustrated in the entity-relationship diagram in Figure 2.

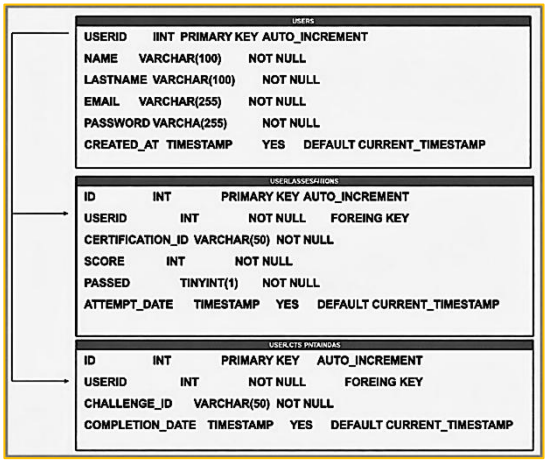

Figure 2
CyberSecureHub Entity-Relationship Diagram

The user interface follows a consistent card-based pattern across all modules. Users browse tiles, open a detail dialog, and access supporting resources, which keeps the focus on learning content rather than navigation. Figure 3 illustrates the navigation flow, showing that every major section can be accessed from the Home Page in a single step.

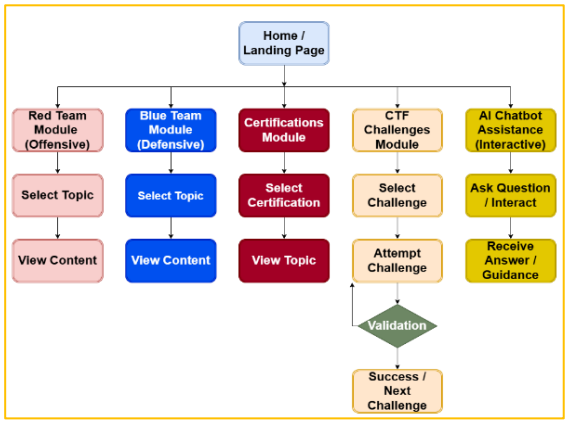

Figure 3
User Interaction and Navigation Flowchart

AI Chatbot Integration

A chatbot was developed using an external large language model API and embedded across the primary parts of the platform: Red Team, Blue Team, certifications, and CTF. The chatbot was designed as a teaching assistant that clarifies concepts, terminology, and workflows without becoming a source of unsafe operation instructions or direct CTF answers [3]. Figure 4 shows the integration workflow.

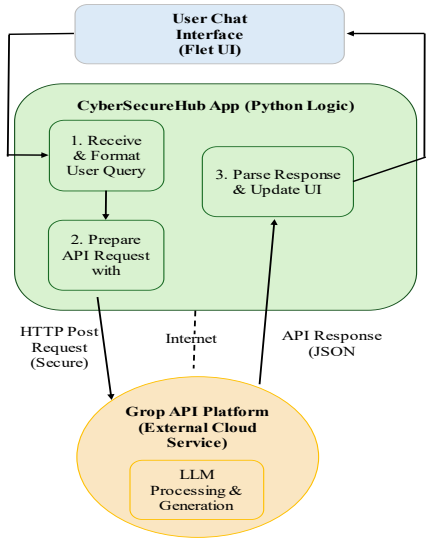

Figure 4
AI Chatbot Integration Workflow

In summary, the integration workflow begins when the user submits a query through the Flet interface. The application logic prepares the request with context, which is sent to the external LLM platform for processing. Finally, the response is parsed and displayed without taking the user away from their current module.

CAPTURE THE FLAG MODULE

The CTF module is a guided learning element rather than an open-ended hacking environment. Each challenge targets a particular concept and reinforces learning through direct engagement, consistent with hands-on cybersecurity education approaches [2, 4]. The module is organized into three difficulty tiers enabling learners to progress from foundational topics to more complex analytical tasks. All challenges include a description, a specific task, a flag-validation step, and hints available on request. The platform contains 31 total challenges across the three tiers; the following sections describe representative challenges from each tier.

TIER 1: CRYPTOGRAPHY AND STEGANOGRAPHY

Tier 1 introduces foundational concepts in encoding, classical cryptography, and hidden data analysis. The challenges help users recognize common transformation methods and understand how information can be concealed or represented in alternative formats.

