CYBERSECURITY PEDAGOGY & PRACTICE JOURNAL

Volume 1, No. 1
July 2022
ISSN: 2832-1006

In this issue:

iscap
Information Systems
& Computing
Academic Professionals

The **Cybersecurity Pedagogy and Practice Journal** (**CPPJ**) is a double-blind peer-reviewed academic journal published by **ISCAP** (Information Systems and Computing Academic Professionals). Publishing frequency is two times per year. The first year of publication was 2022.

CPPJ is published online (https://cppj.info). Our sister publication, the proceedings of the ISCAP Conference (https://proc.iscap.info) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the ISCAP conference. At that point papers are divided into award papers (top 15%), and other accepted proceedings papers. The other accepted proceedings papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the CPPJ journal. Currently the target acceptance rate for the journal is under 35%.

Questions should be addressed to the editor at editorcppj@iscap.us or the publisher at publisher@iscap.us. Special thanks to members of ISCAP who perform the editorial and review processes for CPPJ.

2022 ISCAP Board of Directors

CYBERSECURITY PEDAGOGY AND PRACTICE JOURNAL

Editors

An IoT Based New Platform for Teaching Web Application Security

Zhouzhou Li
zli2@semo.edu

Ethan Chou
echou1s@semo.edu

Charles McAllister
cdmcallister@semo.edu

The Deparment of Computer Science
Southeast Missouri State University
Cape Girardeau, MO 63701, U.S.

Abstract

Web application security is a core issue that must be addressed in cybersecurity degree programs to adequately prepare students for leadership in industry. To teach a “Web Application Security” course, a good exercise platform that can cover the context of Web application is crucial to the learning outcomes. Unfortunately, existing platforms cannot satisfy both cost and efficiency requirements. In this paper, a cost-effective and easy-to-use full-stack Web application platform, ESP32-CAM, is introduced to the course, which is an Internet of Things device with a built-in face recognition Web App. Our major contribution in this paper includes the thoughtful design of an exercise series around the platform, which can provide more hands-on practice in the class, strengthen students’ practical skills, and further inspire the students’ learning interests on a matured technique such as Web applications. Furthermore, through this platform students can explore the cutting-edge technologies in their class projects or capstone project, e.g., “transfer learning” to extend the face recognition to emotion recognition or generative adversarial network to fool the Artificial Intelligence model, which will greatly involve students in academic research.

Keywords: Web Application Security, Internet of Things, Artificial Intelligence, Reverse Engineering, Penetration Testing, Secure Software Development.

1. INTRODUCTION

According to International Telecommunication Union (Buyannemekh & Chen, 2021), by 2019, 53.6% of the world population had stable Internet access and enjoyed the wealth of information. With the Internet, a user who has no technical or engineering background can solve some technical challenges by using his/her World Wide Web (WWW, or just Web) browser to search the Internet for hints or answers. The Web is a critical application for the Internet. Due to its core role played on the Internet, Web application’s security is naturally a significant issue that needs to be addressed in the industry and in academic establishments.

In current usage, the Web becomes a de facto standard for Internet. Other Internet applications (such as email, instant messaging, interactive game, file transfer, cloud storage, etc.) either build themselves upon Web or provide their Web version of solutions. Figure 1 shows the context of Web application, where it depends on the lower-level Internet protocols and supports other Internet applications. Every component in this figure can potentially impact the security. Therefore, the Web Application Security should cover all.

Unfortunately, considering both cost and efficiency, it is difficult to find a suitable platform providing full-stack protocols for students to exercise during course study. The major concerns include:

- Only opening the (Web) application layer for Cybersecurity students to attack. No details for the implementation of the lower layers. Not to mention their vulnerabilities.
- Only provide an over simplified Web layer for Cybersecurity students to attack. Seemingly not a real system.
- Not free if the user wants to experience the advanced functionalities.

