CYBERSECURITY PEDAGOGY & PRACTICE JOURNAL

Volume 2, No. 2
September 2023
ISSN: 2832-1006

In this issue:

4. A Chatbot for Teaching Secure Programming: Usability and Performance Evaluation Study
James Walden, Northern Kentucky University
Nicholas Caporusso, Northern Kentucky University
Ludiana Atnafu, Northern Kentucky University

17. ***Teaching Case***
Applied Steganography: An Interesting Case for Learners of all Ages
Johnathan Yerby, Mercer University
Jennifer Breese, Penn State Greater Allegheny

28. A Case Study in Identifying and Measuring Skills Honed from a Cybersecurity Competition
Ron Pike, Cal Poly Pomona
Jasmine Weddle, Cal Poly Pomona
Sydney Duong, Cal Poly Pomona
Brandon Brown, Coastline College

39. IoT Security Vulnerabilities Analysis by Reverse Engineering: A Face-recognition IoT Application-based Lab Exercises
Sam Elfrink, Southeast Missouri State University
Mario Alberto Garcia, Southeast Missouri State University
Xuesong Zhang, Southeast Missouri State University
Zhouzhou Li, Southeast Missouri State University
Qiuyu Han, Heilongjiang University

68. Recommendations for Developing More Usable and Effective Hands-on Cybersecurity Education Materials Based on Critical Evaluation Criteria
Ahmed Ibrahim, University of Pittsburgh
Vitaly Ford, Arcadia University

82. Utilizing Discord-based Projects to Reinforce Cybersecurity Concepts
Marc Waldman, Manhattan College
Patricia Sheridan, Manhattan College

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.

While the primary path to journal publication is through the ISCAP conference, CPPJ does accept direct submissions at https://iscap.us/papers. Direct submissions are subjected to a double-blind peer review process, where reviewers do not know the names and affiliations of paper authors, and paper authors do not know the names and affiliations of reviewers. All submissions (articles, teaching tips, and teaching cases & notes) to the journal will be refereed by a rigorous evaluation process involving at least three blind reviews by qualified academic, industrial, or governmental computing professionals. Submissions will be judged not only on the suitability of the content but also on the readability and clarity of the prose.

Currently, the 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.

2023 ISCAP Board of Directors

Jeff Cummings Univ of NC Wilmington President	Anthony Serapiglia Saint Vincent College Vice President	Eric Breimer Siena College Past President
Jennifer Breese Penn State University Director	Amy Connolly James Madison University Director	RJ Podeschi Millikin University Director/Treasurer
Michael Smith Georgia Institute of Technology Director/Secretary	David Woods Miami University (Ohio) Director	Jeffry Babb West Texas A&M University Director/Curricular Items Chair
Tom Janicki Univ of NC Wilmington Director/Meeting Facilitator	Paul Witman California Lutheran University Director/2023 Conf Chair	Xihui “Paul” Zhang University of North Alabama Director/JISE Editor

Copyright © 2023 by Information Systems and Computing Academic Professionals (ISCAP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to editorcppg@iscap.us.

CYBERSECURITY PEDAGOGY AND PRACTICE JOURNAL

Editors

Anthony Serapiglia
Co-Editor
Saint Vincent College

Jeffrey Cummings
Co-Editor
University of North Carolina Wilmington

Thomas Janicki
Publisher
University of North Carolina Wilmington

2023 Review Board

Etezady Nooredin
Nova Southern University

Li-Jen Lester
Sam Houston State University

Jamie Pinchot
Robert Morris University

Samuel Sambasivam
Woodbury University

Kevin Slonka
Saint Francis University

Geoff Stoker
University of North Carolina Wilmington

Paul Wagner
University of Arizona

Paul Witman
California Lutheran University

Johnathan Yerby
Mercer University

A Chatbot for Teaching Secure Programming: Usability and Performance Evaluation Study

James Walden
waldenj1@nku.edu

Nicholas Caporusso
caporusson1@nku.edu

Ludiana Atnafu
atnaful1@nku.edu

Department of Computer Science
College of Informatics
Northern Kentucky University
Highland Heights, KY (USA)

Abstract

Security is a fundamental aspect of programming. However, even experienced developers find it difficult to always write secure code or overlook key security flaws. Therefore, it is crucial to provide additional guidance to students who are learning to program in a new language or environment, so that they can understand how to use and write secure code. The goal of our work is to create a chatbot with an authoritative knowledge base on secure programming to help teach student developers how to write secure code, instead of having them rely on common sources of readily available help on the Internet such as tutorials and question and answer sites, which often teach insecure practices and provide example code containing vulnerabilities. To this end, in the first part of our work, we designed, implemented, and evaluated a novel chatbot with a knowledge base covering secure programming in PHP using the Rasa framework, we evaluated the user experience with the chatbot, and compared students' intent to adopt chatbots in comparison with other information sources such as question and answer sites. Participants solved secure web programming problems in a custom web application developed for the experiment with the aid of either the chatbot or their choice of Internet resources. In the second part of our study, we compared the performance of our novel chatbot with that of GPT-3 in terms of comprehension of students' questions, correctness, completeness, and security of the answers. Our findings about the overall user experience suggest that chatbots can be utilized as a more convenient support tool with respect to other systems, such as search engines. In our comparison of the novel chatbot with GPT-3, we found that while GPT-3 performed better in terms of understanding students' questions, our chatbot outperformed it in terms of correctness and security of the answers.

Keywords: Secure programming, software security, chatbots, user experience, GPT-3, OpenAI.

1. INTRODUCTION

Learning to create web applications involves acquiring multiple skills simultaneously, including writing source code in a new programming language, using a development environment for the first time, and ensuring the security of the code and the data being stored. The latter aspect is especially important in the development of server-side and full-stack applications, where novice programmers and students are faced with the additional challenge of exposing their code and users' data to a larger audience and greater cybersecurity risks.

Unfortunately, security often is considered among the least important skills by students, because the impact of insecure code is not immediately evident to them. While it is easy to test the functionality of a feature in a web application (e.g., adding an item to its database), students need additional knowledge and skills to verify that the application performs functions securely.

Even if security is considered equal in importance to functional requirements, it can be difficult to learn secure programming due to the problem of identifying accurate sources of information about security, the difficulty of using Application Programming Interfaces (APIs) securely (Green & Smith, 2016; Olivera et al., 2018) the complexity of testing security flaws in software (Tahaei & Vaniea, 2019), and the evolution of new types of vulnerabilities in web applications (Hiesgen et al., 2022). While incorrect information on security topics is often associated with online sources like Stack Overflow (Fischer et al., 2017; Rahman et al., 2019; Zhang et al., 2021), even college textbooks contain insecure code examples (SANS, 2008).

Helping students avoid sources of code with security flaws has become an even more urgent challenge with the recent introduction of generative AI tools, such as ChatGPT (ChatGPT, 2022), GPT-3 (Brown et al., 2020), and GitHub CoPilot (GitHub, 2022). While ChatGPT and GPT-3 are based on large language models, both tools can generate source code and answer questions about programming. CoPilot on the other hand is a powerful code autocompletion tool. However, as CoPilot's very large training dataset includes both secure and insecure source code, its output can also contain security vulnerabilities (Asare et al., 2022; Pearce et. al, 2022).

Our goal is to provide students with tools that can help them learn secure web development from reliable and secure code. In the first part of our work, we created a chatbot to answer their secure programming questions. The chatbot has a curated knowledge base with accurate information and secure code snippets for web programming in PHP and for connecting to and querying a MySQL database. Designing a chatbot specific to our web programming course enabled us to address both problems associated with learning secure programming. The knowledge base was focused on the security issues and APIs that students encountered in their class, and it was created with accurate information about security issues.

The chatbot was initially introduced in a web development course in the Spring semester of 2021, when we designed an initial experiment and collected data about the overall design of the tool and its integration within a custom website that was used as a learning and development environment. Data from our first study were utilized to evaluate the appropriateness of the system, improve the knowledge base (e.g., add code snippets), and improve the learning environment and its integration into the course. After revising the chatbot and learning environment based on our students' feedback, we tested the revised version with a group of students enrolled in a web programming course in Fall 2021. We reported our findings in a previous study (Walden et al., 2022), where we discussed user experience and adoption dynamics.

OpenAI released ChatGPT, their GPT-enabled chatbot on November 30, 2022, only a few weeks after we presented our study at EDSIGCON. The release of ChatGPT inspired us to compare the performance of our custom secure programming chatbot with that of large language models. As news about ChatGPT was reported widely, the number of users soared and the chatbot became difficult and unreliable to access due to high demand, frequently producing errors or failing to work in the middle of a session. Therefore, we decided to use OpenAI's API to access the underlying GPT-3 language model on which ChatGPT was built for our comparison. We evaluated the performance of our custom chatbot and GPT-3 in terms of comprehension, correctness, completeness, and security. We used the same student queries from our original study to evaluate the performance of GPT-3.

In this paper, we present the results of our complete work. The contributions of our work are as follows:

1. The development of a custom chatbot to support learning secure programming practices in PHP.
2. The evaluation of how effective the chatbot is in helping students write secure code.
3. An analysis of the correctness and security of answers provided by GPT-3 in comparison with a chatbot trained with a custom knowledge base.

This journal paper is an extension of our original conference paper (Walden et al., 2022), adding the comparison of answers provided by our custom secure programming chatbot with answers provided by GPT-3. Sections 3 through 5 of this journal paper are taken verbatim from the original conference paper. The Related Work section is also taken from the conference paper, except for the subsection on GPT-3. The new section 6 focuses on the comparison of our secure programming bot with GPT-3. The introduction and conclusion have been rewritten to include the comparison with GPT-3.

2. RELATED WORK

Resources for Learning Secure Coding

Acar et al. studied the effect of developers' use of different information sources on the functionality and security of the code they produced (Acar et al., 2016). The authors divided developers in their experiment into four groups. The first three groups had access to single sources of information: books only, official Android documentation only, and Stack Overflow only, while the fourth group had free choice of information sources. Developers restricted to only using Stack Overflow produced significantly less secure code than developers using the official documentation or books. However, developers using only official documentation produced significantly less functional code than those using only Stack Overflow.

Stack Overflow is the most popular question and answer site for software developers, including students. Multiple studies have found insecure answers and code snippets in answers for questions on a variety of programming languages and environments on the site (Fischer et al., 2017; Meng et al., 2018; Chen et al., 2019; Fischer et al., 2019; Verdi et al., 2020). One study found that insecure answers received more up votes, comments, favorites, and views than secure answers (Chen et al., 2019).

Automated Tools for Learning Secure Coding

Automated tools can also make it easier for instructors to incorporate cybersecurity into their classes and can provide knowledge and feedback at the precise point in time when students need that. While there are a variety of tools used to assist developers in finding security vulnerabilities through static or dynamic analysis, there are few automated tools designed to help teach students about secure programming. CrypTool has been widely used to assist in the teaching of cryptography (Adamovic et al., 2018), but the focus of the tool is teaching how cryptography works rather than teaching how to write code to securely use cryptographic APIs. CryptoExplorer is a web search application that can provide insecure and secure examples of cryptographic API use, but it is aimed at professional developers (Hazhirpasand et al., 2020).

