Cybersecurity Education with Generative AI: Creating Interactive Labs from Microelectronic Fundamentals to IoT Security Exploitation

1st Kushal Badal
Department of Computer science
North Carolina Agricultural and Technical State University
Greensboro, USA.
kbadal@aggies.ncat.edu

2nd Xiaohong Yuan
Department of Computer science
North Carolina Agricultural and Technical State University
Greensboro, USA.
xhyuan@ncat.edu

3rd Huirong Fu
Computer Science and Engineering Department
Oakland University
Rochester, MI, USA.
fu@oakland.edu

4th Darrin Hanna
Electrical and Computer Engineering Department
Oakland University
Rochester, MI, USA.
dmhanna@oakland.edu

5th Jason Gorski
MicroNova LLC
2505 Pontiac Lake Rd
Waterford, MI 48328 USA.
jason@micro-nova.com

***Abstract*—Creating engaging cybersecurity education materials typically requires months of development time and specialized expertise. This paper describes how we used generative AI to address this challenge. We utilized Claude AI to generate a complete interactive platform that teaches students basic microelectronics through IoT hacking. Through iterative prompting, we generated more than 15,000 lines of functional code, including interactive visualizations, Python security tools, and gamified quizzes with real-time leaderboards. The curriculum guides students through the evolution of computing—from vacuum tubes to modern IoT devices—then helps them apply this foundation to discover real vulnerabilities. We implemented this platform at a GenCyber summer camp with 40 participants, where students identified actual security issues in AmpliPi audio systems—open-source network audio devices designed for multi-room audio distribution—including password weaknesses and denial of service flaws. The entire development process took only three weeks instead of the typical several months. The AI produced quality educational content, although we reviewed everything for technical accuracy and ethical considerations. During the camp, students remained engaged through competitive elements and hands-on labs, learning both theoretical concepts and practical skills. The students used AI-generated tools, including working implementations of SlowLoris and dictionary attacks, to test real systems. Our experience demonstrates that generative AI can efficiently create effective cybersecurity education materials that remain technically current. All materials are publicly available on GitHub for educational use. This approach could help educators stay on track with the rapidly evolving technology despite traditional curriculum development constraints.**

***Index Terms*—Generative AI, Cybersecurity Education, Interactive Learning, IoT Security, Hands-On Labs, Curriculum Development**

I. Introduction

Cybersecurity education faces a fundamental challenge: the rapid evolution of threats and technologies outpaces traditional curriculum development cycles. Educators must prepare students for attacks that did not exist when courses were designed, using defensive techniques that constantly evolve. This temporal disconnect becomes particularly acute when teaching hardware security and IoT vulnerabilities, where new devices enter the market faster than educational materials can be developed [1].

Traditional approaches to creating cybersecurity laboratories require substantial time investment. Developing a comprehensive hands-on module typically requires 3-4 months of faculty effort, including content creation, testing infrastructure setup, and safety validation [2]. By safety validation, we mean ensuring: (i) network isolation to prevent attacks from escaping the lab environment, (ii) system protection to safeguard institutional infrastructure, and (iii) ethical boundaries to prevent misuse of attack techniques. By the time the materials reach students, the specific vulnerabilities and tools may already be outdated. This development bottleneck limits educators' ability to provide current, relevant experiences that reflect real-world security challenges.

The emergence of IoT devices in educational environments presents both opportunity and risk. These devices offer tangible targets for security education, but require careful isolation and monitoring to prevent unintended consequences [3]. For our work, we selected the AmpliPro device [4], a network audio system that runs the open source AmpliPi multichannel audio application [5]. The AmpliPro serves as an ideal educational target because it represents a real-world IoT device with genuine security considerations, while being designed for multiroom audio distribution in residential and commercial settings. Importantly, the open-source nature of the AmpliPi software allows students to examine both the attack surface and the underlying code, providing a comprehensive learning experience. Creating safe but realistic testing environments for such devices requires experience in networking, virtualization, and security controls. By "safe," we mean network safety to prevent attacks from escaping the lab environment, system safety to protect institutional infrastructure, and student safety to prevent accidental self-harm through misuse of tools. These technical skills are particularly lacking at the K-12 level, where cybersecurity education starts to increase [6].

Recent advances in generative artificial intelligence offer a potential solution to these challenges. Large language models can produce functional code, educational content, and interactive visualizations from natural language specifications. However, their application to cybersecurity education raises critical questions about accuracy, safety, and pedagogical effectiveness [7]. Previous work has shown that gamification can significantly improve the mastery of network security concepts [8], and interactive visualization tools have proven to be effective in teaching complex security vulnerabilities [9].

This paper addresses these challenges through the development and deployment of an AI-generated cybersecurity curriculum focused on teaching microelectronics fundamentals through practical IoT security testing. We used generative AI to create a comprehensive educational platform that includes 12 interactive HTML5 presentations that cover the evolution from vacuum tubes to modern IoT security, functional penetration testing demonstrations, and gamified assessment components. Building on successful gamification approaches in cybersecurity education [10], [11], our platform incorporates competitive elements and interactive visualizations to maintain student engagement. The development process required only three weeks instead of the typical 3-4 months, demonstrating significant efficiency gains.

Our contributions include:

- A complete methodology for using generative AI to create cybersecurity educational materials, including prompt engineering strategies and implementation of safety control (rate limiting and target validation).
- Functional penetration testing demonstrations with educational scaffolding that balance realism with safety constraints (network isolation and restricted attack scope).
- The results of a deployment of a GenCyber summer camp with 40 K-12 participants, demonstrating successful vulnerability identification in real IoT devices.
- Open-source release of all materials, enabling reproduction and adaptation by other educators.

The remainder of this paper is organized as follows. Section II reviews related work on cybersecurity education and AI-assisted content generation. Section III details our AI-driven development methodology. Section IV describes the technical implementation of the platform. Section V presents the deployment results and the student results. Section VI concludes the paper and discusses future work.

II. Related Work

This section provides a comprehensive review of existing approaches to cybersecurity education, examining traditional laboratory development methods, the role of gamification and interactive learning, applications of artificial intelligence in computing education, IoT security platforms, and recent advances in AI-enhanced security education. We identify critical gaps in current methodologies that our AI-assisted approach addresses.

A. Hands-on Security Laboratory Development

Cybersecurity education has evolved from theoretical instruction to hands-on experiential learning in the past decade. The NICE Cybersecurity Workforce Framework established standardized knowledge units and competencies for cybersecurity education [12]. However, translating these frameworks into engaging educational experiences remains a challenge. Du and Wang [13] developed the SEED labs, providing hands-on exercises for security education, but noted the substantial time investment required for content development, often exceeding 100 hours per comprehensive lab module.

The GenCyber program, funded by the NSA and NSF since 2014, has been instrumental in bringing cybersecurity education to K-12 students [14]. However, creating effective security laboratories for such programs presents significant challenges. Virtual laboratories reduce infrastructure costs but introduce new complexities. KYPO CRP [15] provides cloud-based cybersecurity training environments but requires substantial technical expertise to deploy and customize. Justice and Vyas [16] developed RunLabs to quickly create virtualized laboratories, but the configuration and maintenance burden remains significant.

Commercial platforms like TryHackMe [17] and HackTheBox [18] leverage gamification principles but are designed for self-directed learning rather than structured classroom instruction. PentesterLab [19] provides guided exercises that focus primarily on Web and network security rather than hardware vulnerabilities.

The Open Web Application Security Project (OWASP) maintains WebGoat [20], an intentionally vulnerable web application for security training. Although valuable, WebGoat focuses solely on web vulnerabilities and lacks hardware security components. Similarly, IoTGoat [21] provides an intentionally vulnerable IoT platform, but requires physical hardware that many institutions cannot afford. Our selection of the AmpliPi system [5] addresses this limitation by using a real-world network audio device commonly used in residential and commercial settings. The open source nature of AmpliPi software allows students to examine both the attack surface and the underlying code, providing realistic IoT vulnerabilities while remaining accessible to educational institutions.

B. Gamification and Interactive Learning in Cybersecurity

Game-based learning has been shown to be effective in cybersecurity education. PicoCTF [22], designed for high school students, demonstrates that complex security concepts can be made accessible by appropriate scaffolding. Schreuders et al. [23] developed Hackerbot, which provides randomized security challenges, although creating new scenarios requires significant expertise.