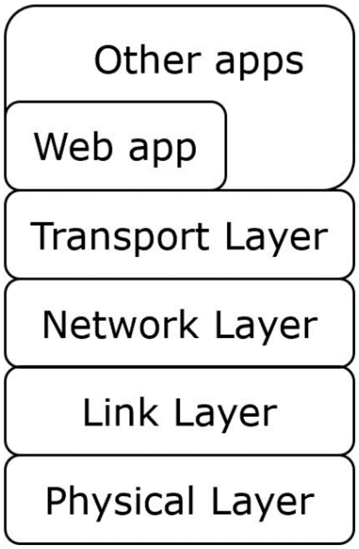


Figure 1: The Context of Web App Security

A good platform for students to practice Web application security but limiting the impact to the real networks need to be well designed.

The remainder of this paper is organized as follows. In the 'Literature Review' section, we review the current courseware or labs designed for teaching Web App Security. In the 'Background' section, the Internet of Things (IoT) device-based face recognition Web platform will be introduced, and its advantages will be explained. In the 'Teaching Objectives' section, the teaching goals of the Web App Security course will be discussed. Then, a list of exercises to support the teaching goal through the platform will be provided. In section 'Student Feedback', students' feedback proved the ESP32-CAM a good platform for learning 'Web App Security' will be given. With the 'Outcomes' section providing quantitative evaluation on the learning effect. A summary of what areas can be improved, as well as a conclusion of discussion will be provided in "Conclusions and Future Work" section.

2. LITERATURE REVIEW

Currently, the most popular platform for teaching Web App Security is Virtual Machine (Chen & Tao, 2011; Schweitzer & Bolang, 2009; Chen et al., 2010; Liegle & Meso, 2005), though in (Yu et al., 2006), the authors still tried using the traditional high-performance Cyber Defender Lab. The advantage of using a VM platform is obvious: cost-effective. In (Oh et al., 2020), a Raspberry Pi 3 based platform was proposed, which also had the cost advantage (cost was about $234) but provided a real platform to the students. A corresponding survey was conducted in a course. The result showed that most students prefer the real-world Web applications for them to attack and defend; they are tired of practicing in a virtual environment.

A Raspberry Pi can be treated as a minicomputer. Most IoT devices are even smaller and less expensive. The insight here is, if we can move the Web App Security teaching platform to a IoT device, we may achieve further cost-saving.

Fortunately, we found one. And its performance is even better.

3. BACKGROUND

ESP32-CAM (Fig .2) is an IoT hardware-based Web App providing quick, accurate and cost-effective "Face Recognition". A typical use case/scenario is given below:

- New users should enroll their face image to the Web App first. A unique ID is then assigned to that face. After that, the face image is saved in the system and the user cannot access it anymore.
- The security department places the ESP32-CAM hardware to the gate/door they want to implement access control by face recognition.
- A user stands in front of the ESP32-CAM hardware. The Web App analyzes the input, i.e., the face image, to generate landmarks for that face and search the image database for matched face. If found, access privilege

will be given to the user; if not, access will be denied.

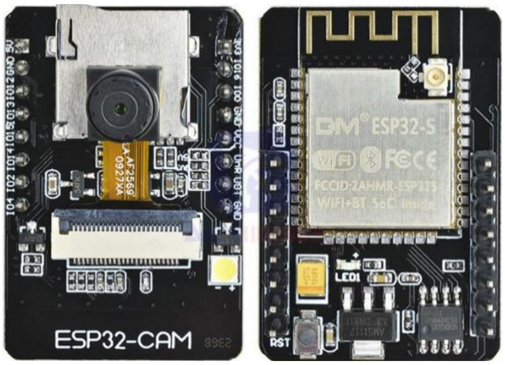


Figure 2: ESP32-CAM Board (Front & Back Views)

The cost of the ESP32-CAM hardware is about $8.00. Considering the peripheral cable and bridge device, the overall cost of one set of exercise hardware is just about $15.00.
And the Web App is ready, within 1 minute, a user can deploy it. Furthermore, it can cold start in 10 seconds, which is much faster than all the known platforms. Not to mention that it is 100% open-source. The instructor doesn't need to verify the potential Intellectual Property issue. And it is easy to maintain and expand, after all, only 4 source files need to be maintained with two of them are header files.
Besides the reasonable price and good performance, another attractive feature this Web App can provide is the integration of IoT and Artificial Intelligence (AI) technologies.