Plugins for Integrated Development Environments (IDEs) can provide secure programming assistance to students in the same environment in which they are writing their code. Whitney et al. incorporated secure Java web programming instruction into an Eclipse plugin called Educational Security in the Integrated Development Environment (ESIDE) (Whitney et al., 2018). ESIDE adds warning icons in Eclipse when problematic code patterns are detected. When students click on the warning, ESIDE provides multiple information options with short explanations and a link to a page that provides a detailed explanation of the potential security issue. ESIDE was based on an earlier plugin, ASIDE, created for professional developers (Xie et al., 2011). Nguyen et al. created a plugin to help professional developers write secure mobile code in Android Studio called FixDroid (Nguyen et al., 2017). While FixDroid was not designed for educational purposes, it would be used for that purpose.

Chatbots for Teaching and Learning

The use of chatbots in education for a wide variety of purposes from providing deadlines to delivering course content is rapidly expanding (Okonkwo & Ade-Ibijola, 2021). A recent survey of chatbots in education found that chatbots serve in four pedagogical roles: learning, assisting, and mentoring (Wollny et al., 2021). The focus of this study is on the learning role. Educational chatbots have been used to help students learn a variety of skills, including computer programming. Both Python-bot (Okonkwo & Ade-Ibijola, 2020) and APIHelper (Zhao et al., 2020) were designed to help students learn how to program.

Evaluation measures for tool adoption

Security tools generally see poor adoption by professional developers, who usually prefer to look up information on the Internet, by visiting developer communities (e.g., Stack Overflow) (Tahaei & Vaniea, 2019), tutorials, and, more recently, videos on YouTube (MacLeod et al., 2015). Xiao et al. interviewed professional developers, exploring how security tool adoption was affected by social environments and communication channels (Xiao et al., 2014). They used the diffusion of innovation theory to evaluate the role of social influence and found that dynamics such as acceptance within the developers' community are among the leading factors that promote the adoption of tools that address security.

Previous studies focusing on the adoption of chatbots in healthcare (Abd-Alrazaq et al., 2020) and finance (Sugumar & Chandra, 2021) found that user experience is a key factor and outlined a variety of technical measures that could be used to assess users' willingness to employ conversational agents. Almahri et al. (Almahri et al., 2020) utilized a revised version of the UTAUT (i.e., UTAUT2), to analyze the specific user dynamics that affect the acceptance, adoption, and use of chatbots in universities in the United Kingdom. They found the performance of the chatbot to be the main predictor of the behavioral intention to use this type of technology. Furthermore, the authors of two studies (Sugumar & Chandra, 2021; Ling et al., 2021) highlighted that when users know that they are entertaining a conversation with a chatbot, their interaction tends to be more opportunistic and utilitarian with respect to their goal and less influenced by aspects that are more typical of a conversation with a human agent (e.g., empathy).

New language models

GPT-3 is the third-generation language prediction model of the Generative Pre-trained Transformer (GPT) model series (Brown et al., 2020). It was the largest language model, with 175 billion parameters (compared to 1.5 billion for GPT-2), when it was released by OpenAI in 2020. Given an initial text prompt, GPT-3 will generate text that would continue that prompt. When given a question as a prompt, the model will typically continue the prompt by providing an answer that question.

On November 30, 2022, OpenAI released a GPT-3 enabled chatbot called ChatGPT (ChatGPT, 2022) that has quickly become popular in many contexts, including education, where its promising applications to improving student writing and coding also raises significant concerns about plagiarism and academic integrity (Zhai, 2022).

While the output of large language models like GPT-3 is syntactically correct and GPT responses can be difficult to distinguish from humanly written text (Brown et al., 2020), it is important to note that GPT-3 does not have any semantic understanding of its input or output. GPT-3 is an unreliable author, that can generate text that is not factual as well as text that is biased. It can even hallucinate facts and references that do not exist in its training data (Maynez et al., 2020). While GPT-3 is unreliable, it can still be useful in a variety of contexts if a human is kept in the loop to validate its output (Dale, 2021).

GPT-3 can also generate source code in multiple programming languages. While GitHub released a dedicated tool called CoPilot (GitHub, 2022), a tool to autocomplete code snippets, GPT-3 and ChatGPT are both capable of producing source code output in response to queries. Unfortunately, studies of CoPilot have shown that generated source code in security relevant contexts frequently contains vulnerabilities (Asare et al., 2022; Pearce et. Al, 2022).

3. DESIGN AND IMPLEMENTATION OF THE CHATBOT

We developed a chatbot (SPbot) to assist students to write secure code in their course on web application development. The purpose of the chatbot was to provide an authoritative source of correct information on secure programming that was also easy to use. The server-side programming language used in the course was PHP, so we wrote all code examples in PHP. The chatbot was deployed as a web widget in a custom web application that presented web application security problems for students to solve.

SPbot was based on the Rasa chatbot framework, an open-source project written in Python. Rasa includes both natural language understanding and dialog management capabilities. The major tasks in creating a chatbot are designing conversation flow, creating a knowledge base, and training the bot to associate questions with the correct answers in the knowledge base. The conversation flow was simple for the secure programming chatbot, tying single questions to single answers.

During our development and testing process, the Rasa framework changed rapidly, including both API and data format changes. After starting development using version 1 of Rasa, we found it impossible to deploy the chatbot to new machines, because it became impossible to install the required dependencies from our saved *conda* environment. This combination of dependency problems and lack of security updates for Rasa 1.x led us to update the bot's code and data to use version 2 of the Rasa framework before performing the experiment.

We designed the secure coding knowledge base to include aspects of secure coding that were directly relevant to the topics students were learning in the web development course, including authentication, input validation, cross-site scripting, and SQL injection. In addition to answering conceptual questions, the bot could also provide code examples when asked. For example, one answer included example code showing students how to perform SQL queries with prepared statements. Figuring out how to include code snippets in Rasa's data files required some trial and error, as the documentation did not support this use case and the data file format changed from JSON to YAML between versions 1 and 2 of the framework.

The final step to creating the secure programming bot was training it to answer secure programming questions. The authors interacted with the bot repeatedly, asking the same questions in a variety of ways to build the initial version of the bot. Once the bot was working, we focused training on teaching the bot to distinguish between similar questions. For example, password security could refer to HTML form input fields, transmitting passwords over HTTPS, or securely storing passwords in a database. While training data for most question and answer pairs consisted of a couple dozen example questions, training data for related topics required approximately twice as many examples to ensure the bot could reliably provide the desired answer.

4. EXPERIMENTAL STUDY: USABILITY ANALYSIS

Materials and methods

We realized an experimental study that evaluated perceived user experience and effectiveness of the chatbot in supporting students and beginner programmers in learning key cybersecurity concepts in client- and server-side web development, including well known security issues in web applications, such as cross-site scripting and SQL injection.

To this end, we designed a custom web application in which users could interact with the chatbot while practicing with code challenges consisting in analyzing and fixing existing code snippets containing cybersecurity flaws. Screenshots of the web application can be seen in Figure 1.

The web application contains five challenges, each addressing a key cybersecurity problem in a web authentication workflow. Participants were required to complete all five challenges.

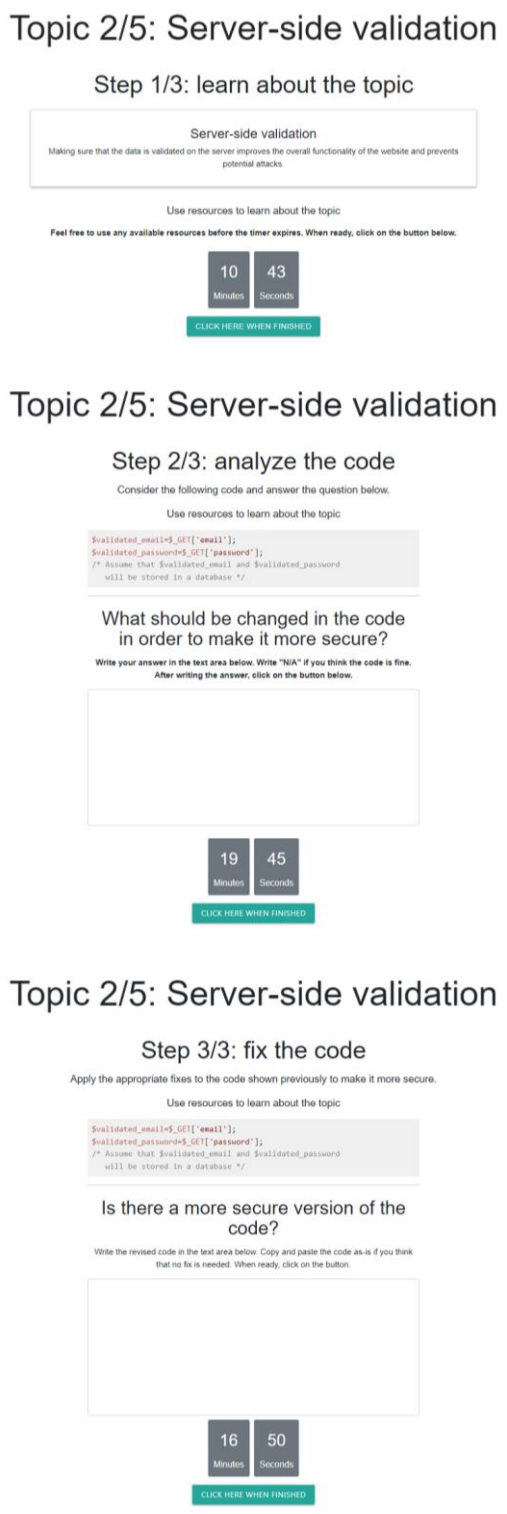
1. *Front-end and web forms*: use of correct input fields to prevent over-the-shoulder attacks when typing a password; HTTP requests and client-server communication (e.g., use of correct and secure HTTP methods and protocols to prevent man-in-the-middle attacks or information leaks).
2. *Server-side data processing*: proper handling of data submitted via HTML forms to prevent missing input and code injection attacks.
3. *Password security*: secure password validation and storage using hashing algorithms.
4. *SQL injections*: use of prepared statements and other mechanisms for preventing potential database attacks.
5. *Cross-site scripting*: use of systems for preventing phishing attacks and injection of scripts and snippets.

For each challenge, the website provided participants with a description of the topic that the challenge focused on and a small piece of source code that contained security flaws.
Subjects were required to complete an experimental task organized into three parts as follows:

1. *Topic and code review*: participants were invited to learn more about the topic and analyze the content of the code snippet.
1. *Code analysis*: subjects were asked to identify and submit a short report in which they described the security flaws.
2. *Bug fixing*: the experimental software showed the original snippet and provided subjects with an editor in which they could write a revised version of the source code.

Before starting the experiment, subjects were asked to complete a pre-survey to collect information about the participants, including their experience with cybersecurity and web development. Participants were given a maximum of 15 minutes to complete each of the three sections (i.e., 45 minutes total for one challenge). During this time, they could work on each of the three parts of the section with the help of either the chatbot or other information resources.

Participants were split into experiment and control groups as follows. Group A was provided with the chatbot for challenges one, three, and five, whereas Group B was provided with the chatbot for challenges two and four (see Figure 2). By dividing the participants into two groups, we provided subjects with the opportunity of using the chatbot as well as other resources in the experimental sections. Conversely, subjects did not get access to the chatbot in control sections. By doing this, they could compare their learning and programming experience and evaluate the value of the chatbot as a learning tool.