CyberCIEGE [24] uses video game mechanics to teach security concepts through scenario-based learning [25], but development costs exceeded a million dollars, highlighting the resource constraints facing educators who want to create engaging security education materials. The effectiveness of the game has been documented in multiple studies [26].

Recent research further demonstrates the effectiveness of gamification in cybersecurity education. Hilliard et al. [8] showed that gamification significantly enhances the mastery of network security concepts, and students demonstrate improved retention when competitive elements are incorporated. Xu et al. [10] developed a game-based approach to teach ARP spoofing attacks, finding that students who learned through games showed a better understanding of attack mechanics compared to traditional instruction.

Interactive visualization has proven particularly effective for complex security concepts. Zhang et al. [9] developed interactive web-based tools to teach buffer overflow concepts, demonstrating that visual representations help students understand vulnerabilities in memory corruption. Similarly, Weanquoi et al. [11] created gamified lessons for access control concepts, showing improved engagement and comprehension among undergraduate students.

Our work builds on these foundations by combining AI-generated content with proven gamification techniques.

C. AI Applications in Computing and Security Education

The integration of artificial intelligence into computing education predates current generative models. Crow et al. [27] surveyed intelligent tutoring systems for programming education, finding mixed results for student learning outcomes. More recently, CodeX [28] and GitHub Copilot [29] demonstrated that AI can generate functional code from descriptions in natural language, fundamentally changing how students approach programming assignments.

Kazemitabaar et al. [30] demonstrated that when properly integrated into the curriculum, AI tools can enhance rather than replace learning. However, concerns about academic integrity persist, as students may complete assignments with minimal understanding of the underlying concepts.

Recent work has begun to explore the application of artificial intelligence in cybersecurity education. Agrawal et al. [31] developed CyberQ to generate questions and answers using an LLM augmented with knowledge graphs. Yamin et al. [32] explored the use of LLMs to generate cybersecurity exercise scenarios, although their focus remained on scenario generation rather than the development of complete educational platforms.

Mukherjee et al. [33] proposed an AI-enhanced intelligent tutoring system for graduate cybersecurity programs, demonstrating improved learning outcomes. Wei-Kocsis et al. [34] advocated for proactive and collaborative learning paradigms in the age of artificial intelligence. However, these works primarily focus on content generation or tutoring support rather than creating functional security testing tools.

D. IoT Security Education Platforms

Teaching IoT security presents unique challenges due to device diversity and rapid evolution. Siboni et al. [35] identified more than 140 distinct IoT protocols, making comprehensive coverage impossible in traditional courses. Pearson et al. [36] developed a low-cost IoT security laboratory, demonstrating that hands-on experiences can be provided with limited resources, although setup complexity remains a barrier.

Rao et al. [37], [38] created IoT security laboratories that emphasize the development of practical skills. Their work highlights the importance of real hardware in security education, but acknowledges the substantial preparation time required for each laboratory exercise.

E. Gap Analysis

The existing literature reveals several persistent challenges in cybersecurity education:

- **Time Investment:** Traditional content development requires 3-6 months for comprehensive modules.
- **Technical Expertise:** Creating realistic but safe environments requires specialized knowledge. Note that while AI assistance reduces this burden, technical expertise is still required to validate AI-generated content and ensure safety controls.
- **Rapid Obsolescence:** Materials become outdated before reaching students.
- **Resource Constraints:** Hardware and infrastructure costs limit accessibility.
- **Engagement:** Static materials fail to maintain student interest compared to gamified approaches.

Our work addresses these gaps by demonstrating that generative AI can create comprehensive educational materials in weeks rather than months, incorporating proven gamification techniques [8] and interactive visualizations [9] while using AmpliPi hardware [5] as an accessible real-world target. Unlike previous applications of AI in education, we generate complete functional tools with built-in safety controls, making cybersecurity education accessible by allowing educators without deep technical expertise to provide authentic hands-on experiences, although technical validation remains essential.

III. Methodology: AI-Assisted Development Process

This section describes our systematic approach to using generative AI to create educational materials on cybersecurity. We detail the development process, interactive content generation methods, security tool implementation, iterative refinement procedures, safety control mechanisms, and validation testing. Our methodology demonstrates how AI can accelerate curriculum development from months to weeks while maintaining educational quality and security standards.

A. Development Process

We used the Claude AI (Anthropic) Opus 4 model to generate a comprehensive cybersecurity education platform over three weeks in June 2025. Our approach treated AI as a collaborative partner rather than a simple code generator, using iterative prompt refinement to produce technically accurate educational materials. Figure 1 illustrates our development pipeline.

The process began with high-level specifications: developing interactive content to teach microelectronics fundamentals, demonstrating IoT vulnerabilities, and providing hands-on penetration testing tools. We decomposed these requirements into modular components that allowed focused development and testing of each element.

B. Interactive Content Generation

We generated 12 HTML5 presentations covering the evolution of computing from vacuum tubes to IoT devices. Each slide was individually developed through specific prompts that requested interactive, simulation-based presentations. For example, our initial prompt for the welcome slide requested: ”I need proper slide or ppt but using js, html, css, with proper simulation animation, visualization everything which is playable. Create a page for each slide, and you can make it scrollable so that text or buttons do not overlap. Make one page at a time so that we can test it properly and later on we can merge all slides pages.” This approach ensured that each slide received focused development attention before integration.

Figure 2 shows our actual prompt interface for creating the first slide of the GenCyber presentation.

Through iterative refinement, we improved the engagement with interactive elements. Listing 1 shows the navigation implementation that connects all 12 slides:

```
1 function loadSlide(slideNumber) {
2     currentSlide = slideNumber;
3     const slideFrame =
4         document.getElementById('slideFrame');
5     slideFrame.src = slideUrls[slideNumber];
6     updateNavigationButtons();
7     updateProgressIndicator();
8 }
```

Listing 1. Slide navigation implementation (AI-generated)

This navigation framework, referenced in Listing 1, enables seamless progression through all educational content. The function manages three components: slide loading based on iframes (lines 2-4), navigation button state management (line 5), and progress indication (line 6). Each slide features unique interactive elements: for example, slide2.html includes an animated vacuum tube simulation where students can click to see electrons flowing from cathode to anode, demonstrating how early computing components functioned before transistors.

Each technical concept was validated. When the AI incorrectly stated Moore’s law as annual doubling, we provided correct biennial data, ensuring accuracy in `slide4.html`.

C. Security Tools Development

We developed two penetration testing tools as Flask web applications, enabling supervised security testing on AmpliPi systems. AmpliPi is an open-source multi-room audio system that provides web-based control for audio zones, sources, and streaming services [5]. The AmpliPro hardware runs this software on a Raspberry Pi platform, making it representative of real-world IoT devices for educational use. We selected Flask as our web framework because it offers lightweight deployment suitable for classroom environments and transparent request handling that students can understand.

We created two primary attack demonstrations: (1) a SlowLoris denial-of-service tool demonstrating resource exhaustion and (2) a dictionary-based password cracker showing authentication vulnerabilities.

1) SlowLoris Implementation: To better help students understand how SlowLoris causes denial of service, we developed a restaurant metaphor through iterative prompting: network connections as tables, partial HTTP requests as customers “holding” tables without ordering. This required four prompt iterations to achieve appropriate complexity:

```
1 def start_attack():
2     """Restaurant: Customers holding tables"""
3     attack_state['active'] = True
4     for i in
      ↪ range(attack_state['max_connections']):
5         sock = socket.socket(socket.AF_INET,
6                              socket.SOCK_STREAM)
7         sock.connect((target_ip, target_port))
8         attack_state['connections'].append(sock)
```

Listing 2. SlowLoris with educational metaphor (AI-generated)

Listing 2 demonstrates the educational approach. The function creates TCP connections (lines 5–6), connects them to the AmpliPi target (line 7), and maintains a list of active connections for management (line 8). The restaurant metaphor (line 2) was generated by AI after we requested educational scaffolding to help teenage learners understand the concepts of resource exhaustion.

