University of Idaho Cyber Security Tutorials

Final Report

University of Idaho, Department of Computer Science

Mentor:

Ananth Jillepalli - Washington State University

Your Team's Name & Logo

Darasimi Ogunbinu-Peters

Isabella Sunderman

CptS 432 Cybersecurity Capstone Project

Fall 2025

TABLE OF CONTENTS -

I. Introduction

Cybersecurity education is increasingly recognized as a critical need; however, many academic programs remain heavily focused on theory rather than practical applications. A gap is created between what students learn in classrooms and the hands-on skills required in industry. The CERES (Cybersecurity Education RESources) project was established in 2015 to address this gap by hosting open-source tutorials and walkthroughs that help students gain practical experience in core cybersecurity domains and practice offensive and defensive techniques in controlled and ethical settings.

Our project continues this effort by developing tutorials in the area of Wireless and Distributed Systems security. Specifically, we aim to design step-by-step, reproducible guides that teach students how to analyze, test, and defend against vulnerabilities in wireless networks and distributed environments. Past CERES teams have developed tutorials on topics including desktop penetration testing, web application testing, network vulnerability scanning, and firewall management. Building on this foundation, our team will expand coverage into new areas that align with industry needs, with a focus on WPA2/3 wireless penetration testing and mobile application security testing.

The motivation behind this work is twofold: first, to help students and professionals practice cybersecurity concepts in a safe and structured lab environment, and second, to create resources that make advanced security concepts more accessible to a broader audience. By bridging the gap between theory and practice, these tutorials contribute to workforce readiness and support the growing demand for applied cybersecurity skills.

The goals of our project are:

1. To produce at least two high-quality, reproducible tutorials that walk learners through a wireless or distributed systems security scenario.

2. To ensure the tutorials are documented in a way that allows replication by future students and professionals.

3. To host the tutorials in an open-source repository where they can be easily shared, improved, and expanded.

Our client is Dr. Daniel Conte de Leon, Associate Professor in the Department of Computer Science at the University of Idaho (contact: dcontedeleon@uidaho.edu), who oversees CERES project collaborations. He has been instrumental in shaping the long-term vision of CERES and ensuring that each iteration of student projects contributes meaningful educational resources.

Our mentor is Dr. Ananth Jillepalli, Scholarly Assistant Professor in the School of Electrical Engineering and Computer Science at Washington State University (contact: ananth.jillepalli@wsu.edu). Dr. Jillepalli provides weekly coaching, guidance on technical direction, and feedback to ensure that our tutorials meet academic and client expectations.

Together, this collaboration between WSU and the University of Idaho provides both academic oversight and real-world applicability, ensuring that the deliverables are practical, replicable, and valuable to the cybersecurity community and well as ethically responsible

I. Project Requirements Specification

Building on the project background, motivation, and goals described in the introduction, we now define the concrete requirements that guide our work. These requirements ensure that our tutorials not only address the gap between academic theory and industry practice but also remain reproducible, accessible, and aligned with the broader mission of the CERES project. The project requires knowledge research to be conducted by ourselves to ensure accurate information is provided as well as the tools are resources are reliable. As a goal for the project to be open source the finding tools that won't harm the systems being tested or used during the activities.

I.1. Project Stakeholders

Client: Dr. Daniel Conte de Leon (University of Idaho, Department of Computer Science)
Needs tutorials that bridge the gap between academic coursework and industry-required hands-on skills. He expects deliverables to be reproducible and directly helpful in teaching cybersecurity.

Mentor: Dr. Ananth Jillepalli (Washington State University, EECS)
Needs consistent updates and demonstrations that show progress. Ensures the tutorials meet academic rigor and are suitable for distribution through CERES.

Students (WSU and University of Idaho, current and future users)
Need easy-to-follow, step-by-step tutorials with screenshots, commands, and explanations that let them practice cybersecurity without requiring extensive prior experience.

Cybersecurity Education Community (CERES project)
Needs open-source, reusable, and scalable resources that expand the CERES library of cybersecurity tutorials for classroom, workshop, and self-study use.

RADICL / WSU EECS Lab Ops (facility owners)
Needs: Clear lab access requests and hardware requirements so the team can reserve VMs, racks, and physical lab space (Dana 117). Requires assurance that experiments and deauth activities are scoped, authorized, and documented.