Caesar Cipher

The Caesar Cipher is one of the oldest and simplest encryption techniques, attributed to Julius Caesar, who used it to protect his private military correspondence [5]. It is a monoalphabetic substitution cipher in which each plaintext letter is replaced by exactly one ciphertext letter, and that mapping is fixed for the entire message. The key is a single integer from 0 through 25, representing the number of positions by which each letter is shifted forward in the alphabet.

The cipher operates on modular arithmetic, where letters of the alphabet are converted into numbers using their zero-indexed position, so that A corresponds to 0, B to 1, and so forth, ending with Z at 25. Encryption is defined by the formula in (1) and decryption by the formula in (2):

$$C = (P + K) \bmod 26 \quad (1)$$

$$P = (C - K + 26) \bmod 26 \quad (2)$$

Where C is the ciphertext letter position, P is the plaintext letter position, and K is the shift key. The +26 in (2) prevents negative results when C < K, since adding a full alphabet cycle does not change the modular value but keeps it non-negative.

Caesar — Encryption

Figure 5 shows the complete letter-by-letter substitution when the key K = 3. Each plaintext letter in the top row is shifted three positions forward in the alphabet to produce the corresponding ciphertext letter in the bottom row.

As a worked example, the word HELLO is encrypted with K = 3 by applying C = (P + K) mod 26 to each letter, producing the ciphertext KHOOR. Figure 6 illustrates the full pipeline with the numeric position below each letter.

Caesar — Decryption

Decryption is the reverse process. Given the ciphertext KHOOR and the key K = 3, the formula P = (C – K + 26) mod 26 recovers each original letter, yielding the plaintext HELLO. Figure 7 shows the decryption pipeline.

Caesar — Brute Force Attack

The Caesar Cipher is broken by modern standards. With only 25 possible keys, an attacker can try every shift in seconds without specialized hardware. This kind of exhaustive search is known as a brute force attack. Table 1 shows the result of applying every shift to the ciphertext KHOOR; only one shift produces a readable English word.

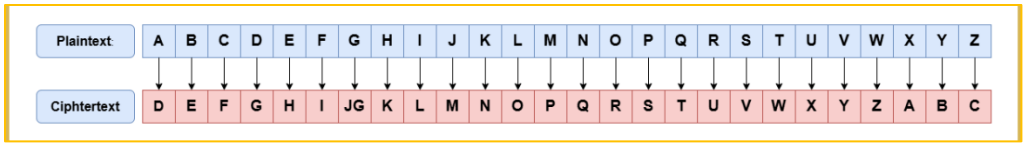

Figure 5
Complete Substitution Table with K = 3

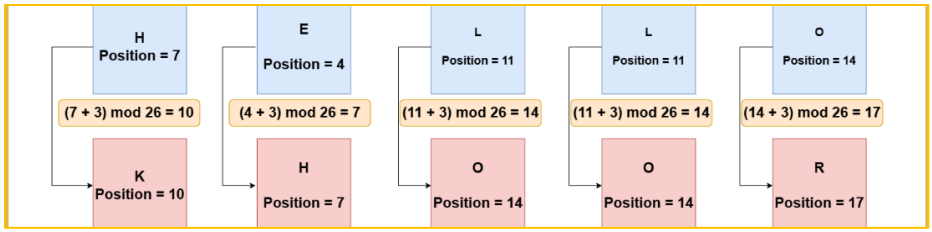

Figure 6
Caesar Encryption Example for HELLO

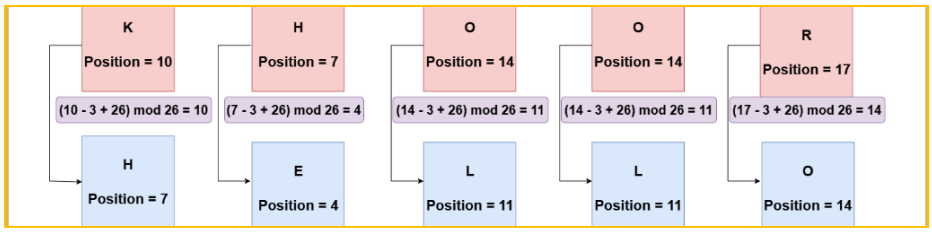

Figure 7
Caesar Decryption Example for KHOOR

Table 1
Brute Force of Caesar Ciphertext KHOOR

Shift (K)	Attempt	Readable?
1	JGNNQ	No
2	IFMMP	No
3	HELLO	Yes
4	GDKKN	No
...	...	No
25	LIPPS	No