2) Password Security Analyzer: The password cracker required a balance between realism and safety. Initial generations were oversimplified—they only attempted single password formats. Real authentication systems like AmpliPi accept parameter names such as username and password. We refined our prompts to handle AmpliPi authentication:

```
def test_login(self, ip, username, password):
    login_attempts = [
        {"username": username, "password":
    ↪ password},
        {"user": username, "password": password},
        {"email": username, "password": password}
    ]
    # Rate limiting: 100ms between attempts
```

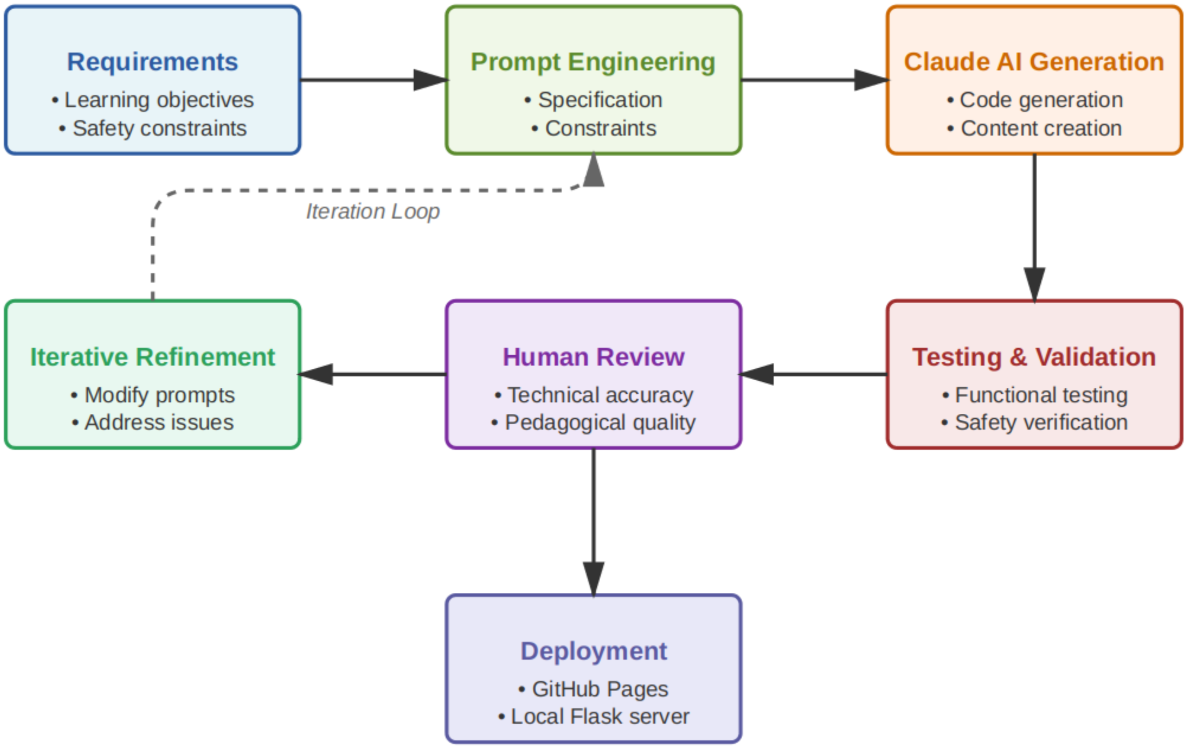


Fig. 1. AI-assisted development workflow showing iterative refinement cycles

```
        time.sleep(0.1)
```

Listing 3. Multi-format authentication testing (AI-generated)

Listing 3 shows the multi-format approach needed for realistic testing against AmpliPi's authentication system.

D. Iterative Refinement and Testing

Each component was tested against AmpliPi hardware (Model: AmpliPro). Table I summarizes the iterations of development.

TABLE I
COMPONENT DEVELOPMENT ITERATIONS

Component	Iterations	Key Issues
Navigation Framework	2	Mobile responsiveness
Interactive Animations	4	Browser compatibility
SlowLoris Tool	6	Socket management
Password Cracker	5	Login detection

The navigation framework required two iterations to fix mobile viewport scaling. The interactive animations needed four iterations to resolve CSS compatibility issues in Chrome and Firefox. The SlowLoris tool took six iterations primarily to handle socket connection cleanup and prevent memory leaks. The password cracker required five iterations because AmpliPi returned HTTP 302 redirects instead of expected 200 status codes, requiring us to modify the success detection logic as shown in Listing 4. We resolved this by providing actual network traces to the AI:

```
success_indicators = [
    response.status_code == 302,
    'zones' in response.text.lower(),
    len(response.cookies) > 0
]
```

Listing 4. Corrected authentication detection

E. Safety Controls

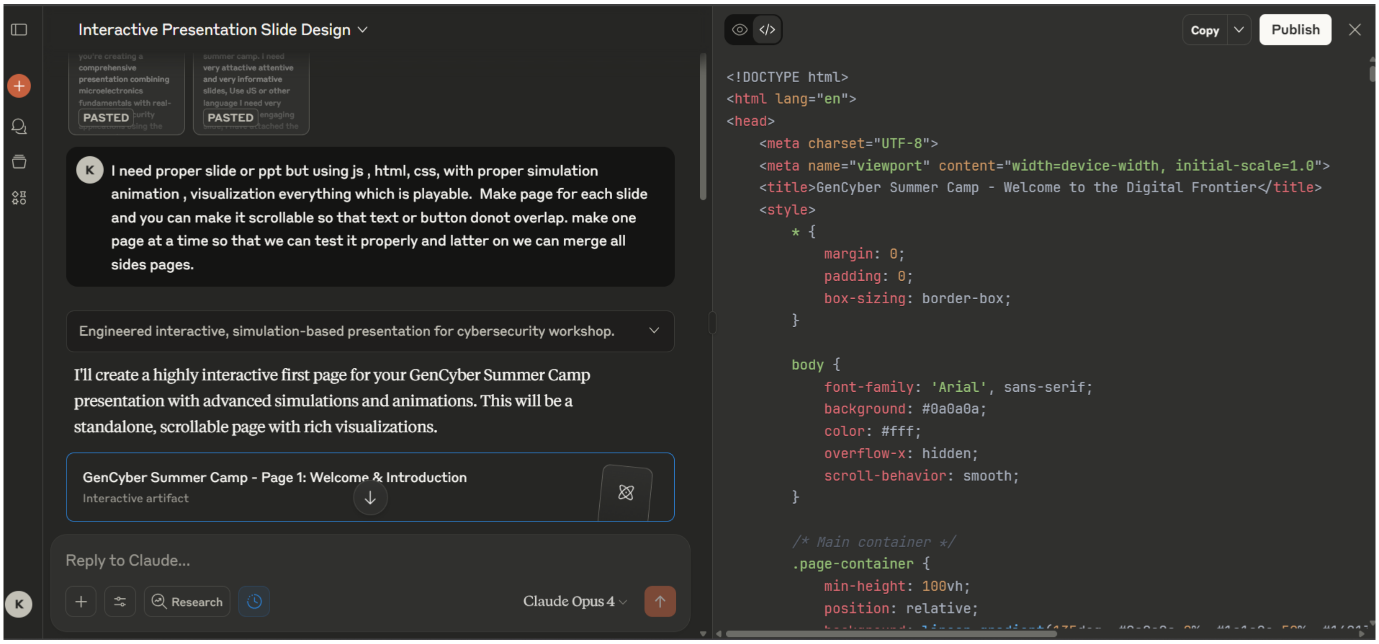
Educational hacking tools require robust safeguards. By "safety", we mean preventing attacks from escaping the lab environment, protecting institutional infrastructure, and ensuring students cannot harm external systems. We implemented the following:

- **Rate limiting:** 10 connections (SlowLoris), 3 workers (password cracking)
- **Session management:** Automatic cleanup on disconnection
- **Prominent warnings:** Educational-use notices on all interfaces

F. Development Metrics

The complete development process yielded the following.

- **Timeline:** 21 days total, 60 active development hours. The "21 days (60 active hours)" indicates that the project spanned 21 calendar days with 60 hours of active development, approximately 3 hours daily
- **Code generated:** 15,247 lines (HTML/JS: 12,156, Python: 3,091)



Interactive Presentation Slide Design

you're creating a comprehensive presentation combining microelectronics fundamentals with real-…curity applications using the

PASTED

summer camp. I need very attactive attentive and very informative slides, Use JS or other language I need very engaging slide, I have attached the

PASTED

K I need proper slide or ppt but using js , html, css, with proper simulation animation , visualization everything which is playable. Make page for each slide and you can make it scrollable so that text or button donot overlap. make one page at a time so that we can test it properly and latter on we can merge all sides pages.