Cyber Security Group / IEEE volunteers (equipment custodians)
Needs: A short hardware wishlist and workshop plan so they can lend parts (ESP32 boards, adapters) and vet the assembly tutorial for reproducibility in an outreach/workshop setting.

Instructor / Course Staff (course administrators)
Needs: Sprint-aligned deliverables (README updates, sprint report, client minutes) and reproducible artifacts that can be graded and re-used in future classes.

I.2. Use Cases

II.2.1 Mobile App Security

Student learners follow the environment setup for the static and dynamic environments appropriately. The student conducts the static analysis using MobSF using the provided IPA and APK test files and is able to interpret the generated reports. The student conducts a dynamic analysis using Android Studio and Frida and the provided APK. The student then can perform code injection and utilize the frida-trace and frida-discover to understand functions of the application.

Primary flow: Student discovers the mobile tutorial → follows environment setup (MobSF, Android Studio, Frida) → performs static + dynamic analysis on provided test apps → documents findings in a lab report.

Acceptance criteria: Provided test APK/IPA run in the lab VM; students can generate and interpret MobSF reports. Successfully sets up Frida server on an emulated device and is able to find functions .

II.2.2 WPA2/3 Penetration Testing

This project concentrates on creating reproducible, pedagogical tutorials for practical wireless security testing, with a primary focus on WPA2 and WPA3. The team's first deliverable is an ESP32 Marauder-based lab tutorial that documents end-to-end assembly, firmware flashing, and controlled handshake capture workflows. The tutorial demonstrates how to capture WPA2 EAPOL handshakes from an authorized test AP, convert captures for GPU cracking (hcxtools → `hashcat`), and analyze the capture using Wireshark. The narrative also contrasts WPA2's 4-way handshake with WPA3's SAE (Dragonfly) mechanism and explains transition-mode vulnerabilities. All steps are designed for use in an authorized lab environment (e.g., RADICL or Dana 117) and are accompanied by reproducible commands, screenshots, and ethics guidance. The detailed tutorial and working drafts are hosted in the project repository:
https://github.com/zoom1338/UofI-CyberTutorials/tree/WPA_2/3-v2

Primary flow: Student finds the WPA2/3 tutorial → sets up Kali VM or uses RADICL VM → follows ESP32 Marauder assembly and flashing steps → captures an authorized handshake → converts capture to hash format → performs offline cracking/analysis → documents ethical scope and remediation suggestions.

Acceptance criteria:

1. At least one reproducible sample capture (`.pcap`) and Wireshark screenshots showing a complete 4-way handshake.
2. A working conversion workflow (hcxtools → `hash.22000`) and a successful Hashcat command example (runs on provided sample).
3. Clear "Ethics & Scope" section and a signed lab authorization checklist for live captures.

II.2.3 Additional use cases (cross-tutorial)

- Instructor / Workshop leader uses the tutorials to run an in-class lab or workshop (materials & parts checklist available).
- CERES community contributor forks the repo to expand the tutorial (e.g., add WPA2-Enterprise, additional wordlists, or hardware options).

I.3. Functional Requirements

Reproducible step-by-step tutorials for each topic (WPA2/3 and Mobile App Security) with explicit prerequisites, terminal commands, and expected outputs.

Tools & Software: aircrack-ng, hcxtools / hcxpcapngtool, hashcat, Wireshark, airodump-ng/aireplay-ng, Nmap, SQLMap, MobSF, Frida, Kali Linux (VM).

Hardware lists & part sourcing: ESP32 Marauder BOM, recommended Wi-Fi USB adapters (monitor + injection support), power bank options, and optional touchscreen wiring diagram.

Artifacts: sample PCAPs in `Experiments/sample-pcaps/`, research logs in `Notes/`, draft tutorials in `Drafts/`, and final tutorials in `FinalTutorial/`.

Documentation: README updated each sprint, sprint report, client meeting minutes, and an ethics checklist included with each tutorial.

Version control & hosting: All content maintained in GitHub (repo + branches) with tags/releases for major tutorial versions.

I.4. Non-Functional Requirement

Reproducibility: Steps must run on a standard Kali VM or RADICL lab VM with minimal environment tweaks.

Usability / Comprehension: Written for learners with basic Linux familiarity; commands are copy-paste friendly; include annotated screenshots and expected outputs.

Portability: Tutorials should be executable in either a local VM, RADICL environment, or a lab workstation (document OS/tool differences).