Internet of Things
IoT is an emerging technology. Though many students have shown their interests in IoT, a course about IoT fundamentals is often still only offered as a course elective in degree programs. If exercises in this Web App Security course can be provided by using an IoT device, then state-of-the-art and valuable content can be added to this old-technique course. It is not necessary for students to take a full IoT course to touch on the embedded hardware as well as wireless communication.

This IoT hardware-based platform is also good for students to conduct edge computing research, which is a hot subarea of cloud computing, by focusing on customized computing to provide prompt responses and accurate results. The AI model integrated in ESP32-CAM was trained by a dataset with different faces. Its generality was already verified. However, when it is applied to a specific face, its recognition accuracy and speed are not perfect, i.e., there is room for new research to improve. In one of the capstone projects, one student group realized the sensitivity of the ESP32-CAM AI model somehow was impacted by personal face features.

Artificial Intelligence
As aforementioned, an AI model is integrated in the ESP32-CAM Web application (Zhang, Zhang, and Qiao, 2016). This generation of AI is an emerging technology, which is based on supervised and unsupervised machine learning. And face recognition belongs to the supervised learning. Most CS/ECE departments already offered Machine Learning/AI courses. Consequently, this course for Web App Security can offer hands-on opportunities for students to comprehensively utilize what they have learned from the ML/AI courses. Due to the interactive character of this AI model, students showed their great interests to the face recognition application.

This AI-integrated platform is also good for students to conduct transfer learning research, which does not change the existing AI model, but builds the new learning framework on top of the existing model. For example, enhance the face recognition to emotion recognition.

Furthermore, with the prevalence of AI models, model-based attacks emerge, which makes the traditional code-based countermeasures outdated. Students will get a chance to learn the newest research in data poisoning, data manipulation, and Generative Adversarial Network.

4. TEACHING OBJECTIVES

After competing this course, students will be able to:

1. Understand HTML and front-end code.
2. Describe the components of a Web App.
3. Deploy a Web App to a specific device.
4. Conduct preliminary reverse engineering & re-engineering.
5. Understand the Software Maturity Model with concentration on Security.
6. Describe different vulnerabilities and their root causes.
7. Conduct pen-testing or attacking by code review, auto vulnerability scanning, and fuzz testing.
8. Describe functional and non-functional requirements and their relationships to security requirements.
9. Conduct threat modeling.

10. Follow secure coding standards to write and review code.
11. Describe the function of a certificate. Apply certificates in Web Apps.
12. Apply Public Key Cryptography in Web Apps.
13. Describe the data impact to Web App Security.

These objectives are the result of decomposing the high-level outcomes of this course into small technical areas and integrating practical skills/tools into these areas. The high-level outcomes source from the NSA CAE-CDE designation requirements.

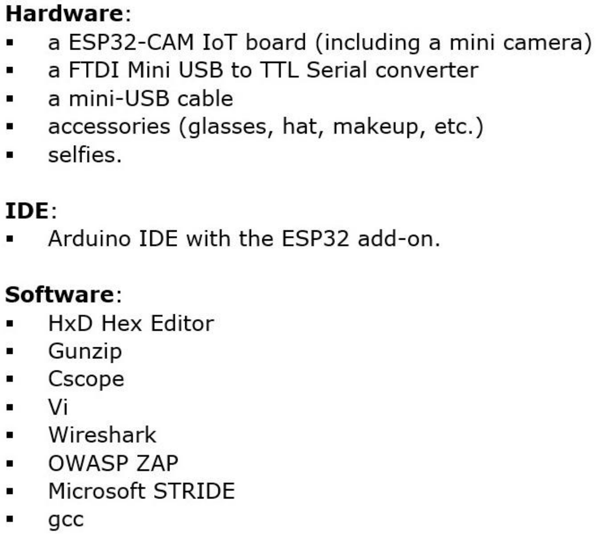
5. EXERCISES

An attempt to fully utilize the proposed platform was made by designing a variety of exercises for students to experience the different aspects of the Web Application Security. In total, 15 independent exercises were prepared, but together, can provide a systematic layout.