Caesar — Frequency Analysis

Frequency analysis attacks substitution ciphers by studying how often each letter appears in the ciphertext. In Standard English text, certain letters appear far more often than others: E is the most common, followed by T, A, O, I, and N. If a Caesar ciphertext shows that the letter H appears most frequently, the analysis begins by hypothesizing that ciphertext H represents plaintext E, which immediately suggests a shift of K = 3.

Figure 8 shows the Standard English letter frequency distribution, while Figure 9 shows the same distribution after a Caesar shift of 3. The shape of the pattern is preserved; only the labels move. This means the Caesar Cipher does not conceal the statistical fingerprint of the language, making the cipher trivially breakable on any sufficiently long message.

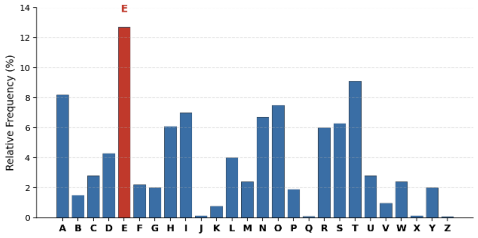

Figure 8
Standard English Letter Frequency

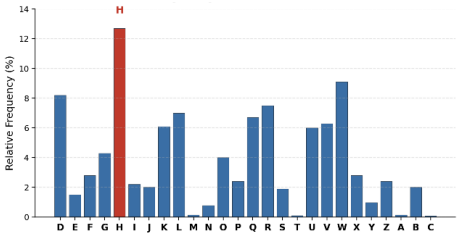

Figure 9
Letter Frequency After Caesar K = 3

Vigenère Cipher

The Vigenère Cipher is a polyalphabetic substitution cipher that addresses the Caesar Cipher vulnerability to frequency analysis. First described in 1553 by Giovan Battista Bellaso and later associated with Blaise de Vigenère, it applies multiple Caesar shifts to a single message [6]. Each letter of the plaintext is shifted by a value determined by a repeating keyword, rather than by a single fixed key. The cipher remained unbroken for more than three centuries and became known as *le chiffre indéchiffrable*, the indecipherable cipher.

Operations are typically performed using a 26-by-26 lookup grid called the Tabula Recta (Figure 10), where each row represents a different Caesar shift: row A represents the alphabet in natural order, row B is shifted to one position, and so on through row Z.

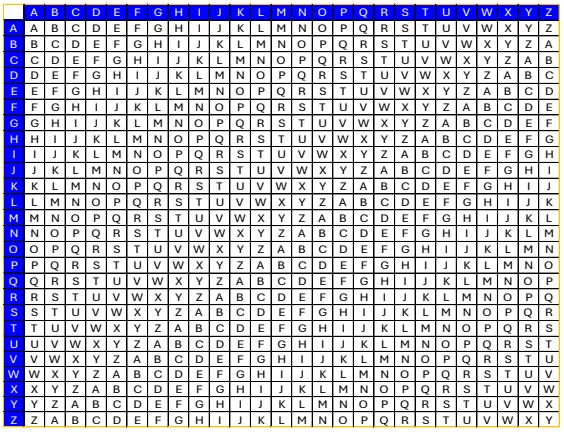

Figure 10
Vigenère Tabula Recta

Encryption and decryption are defined by formulas (3) and (4):

$$C_i = (P_i + K_i) \bmod 26 \quad (3)$$

$$P_i = (C_i - K_i + 26) \bmod 26 \quad (4)$$

Where the subscript i indicates that the shift changes for each position. The keystream is formed by repeating the keyword to match the plaintext length. For example, with keyword KEY (K = 10, E = 4, Y = 24) and a 12-letter plaintext, the keystream becomes KEYKEYKEYKEY.