Engineered interactive, simulation-based presentation for cybersecurity workshop.

I'll create a highly interactive first page for your GenCyber Summer Camp presentation with advanced simulations and animations. This will be a standalone, scrollable page with rich visualizations.

GenCyber Summer Camp - Page 1: Welcome & Introduction
Interactive artifact

Reply to Claude...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>GenCyber Summer Camp - Welcome to the Digital Frontier</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Arial', sans-serif;
            background: #0a0a0a;
            color: #fff;
            overflow-x: hidden;
            scroll-behavior: smooth;
        }

        /* Main container */
        .page-container {
            min-height: 100vh;
            position: relative;
```

Fig. 2. Actual prompt used for generating interactive presentation slides

- **Prompt iterations:** 20 major, ∼30 refinements
- **Manual modifications:** 15% of generated code

G. Validation

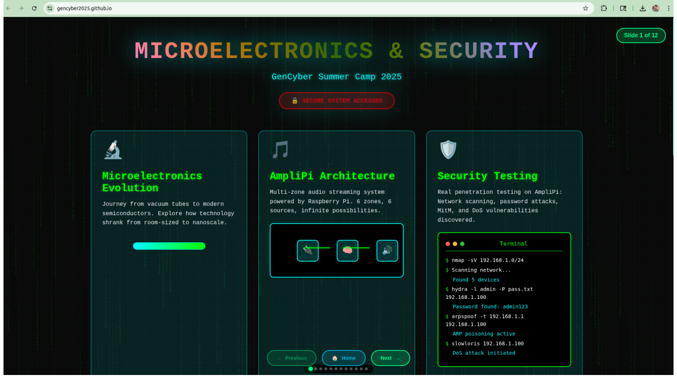
Before deployment, we conducted validation testing with three computer science students and two faculty members. Testing revealed two issues that required remediation: interface latency in the SlowLoris visualization (resolved through CSS optimization) and connection timeouts in the password cracker. Following these optimizations, the platform successfully supported 20 concurrent users during the GenCyber high school student summer camp.

IV. Platform Implementation

This section describes the technical implementation of our cybersecurity education platform. We detail the system architecture, the interactive presentation platform, security testing tools, safety mechanisms, and integration with AmpliPi devices. The implementation demonstrates how AI-generated code was deployed in a real educational environment with appropriate security controls.

A. System Architecture

The platform employs a hybrid deployment architecture that separates educational content from attack tools for security isolation. Interactive presentations are publicly hosted on GitHub Pages (https://microelectronics2025.github.io/), while penetration testing tools are run on an instructor-controlled machine within an isolated local network. Figure 3 illustrates the main interface, which presents three learning modules: (1) Microelectronics Evolution covering the journey from vacuum tubes to modern semiconductors, (2) AmpliPi Architecture explaining the multizone audio system's components and API structure, and (3) Security Testing providing hands-on tools for network scanning and vulnerability discovery.



gencyber2025.github.io

Slide 1 of 12

MICROELECTRONICS & SECURITY

GenCyber Summer Camp 2025

🔒 SECURE SYSTEM ACCESSED

Microelectronics Evolution
Journey from vacuum tubes to modern semiconductors. Explore how technology shrank from room-sized to nanoscale.

AmpliPi Architecture
Multi-zone audio streaming system powered by Raspberry Pi. 6 zones, 6 sources, infinite possibilities.

Security Testing
Real penetration testing on AmpliPi: Network scanning, password attacks, MitM, and DoS vulnerabilities discovered.

```
Terminal
$ nmap -sV 192.168.1.0/24
$ Scanning network...
  Found 5 devices
$ hydra -l admin -P pass.txt
192.168.1.100
  Password found: admin123
$ arpspoof -t 192.168.1.1
192.168.1.100
  ARP poisoning active
$ slowloris 192.168.1.100
  DoS attack initiated
```

Fig. 3. Interactive presentations portal hosted on GitHub Pages, showing the 12-module curriculum with visual navigation

This separation ensures that attack tools remain inaccessible from the public Internet while allowing students to access educational materials from any device. The local network configuration consisted of 20 lab computers, one instructor laptop running Flask applications, and two AmpliPi devices [4] designated as authorized targets. The architecture of the AmpliPi system [4], shown in Figure 4, demonstrates the REST API endpoints that our attack tools target.

As illustrated in Figure 4, the AmpliPi system exposes multiple API endpoints (GET/PATCH/POST/DELETE operations) for controlling audio zones, sources, groups, and streams. These endpoints became our primary attack vectors for demonstrating authentication weaknesses and denial-of-service vulnerabilities. The Raspberry Pi host runs the open-

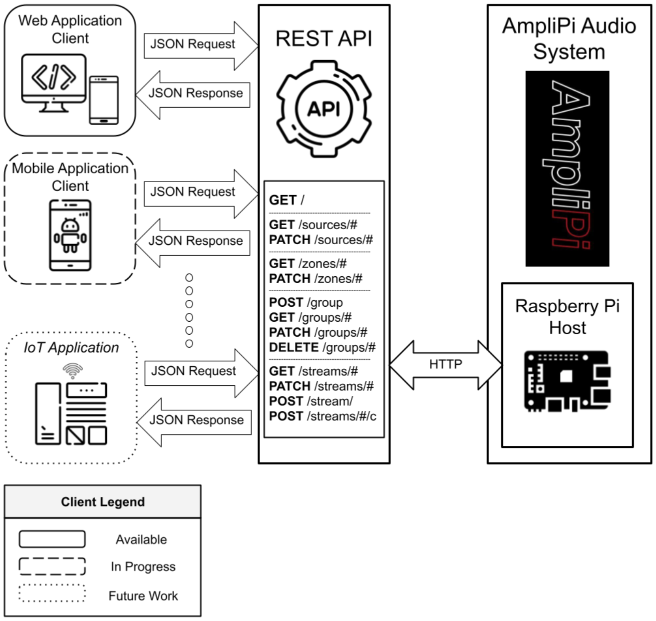


Fig. 4. AmpliPi REST API architecture showing various client types accessing the system through JSON requests/responses over HTTP [4]

source AmpliPi software, making it representative of real-world IoT deployments.

B. Interactive Presentation Platform

1) GitHub Pages Deployment: The presentation system consists of twelve interconnected HTML5 slides accessible at `microelectronics2025.github.io`. Each slide is self-contained with embedded JavaScript and CSS, eliminating external dependencies and ensuring consistent performance across different network conditions. The navigation system provides seamless progression through the curriculum, from vacuum tube fundamentals to modern IoT security concepts.

2) Interactive Visualizations: Each slide incorporates engagement features beyond static content. For example, the network scanning simulation provides visual feedback as devices are discovered, as shown in Listing 5:

```
1  function startNetworkScan() {
2      const devices = [
3          {x: 50, y: 30, ip: '192.168.1.101',
4           name: 'AmpliPi', type: 'audio'},
5          {x: 30, y: 60, ip: '192.168.1.105',
6           name: 'iPhone', type: 'mobile'}
7      ];
8      devices.forEach((device, index) => {
9          setTimeout(() => {
10             createScanDot(device);
11             showDeviceInfo(device);
12             playDiscoverySound();
13         }, index * 1000);
14     });
15 }
```

Listing 5. Network scan visualization with progressive revelation

Listing 5 demonstrates the progressive revelation technique used in our network scanning visualization. Each device appears one second apart (line 13), creating visual dots on the scan display (line 10), showing the device information (line 11), and playing an audio cue (line 12). This sequential discovery helps students understand how network scanning tools identify devices rather than showing all results instantly. Figure 5 shows the resulting visualization.

C. Security Testing Tools

1) Local Flask Deployment: The Flask application (`cyber-edu-app.py`) hosts attack tools on the instructor's machine, accessible only within the laboratory network. Students access tools via the instructor's LAN IP address, preventing external access while enabling supervised classroom use.

2) SlowLoris Implementation: The SlowLoris module demonstrates the concepts of denial of service through the restaurant metaphor discussed in Section III. The implementation includes both technical functionality and educational visualization, as shown in Listing 6:

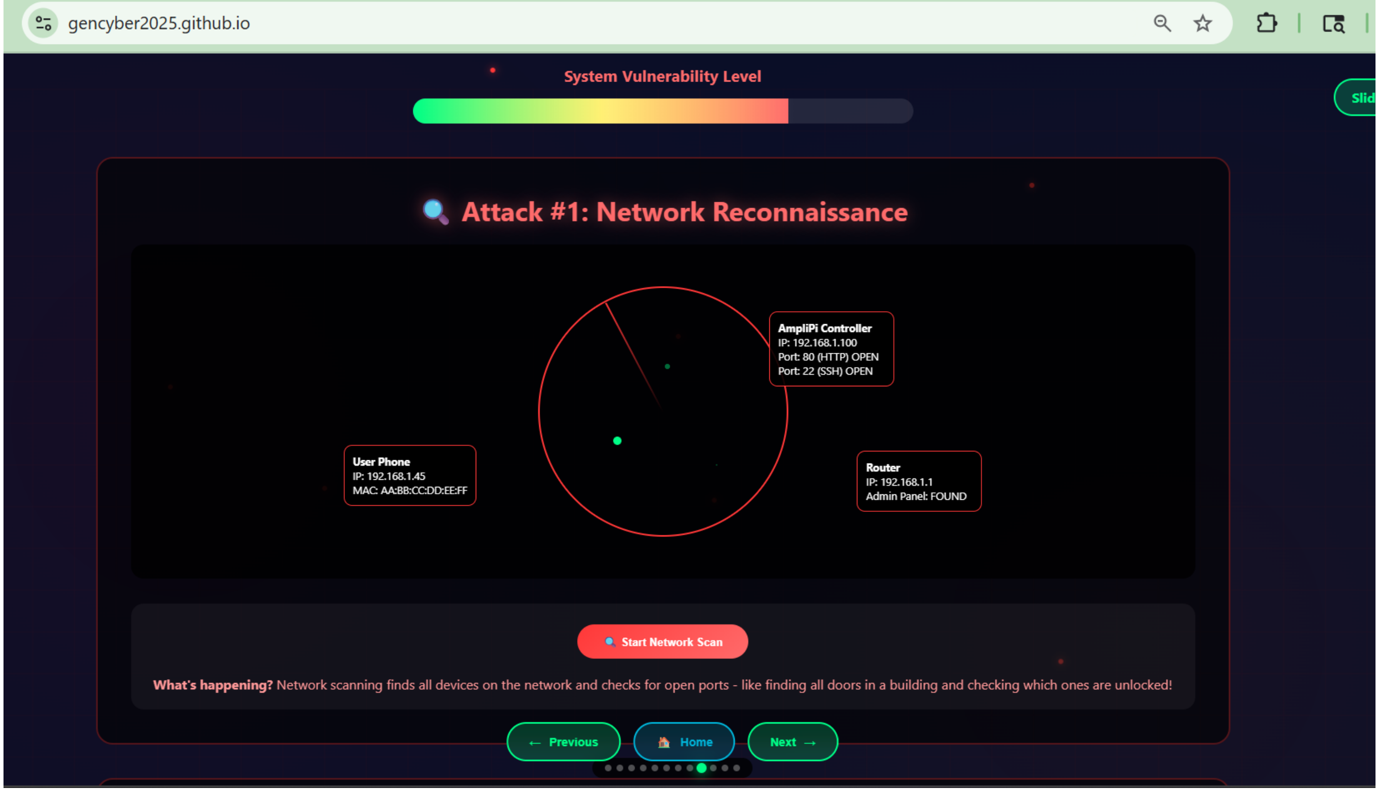
```
1  attack_state = {
2      'active': False,
3      'connections': [],
4      'connections': 50,
5      'target_ip': '192.168.1.143',   # AmpliPi only
6      'restaurant_tables': []   # Visual metaphor
7  }
8
9  def start_attack():
10     """Initiates SlowLoris with restaurant
       ↪ visualization"""
11     if not
       ↪ is_amplipi_target(attack_state['target_ip']):
12         return jsonify({'error': 'Invalid
       ↪ target'})
13
14     attack_state['active'] = True
15     for i in
       ↪ range(attack_state['max_connections']):
16         sock = create_slow_connection()
17         attack_state['connections'].append(sock)
18         # Update restaurant visualization
19
       ↪ attack_state['restaurant_tables'].append({
20             'table_number': i,
21             'status': 'occupied_not_ordering'
22         })
```

Listing 6. SlowLoris with safety controls and educational mapping

Listing 6 implements the SlowLoris attack with built-in safety controls. The code restricts targets to AmpliPi devices only (line 5). The restaurant metaphor maps each connection to a table (line 6), and when the attack runs, it creates slow connections (line 16) while updating the visual representation showing tables as "occupied but not ordering" (lines 19-22). Figure 6 shows the user interface with the visualization of the metaphor of the restaurant.

3) Password Security Analyzer: The password module demonstrates dictionary attacks while teaching password security principles. The tool adapts to AmpliPi's specific authentication behavior:

```
1  def test_login(self, ip, username, password):
2      """Tests authentication against AmpliPi's
       ↪ open-source login"""
```



What's happening? Network scanning finds all devices on the network and checks for open ports - like finding all doors in a building and checking which ones are unlocked!

Fig. 5. Interactive network scanning user interface

```
        self.login_url = f"http://{ip}/auth/login"
        response = self.session.post(
            self.login_url,
            data={"username": username, "password":
        ↪ password},
            timeout=5
        )

        # AmpliPi returns 302 redirect on success,
        ↪ not 200
        if response.status_code == 302:
            if 'zones' in
        ↪ response.headers.get('Location', ''):
                return {"success": True, "password":
        ↪ password}
        return {"success": False}
```

Listing 7. AmpliPi-specific authentication handling

This AmpliPi-specific configuration was necessary because the open-source implementation [5] uses non-standard HTTP responses. The tool includes a strength analyzer that provides real-time feedback on password complexity, as shown in Figure 7.

D. Safety Mechanisms

1) Network Isolation: The laboratory network operates on a dedicated VLAN (192.168.1.0/24) with strict access controls:

- No internet routing during attack exercises
- Instructor machine as sole host for attack tools

2) Application-Level Controls: Target validation ensures that tools can only interact with designated AmpliPi systems:

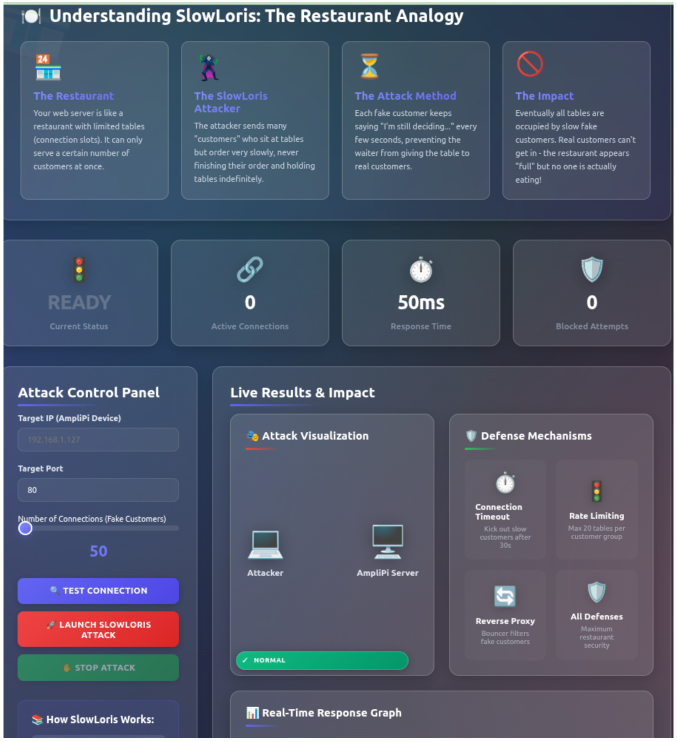
```
ALLOWED_TARGETS = ['192.168.1.143',
    ↪ '192.168.1.144']