Topic 2/5: Server-side validation

Step 1/3: learn about the topic

Server-side validation

Making sure that the data is validated on the server improves the overall functionality of the website and prevents potential attacks.

Use resources to learn about the topic

Feel free to use any available resources before the timer expires. When ready, click on the button below.

Topic 2/5: Server-side validation

Step 2/3: analyze the code

Consider the following code and answer the question below.

Use resources to learn about the topic

```
$validated_email=$_GET['email'];
$validated_password=$_GET['password'];
/* Assume that $validated_email and $validated_password
   will be stored in a database */
```

What should be changed in the code in order to make it more secure?

Write your answer in the text area below. Write "N/A" if you think the code is fine. After writing the answer, click on the button below.

Topic 2/5: Server-side validation

Step 3/3: fix the code

Apply the appropriate fixes to the code shown previously to make it more secure.

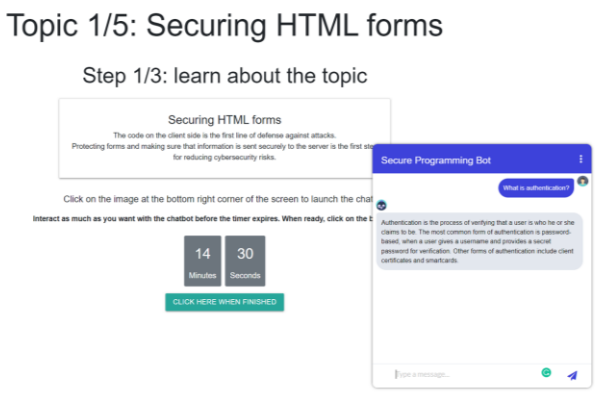
Use resources to learn about the topic

```
$validated_email=$_GET['email'];
$validated_password=$_GET['password'];
/* Assume that $validated_email and $validated_password
   will be stored in a database */
```

Is there a more secure version of the code?

Write the revised code in the text area below. Copy and paste the code as-is if you think that no fix is needed. When ready, click on the button.

Figure 1. The experiment website


Topic 1/5: Securing HTML forms

Step 1/3: learn about the topic

Securing HTML forms

The code on the client side is the first line of defense against attacks.
Protecting forms and making sure that information is sent securely to the server is the first ste
for reducing cybersecurity risks.

Click on the image at the bottom right corner of the screen to launch the chat

Interact as much as you want with the chatbot before the timer expires. When ready, click on the b

Secure Programming Bot

What is authentication?

Authentication is the process of verifying that a user is who he or she claims to be. The most common form of authentication is password-based, when a user gives a username and provides a secret password for verification. Other forms of authentication include client certificates and smartcards.

Figure 2. Chatbot widget

After participants completed each challenge, the website provided them with a short questionnaire. At the end of the entire experiment, after completing all challenges, participants were asked to evaluate their overall experience with the chatbot and to compare it with other resources they used during the experiment.

Specifically, in our study, we analyzed intrinsic and extrinsic aspects that characterize user experience and the willingness to adopt and use technology. To this end, we utilized the UTAUT model, a widely adopted user experience framework that utilizes the five dimensions indicated below as predictors of the intention to adopt and use technology.

– *Performance expectancy*. This aspect refers to the belief that the use of a particular technology will enhance the performance of an individual or will produce some advantage in realizing a task.
– *Effort expectancy*. This is a two-fold measure: on the one hand, it refers to the perceived skillset required to be able to utilize a system and the expected learning curve (human-machine co-evolution). Simultaneously, it relates to the extent of convenience perceived in using the system.
– *Social influence*. This component refers to the user's perception of beliefs or impressions that the product will generate in others (their milieu, their social group, or the external community). This includes the ability of a device to improve the social status of an individual or to create a desired social image. Moreover, this measure involves social acceptance of technology in each context of reference.
– *Facilitating conditions*. Extrinsic factors, such as battery life, device compatibility, and availability of product accessories and

features that render the product more versatile might be a driver for adoption. Also, the presence of technical support and a user's guide might increase the likelihood of acquiring products. Switching costs and longevity are additional aspects that contribute to this dimension.

- *Hedonic motivation*. Intrinsic factors that are not related to product experience are associated with individuals' conditions or beliefs, social background, and education. As this often is a multifaceted aspect, we included open-ended questions to elicit participants' comments and feedback.

Participants
A total of 20 individuals volunteered to participate in the experiment. Participants were recruited from a server-side web development course that taught PHP and MySQL. Subjects were aged 19-32 (21 on average), 18 were males and 2 females. Six were sophomores, eight were juniors, and six were seniors. They all had from one to three years of experience with programming, though they were not familiar with PHP and MySQL prior to the course and had never utilized a chatbot as a learning resource, though they were familiar with chatbot technology in other contexts.

5. RESULTS

Bot chat analysis
We collected 25 student conversations with the chatbot. The number of conversations is greater than the number of participants, because students could exit the chatbot in one section then restart it in a different section of the experiment application. These conversations included 305 questions, 165 questions asked by students in group A and 140 asked by students in group B. We manually analyzed the questions and their answers to determine if questions were related to secure programming and whether the bot provided relevant answers to the questions.

We found that 215 out of the 305 questions students asked the bot were related to secure programming. The remaining 30% of questions included technical questions about PHP, JavaScript, or SQL that were not related to security, greetings like "hello", tests of the bot like "are you secretly a human?", and general chit chat.

The bot answered 213 (70%) questions correctly, including both secure programming and non-programming questions like requests for information about the bot. Out of the 92 questions answered incorrectly, 26 were not questions about secure programming. Incorrect answers for questions about secure programming questions fell into four categories: nonspecific questions (10), questions about topics not in the bot's knowledge base (33), questions where the bot provided a wrong answer (22), and questions consisting solely of source code (1).

Five of the nonspecific questions were requests for more information on the question that the bot had just answered. As the bot does not retain context, it is impossible for it to answer such questions. Other nonspecific questions including asking for code examples without specifying a topic and completely open questions like "How?" Student questions included 688 words. The bot responded to these questions with 13,893 words. The top twenty most common words used by the students and the bot are listed in Table 1, while the word cloud diagram (see Figure 3) visualizes the frequency of student word use.

User word	**User word count**	**Bot word**	**Bot word count**
password	42	code	723
secure	34	password	521
validate	31	input	393
html	26	user	275
email	20	data	218
php	20	web	216
code	18	php	211
passwords	18	validation	183
cross	16	passwords	172
scripting	15	hash	165
site	14	email	163
forms	13	application	144
sql	12	secure	144
form	11	sql	136
server	11	output	120
xss	10	post	109
get	9	filter	108
input	9	function	108
protect	9	length	107

Table 1. Top 20 words

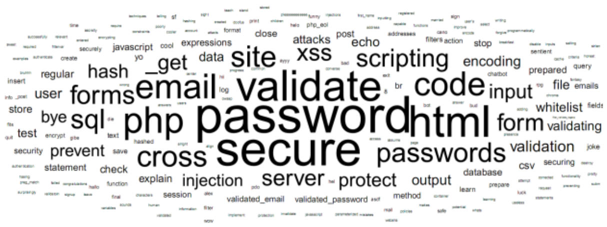


Figure 3. Word cloud of chatbot interactions

The 26 conversations that students had with the chatbot consisted of 688 words. The bot responded to these conversations with 13,893 words. The top twenty most common words used by the students and the bot are listed in Table 1, while the word cloud diagram below (see Figure 3) visualizes the frequency of student word use. The three most common words (password, secure, and validate) are all relevant to security queries, indicating student concerns about how to use and store passwords and how to validate user input.

Survey Analysis

A total of 19 participants completed the experiment and responded to the surveys. One subject only finished two sections and, thus, we did not include their data in our analysis. First, we analyzed the overall perceived level of interaction with all the available resources, which is shown in Figure 4. Compound data from surveys about sections from one to five show that, when subjects were provided with it, the chatbot was the first type of resource used, as it represented 35% of the queries. Search engines were the second preferred resource, utilized in 30% of the cases. Developer communities were ranked third in terms of preference, with 17% usage rate, followed by tutorials (9%), YouTube video (4%), other resources (3%), and books (2%). No statistically significant trends, differences, or training effects were found between individual sections, which shows that subjects did not increase or decrease the use of a specific resource throughout the experiment.

Although our data show that subjects preferred to interact with the chatbot even if they were allowed to use other types of materials, students commented that they sometimes had to use additional resources to complete the challenge. This could be due to a lack of familiarity with interacting with such a chatbot and to the inherent switching cost with respect to systems that they already are familiar with and use.

Subsequently, we analyzed participants' responses with respect to the specific User Experience dimensions defined by the UTAUT model. As shown in Figure5, it is possible to identify two groups of resources based on participants' responses. Search engines, the chatbot, and developer communities received very high scores in terms of adoption metrics, with average results of 74%, 78%, and 70%, respectively. On the contrary, the other types of materials were ranked lower. Specifically, tutorials, YouTube videos, other resources, and books, had an average of 51%, 48%, 35%, and 23%, respectively.

Data about individual user adoption dimensions indicate that the chatbot was perceived as having a lower performance expectancy (67%) with respect to search engines (80%) and developer communities (70%). This could be due to the lack of familiarity with the system and how to query the knowledge base. Also, this could be caused by the content of the knowledge base itself, which can be improved. Tutorials, YouTube videos, other resources, and books were ranked lower, with 60%, 55%, 40%, and 14% preference.

Based on previous studies (Almahri et al., 2020), performance expectancy is a critical aspect in the adoption and use of chatbot technology. Thus, our results suggest that further work is needed before using the chatbot more consistently in web programming courses, because its current perceived performance expectancy might be a cause of discontinuation.

As far as effort expectancy is concerned, the chatbot ranked first (87%) compared to search engines (73%), other resources (61%), developer communities (55%), YouTube videos (49%), tutorials (22%), and books (12%). Students indicated that the chatbot was the most convenient resource to gain an initial understanding of the topic. Although this could be because the chatbot was integrated into the website, participants' comments mostly report the ease of typing questions in natural language and the responsiveness of the system, which returned either relevant results or answers that were clearly inaccurate. On the contrary, other systems require students to evaluate the response, identify the significant portion within a larger block of text or code, or filter out solutions that are imprecise or did not address the security requirements mentioned in the challenge.

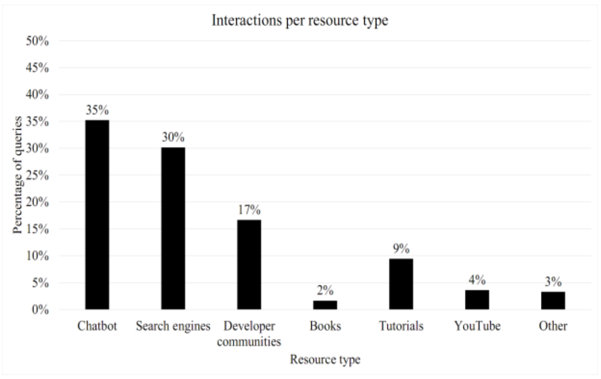


Figure 4. Interactions per resource type

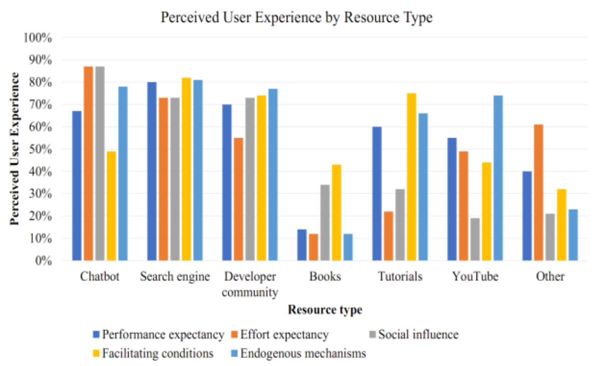


Figure 5. Perceived User Experience by Resource Type

6. PERFORMANCE EVALUATION: SPBOT VS GPT-3

Soon after the presentation of our original conference paper, OpenAI released their GPT-3 enabled chatbot, ChatGPT, on November 30, 2022. The provision of a user-friendly web-based chatbot interface resulted in skyrocketing growth of awareness of the capabilities of large language models. ChatGPT quickly became popular among software developers due to its ability to answer programming questions and providing users with code examples and snippets.