1. The first two exercises are related to user experience – before attacking or defending a Web App, the students will need to get familiar with it.
2. The next six exercises cover how to attack a Web App and fundamental skills and tools. Among them, four exercises are related to finding the vulnerabilities of the Web App by studying the code. Followed are auto vulnerability scanning & fuzz testing exercises.
3. After the students understand how to attack a Web App, countermeasures (defending skills and tools) can be introduced. Four exercises related to Secure Software Development Life Cycle and one exercise related to symmetric encryption are provided in this part to students.
4. The next two exercises address the non-code vulnerabilities caused by AI models.

A corresponding optional project was designed to respond to the requests from a few of students, who would like to do correlated research in this Web App Security course, an independent study, or in their capstone course.

As a summary, here is the list of hardware, Integrated Development Environment (IDE), and software used in the exercises:



Hardware:
- a ESP32-CAM IoT board (including a mini camera)
- a FTDI Mini USB to TTL Serial converter
- a mini-USB cable
- accessories (glasses, hat, makeup, etc.)
- selfies.

IDE:
- Arduino IDE with the ESP32 add-on.

Software:
- HxD Hex Editor
- Gunzip
- Cscope
- Vi
- Wireshark
- OWASP ZAP
- Microsoft STRIDE
- gcc

Figure 3: Resources Used in Exercises

Deploy the face recognition Web application to an ESP32-CAM IoT board

Students will need to learn how to **deploy** a Web application to an IoT device.

1. The application code is ready in Arduino IDE after installing the appropriate ESP32 board's add-on.
2. Students need to connect the ESP32-CAM board to a host (where Arduino IDE is running), then cross-compile the code in Arduino and download the executable from the host to the board.
3. After reset, the board is up with the face recognition Web application ready.
4. Access a pre-defined URL to reach the application's control panel, where a user can enroll a face and see if the board can recognize it later when the same face appears in front of the camera of the ESP32-CAM board.

This is a team project with 4 or 5 members in the team. Exercise hardware includes an ESP32-CAM board, an FTDI Mini USB to TTL Serial converter, and a mini-USB cable.

For most of the students, this is the first time they touch an IoT device or an embedded system. Students are curious and worried. A clear instruction manual can help them quickly accomplish this exercise so that they will build their confidence on learning a new technique/skill/method.

Through this exercise, students can explore fundamentals of IoT development and application deployment, and they can identify the basic components of a Web application, which could be deployed on any hardware. They have learned Web application development concepts in their freshman or sophomore year but deploying a Web application to independent physical hardware is the first time for most of them.

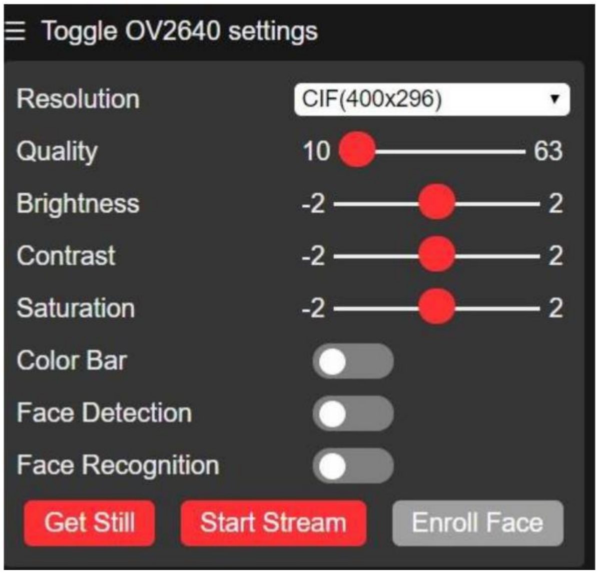
Furthermore, this is a good chance for students to experience IoT in a Web App Security course.

Fool the AI model
The AI model integrated to the Web application can provide quick and accurate face recognition. However, it cannot guarantee 100% correctness. This exercise encourages the students to stably reproduce false positive and false negative situations, which will inspire the students to think about the deeper logic in the AI model though it appears a black box so far. This is a very whimsical yet important exercise, and how creative students are can be observed. Some students tried making funny faces to cause false negative cases. Others tried wearing glasses, hats, or even a fake beard to fool the AI model, like a fashion show. Creatively, some students directly used a printed photo and successfully made the Web application believe this is the enrolled person. Students can sample the problem the AI model has, but do not know why. Having the question not directly answered here will keep students' curiosity piqued until later they are asked to get hints from the papers about Data Manipulation, Data Poisoning, and Generative Adversarial Networks. This is a good chance for students to experience AI in a Web App Security course.