Maintainability: Modular repo structure (Notes/, Drafts/, Experiments/, FinalTutorial/) and conventions for adding future content (templates for logs, checklists, screenshots).

Security & Ethics: Each tutorial must include explicit authorizations for live testing, a disclaimer on legal/ethical constraints, and safe default settings (work with sample files when hardware is unavailable).

Licensing & Distribution: Tutorials released under an open educational license appropriate for CERES (recommendation: CC-BY-NC-ND for educational reuse; document chosen license in `LICENSE.txt`).

II. Software Design - From Solution Approach

This section describes the final design of our project, which is structured around producing reproducible cybersecurity tutorials for the CERES community. The system is not a traditional software application but a framework of tutorials, tools, and supporting resources organized in a GitHub repository. Our design emphasizes reproducibility, modularity, and educational usability.

II.1. Workflow Design

Revise and include Section II from your Solution Approach report here. Provide the block diagram of your architecture and give a brief description of it.

II.1.1. Overview

At a high level, our project workflow follows these steps:

1. **Research and Scoping:** Identify relevant topics in wireless penetration testing (WPA2/3) and mobile application security (static and dynamic analysis).

2. **Experimentation and Data Collection:** Perform controlled lab experiments, using sample packet captures for WPA2/3 and sample APKs for mobile app analysis.

3. **Tutorial Development:** Translate technical steps into tutorials with screenshots, annotated commands, and ethics disclaimers.

4. **Review and Iteration:** Share drafts with mentor and client for feedback, update based on academic and technical requirements.

5. **Delivery:** Publish tutorials in the open-source GitHub repository, organized into Notes, Drafts, Experiments, and Final Tutorial directories.

This workflow allows both team members to develop tutorials in parallel while maintaining a shared structure and quality standard.

In this sprint, the workflow was exercised primarily through the ESP32 Marauder build process. The steps included sourcing and soldering hardware, wiring the TFT display to the ESP32 board, installing the CP210x driver, flashing the Marauder firmware, and verifying the first successful boot sequence. These actions correspond to the "Experimentation and Data Collection" and "Tutorial Development" phases of the workflow, forming the foundation for the complete WPA2 handshake capture and analysis tutorial.

II.1.2. Subsystem Decomposition

Our project decomposes naturally into 3 tutorial subsystems:

- **Subsystem A: ESP32 Marauder (Darasimi)**
  This subsystem focuses on the **hardware component** of the wireless tutorial. It involves physically constructing, wiring, and flashing an ESP32 Marauder device capable of capturing WPA2/3 network handshakes. The device uses a 2.8" TFT touchscreen (ILI9341), SD card, and jumper/breadboard connections. The subsystem includes a detailed wiring diagram (`Deliverables/images/wiring_cheat_sheet.png`), photos of the build process, and a step-by-step assembly tutorial. The goal is to provide students with an accessible way to replicate a low-cost wireless penetration testing tool and understand its role in network security analysis.
- **Subsystem B: WPA2/3 Wireless Penetration Testing (Darasimi)**
  This subsystem builds directly on the ESP32 Marauder device and focuses on **software-driven experimentation**. It documents the full workflow: scanning networks, capturing the WPA2 4-way handshake, transferring `.pcap` files from the Marauder SD card, converting them using `hcxtools` or `cap2hashcat`, and attempting password recovery using `hashcat`. Each step includes reproducible commands, ethics disclaimers, and validation through Wireshark packet analysis. Sample captures and conversion outputs are stored under `Experiments/sample-pcaps/`. This subsystem also outlines the upcoming extension into WPA3 SAE handshake testing and KRACK reinstallation demonstrations.
- **Subsystem C: Mobile Application Security Testing (Isabella)**
  Focus on static and dynamic analysis of Android applications and iOS applications. Tools include Mobile Security Framework (MobSF) , Frida, and Android Studio.
  Tutorial will walk learners through inspecting IPAs and APKs for insecure configurations. The static analysis focuses on report generation and interpretation while the dynamic analysis focuses on monitoring runtime behavior in an emulator tracing functions of an application and code injection.

Each subsystem feeds into the **FinalTutorial** directory as polished, step-by-step, reproducible guides.