The appearance of ChatGPT inspired us to evaluate the performance of a large language model in answering secure programming questions and compare its performance with that of our custom secure programming chatbot. Due to the tremendous demand for ChatGPT, it was often inaccessible or would error out during chats, so we performed a comparison of our chatbot with the underlying GPT-3 model using OpenAI's API.

GPT-3 is orders of magnitude larger than our custom chatbot and was trained on 499 billion tokens (Dale, 2021). Our secure programming bot was trained on slightly more than 16,000 tokens. While the tremendous size difference might suggest that GPT-3 should perform far better than our custom chatbot, the restriction of the domain of interest to secure programming in PHP reduces that advantage. Furthermore, large language models like GPT-3 have well known flaws (Brown et al., 2020), including bias, errors in factualness, and a tendency to hallucinate text that was not in their training data (Maynez et al., 2020).

The objective of our study was to compare the performance of SPbot and GPT-3 in answering questions on secure programming. We evaluated both systems using the 215 questions related to secure programming that were asked by the students in our first experimental study (see Section 5) We ignored chitchat and other student questions that were not relevant to the topic of secure programming. We already had answers to the questions from SPbot from our original study. We use OpenAI's API to ask GPT-3 the same questions using the latest version of the model, *text-davinci-003*, which contains improvements designed to provide software developers with more accurate answers.

Pre-existing chatbot evaluation frameworks focus on Turing tests and usability studies or present different approaches for analyzing chatbots (similar to our usability study described in Section 4) (Maroengsit et al., 2019), while we wanted to evaluate the performance of chatbots in the domain of learning secure programming. Therefore, we created a rubric that enabled us to specifically take into consideration four key performance factors, which we considered the most relevant for helping students learn to write secure web applications:

- *Comprehension*: this measure evaluates whether the system was able to understand the question asked by the student. To this end, we score how well the answer of the system aligns with the question.
- *Correctness*: the degree to which the answer provided by the system is correct. In addition to evaluating the presence of errors, this considers the ability of SPbot and GPT-3 to output a response relevant to secure programming in the domain of web development.
- *Completeness*: this dimension evaluates how detailed the answer is, based on aspects that are left out, presence of code examples, and reference to additional resources.
- *Security*: this dimension evaluates whether the answer and code provided by the system contain any security flaws or overlooks or disregards important cybersecurity considerations.

The answers from GPT-3 and SPbot were independently graded by two experts in cybersecurity and full-stack development who graded each aspect using a 5-point Likert scale.
Our chatbot and GPT-3 were both evaluated with an overall average score of 4.2 out of 5, which indicates that they generally performed well in understanding the questions asked by the students, answering them correctly and completely, and addressing security aspects appropriately.

However, an analysis of the individual dimensions (see Figure 6) shows that SPbot and GPT-3 performed very differently. In the dimension of comprehension, SPbot scored 3.6/5, whereas GPT-3 scored 4.4/5, which indicates that OpenAI's system had better understanding of the questions asked by the students. SPbot's comprehension scored the lowest (i.e., 1/5) on 25% of the questions. On the contrary, GPT-3 failed understanding the questions in 12% of the cases only. GPT-3's huge training data set gives it a level of language fluency that SPbot cannot match.

In terms of correctness, SPbot and GPT-3 achieved the same score, that is, 4.4. Their highest and lowest performances are also comparable. However, SPbot scored 4.4 in completeness and performed better than GPT-3 (which scored 3.9). SPbot and GPT-3 performed the lowest in 6% and 9% of the cases, respectively. Finally, as far as security is concerned, SPbot outperformed GPT-3, with a score of 4.6 and 3.9, respectively. GPT-3's score was impacted by the fact that it scored the lowest in 15% of the cases, whereas SPbot failed to provide a secure answer in 6% of the cases.

The difference between the two systems is statistically significant at an alpha level of .005 for the dimensions of comprehension, completeness, and security, whereas there is not a statistically significant difference between SPbot and GPT-3 for correctness.

We also examined the 15 code examples provided by GPT-3. While all code examples provided by SPbot were written in PHP, 5 of the code examples provided by GPT-3 were written in JavaScript and 3 were written in Java. Of the 7 code examples written in PHP, all but one were responses to queries that either contained the word PHP or contained bits of PHP code.

Based on our findings, we can conclude that the effectiveness of GPT-3's language model in understanding users' questions is an appealing feature that immediately increases the degree of perceived usability, especially in terms of performance expectancy. This ability might be one of the primary reasons for its popularity and increasing user adoption. Even when provided with partial questions, GPT-3 was able to understand the context, complete the question, and answer it correctly.

Conversely, language fluency was SPbot's weakest point. From an educational standpoint, one of the pros of GPT-3's language model is that even if a student does not exactly know how to formulate the right question, they will be provided with an answer that is somewhat relevant to their prompt. On the contrary, SPbot requires students to have a preliminary understanding of the subject such that they can use key terms appropriately. However, these weaknesses could also be an incentive for students to learn the theory so they can use the correct terms when interacting with the chatbot.

We did not expect SPbot and GPT-3 to score similarly in terms of correctness, considering the differences in complexity of the two language models. In this regard, both systems could be an effective alternative for education, though SPbot's limited yet focused knowledge base represents a more efficient alternative to GPT-3. However, the statistically significant differences in the average scores achieved by the two systems regarding completeness and security raise the most concerns about the adoption of GPT-3 as a tool for teaching students how to develop secure software.

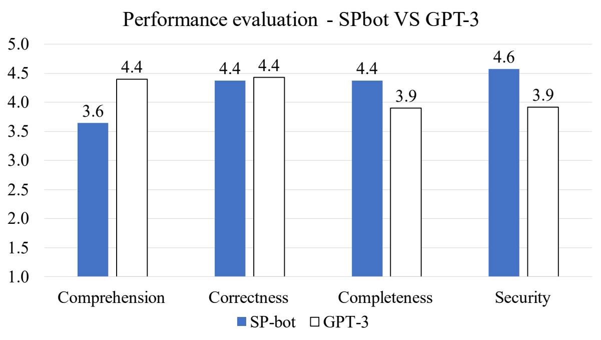


Figure 6. Performance of SPbot and GPT-3

Although the overall user experience with OpenAI's language model might be more appealing to a novice programmer who wants to achieve a basic and more general understanding of a topic, OpenAI's use of publicly available source code without any control or quality assurance process in place results in a superficial knowledge base that often outputs answers that expose applications and data to significant consistency issues, including code examples in programming languages other than PHP and security flaws. Consequently, using GPT-3 as a teaching tool could be harmful for students who would learn concepts from correct answers that do not address the topic thoroughly, overlook key cybersecurity aspects, or produce source code that contains security vulnerabilities. For instance, in several cases, GPT-3's answers contained code examples that are vulnerable to SQL injection, store password values without hashing them, or do not securely validate user input.

As a result, although GPT-3 is a significant milestone, we would not encourage its adoption for teaching secure software development. Nevertheless, its compelling user experience could be leveraged for introducing students programming topics for the first time and providing them with the opportunity to initially explore and understand concepts. Subsequently, after an first exposure, students could use more reliable systems based on language models trained ad hoc, with code examples from reputable sources, and with a more rigorous approach to the information in the knowledge base.

6. CONCLUSION

In this paper, we detailed the design, use, and user experience evaluation of a chatbot aimed at teaching secure programming concepts to students enrolled in web development courses. Our objective is to provide students with a user-friendly learning environment that simultaneously is a reputable source of information. Our findings align with previous studies (Abd-Alrazaq et al., 2020), which found that user experience is one of the key adoption factors of chatbots. Therefore, in our experiment, we focused on evaluating the overall user experience of the system based on the dimensions defined in the Unified Theory of Acceptance and Use of Technology (UTAUT) (Venkatesh et al., 2003. In contrast to other studies, including (Abd-Alrazaq et al., 2020; Mokmin & Ibrahim, 2021), which evaluated whether users enjoyed the conversational aspect of chatbot, we focused on the performance of the chatbot in providing accurate answers and on user experience metrics directly related with the goal of learning key aspects of secure programming via inductive reasoning guided by coding challenges.

We found that students interacted with the chatbot throughout the experiment more than with other information sources to learn about security topics and solve web programming challenges. Although the perceived performance of the chatbot was lower than other systems, such as search engines, its effort expectancy was ranked as a higher factor for adoption. Furthermore, although search engines and developer communities provide materials that were perceived as more accurate, users reported that screening resources requires additional effort in addition to the uncertainty of the quality. Furthermore, quantitative and qualitative data from our survey show that participants considered their user experience with the chatbot as extremely positive, which also suggests that the chatbot can be utilized as a convenient teaching, learning, and support tool for novice programmers.

We compared our secure programming chatbot with GPT-3. Quantitative data from our performance evaluation show that both systems were able to address most questions correctly, though GPT-3's language model was more effective in understanding the questions asked by the students. However, we also found a statistically significant difference in SPbot outperforming GPT-3 in providing more complete and secure answers. Therefore, we would conclude that chatbots built using a more reliable knowledge base should be preferred to systems that leverage publicly available data sets without adequate information quality assurance processes.

In the future, we plan to expand the bot's knowledge base to answer secure programming questions asked by students that currently have no answer. We also plan to improve the bot's training using the data provided by the students during the experiment. To help students with requests for additional information on a question, we plan to add suggestions for additional questions that the bot can answer in answers provided by the bot. We may also look at using a large language model as a foundation model to build a secure programming focused model, which could enable a chatbot to both have the language fluency of GPT-3 and the accuracy and security of our custom chatbot.

7. REFERENCES

Abd-Alrazaq, A., Safi, Z., Alajlani, M., Warren, J., Househ, M., Denecke, K., et al. (2020). Technical metrics used to evaluate health care chatbots: Scoping review. Journal of medical Internet research, 22 (6), e18301.

Acar, Y., Backes, M., Fahl, S., Kim, D., Mazurek, M. L., & Stransky, C. (2016). You get where you're looking for: The impact of information sources on code security. 2016 IEEE Symposium on Security and Privacy (SP), 289–305.

Adamovic, S., Sarac, M., Stamenkovic, D., & Radovanovic, D. (2018). The importance of the using software tools for learning modern cryptography. International Journal of Engineering Education, 34 (1), 256–262.