Vigenère — Encryption

To encrypt HELLO with the keyword KEY, the keystream is first constructed as KEYKE. Each plaintext letter is then encrypted using the Tabula Recta: the row is selected by the plaintext letter, the column is selected by the keystream letter, and the intersection cell is the ciphertext letter. Figure 11 shows the lookup process for the first letter pair (plaintext H, keystream K), which yields the ciphertext letter R. The same procedure is applied to the remaining letters, producing the full ciphertext RIJVS.

A longer example using the keyword KEY on the plaintext ATTACKATDAWN produces the ciphertext KXRKGIKXBKAL. Repeated plaintext letters can still produce repeated ciphertext when they align with the same part of the keystream, which is the property the Kasiski examination later exploits.

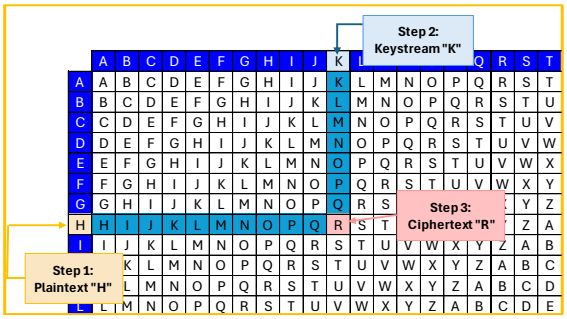

Figure 11
Tabula Recta Encryption Lookup

Vigenère — Decryption

Decryption reverses the procedure. Given the ciphertext RIJVS and the keyword KEY, the keystream KEYKE is rebuilt. For each pair, the keystream letter selects the column in the Tabula Recta, and the column is scanned downward until the ciphertext letter is found. The row label on the left is the recovered plaintext letter. Figure 12 shows this lookup for the first pair (keystream K, ciphertext R), which recovers the plaintext letter H. Repeating the process for all five pairs yields HELLO.

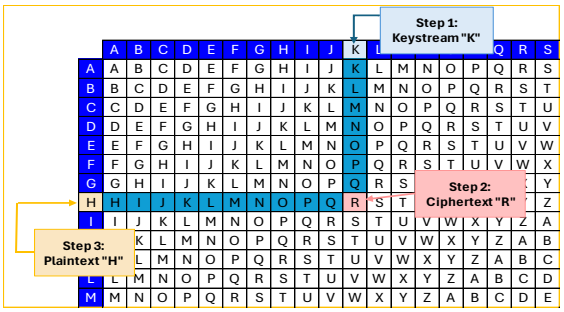

Figure 12

Tabula Recta Decryption Lookup

Vigenère — Kasiski Examination

The Vigenère Cipher remained secure for centuries because a single ciphertext letter no longer corresponds to a single plaintext letter, which defeats simple frequency analysis. However, it is breakable through the Kasiski examination, which exploits the fact that the keystream repeats. When the same plaintext sequence aligns with the same key letters, it produces an identical ciphertext sequence, and the distance between such repeats is a multiple of the keyword length [7].

In this approach, an attacker collects repeated trigrams, measures the distance between each pair, and computes the greatest common divisor (GCD) of those distances. The resulting GCD reveals the keyword length, or a small multiple of it. Table 2 illustrates this process using a sample ciphertext.

Table 2

Kasiski Distance Analysis

Trigram	Positions	Distance
ABC	12, 27	15
XYZ	8, 23	15
DEF	5, 20	15

Since GCD(15, 15, 15) = 15, the keyword length is a divisor of 15: 1, 3, 5, or 15. Once the keyword length is recovered, the ciphertext is split into that many interleaved streams; each stream is a simple Caesar cipher and yields to ordinary frequency analysis. To confirm the keyword length, the Index of Coincidence (IC) can be computed for each candidate length. Standard English text typically produces an IC of approximately 0.065, while a random distribution produces approximately 0.0385. The candidate keyword length whose interleaved streams produce the highest average IC is the most likely keyword length [7].

Playfair Cipher

The Playfair Cipher is a digraphic substitution cipher invented in 1854 by Charles Wheatstone and promoted by Lord Lyon Playfair to the British government. It saw active military use during the Second Boer War and World War I. Unlike the Caesar and Vigenère ciphers, Playfair encrypts pairs of letters (digraphs) simultaneously, which provides resistance to standard single-letter frequency analysis since the cipher obscures single-letter frequencies by encrypting two letters together [8].