def verify_amplipi_signature(ip):
    """Confirms target is genuine AmpliPi
    ↪ device"""
    try:
        response =
    ↪ requests.get(f"http://{ip}/api/",
                        timeout=2)
        # Check for AmpliPi API signature
        return 'amplipi' in
    ↪ response.json().get('name', '').lower()
    except:
        return False
```

Listing 8. Target validation with AmpliPi verification

E. Integration with AmpliPi

The tools specifically focus on AmpliPi's authentication endpoints. Initial testing showed that AmpliPi's login system returns HTTP 302 redirects on success rather than 200 OK responses. We adjusted the detection logic accordingly as shown in Listing 4. This platform-specific setup ensured an accurate demonstration of vulnerability while keeping the educational value intact. Students successfully identified weak default passwords and demonstrated denial-of-service vulnerabilities during supervised exercises.



Understanding SlowLoris: The Restaurant Analogy

The Restaurant

Your web server is like a restaurant with limited tables (connection slots). It can only serve a certain number of customers at once.

The SlowLoris Attacker

The attacker sends many "customers" who sit at tables but order very slowly, never finishing their order and holding tables indefinitely.

The Attack Method

Each fake customer keeps saying "I'm still deciding..." every few seconds, preventing the waiter from giving the table to real customers.

The Impact

Eventually all tables are occupied by slow fake customers. Real customers can't get in - the restaurant appears "full" but no one is actually eating!

READY — Current Status

0 — Active Connections

50ms — Response Time

0 — Blocked Attempts

Attack Control Panel

Target IP (AmpliPi Device): 192.168.1.127

Target Port: 80

Number of Connections (Fake Customers): 50

Live Results & Impact

Connection Timeout — Kick out slow customers after 30s

Rate Limiting — Max 20 tables per customer group

Reverse Proxy — Bouncer filters fake customers

All Defenses — Maximum restaurant security

Fig. 6. SlowLoris attack tool interface showing restaurant metaphor: occupied tables represent held connections, with real-time updates as the attack progresses

V. Results and Discussion

A. Deployment Context and Participant Demographics

The platform was deployed during a two-week GenCyber summer camp with 40 K-12 students, divided into two cohorts of 20 students per week. The students had a variety of technical backgrounds. The camp used a controlled laboratory environment with isolated network infrastructure as described in Section IV.

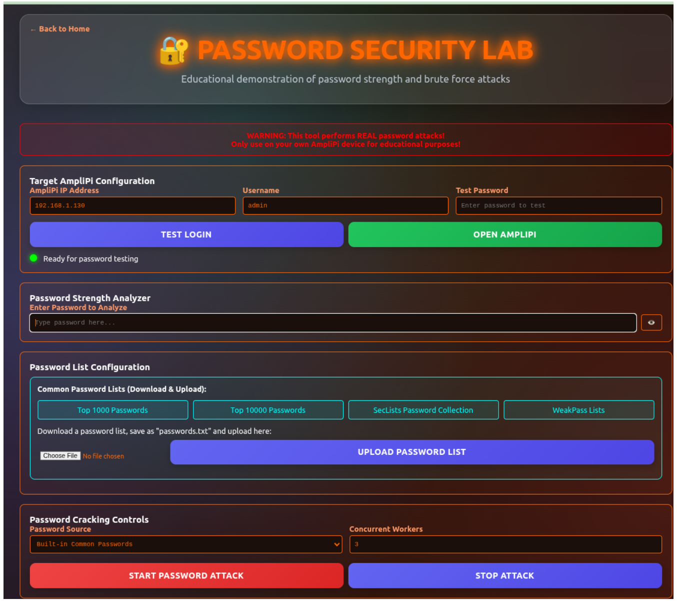
B. Learning Outcomes Assessment

1) Knowledge Acquisition: We implemented a real-time quiz system on our website (microelectronics2025.github.io) integrated with Google Sheets for immediate data collection. The assessment consisted of questions covering the microelectronics fundamentals. Figure 8 shows the quiz interface and Figure 9 shows the scores of different students on Google sheets.

The students demonstrated the strongest comprehension in areas where interactive visualizations were used.

2) Practical Skills Demonstration: During hands-on sessions, students successfully identified vulnerabilities in the AmpliPi system:

- **Authentication Weaknesses:** All students successfully used the dictionary attack tool to discover weak default passwords on testing AmpliPi devices. Figure 10 shows the user interface of the dictionary attack tool, where a password was found successfully and the attack was completed successfully.
- **DoS Vulnerabilities:** All students successfully executed SlowLoris attacks, observing service degradation. Figure 11 shows the user interface of the SlowLoris attack in progress.



PASSWORD SECURITY LAB

Educational demonstration of password strength and brute force attacks

WARNING: This tool performs REAL password attacks!
Only use on your own AmpliPi device for educational purposes!

Target AmpliPi Configuration

AmpliPi IP Address: 192.168.1.130

Username: admin

Test Password: Enter password to test

Ready for password testing

Password Strength Analyzer

Enter Password to Analyze: Type password here...

Password List Configuration

Common Password Lists (Download & Upload):

Download a password list, save as "passwords.txt" and upload here:

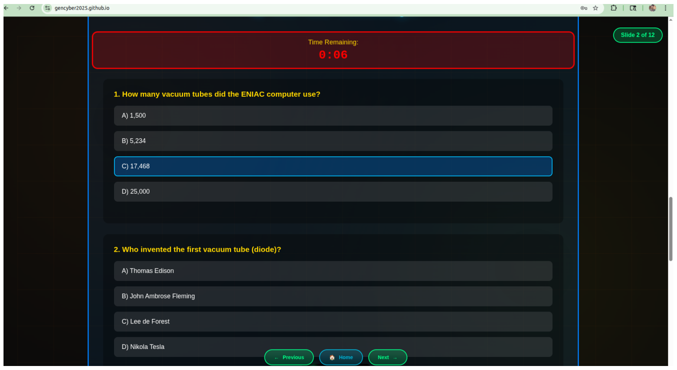
No file chosen

Password Cracking Controls

Password Source: Built-in Common Passwords

Concurrent Workers: 3

Fig. 7. Password security analyzer interface displaying real-time strength metrics, attempted passwords list, and educational feedback about password vulnerabilities



gencyber2025.github.io

Slide 2 of 12

Time Remaining:
0:06

1. How many vacuum tubes did the ENIAC computer use?

A) 1,500

B) 5,234

C) 17,468

D) 25,000

2. Who invented the first vacuum tube (diode)?

A) Thomas Edison

B) John Ambrose Fleming

C) Lee de Forest

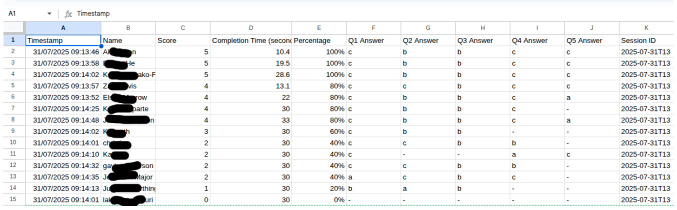
D) Nikola Tesla

Fig. 8. User Interface of Quiz

C. Educational Effectiveness

1) Conceptual Understanding: The metaphor of a restaurant for SlowLoris attacks proved particularly effective. Post-session discussions revealed that students could accurately explain denial-of-service concepts using metaphor, with one participant noting: “It is like someone sitting at all tables but never ordering food.” This conceptual mapping facilitated understanding of resource exhaustion attacks without requiring deep networking knowledge.

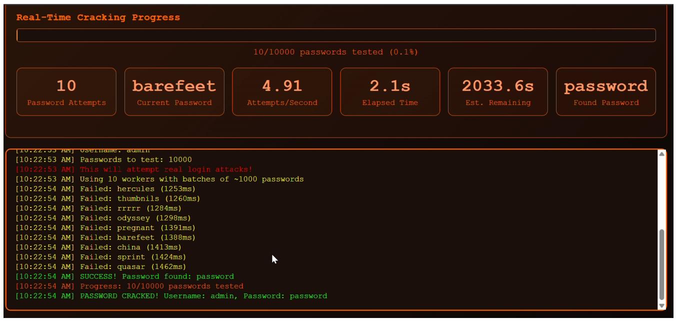
2) Engagement Sustainability: Unlike traditional lecture-based security education, the platform maintained consistent



A1 | fx Timestamp

	A	B	C	D	E	F	G	H	I	J	K
1	Timestamp	Name	Score	Completion Time (second	Percentage	Q1 Answer	Q2 Answer	Q3 Answer	Q4 Answer	Q5 Answer	Session ID
2	31/07/2025 09:13:46	[redacted]	5	10.4	100%	c	b	b	c	c	2025-07-31T13
3	31/07/2025 09:13:58	[redacted]	5	19.5	100%	c	b	b	c	c	2025-07-31T13
4	31/07/2025 09:14:02	[redacted]	5	28.6	100%	c	b	b	c	c	2025-07-31T13
5	31/07/2025 09:13:57	[redacted]	4	13.1	80%	c	c	b	c	c	2025-07-31T13
6	31/07/2025 09:13:52	[redacted]	4	22	80%	c	b	b	c	a	2025-07-31T13
7	31/07/2025 09:14:25	[redacted]	4	30	80%	c	b	b	c	-	2025-07-31T13
8	31/07/2025 09:14:48	[redacted]	4	33	80%	c	b	b	c	a	2025-07-31T13
9	31/07/2025 09:14:02	[redacted]	3	30	60%	c	b	b	-	-	2025-07-31T13
10	31/07/2025 09:14:01	[redacted]	2	30	40%	c	c	b	b	-	2025-07-31T13
11	31/07/2025 09:14:10	[redacted]	2	30	40%	c	-	-	a	c	2025-07-31T13
12	31/07/2025 09:14:32	[redacted]	2	30	40%	c	c	b	b	-	2025-07-31T13
13	31/07/2025 09:14:35	[redacted]	2	30	40%	a	c	b	c	-	2025-07-31T13
14	31/07/2025 09:14:13	[redacted]	1	30	20%	b	a	b	-	-	2025-07-31T13
15	31/07/2025 09:14:01	[redacted]	0	30	0%	-	-	-	-	-	2025-07-31T13

Fig. 9. Real-time quiz results showing student performance



Real-Time Cracking Progress

10/10000 passwords tested (0.1%)

10	barefeet	4.91	2.1s	2033.6s	password
Password Attempts	Current Password	Attempts/Second	Elapsed Time	Est. Remaining	Found Password