Almahri, F. A. J., Bell, D., & Merhi, M. (2020). Understanding student acceptance and use of chatbots in the United Kingdom universities: A structural equation modelling approach. 2020 6th International Conference on InformationManagement (ICIM). https://doi.org/10.1109/icim49319.2020.244712

Asare, O., Nagappan, M., & Asokan, N. (2022). Is GitHub's Copilot as bad as humans at introducing vulnerabilities in code? ArXiv Preprint ArXiv:2204. 04741.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., … Others. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

ChatGPT: Optimizing Language Models for Dialogue. (2022, November). OpenAI. Retrieved from https://openai.com/blog/chatgpt/

Chen, M., Fischer, F., Meng, N., Wang, X., & Grossklags, J. (2019). How reliable is the crowdsourced knowledge of security implementation? 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), 536–547.

College software texts found to teach insecure coding. (2008). SANS. https://web.archive.org/web/20210127172018/ https://www.sans.org/newsletters/newsbites/x/57#313

Dale, R. (2021). GPT-3: What's it good for? Natural Language Engineering, 27(1), 113–118.

Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., & Fahl, S. (2017). Stack Overflow considered harmful? the impact of copy&paste on Android application security. 2017 IEEE Symposium on Security and Privacy (SP), 121–136.

Introducting GitHub CoPilot: your AI pair programmer. (2022, June). GitHub. Retrieved from https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/

Green, M., & Smith, M. (2016). Developers are not the enemy!: The need for usable security APIs. IEEE Security & Privacy, 14 (5), 40–46.

Hazhirpasand, M., Ghafari, M., & Nierstrasz, O. (2020). CryptoExplorer: An interactive web platform supporting secure use of cryptography APIs. 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 632–636.

Hiesgen, R., Nawrocki, M., Schmidt, T. C., & W¨ahlisch, M. (2022). The race to the vulnerable: Measuring the log4j shell incident. arXiv preprint arXiv:2205.02544.

Ling, E. C., Tussyadiah, I., Tuomi, A., Stienmetz, J., & Ioannou, A. (2021). Factors influencing users' adoption and use of conversational agents: A systematic review. Psychology Marketing.

MacLeod, L., Storey, M.-A., & Bergen, A. (2015). Code, camera, action: How software developers document and share program knowledge using youtube. 2015 IEEE 23rd International Conference on Program Comprehension, 104–114.

Maroengsit, W., Piyakulpinyo, T., Phonyiam, K., Pongnumkul, S., Chaovalit, P., & Theeramunkong, T. (2019). A survey on evaluation methods for chatbots. Proceedings of the 2019 7th International Conference on Information and Education Technology, 111–119.

Maynez, J., Narayan, S., Bohnet, B., & McDonald, R. (2020). On Faithfulness and Factuality in Abstractive Summarization. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 1906–1919.

Meng, N., Nagy, S., Yao, D., Zhuang, W., & Argoty, G. A. (2018). Secure coding practices in Java: Challenges and vulnerabilities. Proceedings of the 40th International Conference on Software Engineering, 372–383.

Mokmin, N. A. M., & Ibrahim, N. A. (2021). The evaluation of chatbot as a tool for health literacy education among undergraduate students. Education and Information Technologies, 1–17.

Nguyen, D. C., Wermke, D., Acar, Y., Backes, M., Weir, C., & Fahl, S. (2017). A stitch in time: Supporting Android developers in writing secure code. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 1065–1077.

Okonkwo, C. W., & Ade-Ibijola, A. (2020). Python-Bot: A chatbot for teaching python programming. Engineering Letters, 29 (1).

Okonkwo, C. W., & Ade-Ibijola, A. (2021). Chatbots applications in education: A systematic review. Computers and Education: Artificial Intelligence, 2, 100033.

Oliveira, D. S., Lin, T., Rahman, M. S., Akefirad, R., Ellis, D., Perez, E., Bobhate, R., DeLong, L. A., Cappos, J., & Brun, Y. (2018). Api blindspots: Why experienced developers write vulnerable code. Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018), 315–328.

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? assessing the security of GitHub Copilot's code contributions. 2022 IEEE Symposium on Security and Privacy (SP), 754–768. IEEE.

Rahman, A., Farhana, E., & Imtiaz, N. (2019). Snakes in paradise?: Insecure python-related coding practices in stack overflow. 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), 200–204. IEEE.

Sugumar, M., & Chandra, S. (2021). Do I desire chatbots to be like humans? exploring factors for adoption of chatbots for financial services. Journal of International Technology and Information Management, 30 (3), 38–77.

Tahaei, M., & Vaniea, K. (2019). A survey on developer-centred security. 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 129–138.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. MIS quarterly, 425–478.

Verdi, M., Sami, A., Akhondali, J., Khomh, F., Uddin, G., & Motlagh, A. K. (2020). An empirical study of C++ vulnerabilities in crowd-sourced code examples. IEEE Transactions on Software Engineering.

Walden, J., Caporusso, N., & Atnafu, L. (2022). A Chatbot for Teaching Secure Programming. Proceedings of the EDSIG Conference. ISCAP.

Whitney, M., Lipford, H. R., Chu, B., & Thomas, T. (2018). Embedding secure coding instruction into the IDE: Complementing early and intermediate CS courses with ESIDE. Journal of Educational Computing Research, 56 (3), 415–438.

Wollny, S., Schneider, J., Di Mitri, D., Weidlich, J., Rittberger, M., & Drachsler, H. (2021). Are we there yet?-a systematic literature review on chatbots in education. Frontiers in artificial intelligence, 4.

Xiao, S., Witschey, J., & Murphy-Hill, E. (2014). Social influences on secure development tool adoption: Why security tools spread. Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, 1095–1106.

Xie, J., Chu, B., Lipford, H. R., & Melton, J. T. (2011). ASIDE: IDE support for web application security. Proceedings of the 27th Annual Computer Security Applications Conference, 267–276.

Zhai, X. (2022). ChatGPT user experience: Implications for education. Available at SSRN 4312418.

Zhang, H., Wang, S., Li, H., Chen, T.-H., & Hassan, A. E. (2021). A study of c/c++ code weaknesses on stack overflow. IEEE Transactions on Software Engineering, 48(7), 2359–2375.

Zhao, J., Song, T., & Sun, Y. (2020). Apihelper: Helping junior android programmers learn api usage. IAENG International Journal of Computer Science, 47 (1), 92–9

Editor's Note:

This paper was selected for inclusion in the journal as an ISCAP 2022 Distinguished Paper. The acceptance rate is typically 7% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a paper in 2022.

Teaching Case

Applied Steganography: An Interesting Case for Learners of all Ages

Johnathan Yerby
yerby_jm@mercer.edu
Computer Science
Mercer University
Macon, GA 31005, USA

Jennifer Breese
jzb545@psu.edu
Information Systems Technology
Penn State Greater Allegheny
McKeesport, PA 15132, USA

Abstract

There is a need for interesting demonstrations to capture the attention of learners in the fields of cybersecurity and cyber forensics. The number of new systems and amount of data grows constantly and needs to be secured using cybersecurity proactively and cyber forensics reactively. There is a large and growing gap between people aware, interested, and skilled enough to meet the needs. The United States is working to create a pipeline of learners that go into cyber related fields. This case study is an example of an exercise that gains the attention, interest, and positive feedback of students from middle school to college graduates. The exercise is a demonstration of steganography and only requires a computer and free software. The activity is tailorable for each audience with little preparation. The concept is simple, take a file, hide a message inside that is undetectable, then have the learners find the hidden information using software and passwords. Although the exercise is simple, it effectively gains students' attention and interest in the fields of cybersecurity and forensics and becomes the catalyst to have them imagine a new future. This case study discusses how to execute the exercise, the evolution of this exercise over the last decade, and how to use the case to allure new learners into the cyber field of study.

Keywords: forensics, cybersecurity, steganography, learning, case study

1. INTRODUCTION

There is an immense need for skilled cybersecurity professionals. Currently, K-12 teachers do not have exposure or skills to provide the inspiration and awareness for a career in cybersecurity. Time and funding are constraints that prevent cybersecurity from being introduced and practiced in K-12 and many workplaces. Students need to be introduced to, trained, and educated in the field of cybersecurity. There are barriers for K-12 teachers to be prepared to teach a field that is foreign to them as teachers, but also within the systems in which they are employed. Education is one of the least secure industries as a whole and a common target (Yerby & Floyd, 2018; Zimmerman, 2018). Practicing and teaching cybersecurity practices within educational institutions has not been the top priority for teachers and staff; instead they are focused on the roles that they are directly tasked with (Yerby & Floyd, 2018). From teaching to practicing principles, cybersecurity has not been a priority. Schools are looking for ways to fix the issues through training, awareness, education, and using modern technologies.

In the article *Children are the future of cybersecurity,* Webster (2017) described the paradox of high salary, secure jobs with great benefits, and a massive gap of skilled workers to fill the needs. Teenagers are not aware or confident that they could become a cybersecurity professional. Webster (2017) pointed out that the cybersecurity field needs more hands-on and fun activities for young audiences. There is an increasing emphasis on getting learners from K-12 to college interested in and skilled enough to meet the cyber needs of our nation. In 2015 the National Security Agency started a program called GenCyber, with eight prototype camps (Dark, Daugherty, Dark, Albright, Brown, Emry, & McCallen, 2021). Camps goals were to increase interest and diversity in the cybersecurity workforce of the nation, understand safe online behavior, and improve teaching methods for delivering cybersecurity content in K-12 curricula (Dark et al., 2021).

In the five years since the inception of GenCyber, 15,545 students attended cybersecurity camps at no cost. Another ongoing effort is Cyber.org, funded by the Department of Homeland Security (DHS) and the Cybersecurity Infrastructure and Security Agency (CISA)'s Cybersecurity Education Training Assistance Program (CETAP) grant (Noland, 2020). The group is working on preparing teachers, to prepare their K-12 students to meet the cybersecurity needs of the nation (Noland, 2020).

Cybersecurity is becoming more mainstream as represented in the plots of movies, television, and streaming service shows. Often media misrepresents reality of the field, which could help or hinder attaining a person's interest in the field, especially when reality falls short of dramatized expectations. A study examined numerous publications and found the literature to be mixed if there was a Tech-effect when reality and media representations mismatched (Paullet, Davis, McMillion, & Yerby, 2013). Educators should offer interesting and honest hands-on learning opportunities to reach deeper levels of learning and engagement. A 2013 (Paullet et al.) study examined how a potential tech-effect influenced jurors and found that an increased exposure to technology correlated with higher acquittal rates where digital evidence was used. Careful work must be done to introduce and positively present options in the broad field of cybersecurity. If reality is never presented to students from K-12 to college, they will not know if it is a field for which in which they may excel.

While students are being introduced to the concepts and ideas of what the field of cybersecurity entails, they need a realistic honest conceptual framework. Educators must be honest about the pros, cons, opportunities, and challenges of pursuing a life of learning and working in the field of cybersecurity. Cybersecurity is broad and ranging from technical to non-technical, frameworks, theoretical models, governance, physical security, sales, engineering, security architecture, forensics, to social engineering to name just some of the niches of the field. Again, how concepts or the field is introduced can influence the likelihood of sustained interest in becoming a cybersecurity professional. In a study with UK teachers of students 12–16 years-old, researchers found students feel more tech savvy than the teachers because they have grown up with technology, believe that hacking is glamorous, and are naive about how invincible they are online (Pencheva, Hallett, & Rashid, 2020).