II.2. Data design

Our project does not involve designing custom databases, but does require handling structured experiment data and captures:

- **For ESP32 Marauder tutorials:**
  - **Flashing logs (`.txt`) and screenshot documentation of `esptool.py` output for firmware validation.**

- **Hardware inventory YAML file (`hardware_config.yaml`) to help future students replicate exact component mappings.**
- **Image assets (JPG/PNG) showing wiring setup, soldering corrections, and display connection issues documented under `Deliverables/images/`.**
- **Parts list**: Required components for device and wiring diagram
- **CSV exports** from Airodump-ng: Include network metadata (BSSID, channel, encryption type, signal strength).
- **Wordlists**: Approved password lists for cracking demonstrations (stored securely and referenced in documentation).

- **For WPA2/3 tutorials:**
  - **PCAP files**: Contain WPA2 handshakes for Wireshark and Aircrack-ng analysis.
  - **CSV exports** from Airodump-ng: Include network metadata (BSSID, channel, encryption type, signal strength).
  - **Wordlists**: Approved password lists for cracking demonstrations (stored securely and referenced in documentation).
- **For Mobile App tutorials:**
  - **IPA and APK files:** Application files for static analysis
  - **APK Application:** Used for Dynamic analysis to observe application function calls during runtime and code injection

All data is stored in `Experiments/` and linked to specific tutorials, ensuring reproducibility.

II.3. Reproducibility design

The user interface for our project is not a graphical application but a structured repository and tutorial design. To maximize reproducibility, we have made the following design decisions:

- **GitHub Repository Structure:**
  - `Notes/`: Raw research logs and meeting notes.
  - `Drafts/`: Work-in-progress tutorials and requirements documents.
  - `Experiments/`: Lab setup instructions, PCAPs, APKs, and test outputs.
  - `Final Tutorial/`: Polished tutorials ready for delivery.
- **Tutorial Format:**

  - Markdown files with consistent section headers (Overview, Lab Setup, Steps, Ethics & Scope).
  - Embedded screenshots with labeled placeholders until final testing is complete.
  - Code snippets and commands are formatted in fenced blocks for easy copy/paste.
  - Ethics disclaimer included in every tutorial.
- **Hardware guides:**
  - Tutorials will include notes about how to replicate experiments with loaned hardware (e.g., Wi-Fi adapters, Android devices/emulators).
  - Students outside the club can still reproduce results using sample files and emulator setups provided in the repo.

For the ESP32 Marauder tutorial, reproducibility was tested by following the documentation on two different machines: a Windows 11 laptop (with CP210x driver and Python environment) and a Kali Linux VM. This confirmed cross-platform capability for firmware flashing and data transfer. Future iterations will include a Docker-based reproducibility test script that verifies environment setup, package dependencies, and the presence of required firmware binaries in the `bins/` directory.

This design ensures that tutorials remain accessible, reproducible, and maintainable over time.

III.Test Case Specifications and Results

III.1. Testing Overview

Include a summary of your test objectives and test plans.

Describe the overall approach to testing and provide the overall flow of the testing process. An example is provided in an Appendix.

Are you using Continuous Integration (CI) and/or Continuous Delivery (CD) in your testing? What tools are you using, and what kinds of testing do they provide? Unit tests, integration, system, user interface, speed/non-functional requirement testing, user testing, etc.

In all of these major categories, include at least some material about your implementation, how it is currently executing, and/or what future work would be needed to ensure it happens for this project.

- Activities and non-guided challenges solution and answers
- Objectives
- Obtainment testing
- Environmental testing

The primary objective of testing in our project is to ensure that each tutorial is reproducible, accurate, and educationally valuable for students with limited prior experience. Because our "software system" is structured as tutorials rather than a single application, testing focuses on verifying that each documented workflow executes successfully in a controlled environment.

Our testing approach includes:

During Sprint 2, testing primarily validated hardware assembly and firmware communication. The ESP32 Marauder successfully enumerated as COM3 after driver installation, and the flashing script (`c5_flasher.py`) ran through dependency installation and device detection. Flashing initially failed due to a mismatch between the binary (ESP32-C5) and the detected chip (ESP32-WROOM). This will be corrected by using the proper Marauder firmware variant. Once flashed, functional testing will include verifying touchscreen response, SD card logging, and capture initiation from the "Wi-Fi Sniffers" menu.

- **Prototype validation:** Confirming that all commands and steps in the tutorials can be run without errors in a fresh environment.
- **Environment verification:** Ensuring tools install correctly in Kali Linux and sample data (e.g., PCAPs, APKs) can be analyzed.
- **Reproducibility testing:** Running tutorials multiple times on clean VMs to verify consistent outcomes.
- **Educational validation:** Informal peer testing, where students follow the tutorials and provide feedback on clarity and usability.