The Playfair Cipher operates on a 5 × 5 grid called the key square, which introduces modular arithmetic base 5 instead of the base 26 used in the previous ciphers. Because the English alphabet contains 26 letters and the grid has only 25 cells, the letters I and J are combined into a single cell. The key square is constructed from a keyword by writing the keyword without duplicates, replacing every J with I, and then filling the remaining cells with the unused letters of the alphabet in order. Figure 13 shows the construction of a 5 × 5 key square using the keyword MONARCHY.

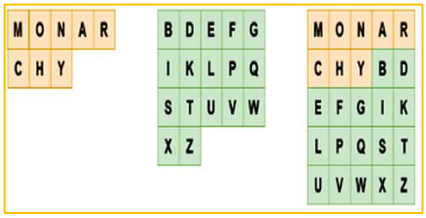

Figure 13

Playfair Key Square Construction (Keyword: MONARCHY)

Once the key square is built, encryption proceeds by splitting the plaintext into digraphs, inserting an X between identical letters in the same pair, padding an odd-length final pair with X, and then applying one of three rules to each digraph based on the relative position of the two letters in the grid.

The first rule (Same Row) replaces each letter with the letter immediately to its right, wrapping

around to the leftmost column when needed. Figure 14 shows this rule applied to the pair N and A in the same row, which produces A and R.

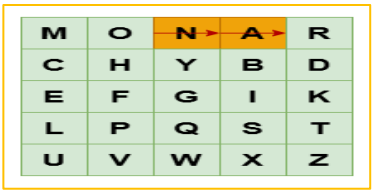

Figure 14
Playfair Rule 1 — Same Row

The second rule (Same Column) replaces each letter with the letter immediately below it, wrapping around to the top row when needed. Figure 15 shows this rule for the pair M and C in the same column, which produces C and E.

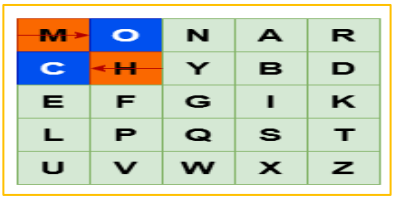

Figure 15
Playfair Rule 2 — Same Column

Finally, the third rule (Rectangle) applies when the two letters are in different rows and different columns, they form the corners of a rectangle, and each letter is replaced by the letter in the same row but at the column of the other letter. Figure 16 shows this rule for the pair M and H, which produces O and C. This rule is its own inverse, so the same operation is used for both encryption and decryption.

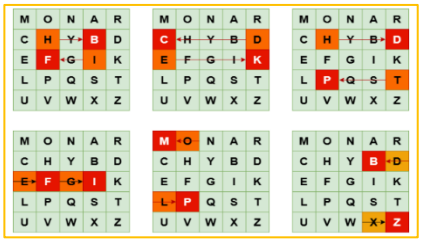

Figure 16
Playfair Rule 3 — Rectangle

Playfair — Encryption

To encrypt the phrase, HIDE THE GOLD using the keyword MONARCHY, the plaintext is first prepared as the six digraph pairs HI, DE, TH, EG, OL, and DX (an X is appended to pad the final pair). Each digraph is then encrypted using the key square and the appropriate rule. Figure 17 shows the complete encryption: pairs HI and DE form rectangles producing BF and CK; TH is a column case producing PD; EG is a row case producing FI; OL forms a rectangle producing MP; and DX forms a rectangle producing BZ. The resulting ciphertext is BFCKPDFIMPBZ.

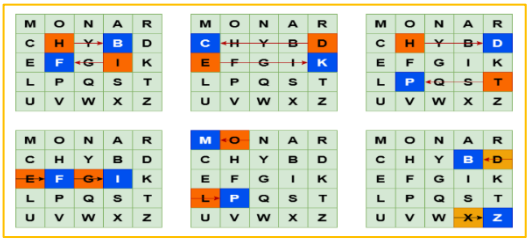

Figure 17
Playfair Encryption of HIDE THE GOLD

Playfair — Decryption

Decryption applies the inverse of each rule: Rule 1 shifts left instead of right, Rule 2 shifts up instead of down, and Rule 3 is its own inverse. Applying these inverse rules to the ciphertext BFCKPDFIMPBZ recovers HIDETHEGOLDX. The trailing X is removed as padding to produce the original message HIDE THE GOLD; the full lookup pipeline is shown in Figure 18.