Reverse Engineering
Reverse engineering is an important practical skill that can be used in attacking a system or pen-testing. Through reading and analyzing the source code or binary program, the attacker or tester can infer the original design ideas and architecture. In this designed exercise students will be asked to determine the design of the face recognition Web application from the published C code. Either the architecture or the pseudo code of the Web application should be submitted.

There will be several challenges for students to overcome. The first one is the library code, which was not published by the application developers. Only four C source files were published, but the most fundamental functions were provided by the libraries with (debugging) symbols stripped off. Students will need either read the assembly code (not recommended to them due to the difficulty level) or perform a Google search for the source code of the libraries.

Re-engineering
The ESP32-CAM Web App provided a complicated control panel to configure the attached camera and the face recognition parameters. However, half of these parameters are too professional to be changed by most of the students. Therefore, simplifying the control panel can reduce the confusion and distraction on the face recognition application itself. Figure 4 shows the simplified version of the control panel, which is much simple.



Resolution CIF(400x296) ▾
Quality 10 — 63
Brightness -2 — 2
Contrast -2 — 2
Saturation -2 — 2
Color Bar
Face Detection
Face Recognition

Figure 4: Simplified Control Panel of the Web App

To accomplish this exercise, students will need to overcome several small challenges:

1. Understand the original HTML code and identify the unnecessary elements on the HTML page. Because several elements have dependency relationships, before deleting one unnecessary element, students must first resolve its dependencies.
2. Because the original HTML page was compressed then saved in the ESP32-CAM flash, to replace it with the simplified page, students will need to know how to convert their HTML code to .gzip format.
3. Also, the compressed HTML page is saved as an array of hex bytes in ESP32-CAM. Students will need to convert the raw bytes of the .gzip file to a hex byte array. To complete this task, students must master one Hex Editor.

These are practical skills related to Web App Security.

Determine vulnerabilities of the Web App

Before taking this course, the students already had a solid foundation in Cybersecurity from earlier courses. Thus, it is easy for them to detect several vulnerabilities of the face recognition Web application. However, to provide full coverage, they will need to have a systematic view and comprehensively utilize their knowledge, skill, and inference capability. This exercise will provide a record of how many vulnerabilities they can find without further education. After they finish this course study, they can retry this exercise to identify what additional vulnerabilities they can find.

Fix the Buffer Overflow vulnerability demonstrated by a video

Buffer Overflow once was a top vulnerability. And the original code of the face recognition App suffered from this vulnerability. A recorded video can show how the attack vector "http://{IP}/control?var=framesize&val=512" could corrupt the face recognition Web application because the variable used to save the 'framesize' parameter is just an 8-bit integer. This attack vector was not determined through the auto vulnerability scan nor the reverse engineering because its URL is a hidden one. When a user changes a parameter through the control panel, the front-end code will generate a similar but hidden URL to update the parameter saved at the Web server. To expose the hidden URL, students will need to understand the front-end code (i.e., the HTML page), use Wireshark to capture the network traffic for analysis, or understand the back-end code.

At minimum, a student will need to master one of the following skills before they can find the Buffer Overflow vulnerability:

- Efficiently trace the front-end code in HTML and Java Script.
- Know how to filter network traffic by Wireshark and narrow down the packets of interest.
- Efficiently trace the back-end code in C and C++.

Unfortunately, it is not easy, but students will realize tools alone are not the most important factor in Web App Security. Both understanding the target's code and using automatic tools are crucial.

Auto Vulnerability Scanning & Fixing

There are many automatic scan tools for Web App vulnerabilities, which can greatly save the attacker or tester's effort during target vulnerability scanning. The Open Web Application Security Project (OWASP) Zed Attack Proxy (ZAP) (Wikipedia Contributors, 2021a) is a good one for Web App attacking or testing. Using it, students can scan the vulnerabilities of the face recognition Web App in an automatic style. Based on hints provided by the ZAP report, students will need to explore the back-end code for the best place to put the fix.