We are not implementing Continuous Integration (CI) pipelines for this project, as it is not code-driven in the traditional sense. However, we use GitHub version control to track changes and updates to tutorial files. Future work could include automated reproducibility tests (e.g., scripted environment setup and file verification).

Categories of testing:

- **Unit Testing (tutorial-level):** Each command and workflow step is executed independently to verify correctness.
- **Integration Testing:** Steps are run in sequence (e.g., capturing → analyzing → cracking a WPA2 handshake).
- **System Testing:** Entire tutorial run-throughs confirm that a learner can start from the setup stage and reach the final outcome.
- **Usability Testing:** Informal peer testing ensures instructions are understandable and free of ambiguity.
- **Non-Functional Testing:** Reproducibility and portability confirmed by executing tutorials in multiple environments (personal VMs and, later, Cybersecurity Club labs).

III.2. Environment Requirements

Specify both the necessary and desired properties of the test environment. The specification should contain the physical characteristics of the facilities, including the hardware, the communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Identify special test tools needed.

WPA2/3 Tutorial (Darasimi):

- **Hardware:** USB Wi-Fi adapter with monitor and injection support (e.g., Alfa AWUS036NHA).
- **Special hardware note:** Breadboard layout follows pin mappings in the original ESP32 Marauder schematic (`Deliverables/images/wiring_cheat_sheet.png`). Portable power source integration (5V USB bank) is planned to enable field demonstrations without tethering to a PC.
- **Software:** Kali Linux VM (2025.2), VirtualBox/VMware, Aircrack-ng, Wireshark, Hashcat, hcxtools.
- **Data:** Sample WPA2 handshake PCAPs (public) prior to live captures.
- **Mode of usage:** Stand-alone VM or Cybersecurity Club lab with authorized APs.
- **Special tools:** Wordlists for password cracking (approved and scoped ethically).

Mobile App Security Tutorial (Isabella):

IV. **Hardware:** Standard laptop with virtualization support.

V. **Software:** Android Studio, MobSF (Mobile Security Framework), Docker, Node.js, Android Studio, Frida. VM can be used but tutorial is tested on Windows 11 and Ubuntu 24.

VI. **Data:** Sample APK and IPA files provided by MobSF and APK

VII. **Mode of usage:** Emulator-based analysis (no live production apps).

VIII. **Special tools:** Debugging frameworks and mobile runtime instrumentation utilities.

VIII.1. Test Results

Include your prototype test results from your “Test Case Specifications and Results” document, as updated with your current project status.

As of this sprint, the ESP32 Marauder has been fully assembled on the breadboard, the TFT display pins soldered, and all wiring verified using the provided schematic. The CP210x driver installation enabled COM port recognition, and the flasher executed up to chip verification. The next immediate milestone is to flash the correct firmware version and validate that the Marauder boots into the expected on-screen interface. Subsequent testing will verify network scanning, deauthentication simulation (in authorized lab settings), and successful handshake capture saved to the SD card. Preliminary screenshots and assembly documentation are stored in the `Deliverables/images/` directory.

IX. Projects and Tools used

Include a summary of the libraries, frameworks, and tools you used to implement your project. For example, you could have used some web frameworks, a database, a network message passing tool, graphics generation libraries, and various operating system platforms. Please list these out with a short one-sentence note about what it was used to build/support in your project.

Tool/library/framework	Quick note on what it was for
MobSF	Used for the Mobile App Security Tutorial, Static
Docker	Required for MobSF
Frida	Used for the Mobile App Security Tutorial, Dynamic
Node.js	Required for Frida
Android Studio	Used for the Mobile App Security Tutorial, Dynamic
Wire Shark	Used to inspect WPA2 handshake captures and verify EAPOL message sequences
Aircrack-ng Suite	Used for WPA2 capture and crack workflow validation
hcxtools / hcxpcapngtool	Converts captured WPA2/3 handshakes to hash formats for offline analysis
Hashcat	Performs offline password cracking on converted WPA2 hashes
ESP32 Marauder Firmware	Provides on-device Wi-Fi scanning, deauthentication, and packet capture features
esptool.py / c5_flasher.py	Used to flash firmware onto the ESP32 module
VirtualBox / VMware	Hosts the Kali Linux and Ubuntu environments for testing and reproducibility
CP210x USB Driver	Enables ESP32 serial communication for firmware flashing
Markdown / GitHub Pages	Used for tutorial documentation, hosting, and collaboration