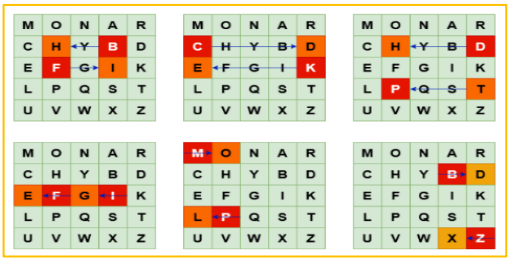

Figure 18
Playfair Decryption of BFCKPDFIMPBZ

Playfair — Frequency Analysis

The Playfair Cipher resists single-letter frequency analysis because it encrypts letter pairs rather than individual letters, so the distribution of single characters in the ciphertext does not match the distribution of single characters in English. The cipher is, however, vulnerable to diagram frequency analysis: in English, common diagrams such as TH, HE, IN, ER, and AN are far more frequent than

others, and a sufficiently long Playfair ciphertext reveals diagram frequencies that can be matched against known English tables [8]. Table 3 illustrates this approach.

Table 3
Digram Frequency Mapping Hypothesis

Ciphertext Digram	Frequency	Likely English Digram
BM	12	TH
OD	9	HE
ZB	7	IN
XD	6	ER
NA	5	AN

Once a diagram mapping is hypothesized, the attacker constrains the positions of those letters in the key square and tests candidate squares against known English patterns until the keyword is recovered. With about 200 ciphertext characters this attack succeeds reliably, and if a portion of the plaintext is known, even 30 characters of known plaintext can leak enough row and column structure to reconstruct the keyword.

TIER 2: WEB VULNERABILITY SIMULATION

Tier 2 focuses on common web application security weaknesses and presents them through controlled challenge scenarios. These exercises help users understand how insecure input handling, weak validation, and broken authorization logic can create serious security problems in real systems.

SQL Injection

SQL Injection occurs when user-supplied input is incorporated into a database query without sanitization or parameterization. The attacker alters the query logic to bypass authentication, extract data, or modify database contents. SQL Injection has remained on the OWASP Top 10 for over two decades [9]. Figure 19 illustrates the attack flow used in the challenge: the user enters credentials into a login form whose normal query reads SELECT * FROM users WHERE username = 'admin' AND password = 'secret'. The attacker injects an input that comments out the password check, leaving a condition that is always true, and the database executes the modified query and returns successful authentication. This challenge helps learners recognize how SQL special characters and operators alter query logic and understand parameterized queries as the definitive mitigation.

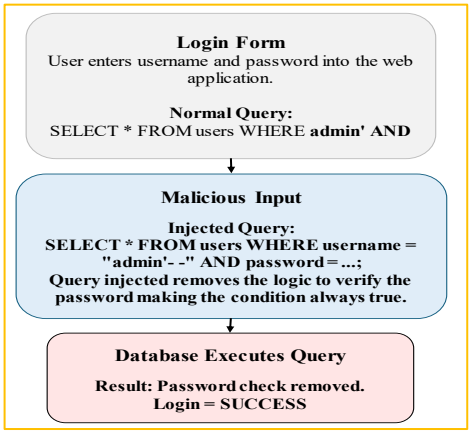

Figure 19
SQL Injection Attack Flow

Reflected XSS

Reflected Cross-Site Scripting (XSS) occurs when a web application includes unvalidated user input directly in its HTML response, causing the victim's browser to execute embedded scripts. Unlike SQL Injection, which targets the server, XSS targets the victim's browser and can steal session cookies, redirect users, or capture keystrokes. Figure 20 illustrates the attack flow used in the challenge.

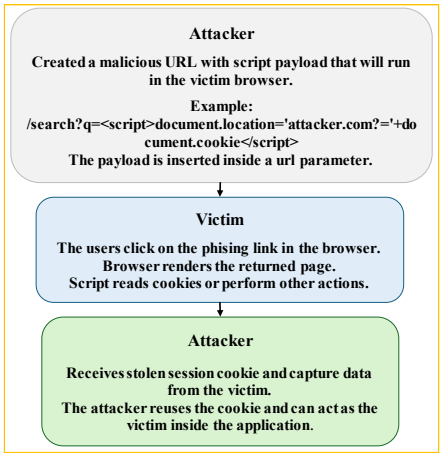

Figure 20
Reflected XSS Attack Flow

In this challenge the attacker crafts a malicious URL with a script payload in a query parameter, the victim clicks the link, the browser renders the returned page and executes the script, and the attacker receives the stolen session cookie and can act as the victim inside the application.