Fuzz Testing

Fuzz testing or Fuzzing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program (Wikipedia Contributors, 2021b). Its purpose is to verify the reliability of the target, and it can verify the coverage of the implemented code. It is a good tool for Web App attacking or testing. In previous exercises, the 'control' hidden URL has been exposed. Thus, students can direct OWASP ZAP to feed a wide range of inputs to the face recognition Web App to see if some inputs can trigger exceptions to the App. Students should be able to experience automated testing and realize its efficiency.

Secure Software Development Life Cycle (SDLC)

To prevent vulnerabilities from being integrated into the Web App from scratch, the secure development process is crucial, which can monitor the quality of Web App development. And security is just one aspect of the product quality metrics. Thus, knowing the impacts from non-security requirement is also important. The goal of this exercise is to give the students a systematic view about the security. OWASP Software Assurance Maturity Model (SAMM) allows teams and developers to assess, formulate, and implement strategies for better security which can be easily integrated into an existing organizational Software Development Life Cycle (SDLC). This is especially important when students run/join software companies in the future.

Students are expected to read the OWASP SAMM Quick Start Guide (Wen, 2017).

Secure Software Design

Producing secure software requires conducting secure practices as early in the SDLC as possible. Design is the next phase after the customer requirement analysis. At this phase, platform, environment, constraints, components, and their relationships as well as interactions are decided. Integrating security consideration at this phase can greatly reduce software vulnerabilities. Therefore, it can avoid the most cost for finding and fixing the vulnerabilities in downstream. In this exercise, students will need to analyze the security requirements of the Web App, then propose an architecture (update) and detail the interactions between the components in the architecture. Both sunny-day and rainy-day scenarios should be performed to exclude potential vulnerabilities.

Sequence diagrams that focus on different aspects of security should be submitted as the result of Secure Software Design.

Threat Modeling
Threat modeling is a powerful tool, which can be used to determine the attack surface of the Web App. It is useful for

- Ensuring the design complements the security objectives.
- Making trade-offs and prioritizing efforts
- Reducing the risk of security issues during development and operation.

In this exercise, students will try Microsoft’s threat modeling framework, STRIDE (Spoofing, Tampering, Repudiation, Information, DoS, Elevation of privilege) to determine the attack surface of their Web App.

Best Coding Practice
Best coding practice is a kind of accumulation of experience from existing events. Though it cannot defeat all attacking attempts, it can fix most severe vulnerabilities and mitigate the attacking consequence. Students are expected to go through a check list (i.e., OWASP Secure Coding Practices Quick Reference Guide (Lala, Kumar, & Subbulakshmi, 2021)) to review and evaluate the overall security of their code.

Asymmetric Cryptography
To protect the confidentiality of the Web traffic, encryption should be conducted. Usually, asymmetric cryptography is used to generate public and private keys for symmetric key and signature distribution. The core part of the asymmetric cryptography is the difficult mathematic problem, such as the big integer factoring problem.

A Fermat Sieve-based 64-bit C program was given to students to demonstrate the big integer factoring algorithm as well as the time consumption. Because the program cannot handle numbers larger than 64 bits, the students are expected to port the logic to a Python program, which can handle larger numbers.

Data Manipulation & Poisoning
During training, machine learning algorithms search for the most accessible pattern that correlates pixels to labels. But when a common yet trivial pattern is given a higher weight, a noise or a small piece of polluted data could cause the wrong judgement of the trained AI model. Students will need to read two articles to realize and understand the non-code impact to Web App Security:

1. The Threat of Adversarial Attacks on Machine Learning in Network Security--A Survey (Ibitoye et al., 2019).
2. Adversarial machine learning (Vorobeychik & Kantarcioglu, 2018).

Generative Adversarial Network
Generative modeling discovers and learns the patterns in input data in such a way that the model can be used to generate new examples that plausibly could have been drawn from the original dataset. In a GAN, two sub-models (the generator model for new examples and the discriminator model for classification) are trained together adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples. Students will need to read one article to realize and understand the GANs’ impact to Web App Security: Generative Adversarial Networks (GAN) A Gentle Introduction (Wang, 2017).

Capstone Projects or Research Directions
Based on the compact ESP32-CAM IoT hardware and the integrated face recognition AI model, there are three capstone projects, or three research directions suggested for students who want to try different things beyond this course study.