Discord & Zoom	Used for team communication, sprint check-ins, and live debugging

X. Description of Final Prototype

XI.1 ESP32 Marauder Assembly

The ESP32 Marauder was constructed on a breadboard using male header pins, jumpers, and a 2.8” ILI9341 TFT display. The pins were soldered and wired according to the official Marauder schematic (referenced as `Deliverables/images/wiring_cheat_sheet.png`). After verifying all connections, the board was flashed using `esptool.py` with the CP210x USB driver installed. Photos of each stage (soldering, wiring, first boot) are included in the Deliverables folder.

XI.1.1 Firmware and Verification

Once flashed, the Marauder’s touchscreen interface displayed the main menu, confirming successful setup. The prototype includes functionality for scanning nearby Wi-Fi networks, capturing EAPOL handshakes, and saving `.pcap` files to the onboard SD card. These captures can be exported via USB for further analysis using Wireshark and `hcxtools`.

XI.1.2 WPA2 Capture & Analysis Workflow

1. **Capture:** The Marauder is used to select an access point and capture a handshake when a client reconnects.
2. **Transfer:** The captured `.pcap` file is moved from the SD card to a Kali VM.
3. **Convert:** `hcxpcapngtool` converts the capture into a Hashcat-compatible `.22000` format.
4. **Crack:** Hashcat performs offline password recovery using an approved educational wordlist.
5. **Analyze:** Wireshark confirms the presence of the 4-way EAPOL handshake.

Each of these steps has been documented in Markdown with screenshots, commands, and expected outputs to ensure reproducibility.

XI.1.3 Planned Additions

Future enhancements include adding a **portable power bank** for standalone use, testing WPA3 (SAE) handshakes, and implementing SD card logging for experiment reproducibility. All prototype-related assets, including wiring photos (`wired_marauder.jpg`), first boot screenshots, and setup notes, are stored under `Deliverables/images/`.

XI.2 Mobile App Security

The tutorial covers an environment setup for both the static and dynamic analyses of mobile applications. The tutorial utilizes MobSF and Frida for conducting the security analysis with Docker and Android for the environment. The tutorial has been testing on VM and personal host machines with success in the environment setup. Android applications are the focus of the tutorials due to the availability of testing materials. MobSF's use in the static tutorial highlights vulnerabilities found in the APK and IPA files and how to interpret the sections of the reports that can lead into the Dynamic analysis using frida-discover and frida-trace to find hidden functions of the application and inappropriate memory accesses.

X. Social Responsibility and Broader Impacts

The collection of tutorials produced are educational uses for learners to utilize and build their cyber security skillset to meet industry standards. The topics covered in the tutorials are built ethically utilizing vetted tools and software. Cyber security education is a continuous uphill battle to catch up with the industry but to do so safely and legally while having access to limited resources.

In particular, the WPA2/3 tutorial emphasizes ethical hacking practices and scope restriction. All experiments are performed on authorized access points under controlled lab conditions. The ESP32 Marauder's capabilities are framed purely for educational demonstration, helping students understand wireless security threats while reinforcing responsible disclosure and defensive mitigation practices.

XI. Product Delivery Status

When was/will the project be delivered to your client? Was it demonstrated, and who did you hand it off to?

Also include project location information here. This is for both the instructor and your client as future reference. This should include:

1) Source repositories
2) Equipment storage location - where did you leave any client or course materials
3) Any other materials needed to rebuild your project from the ground up

Ensure that either here or in your detailed description, you include any instructions or where the instructions are to install and set up your project, so future users know where to start.

The Sprint 2 deliverable, consisting of the ESP32 Marauder assembly tutorial and supporting documentation, has been uploaded to the project GitHub repository under the `WPA_2/3-v2` branch:

Repository: https://github.com/zoom1338/UofI-CyberTutorials/tree/WPA_2/3-v2

Delivery Date: October 11, 2025
Demonstration: The hardware progress and tutorial walkthrough were demonstrated during the client meeting with Dr. Jillepalli and Aj at WSU EECS.

Equipment Storage: The ESP32 Marauder device and TFT display are currently stored in the WSU Cybersecurity Club lab space (Dana 117). Spare jumper wires, SD cards, and the flasher cable remain in the same workspace.