```
[10:22:53 AM] Username: admin
[10:22:53 AM] Passwords to test: 10000
[10:22:53 AM] This will attempt real login attacks!
[10:22:53 AM] Using 10 workers with batches of ~1000 passwords
[10:22:54 AM] Failed: hercules (1253ms)
[10:22:54 AM] Failed: thumbnils (1260ms)
[10:22:54 AM] Failed: rrrrr (1284ms)
[10:22:54 AM] Failed: odyssey (1298ms)
[10:22:54 AM] Failed: pregnant (1391ms)
[10:22:54 AM] Failed: barefeet (1388ms)
[10:22:54 AM] Failed: china (1413ms)
[10:22:54 AM] Failed: sprint (1424ms)
[10:22:54 AM] Failed: quasar (1462ms)
[10:22:54 AM] SUCCESS! Password found: password
[10:22:54 AM] Progress: 10/10000 passwords tested
[10:22:54 AM] PASSWORD CRACKED! Username: admin, Password: password
```

Fig. 10. User interface of the dictionary attack tool

engagement. The competitive quiz element, where the results were displayed in real time through Google Sheets projection, created positive peer competition and motivation.

D. Technical Performance

1) Platform Stability: GitHub Pages hosting provided reliable access to educational content, while the local Flask server managed attack tools without performance degradation. One notable issue occurred when multiple students simultaneously launched SlowLoris attacks, causing brief network congestion that required instructor intervention.

2) AI-Generated Code Quality: The generative AI produced functional code that required minimal post-generation modification. However, several issues appeared during the deployment.

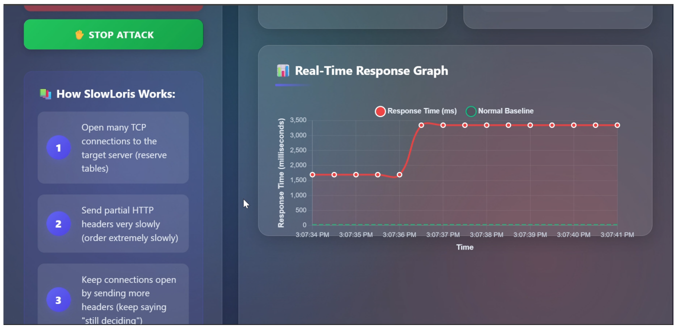
- Initial password detection logic failed with AmpliPi's 302 redirect responses
- CSS animations displayed inconsistently across different browsers

These issues were resolved through iterative testing and manual correction, highlighting the need for human oversight in AI-generated educational tools.

E. Security and Ethical Considerations

1) Controlled Environment: Network isolation was successful in preventing any attacks from affecting external systems. All dictionary attack attempts targeted only the designated AmpliPi devices (192.168.1.143-144), confirming the effectiveness of built-in target validation.

2) Student Behavior: Despite having functional attack tools, the students demonstrated responsible use. No attempts were made to bypass safety controls or target unauthorized systems. This suggests that the educational framework and ethical discussions successfully conveyed the appropriate boundaries.



1 Open many TCP connections to the target server (reserve tables)

2 Send partial HTTP headers very slowly (order extremely slowly)

3 Keep connections open by sending more headers (keep saying "still deciding")

Fig. 11. User interface of the SlowLoris attack tool

F. Limitations

Several limitations became apparent during deployment:

1) **Scalability:** The Flask server configuration limited concurrent users to 20, requiring session rotation for larger groups
2) **Hardware Dependency:** The curriculum assumes availability of AmpliPi devices, limiting reproducibility. However, other IoT devices can be used instead of AmpliPi devices. The modular design of the platform allows for adaptation to alternative targets such as smart home hubs, IP cameras, or other network-connected audio systems that expose web interfaces. Instructors would need to modify the target validation code and adjust attack parameters to match their chosen device's authentication mechanisms.
3) **Network Configuration:** Setting up isolated VLANs required significant instructor experience. Specifically, instructors need knowledge of: (a) VLAN configuration on managed switches to create network isolation, (b) firewall rules to prevent attacks from escaping the lab environment, (c) DHCP server setup to assign appropriate IP ranges to student machines, (d) routing table configuration to ensure traffic remains within the isolated network, and (e) monitoring tools to observe network traffic during exercises. In addition, instructors must understand how to disable the internet during attack exercises while maintaining local network functionality. These technical requirements may pose challenges for educators without networking backgrounds, particularly institutions lacking dedicated IT support for educational laboratories.

G. Comparison with Traditional Approaches

Although direct comparison with traditional teaching methods was outside our scope, informal feedback from instructors familiar with previous GenCyber camps suggested improved engagement and retention. The ability to immediately apply learned concepts through functional tools appeared to reinforce understanding more effectively than simulation-only approaches.

H. Generative AI as Educational Tool Developer

The use of generative AI reduced the development time from an estimated 3-4 months to 3 weeks. However, this efficiency came with trade-offs. The AI struggled with maintaining consistency across multi-file projects and occasionally produced pedagogically inappropriate explanations that required human intervention. The iterative refinement process, while faster than traditional development, still required significant technical expertise to evaluate and correct generated content.

I. Implications for Cybersecurity Education

This deployment demonstrates that AI-assisted development can produce effective cybersecurity educational materials. The combination of theoretical foundation (microelectronics evolution) with practical application (vulnerability testing) created a comprehensive learning experience. However, the requirement for technical review and ethical oversight suggests that generative AI serves best as an accelerator rather than a replacement for human expertise in educational content creation.

VI. Conclusion and Future Work

This work demonstrates that generative AI can substantially accelerate cybersecurity curriculum development while maintaining educational effectiveness. Using Claude AI [39], we reduced the development time from typical 3-4 months to 3 weeks while producing functional educational materials that successfully taught 40 K-12 students during GenCyber summer camps. The platform combined theoretical foundations in microelectronics with practical security testing, enabling students to identify real vulnerabilities in AmpliPi IoT devices through guided exercises. While these results are promising, the 40-participant sample represents an early-stage case study; future work with larger cohorts and longitudinal assessment would strengthen generalizability claims.

The deployment revealed both the potential and limitations of AI-assisted educational content generation. While Claude produced structurally sound code requiring minimal modification, domain-specific edge cases such as AmpliPi's non-standard HTTP response codes required human intervention. Artificial intelligence demonstrated unexpected pedagogical creativity through metaphors such as representing network connections as restaurant tables, which proved to be more effective than traditional technical explanations for teenage learners. This restaurant metaphor was autonomously generated by Claude when prompted to create educational scaffolding for denial-of-service concepts, demonstrating that generative AI can contribute novel pedagogical approaches beyond code generation. Safety considerations were successfully addressed through Claude's constitutional AI training [40], which consistently refused to generate unbounded attack code while still producing functional demonstrations.

In the future, several areas warrant investigation. Longitudinal studies should examine whether AI-generated materials produce knowledge retention comparable to traditionally developed curricula over 6-12 month periods. The current platform provides uniform content regardless of student background, suggesting opportunities for adaptive difficulty scaling based on individual performance metrics. Although we used Claude exclusively, systematic comparison with other models such as GPT-4 [41] or Gemini [42] could identify model-specific strengths for different educational objectives.