The teachers point out that students are not technically aware of foundational cybersecurity concepts and that there is no adequate adult support network (Pencheva, et al., 2020). Using application-level learning has proven to engage students, especially when learning cybersecurity concepts (Nygard, Chowdhury, Kambhampaty, & Kotala, 2018). Using hands-on exercises following skills that would be used in real work is helpful. Deeply engaging students transfer knowledge, use it, and synthesize what they have learned (Nygard et al., 2018). Educators should strive to have engaged students and find the methods that reach their learners.

This case study explains how this unique exercise has been successfully used on learners from middle and high school through college students and parents. We began the study out of one exercise of over twenty in a cyber forensics course but noticed that there was something different about how this particular topic and technique resonated with learners. Future needs for recruiting events and building cyber related programs at multiple other schools led the authors and several colleagues to turn to this exercise.

The literature shows that Information Technology is a program that can be equally effectively delivered online or face-to-face (Yerby & Floyd, 2013). This is an exercise that is adaptable to in-person and online either synchronous, asynchronously, as an assignment, and as an interactive discussion. Overall, the entire exercise is very adaptable.

2. OVERVIEW

This exercise was created to meet several needs of a growing cybersecurity program, cyber forensics program, recruiting and outreach to students from middle school to college, and to increase interest and awareness of cybersecurity and cyber forensics as career opportunities. This is a simple lab to grab the attention of learners from middle school to adults and introduce them to something interesting and new. This steganography exercise uses a free website, a file, three passwords, and free software that can be run on a Windows PC.

The exercise began as one of many labs in an introduction to cyber forensics course. The reaction from the students was so overwhelmingly positive that the exercise went on to be demonstrated over 50 times in the intro to forensics class, cybersecurity classes, and a modified version in the intro to IT class as a preview of what students would learn if they decided to pursue the concentration in forensics or cybersecurity.

The exercise affords the demonstrator a large amount of flexibility on the inputs, outputs, and time that they wish to spend on the topic.

Before starting the assignments, the instructor should define and discuss the term steganography. Steganography is a technique to hide information in plain sight. In our case we will be using an image file to hide text, utilizing software.

Students of all ages are creative with thinking of nefarious and honest uses for hiding information within another file. Some examples that instructors may bring forth to discuss include:

1. Avoiding censorship
2. Watermarking to prove ownership or authenticity
3. Intellectual property protection or tracking
4. Store information on location
5. Terrorism
6. Exploitation

Students can begin with a prescribed set of inputs and outputs, detailed in Assignment 1 (Appendix B), then extend the exercise such as in Assignment 2 (Appendix C). After completing the exercise, the instructor has additional flexibility to introduce new related topics for the scenario of used in recruiting, capture-the-flag cyber competitions, in courses such as intro to IT, intro to forensics, or cybersecurity. The demonstration that has worked well over 50 times by one of the authors, and the second author over 20 times. Other instructors at other institutions have all reported positive findings and student interactions as well.

3. STEGANOGRAPHY

Steganography, the term translated from Greek means “covered writing” and is different from encryption which means “secret writing” in Greek (Khan, 1996). The introduction of this exercise was the first instance most students have ever heard of the term steganography, and commonly misheard it as stenography, which is recording spoken word, typically in a courtroom setting. Steganography is a technique to hide information in plain sight, it is a science of invisible communication (Singh & Singh, 2018). The term dates to the fifth century BC, where according to author Herodotus, messages would be tattooed onto shaved heads of a servant, and then once the hair grew back, the message would be sent (Khan, 1996). Steganography is a new fascinating topic to students and teachers. Diving into the possibilities of the ancient data hiding method inspires engaged discussions related to spy craft, invisible ink, criminal behavior, watermarking, and animal behavior before moving into anything related to technology.

In cyber related terms, steganography uses a cover, the hidden message, and sometimes also requires a key. The cover is an innocent looking file or set of files, which would be of no apparent value or danger if it were intercepted and viewed by an adversary. The cover file is intentionally designed to appear to have nothing suspicious. The hidden message can be text, cypher-text, other images, or anything that can be embedded into the cover file (Johnson & Jajodia, 1998; Hamid, 2018). The best files to use as covers are lossless files such as BMP and GIF, because there is information that can be replaced, compressed, or removed with less noticeable results to the image (Singh & Singh, 2018).

Students need the correct software to complete the process. There are many programs that perform steganography detailed in Table 1. Openpuff was used for this exercise; however, it can be adapted to use any of the software listed above. The interface of each program varies. The software typically has the cover file identified by the user first, then the hidden information is selected. Users complete the process by entering a key, which is a single or series of passwords. Then the software executes to create a stego-file. The stego-file looks remarkably like the cover file; however, it now has hidden information. The stego-file can be examined and usually the difference in imperceptible to the human eye. Some of the properties of the file may have changed and changes depending on several variables such as the software, size and format of the cover and hidden message.

Tool	Windows	Linux	Mac	Function
Camouflage	x			Data Hiding, Encryption
Crypture	x			Encrypt, Header modification, save as bmp
DeepSound	x			Data Hiding in Audio
Hide & Send	x			Data Hiding, Encryption
Image Steganography	x			Data Hiding, Encryption
iSteg			x	Data Hiding, Encryption
matroschka		x		Command line Data Hiding, Encryption
Openpuff	x			Data Hiding, Encryption
OpenStego	x	x	x	Data Hiding, Watermarking
OurSecret	x			Data Hiding, Encryption
Outguess			x	Data Hiding, Encryption
Rizzy built on Stepic		x		Data Hiding
SSuite Picsel Security	x			Data Hiding, Encryption
Steg		x		Data Hiding, Encryption
Steganograph XPlus	x			Data Hiding, Encryption
SteganPEG	x			Data Hiding multiple files
StegCloak	x	x	x	Web Interface, Data Hiding, Watermarking
Steghide	x	x		Data Hiding including audio files
Stegolego		x		Data Hiding in bmp only
S-tools	x			Data Hiding, Encryption
Xiao Steganography	X			Data Hiding in BMP and WAV, Encryption

TABLE 1: Free Steganography Tools by Platform and Functionality

Although the concept of steganography is ancient, it continues to be an effective method to covertly communicate or hide information. Just as there are dozens of software solutions dedicated to performing steganography (Table 1), there are also several programs designed to at least detect that a file has been subjected to steganography. At that point, it becomes a chore of trying to solve the type of encoding, which software, and version sometimes, and the keys. Steganography is often undetectable, and even when it is, the detection does not always result in uncovering the hidden message. For this exercise, students' imagination runs wild on the nefarious uses that they envision using this new skill to do good and harm in their world. The runaway conversations and freedom to use what they are beginning to understand hooks their attention.

4. EVOLUTION OF EXERCISE

The exercise began as a small piece of a greater lesson about data hiding techniques for a cyber forensics course in 2010. The instructor discussed several techniques during a lecture and then had multiple groups of students find a tool and perform the process as part of an in-class presentation. Something about this resonated more than most other topics. Students were captivated and continued discussing the lesson, tools, uses, and their genuine enjoyment. Soon students not even enrolled in the course were discussing steganography and using the method in other courses and created some online challenges for a capture-the-flag contest they were developing.

This simple exercise became a rite-of-passage when taking the intro to cyber forensics course. As time went on, there were some students in the class that already knew about the exercise and had practiced, because they found out from a friend. However, that did nothing to detract from the interest of other students and in fact created a more robust discussion and a drive for the students to push the technique further and challenge one another. Around the same time the instructor was working on creating something as a short demonstration for open house recruiting events and decided to use steganography.

Comments from students:

> Steganography -- It was interesting to see how the OpenPuff application worked and get some hands-on experience.
>
> When it came to the steganography module, I found it interesting how hidden information can be placed within low-quality images and other files. After watching the videos on how steganography works, I was able to determine that some organizations or criminals may use steganography to hide licensing information or secret files that will be concealed from the naked eye unless the individual has the software to find the hidden information. I enjoyed how we were able to use a hands-on tool to hide a document within a photo and also find hidden information in my classmates' files.
>
> I was a criminal justice major when I learned about how to do steganography, I had to switch majors to forensics this semester.

The authors have used this demonstration or exercise more than seventy times with remarkable success. It has been used in courses focused on cyber forensics, cybersecurity, and introduction to IT. The abbreviated lesson can be demonstrated as quickly as 5 minute "trick" and then discussion can ensue afterwards; the lesson worked. The activity has been performed for and by hundreds of people open house recruiting events, K-12 outreach, assignments, and sharing with colleagues as something clever that has students engaged.

In 2013, the instructor created a Weebly website with three videos and a hands-on portion for student to complete (Yerby, 2013). The first video (2-minutes) covered the background of steganography. The second video (9-minutes) was an instructor walk through of a similar exercise using a different, but analogous tool that students were expected to utilize in their hands-on portion. The exercise in the demonstration was based on a hands-on exercise from the textbook *Guide to Computer Forensics and Investigations* (Nelson, Phillips, & Steuart, 2009). The third and final video was a follow-up instructor led video providing additional steganography learning material.

The reason this method was used is because it is impactful yet simple to grasp. It is hands-on and does not require a lengthy period of planning or technology acquisition. This exercise adds time for critical thinking and creativity which allows learners to have fun with the topic and even think as an adversary. In the second portion of the assignment students have flexibility to turn the challenge into a game to see how difficult or how fast they can solve one another's challenges.

The hands-on exercise on the website provides hints for completion but allows for free-thinking and problem solving by student participants. The exercise underwent peer review and was then reproduced with permission and shared multiple other channels including the Curriculum Standards series (National CyberWatch Center Curriculum, 2018). Educators have free access to utilize this activity.

Variations of the exercise included chapter readings in assigned course textbooks, instructor created capture-the-flag competition in class, and in class exercises (Nelson et. al., 2009 & 2018). Additionally, senior-level students were required to find their own cover image, hide a message, and provide the hints, including passwords, for peers to access the message with a response in a discussion post format with their Learning Management System. After posting the stegged images and enough hints to solve them, the students were expected to provide synopses about their thoughts on the pros and cons of using steganography. In-class or online discussions would ensue about the methods or tools each student chose to use. Sometimes the students would complain that there were not enough hints provided to decipher the stegged file. Students would discuss the challenges that they uncovered based on factors such as file size, file type, specific tool limitations, and the methods that each type of steganography tool was attempting to use. In some courses, the instructor gamified the submissions, to see which student could solve or uncover the most stegged files during a period of time. Although the method dates to the fifth century and this version of this exercise has been utilized for over a decade, there is something about it that remains fresh and engaging. Students are excited to win the challenge, to be first, or to make variations to make their version the best.

5. SET-UP

The straightforward implementation of the exercise involves reading, defining, and discussing what steganography is and additional applications of use. Students are encouraged to think about positive and negative utilization of the method. Students are asked if they believe they would be able to determine if there was hidden information in a file.

Students were directed to the Weebly page, http://yerby.weebly.com/forensics.html, as shown in Appendix A, or was embedded in the course Learning Management System (LMS). Then students use in-class time to try to be the first person to solve the challenge just using the limited information on the page. This gamification further creates a sense of interest and competitiveness.