- **Transfer Learning** (Wikipedia Contributors, 2021c) – the AI model will generate five landmarks (points) for each input/face image. The face recognition Web App will compare enrolled one with the current input/face image to evaluate their similarity by calculating a correlation coefficient between the landmarks. If the landmarks are used as the starting point for further emotion recognition, the function of the Web App is enhanced while leaving the integrated AI model intact. Emotional information is a supplement to the face information, which will enhance the security when they are used in access control and authentication/ authorization scenarios.

- **Edge Computing** (Wikipedia Contributors, 2021d) – the face recognition AI model was trained by public datasets. However, different people have different facial features. When this AI model is deployed in a specific target environment, its application context may be limited to a small group of people. Then, one thing perhaps may be enhanced: customize the AI model for the target environment to provide quicker and more accurate response. This direction belongs to the scope of Edge Computing.

- **Data Manipulation, Data Poisoning, & GAN** (Chen et al., 2017; Ledig et al, 2017) – Examples have been seen that adding some trivial noise to the input image can mislead the AI model. Due to the black-box character of the AI model, these are hard to explain. Moreover, the traditional countermeasures for code-based vulnerabilities cannot be reused for the model-based (or data-based) vulnerabilities. Evaluating the impact and finding a solution is a good project topic or research direction.

6. STUDENT FEEDBACK

In Spring 2021, one of the authors delivered this proposed platform-based courseware to 37 senior Cybersecurity undergraduate students in the CY410 Web App Security online course. CY410 is a major core course. Its prerequisites include "Python Programming", "Java Programming", "Introduction to Cybersecurity", "Web Development", "Data Protocol Security", and "Information Security in System Administration". At the end of the semester, 59% students provided their feedbacks to CY410. Overall, the feedbacks are positive, and the average 'grade' the students gave to the instructor was 4.55 (the program's average was 4.27 and the school's average was 4.29). The students also realized the depth of this course because of so many tricky hands-on exercises. Though they never admitted it. What the students complained most was they couldn't get enough hardware for the team projects. Only one set of devices were given to a project team. Due to the COVID-19 pandemic, projects or teamwork is not sufficiently organized. Each team member individually wanted to use the hardware. Therefore, assigning a set of equipment to each student rather than each project team may further improve their feedbacks.

7. OUTCOMES

Table 1 (in the Appendix) shows the learning outcomes corresponding to the teaching objectives in section 3.

8. CONCLUSIONS AND FUTURE WORK

The ESP32-CAM IoT and AI platform provides rich features from almost every aspect for students to experience Web App Security and attracts students to touch the cutting-edge research in IoT, Edge Computing, and Transfer Learning. Totally it can support more than 16 corresponding hands-on exercises. In the future, we plan to connect database to this platform or implement a 'little' DB in it. A prototyping has been done. We will provide more details in our future paper. Furthermore, at a cost of $15/student, this IoT platform provides a cost-effective solution for teaching Web App Security, which is the lowest-cost platform so far to our best knowledge. This means the instructors can offer sufficient hardware to the students. The teaching effect showed students gave very positive feedback to the new teaching/exercise platform. We expect further improvement in the student feedback (currently 4.55) when we equip every student with one set of the device.

9. REFERENCES

Buyannemekh, B., & Chen, T. (2021). Digital governance in Mongolia and Taiwan: A gender perspective. Information Polity. IOS Press BV. https://doi.org/10.3233/IP-219005

Chen, L., Tao, L., Li, X., & Lin, C. (2010). A tool for teaching web application security. In Proceedings of the 14th Colloquium for Information Systems Security Education (pp. 17-24).

Chen, L. C., & Tao, L. (2011). Teaching web security using portable virtual labs. In Proceedings of the 2011 11th IEEE International Conference on Advanced Learning Technologies, ICALT 2011 (pp. 491–495). https://doi.org/10.1109/ICALT.2011.153

Chen, X., Liu, C., Li, B., Lu, K., & Song, D. (2017). Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526.