TIER 3: THREAT ANALYTICS AND FORENSICS

Tier 3 emphasizes defensive analysis, evidence interpretation, and investigation-oriented thinking. These challenges resemble activities associated with security operations, incident response, and forensic review, and are aligned with the tiered SOC model described in the cybersecurity literature [10].

Brute Force Logs

A brute force attack attempts unauthorized access by trying password combinations until one succeeds. In authentication logs, it leaves a clear fingerprint: a high volume of failed logins from a single IP targeting the same account within a short time window [11]. Figure 21 illustrates the analysis flow used in the challenge:

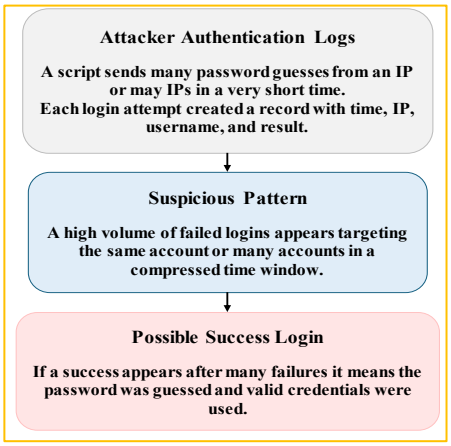

Figure 21
Brute Force Attack Pattern in Authentication Logs

In this challenge, the attacker generates many login attempts that each produce a log record with time, IP, username, and result; a suspicious pattern emerges as failures accumulate; and successful login after many failures indicates that credentials were guessed. The challenge helps learners read structured authentication logs as security analysts and distinguish brute force from normal login failures, as well as understand related variants such as dictionary attacks, credential stuffing, and password spraying.

Email Spoofing

Email spoofing changes the sender's information of a message, so it appears to come from another address, and is commonly used to impersonate a trusted source. The attack exploits the fact that SMTP does not authenticate sender identity. Figure 22 shows the attack flow used in the challenge: the attacker sends an email that looks like it came from IT, a bank, or a trusted contact; the victim's inbox displays a legitimate-looking message and the user clicks a link, opens an attachment, or replies with sensitive information; and the attacker collects credentials, sensitive data, or system access. This challenge helps learners interpret email headers to trace the true origin of a message, recognize common phishing indicators, and understand authentication controls such as SPF, DKIM, and DMARC as defenses.

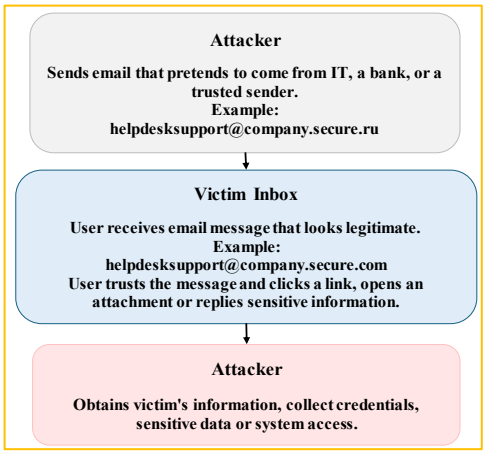

Figure 22
Email Spoofing Attack Flow

RESULTS AND DISCUSSION

All modules were implemented and operated as intended. The Red Team module delivers seven topic cards and the Blue Team module six, spanning offensive and defensive fundamentals; the Certifications module organizes ten certifications

into three career paths (Penetration Tester, SOC Analyst, and Governance, Risk and Compliance). The CTF module loaded all 31 challenges correctly across the three tiers, with flag validation and on-request hints working as expected. The tier structure performed as intended: Tier 1 was approachable for users with no prior CTF experience, while Tier 3 required analytical thinking and log interpretation. The AI chatbot remained accessible throughout all sections and maintained appropriate boundaries, explaining instructions, confirming that it functioned as a learning support layer rather than a shortcut [3], [12].