Rebuild Instructions:

XII. Firmware flashing guide and wiring diagrams: `FinalTutorial/esp32_marauder_tutorial.md`

XIII. Sample captures and conversion examples: `Experiments/sample-pcaps/`

XIV. Dependencies and drivers: linked in the README.md under “Hardware Setup”

XV. Conclusions and Future Work

XV.1. Limitations and Recommendations

Limitations to this project include resource access; although the tools and tutorials themselves are open-source, performing the tasks requires specific hardware.

The ESP32 Marauder currently runs only WPA2 capture and deauthentication modes. WPA3 (SAE) support requires a firmware update and additional testing. The device also has limited logging capabilities currently only storing packet capture files. The fragility of the hardware is also of concern and requires knowledge and tools of proper soldering and a device enclosure to protest the breadboard setups.

The Mobile App security testing is limited to Android due to the restrictions from Apple. Several of the Apple IPA files available from trust-worthy sources weren't vulnerable due to the low complexity of the application. Dynamic testing of an iOS application also requires a physical device which would put users' devices at risk or using a paid service such as Corellium with an extensive vetting process to obtain an account.

XV.2. Future Work

Include a conclusion and discuss the future work. Future work should include a discussion on how to extend this project. For those of you who are self or EECS-sponsored, discuss commercialization possibilities.

Finalize firmware flashing with the correct ESP32 variant to resolve chip mismatch issues.

Implement automated SD card logging and hash export to reduce manual transfer errors.

Expand the tutorial to include **KRACK attack** demonstrations and WPA3 transition mode testing.

Integrate a portable battery source for standalone operation.

Produce a detailed **step-by-step assembly video** for educational reuse in CERES workshops.

Conclusion

The project demonstrates the feasibility of combining open-source hardware and reproducible lab documentation to teach advanced wireless security concepts. By integrating the ESP32 Marauder into the CERES framework, students can gain hands-on penetration testing experience in a safe, ethical, and cost-effective way.

XVI. Acknowledgements

Thank the people who have contributed to your project. Also, thanks to your sponsors.

Special thanks to Aj Jillepalli for the guidance on integrating hardware-based tutorials into CERES and for feedback on the Marauder build process.

Thank you to the Washington State Cyber Security Club for the use of resources and club room for this project.

XVII. Glossary

XVIII. References

XIX. Appendix A – Team Information

Team Members & Bios

Darasimi Ogunbinu-Peters is a cybersecurity student at Washington State University pursuing a Bachelor of Science in Cybersecurity. His technical interests include wireless security, penetration testing, and digital forensics. For this project, Darasimi was responsible for designing and implementing the WPA2/3 Wireless Penetration Testing tutorial. His contributions include researching WPA2/3 handshake vulnerabilities, setting up lab experiments with

Aircrack-ng, Wireshark, and Hashcat, and documenting reproducible workflows that emphasize ethical use and reproducibility.

Isabella Sunderman is a cybersecurity student at Washington State University pursuing a Bachelor of Science in Cybersecurity. She is interested in machine learning, network architecture, and system administration. For the project, Isabella was responsible for designing the Mobile Application Security Tutorial on both static and dynamic scanning. Contributions include researching Mobile app security tools and resources, setting up Virtual Machines, and setting learning objectives and outcomes of tutorials.

XX. Appendix B - Example Testing Strategy Reporting

1) Identify the requirements being tested.
2) Either link to available online test results/or take screenshots of the various system testing results
3) User testing can be demonstrated via survey results, quotes, and a discussion of the feedback received

Test reporting was conducted through informal feedback between team members on tutorials, focusing on the categories of readability and reproducibility.

XXI. Appendix C - Project Management

Weekly Meetings with the instructor/mentor to discuss the previous week's progress and set goals for the current week, task progress, and non-immediate roadblocks. These meetings are designed to touch base. The project mentor also serves in the role of sponsor, clarifying project deliverables and maintaining a close connection to the project and previous contributions.

Weekly Team-only meetings to discuss more administrative, non-technical project tasks. These meetings provide an open space for communication about the mentor meeting and to update the project board, individual sprint deliverables, and allocate work amongst team members. The open forum of discussion also addressed non-immediate questions and communication between the team and mentor.

The team utilized GitHub, Discord, Zoom, and email for communication and project organization. The team's project GitHub repository also includes a GitHub Project where the team creates issues in the repository and tracks them on GitHub. This allowed team members to see what each other were working on without pulling up the commit log or interfering with each other's branches.