Ibitoye, O., Abou-Khamis, R., Matrawy, A., & Shafiq, M. O. (2019). The Threat of Adversarial Attacks on Machine Learning in Network Security--A Survey. arXiv preprint arXiv:1911.02621

Lala, S. K., Kumar, A., & Subbulakshmi, T. (2021). Secure web development using OWASP guidelines. In Proceedings - 5th International Conference on Intelligent Computing and Control Systems, ICICCS 2021 (pp. 323–332). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICICCS51141.2021.9432179

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., … Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 (Vol. 2017-January, pp. 105–114). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/CVPR.2017.19

Liegle, J., & Meso, P. (2005). Evaluation of a virtual lab environment for teaching web-application development. In Proceedings of ISECON.

Oh, S. K., Stickney, N., Hawthorne, D., & Matthews, S. J. (2020). Teaching Web-Attacks on a Raspberry Pi Cyber Range. In SIGITE 2020 - Proceedings of the 21st Annual Conference on Information Technology Education (pp. 324–329). Association for Computing Machinery, Inc. https://doi.org/10.1145/3368308.3415364

Schweitzer, D., & Boleng, J. (2009). Designing web labs for teaching security concepts. Journal of Computing Sciences in Colleges, 25(2), 39-45.

Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-169.

Wang, S. (2017). Generative Adversarial Networks (GAN) A Gentle Introduction. Tutorial on GAN in LIN395C: Research in Computational Linguistics.

Wen, S. F. (2017, November). Software security in open source development: A systematic literature review. In 2017 21st conference of open innovations association (fruct) (pp. 364-373). IEEE.

Wikipedia contributors. (2021a, May 9). OWASP ZAP. In Wikipedia, The Free Encyclopedia. Retrieved 01:10, April 18, 2022, from https://en.wikipedia.org/w/index.php?title=OWASP_ZAP&oldid=1022316463

Wikipedia contributors. (2021b, May 21). Fuzzing. In Wikipedia, The Free Encyclopedia. Retrieved 01:11, April 18, 2022, from https://en.wikipedia.org/w/index.php?title=Fuzzing&oldid=1024357049

Wikipedia contributors. (2021c, May 10). Transfer learning. In Wikipedia, The Free Encyclopedia. Retrieved 01:53, April 18, 2022, from https://en.wikipedia.org/w/index.php?title=Transfer_learning&oldid=1022394104

Wikipedia contributors. (2021d, June 6). Edge computing. In Wikipedia, The Free Encyclopedia. Retrieved 01:55, April 18, 2022, from https://en.wikipedia.org/w/index.php?title=Edge_computing&oldid=1027237845

Yu, H., Liao, W., Yuan, X., & Xu, J. (2006). Teaching a web security course to practice information assurance. ACM SIGCSE Bulletin, 38(1), 12–16. https://doi.org/10.1145/1124706.1121348

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Processing Letters, 23(10), 1499–1503. https://doi.org/10.1109/LSP.2016.2603342

.

Appendices and Annexures

Objective ID	Objective Description	Pass Rate (grade > 4/5)	Exercise(s)
1	Understand HTML and front-end code.	86.5%	Re-engineering
2	Describe the components of a Web App.	100%	Deploy App & Reverse Engineering
3	Deploy a Web App to a specific device.	100%	Deploy App
4	Conduct preliminary reverse engineering & re-engineering.	86.5%	Reverse Engineering & Re-engineering
5	Understand the Software Maturity Model with concentration on Security.	100%	Secure SDLC (Read OWASP SAMM)
6	Describe different vulnerabilities and their root causes.	89.2%	Determine vulnerabilities & Fix Buffer Overflow
7	Conduct pen-testing or attacking by code review, auto vulnerability scanning, and fuzz testing.	91.9%	Auto scanning & Fuzz testing
8	Describe functional and non-functional requirements and their relationships to security requirements.	97.3%	Secure software design
9	Conduct threat modeling.	81.1%	Threat Modeling
10	Follow secure coding standards to write and review code.	91.9%	Best Coding Practice
11	Describe the function of a certificate. Apply certificates in Web Apps.	N/A	N/A
12	Apply Public Key Cryptography in Web Apps.	Non-graded	Asymmetric Cryptography
13	Describe the data impact to Web App Security.	100%	Data manipulation & poisoning, and GAN papers

Table 1. Learning Outcomes