After some students figured out the challenge, the instructor reviews the steps collectively. First, the students had to read carefully to download the correct software, which was Openpuff version 4. Some students may experience problems if they downloaded a different version or the wrong software altogether, such as S-tools, which was used in the demonstration video and is also available on the webpage. To solve the issue of the wrong version, simply guide the student to read the prompts more closely and use the correct version. Paying attention to details is a skill required to work in the fields of security and forensics. Once students run the correct version of the software, they need to provide the stego-file, which is a photo on the webpage. The photo is a German Shepherd dog, which came from an episode of *It's Always Sunny in Philadelphia* (Day, Howerton, & McElhenney, 2012). In the episode one of the characters used an old canvas to paint a new picture on, hence the nod to steganography. Students need to right-click on the webpage and save the image to their computer as secret.bmp, which is the existing name of the stego-file.

Next students execute the OpenPuff.exe file and select Unhide. They enter the three passwords which were given as "clues" on the webpage. If the passwords were copied incorrectly or an extra space is included, the uncovering does not work. To solve this issue, simply make sure that students are getting the passwords verbatim, with no additional spaces. Next students select Add carrier then select the previously downloaded secret.bmp file. At this point select Unhide on the Openpuff 4.0 interface, select a save location for the hidden message, and users are given a confirmation that "1/1 carrier processed." Clicking "OK" gives a summary of the process, including the name of the hidden data and the size. The file is saved, and the challenge is solved.

The entire process can be completed in multiple variations including having students use different software or creating their own stego-files.

6. CONCLUSION

While this exercise in its variations has been a small part of both intro and advanced courses it consistently ranked among the top experiences by students in their course feedback. Students can recall the discussions related to and hands-on use of this activity above others. The lesson can be as short or long as desired to provide an engaging introduction or as a leveling skill. Steganography does not stop with images, audio files can also be used, but the process and software are slightly different. Steganalysis is an interesting tangent related to this exercise. This short exercise is the opening to begin exposing students to the possibility of a future meeting the needs in cybersecurity and forensics work.

Steganography is a fantastic way to introduce the field and potential future of working in cybersecurity to learners from diverse ages, backgrounds, and educational interests. Image steganography is not the stopping point. You can extend this skill and conversation into documents, audio, and video. This activity is easy to turn into gamified learning, to further increase interest. Instructors can learn enough to feel confident in delivering the lesson and leading an in-class discussion. This small exercise is really something that works and is an interesting, quick set-up for learners of all ages.

7. REFERENCES

Dark, M., Daugherty, J., Dark, R., Albright, H., Brown, D., Emry, M., & McCallen, A. (2021) GenCyber 5-Year evaluation 2015-2019. https://tinyurl.com/gencyb

Day, C. (Writer), Howerton, G. (Writer), McElhenney, R. (Writer), & Shakman, M. (Director). (2012, October 11). Pop-Pop: The Final Solution (Season 8, Episode 1) [TV series episode]. *It's Always Sunny in Philadelphia*. 3 Arts Entertainment, FX Productions, & RCG Productions (II).

Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *Computer*, *31*(2), 26-34. https://doi.org/10.1109/MC.1998.4655281

Kahn, D. (1996, May). The history of steganography. *International workshop on information hiding* (pp. 1-5). Springer, Berlin, Heidelberg.

National Cyberwatch Center Curriculum Standards Panel (2018). National Cyberwatch Center. https://www.nationalcyberwatch.org/csp/

Nelson, B., Phillips, A., & Steuart, C. (2009). *Guide to computer forensics and investigations*. Cengage Learning.

Nelson, B., Phillips, A., & Steuart, C. (2018). *Guide to computer forensics and investigations*. Cengage Learning.

Noland, H. (2020, June 25). *National Integrated Cyber Education Research Center Unveils Rebrand to CYBER.ORG*. Cyber.org. https://cyber.org/news/nicerc-unveils-rebrand-cyberorg

Nygard, K., Chowdhury, M., Kambhampaty, K., & Kotala, P. (2018, April). Cybersecurity materials for K-12 education. *The Midwest Instruction and Computing Symposium 2018*.

Paullet, K. L., Davis, G. A., McMillion, S. K., & Yerby, J. (2013). The new tech effect: A comparative analysis of two universities. *Issues in Information Systems*, *14*(2), 12-22.

Pencheva, D., Hallett, J., & Rashid, A. (2020). Bringing cyber to school: Integrating cybersecurity into secondary school education. *IEEE Security & Privacy*, *18*(2), 68-74.

Singh, H. & Singh, M. (2018). Color image steganography techniques - A review. *RA Journal of Applied Research, 4(1)*. https://doi.org/10.18535/rajar/v4i1.01

Webster, M. (2017, June 5). *Children Are the Future of Cybersecurity*. EDUCAUSE. https://er.educause.edu/blogs/2017/6/children-are-the-future-of-cybersecurity

Yerby, J., & Floyd, K. (2018, August). Faculty and staff information security awareness and behaviors. *Journal of The Colloquium for Information Systems Security Education, 6(1),* 1-23. https://cisse.info/journal/index.php/cisse/article/view/90

Yerby (2013). Forensics - Steganography. http://yerby.weebly.com/forensics.html

Yerby, J., & Floyd, K. (2013). An investigation of traditional education vs. fully online education in information technology. SAIS 2013 Proceedings. http://aisel.aisnet.org/sais2013/40

Zimmerman, E. (2018, December 14). *Three cybersecurity focus areas for education institutions in 2019.* EdTech. https://edtechmagazine.com/higher/article/2018/12/3-cybersecurity-focus-areas-education-institutions-2019

APPENDIX A
Website containing instructions, file, hints, and tools

Steganography

To begin to understand what steganography is, watch the three short videos below. You will likely want to watch them in full screen mode to see them at the best resolution.
The first video will give you an understanding of what steganography is, the second video will give you a demonstration of using S-tools to complete the challenge below, and the final video gives a review that will help you understand the process in more detail.

Find the hidden information in the attached file...

There is a hidden message hidden within the image below using steganography.
Please do not share the solution to keep it fair to everyone.
Here are your only clues:

- forensics techniques hidedata
- I used this software - version 4.0
  - http://embeddedsw.net/OpenPuff_Steganography_Home.html

openpuff.zip
Download File

s-tools4.zip
Download File

Source: http://yerby.weebly.com/forensics.html

APPENDIX B
ASSIGNMENT ONE

ASSIGNMENT ONE is prescribed, and the instructor will be able to determine what the secret message is. Every student who completes the challenge will get the same message.

1. Students begin by going to http://yerby.weebly.com/forensics.html
2. Students should use their imagination for the uses of steganography. Both with and without involving computers. Write two responses.
3. On the webpage above there is a 9-minute video demonstration of how to use steganography.
    a. Download steganography software.
    b. Select a cover file (an image file)
    c. Drag the .bmp file into the software.
    d. Select a .txt file.
    e. Drop it on top of the .bmp file.
    f. Enter a passphrase.
    g. Select the encryption algorithm.
    h. Select OK
    i. A new identical looking image will appear with a window titled "hidden data"
    j. Right-click the file and save the file as Steg.bmp
    k. Examine the file with an image viewer.
4. After viewing the demonstration, students will attempt to unhide data in another file that is displayed.
5. They will need to right-click the large image of a dog on the webpage, and save it as "secret.bmp"
6. Then they will download OpenPuff version 4.0. They must get the correct version for the exercise to work.
7. OpenPuff 4.0 is an executable that does not need to be installed – just run it.
8. Once you run the software you will see the menu displayed in Figure 1. Select "Unhide."

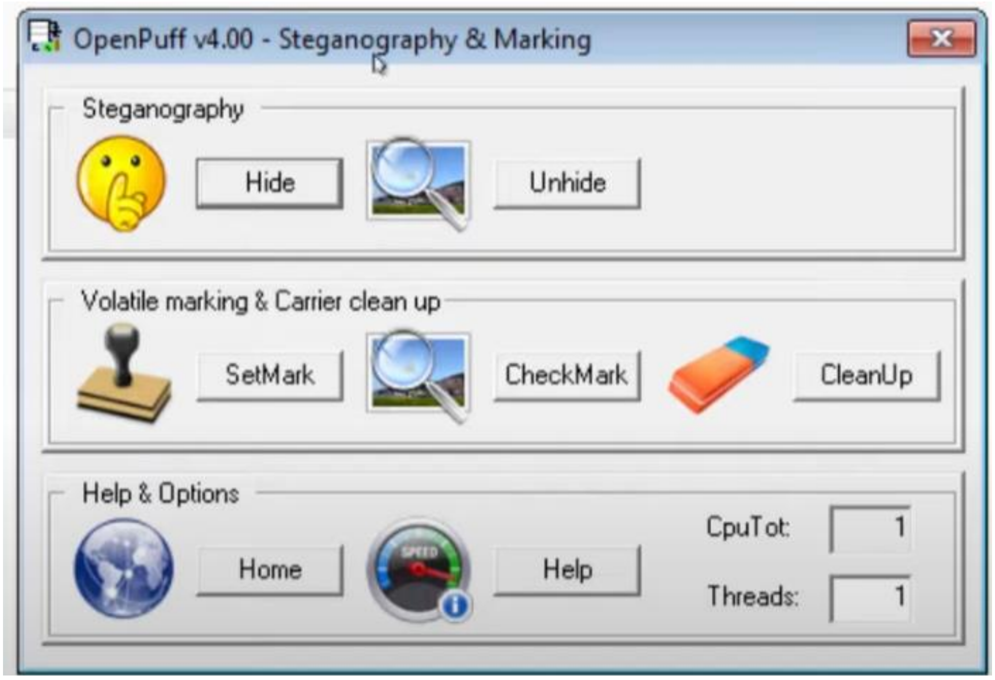
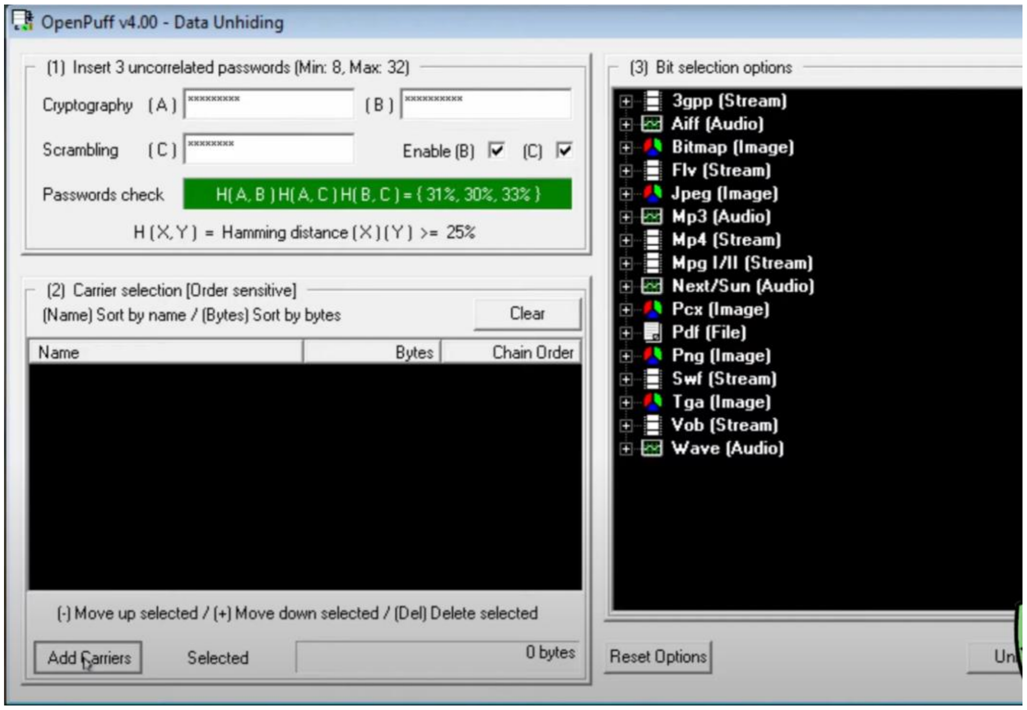


Figure 1. OpenPuff 4.0 select Unhide

9. Students should navigate back to the webpage to copy the three passwords precisely. If they add an extra space or misspell any of the words or put them in the wrong order the Unhiding will not succeed.