The results address the project's research questions. A single information architecture can present offensive and defensive cybersecurity concepts without confusion between them; the modular structure keeps Red Team and Blue Team content distinct while the shared navigation makes clear that both belong to the same field. The Certifications module places credential guidance in the same environment as the technical content it relates to, reducing the research time required for students to explore career paths.

CONCLUSION

CyberSecureHub achieved its primary goal of centralizing cybersecurity education into a single, accessible web-based platform. The project confirms that structured content, guided practical exercises, career-oriented certification pathways, and real-time AI support can be successfully integrated without fragmenting the user experience. The most significant contribution of this project is the demonstration that a lightweight Python and Flet application can serve as a viable foundation for structured cybersecurity education, and that the modular JSON-driven content architecture is flexible enough to support five distinct learning areas.

FUTURE WORK

Future development should focus on conducting formal user studies to validate the platform's educational effectiveness and measure learning outcomes, as well as expanding the CTF content into additional domains such as reverse engineering, and cloud security misconfigurations.

REFERENCES

[1] K. Abhari, M. Safaei Pour, and H. Shirazi. “How to design a better cybersecurity readiness program,” in *MIS Quarterly Executive*, vol. 23, no. 4, Art. 8, 2024. Doi:10.17705/2msqe.00107.

[2] L. Wang, Z. Tian, Z. Gu, and H. Lu. “Crowdsourcing approach for developing hands-on experiments in cybersecurity education,” in *IEEE Access*, vol. 7, pp. 169066–169071, 2019. Doi: 10.1109/ACCESS.2019. 2952585.

[3] M. Roshanaei and M. G. Jachura, “Integrating AI in cybersecurity higher education: A path to workforce readiness,” in *Journal of Intelligent Learning Systems and Applications*, vol. 17, no. 2, pp. 45–67, 2025. Doi: 10.4236/jilsa.2025.172005.

[4] N. R. Al-Kazaz, S. A. Irvine, and W. J. Teahan. “An automatic cryptanalysis of Playfair ciphers using compression,” in *Proc. 1st Int. Conf. Historical Cryptology (HistoCrypt 2018)*, Uppsala, Sweden, 2018, pp. 115–124. Available: https://ep.liu.se/konferensartikel.aspx?series= ecp&issue=149&Article_No=21.

[5] V. Švábenský, P. Čeleda, J. Vykopal, and S. Brišáková. “Cybersecurity knowledge and skills taught in capture the flag challenges,” in *Computers & Security*, vol. 102, 102154, 2021. Doi: 10.1016/j.cose.2020.102154.

[6] O. E. Omolara, A. I. Oludare, and S. E. Abdulahi. “Developing a modified hybrid Caesar cipher and Vigenere cipher for secure data communication,” in *Computer Engineering and Intelligent Systems*, vol. 5, no. 5, pp. 34–46, 2014. Available: https://iiste.org/Journals/index.php/ CEIS/article/view/12810.

[7] P. Cichonski, T. Millar, T. Grance, and K. Scarfone. “Computer security incident handling guide,” in NIST Special Publication 800-61 Rev. 2, 2012. Doi: 10.6028/NIST.SP.800-61r2.

[8] M. Elkhodr and E. Gide, “Integrating generative AI in cybersecurity education: Case study insights on pedagogical strategies, critical thinking, and responsible AI use,” in *arXiv*, 2025. Available: https://arxiv.org/abs/2502.15357.

[9] A. Saraswat, C. Khatri, P. Thakral, and P. Biswas. “An extended hybridization of Vigenere and Caesar cipher techniques for secure communication,” in *Procedia Computer Science*, vol. 92, pp. 355–360, 2016. Doi: 10.1016/j.procs.2016.07.393.

[10] A. L. Hananto, A. Solehudin, A. S. Y. Irawan, and B. Priyatna, “Analyzing the Kasiski method against Vigenere cipher,” in *arXiv*, 2019. Available: https://arxiv.org/abs/1912.04519.

[11] OWASP. (2026). *Web Security Testing Guide (WSTG) v4.2* [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/v42/.

[12] M. Vielberth, F. Böhm, I. Fichtinger, and G. Pernul. “Security operations center: A systematic study and open challenges,” in *IEEE Access*, vol. 8, pp. 227756–227779, 2020. Doi: 10.1109/ACCESS.2020.3045514.