The methodology's success with AmpliPi systems suggests potential expansion to additional IoT platforms, although maintaining safety while diversifying targets presents challenges when considering devices like personal smart speakers that raise privacy concerns. Rather than using artificial intelligence for initial generation followed by human review, future iterations should explore true collaborative development, where educators provide pedagogical frameworks, while AI generates technical implementations in real time. This approach could combine human expertise in learning theory with AI's capacity for rapid content generation.

Acknowledgment

This work is supported in part by the National Science Foundation (NSF) under the award numbers 2146280, 2434784, 2434785, and 2146498. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

References

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] C. Willems and C. Meinel, "Practical network security teaching in an online virtual laboratory," in *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer ..., 2011, p. 1.

[3] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer networks*, vol. 76, pp. 146–164, 2015.

[4] MicroNova, *AmpliPi/AmpliPro User Manual*, 2024, accessed: June 2025. [Online]. Available: https://www.micro-nova.com/amplipi

[5] ——, "AmpliPi: Open Source Whole House Audio System," https://github.com/micro-nova/AmpliPi, 2024, open source multi-channel audio application.

[6] V. Švábenský, P. Čeleda, J. Vykopal, and S. Brišáková, "Cybersecurity knowledge and skills taught in capture the flag challenges," *Computers & Security*, vol. 102, p. 102154, 2021.

[7] P. Denny, J. Prather, B. A. Becker, J. Finnie-Ansley, A. Hellas, J. Leinonen, A. Luxton-Reilly, B. N. Reeves, E. A. Santos, and S. Sarsa, "Computing education in the era of generative ai," *Communications of the ACM*, vol. 67, no. 2, pp. 56–67, 2024.

[8] K. Hilliard, X. Yuan, K. S. Bryant, J. Xu, and J. Zhang, "Using gamification to enhance mastery of network security concepts," *Journal of Cybersecurity Education, Research and Practice*, vol. 2024, no. 1, July 2024.

[9] J. Zhang, X. Yuan, J. Johnson, J. Xu, and M. Vanamala, "Developing and assessing a web-based interactive visualization tool to teach buffer overflow concepts," in *Proceedings of the Frontiers in Education Conference*, Uppsala, Sweden, October 2020.

[10] J. Xu, X. Yuan, S. Nallela, K. Hilliard, and J. Zhang, "Learn ARP spoofing attack in a game," in *Proceedings of the Frontiers in Education Conference*, Uppsala, Sweden, October 2022.

[11] P. Weanquoi, J. Zhang, X. Yuan, J. Xu, and E. J. Jones, "Learn access control concepts in a game," in *Proceedings of the Frontiers in Education Conference*, Lincoln, NE, USA, October 2021.

[12] N. C. W. Framework, "Nice cybersecurity workforce framework (ncwf): Draft nist special publication 800-181," Tech. Rep.

[13] W. Du and R. Wang, "Seed: A suite of instructional laboratories for computer security education," *Journal on Educational Resources in Computing (JERIC)*, vol. 8, no. 1, pp. 1–24, 2008.

[14] National Security Agency. (2024) GenCyber program overview. [Online]. Available: https://www.gen-cyber.com

[15] J. Vykopal, R. Ošlejšek, P. Čeleda, M. Vizvary, and D. Tovarňák, "Kypo cyber range: Design and use cases," 2017.

[16] C. Justice and R. Vyas, "Cybersecurity education: Runlabs rapidly create virtualized labs based on a simple configuration file," in *2017 ASEE Annual Conference & Exposition*, 2017.

[17] TryHackMe. (2024) TryHackMe learning paths. [Online]. Available: https://tryhackme.com

[18] Hack The Box. (2024) Hack The Box academy. [Online]. Available: https://academy.hackthebox.com

[19] PentesterLab. (2024) PentesterLab exercises. [Online]. Available: https://pentesterlab.com

[20] OWASP Foundation. (2024) WebGoat project. [Online]. Available: https://owasp.org/www-project-webgoat/

[21] ——. (2020) IoTGoat - deliberately insecure firmware. [Online]. Available: https://github.com/OWASP/IoTGoat

[22] Carnegie Mellon University. (2024) picoCTF - computer security competition. [Online]. Available: https://picoctf.org

[23] Z. C. Schreuders, T. Shaw, A. Mac Muireadhaigh, and P. Staniforth, "Hackerbot: Attacker chatbots for randomised and interactive security labs, using {SecGen} and {oVirt}," in *2018 USENIX Workshop on Advances in Security Education (ASE 18)*, 2018.

[24] C. E. Irvine, M. F. Thompson, and K. Allen, "CyberCIEGE: gaming for information assurance," in *IEEE Security & Privacy*, vol. 3, no. 3. IEEE, 2005, pp. 61–64.

[25] B. D. Cone, C. E. Irvine, M. F. Thompson, and T. D. Nguyen, "A video game for cyber security training and awareness," *computers & security*, vol. 26, no. 1, pp. 63–72, 2007.

[26] C. E. Irvine, M. F. Thompson, and K. Allen, "Cyberciege: an information assurance teaching tool for training and awareness," in *Federal information systems security educators' association conference, North Bethesda, MD*, 2005.

[27] T. Crow, A. Luxton-Reilly, and B. Wuensche, "Intelligent tutoring systems for programming education: a systematic review," pp. 53–62, 2018.

[28] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[29] GitHub. (2024) GitHub Copilot. [Online]. Available: https://github.com/features/copilot

[30] M. Kazemitabaar, J. Chow, C. K. T. Ma, B. J. Ericson, D. Weintrop, and T. Grossman, "Studying the effect of ai code generators on supporting novice learners in introductory programming," in *Proceedings of the 2023 CHI conference on human factors in computing systems*, 2023, pp. 1–23.

[31] G. Agrawal, K. Pal, Y. Deng, H. Liu, and Y.-C. Chen, "Cyberq: Generating questions and answers for cybersecurity education using knowledge graph-augmented llms," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 21, 2024, pp. 23 164–23 172.

[32] M. M. Yamin, E. Hashmi, M. Ullah, and B. Katt, "Applications of llms for generating cyber security exercise scenarios," *IEEE Access*, 2024.

[33] M. Mukherjee, J. Le, and Y.-W. Chow, "Generative ai-enhanced intelligent tutoring system for graduate cybersecurity programs," *Future Internet*, vol. 17, no. 4, p. 154, 2025.

[34] J. Wei-Kocsis, M. Sabounchi, G. J. Mendis, P. Fernando, B. Yang, and T. Zhang, "Cybersecurity education in the age of artificial intelligence: A novel proactive and collaborative learning paradigm," vol. 67, no. 3. IEEE, 2023, pp. 395–404.

[35] S. Siboni, V. Sachidananda, Y. Meidan, M. Bohadana, Y. Mathov, S. Bhairav, A. Shabtai, and Y. Elovici, "Security testbed for internet-of-things devices," *IEEE transactions on reliability*, vol. 68, no. 1, pp. 23–44, 2018.

[36] B. Pearson, L. Luo, C. Zou, J. Crain, Y. Jin, and X. Fu, "Building a low-cost and state-of-the-art iot security hands-on laboratory," in *IFIP International Internet of Things Conference*. Springer, 2019, pp. 289–306.

[37] A. Ravishankar Rao and D. Clarke, "Capacity building for a cybersecurity workforce through hands-on labs for internet-of-things security," in *National Cyber Summit*. Springer, 2019, pp. 14–29.

[38] A. R. Rao and A. Elias-Medina, "Designing an internet of things laboratory to improve student understanding of secure iot systems," *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 154–166, 2024.

[39] Anthropic. (2025) Claude 4 model card. [Online]. Available: https://www.anthropic.com/claude

[40] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon *et al.*, "Constitutional ai: Harmlessness from ai feedback," *arXiv preprint arXiv:2212.08073*, 2022.

[41] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[42] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.