10. Next select "Add Carriers" as shown in Figure 2.


OpenPuff v4.00 - Data Unhiding

(1) Insert 3 uncorrelated passwords (Min: 8, Max: 32)

Cryptography (A) ********** (B) ***********

Scrambling (C) ********* Enable (B) ✓ (C) ✓

Passwords check H(A, B) H(A, C) H(B, C) = { 31%, 30%, 33% }

H (X, Y) = Hamming distance (X) (Y) >= 25%

(2) Carrier selection [Order sensitive]

(Name) Sort by name / (Bytes) Sort by bytes

Name	Bytes	Chain Order

(-) Move up selected / (+) Move down selected / (Del) Delete selected

Selected 0 bytes

(3) Bit selection options

- 3gpp (Stream)
- Aiff (Audio)
- Bitmap (Image)
- Flv (Stream)
- Jpeg (Image)
- Mp3 (Audio)
- Mp4 (Stream)
- Mpg I/II (Stream)
- Next/Sun (Audio)
- Pcx (Image)
- Pdf (File)
- Png (Image)
- Swf (Stream)
- Tga (Image)
- Vob (Stream)
- Wave (Audio)

Figure 2. After inserting passwords, Add carriers in OpenPuff 4.0

11. Next select the secret.bmp file that was previously downloaded from the website.
12. Click "Unhide"
13. Save the file to the desktop. Students will get a success message and task report indicating that a new file secret.txt was created.
14. Students open the secret.txt file, view the secret message and submit it as evidence that they solved the challenge.

APPENDIX C
ASSIGNMENT TWO

ASSIGNMENT TWO allows students to branch out and create their own version using this skill. Every student that completes the challenge will get the message that was added by another student.

For ASSIGNMENT TWO, students can create and solve their own steganography puzzle using software detailed in Table 1 in this paper. The instructions have less structure for this second portion of the assignment to grant additional opportunity for students to be creative and employ the skill using a different method and software.

1. Run a new steganography tool. The students will need to examine what is available and compatible with the computer that they are using. A few on the list are Linux utilities and many others run in modern Windows operating systems. Students may start with a tool that does not work for them. They are allowed to choose a different tool.
2. Choose a low-resolution image that is in a file format supported by the tool selected by the student. Then add a (short) .txt sentence/message using the tool/program.
3. Choose a low-resolution image that is in a file format supported by the tool selected by the student. Then add a (short) .txt sentence/message using the tool/program.
4. Visually inspect the two images and describe the differences before and after hiding information in the image. Describe any differences you see in 2-3 sentences. Submit your images for comparison and your comparison.
5. Students will usually see no difference in the high resolution and will sometimes see degraded image detail in the low-resolution images.

A Case Study in Identifying and Measuring Skills Honed from a Cybersecurity Competition

Ron Pike
rpike@cpp.edu

Jasmine Weddle
jweddle@cpp.edu

Sydney Duong
slduong@cpp.edu

Department of Computer Information Systems
Cal Poly Pomona
Pomona, CA United States

Brandon Brown
bbrown118@coastline.edu

Department of Computer Service Technology
Coastline College
Fountain Valley, CA

Abstract

Many tools have evolved in the extracurricular space for the cybersecurity field that could belong in cybersecurity education. Additionally, the measurable learning students complete outside of formal classroom education before entering college or university needs to be formally recognized and awarded academic credit. These strategies are novel in their approach for learners to gain or demonstrate their cybersecurity skills meaningfully. The following case explores a dynamic example about the benefits of these tools through a cybersecurity competition named Red vs. Blue. The case outlines methods of assessment relative to the competition and exemplifies the impact on students from a test case competition in their own words.

Keywords: Cybersecurity, Competency-Based Education, Assessment, Cyber Competitions.

1. INTRODUCTION

The focus of this paper is to highlight the efforts of students and faculty in measuring skills derived from a cybersecurity competition created and operated by students at a state university in the Southwestern United States. The university is part of the National Centers of Academic Excellence (CAE) program and focuses on students putting their learning into practice.

The students are engaged in cybersecurity academic programs and are contributors to co-curricular and extra-curricular cybersecurity programs and research. One of the students is studying Computer Science within the College of

Science and the other Computer Information Systems within the College of Business. There are two perspectives presented in the paper: the students examining the impact of a particular cybersecurity competition on the learning environment and the faculty exploring the process of assessing and mapping competencies in a fashion that supports robust learning pathways from middle school to career.

Cybersecurity, like many other professions, has a need for lifelong learning. However, we argue that cybersecurity faces a greater challenge than most disciplines related to lifelong learning, due to four key challenges. First, cybersecurity is in an early phase of maturation as a discipline, which creates a need for constant change and development. Second, cybersecurity is influenced by ongoing change within the IT infrastructure domain. Third, cybersecurity faces an adversarial relationship with perpetrators who make a profession out of undermining and evading cybersecurity controls. Finally, the fourth challenge is the multi-disciplinary aspect of cybersecurity, which encompasses multiple technological fields not limited to computer engineering and computer science, but also encompassing business, political science, psychology, and more. All these fields are required to develop security and privacy controls necessary to empower and protect organizations and individuals. Each of these challenges contributes to a constantly changing landscape for the cybersecurity field and the academic programs that service the field. Such rapid changes create a need for flexibility in cybersecurity programs to adapt to new developments within the IT infrastructure domain while also creating and maintaining an academic core that ensures the consistent growth of the field. A grand theory of cybersecurity is needed, but none are visible on the horizon.

Academic programs in cybersecurity have drawn on numerous tools and pedagogies to address these challenges. We explore key tools and pedagogies that allow for academic programs to interconnect with one another and external learning opportunities, creating pathways for learners from their first exposure with cybersecurity to the end of a successful cybersecurity career. We also draw on the concept of holistic assessment to help learners demonstrate their cybersecurity learning achievements to academia and employers, as well as enable academic programs to honor learning already achieved (Pike, 2022). Academic tools such as articulation, transfer of credit, competency-based education, and credit for prior learning will enable academic providers to pattern their programs in a way that builds upon prior learning, as opposed to requiring students to retake classes on concepts they have already learned in order to fit into a particular academic program. The freedom to build upon learning already achieved empowers students as they navigate through learning pathways toward cybersecurity competencies.

2. METHOD OF ASSESSMENT

The methodology of holistic assessment is to broaden the types of cybersecurity learning environments that can be formally assessed. This not only increases the activities that can be assessed, but also offers more authentic forms of assessment with a focus on measurable competencies. For example, asking a student to define how they would enhance firewall protections in the event of an emergency by writing an essay is not effective, especially for learning with language or writing deficiencies. A much stronger form of assessment is to place a student in an environment, such as a competition, where they are confronted with an emergency, and the firewall must be configured to mitigate the current threat.

Experiential learning may be foreign to many students but is a core principle at this university. The university's motto is "Learn by Doing" and all programs work to provide experiential learning in their domains. Davenport and Markus, in their paper on rigor vs. relevance, suggest that the IS discipline should use the fields of medicine and law as exemplars to follow, in large part due to their commitment to experiential learning (Davenport & Markus, 1999). While Davenport and Markus focus their paper on research, they also call for the discipline's research to be incorporated into academic teaching which means the classroom requires relevance made possible by experiential learning as well. Cybersecurity fits Davenport and Markus's model of a discipline that requires experiential components in academic programs to provide relevance.

Competitions, and other forms of experiential learning, are more authentic both because the student is doing the work rather than writing about it, and because there is a live context that helps the student to make sense of the task at hand. While writing is an effective learning tool in many cases, it is not effective in evaluating students' ability to adapt to active-attack scenarios. Experiencing an active-attack scenario in a competition compels the student to utilize cybersecurity tools in real-time which

demonstrates an understanding of the threat, an understanding of an appropriate mitigation, and the ability to wield a mitigation tool effectively. As mentioned above, this is an authentic assessment of a student's knowledge and skills and uses the skills under study to demonstrate the learning. A student with a learning disability which limits their writing can demonstrate their learning without an outside hinderance (poor writing) diminishing their outcome.
Some students may find the time-pressure of competitions to be difficult and even stressful; however, experiencing this discomfort in an academic setting allows a student to develop strategies to handle such discomfort and perhaps even guide a student toward roles in the cyber field that do not require such activities. Other students find they excel in active-attack scenarios and pivot their ambitions toward career roles where real-time attack and defend activities are part of the job. In either case, students are given an opportunity to better understand roles in the cybersecurity field and prepare themselves for related challenges. Our example of the dynamic Red vs. Blue competition (RvB) sets forth a case study in the analysis of skills attainment through Competency Based Education, Life-Long Learning, and Skills-Based Education practices.

COMPETENCY BASED EDUCATION (CBE) ASPECTS TO EVALUATION OF SKILLS
Competency Based Education (CBE) is defined as an approach which allows a student to advance based on their ability to master a skill or competency at their own pace, regardless of environment. The CBE method can be tailored to meet different learning abilities enabling more efficient student outcomes. (Educause, 2022). This modality of learning is focused on measuring the skill level already attained or through flexible learning methodologies. Students can progress through courses as soon as they can prove that they have mastered the material. CBE replaces the more traditional modality of higher education, where they would normally advance only when a term has ended and allows the student to progress at a faster rate since they can spend more time on skills and materials that are not already mastered.

Competitions offer the ability to measure the level of skill that a student possesses in a simulated/emulated real-world environment. The challenge here is the methodology of measuring the skill and setting a level to it. Beginner competitions, such as Capture the Flag events, are structured in a fashion where student learning outcomes can be measured efficiently since the challenges and problems are more static. However, many competitions that are open and flexible, such as Red vs. Blue style scenarios, are more dynamic and harder to measure. This is especially true for team-based competitions, as multiple students can work on the same problem or issue, and the ability to disambiguate which student demonstrated a particular competency is required.

3. CYBERSECURITY LIFE-LONG LEARNING

In a recent conversation among the authors of the paper, the maturity of the cybersecurity field was likened to a 40-year-old living in the basement of a parent's home. The field gets older but has not established many of the traits of a mature discipline, such as research streams building upon one another and the use of ongoing research from the field being a strategic asset in the teaching and learning process. While critical work has been accomplished in defining core components of cybersecurity through joint work between academia and government, this work is still in development and is not broadly used across academia (Petersen, et al, 2020). In fact, it seems there are many versions of core content for cyber academic programs, meaning that there is no true core, and a single academic core is likely not possible given the need for contributions related to computer hardware, software and protocols, implementation platforms and systems, organizational structures and ethical and psychological considerations which span numerous disciplines.

We do not seek to provide an answer to the question of a core for cybersecurity education, but rather suggest a systematic exploration and methodical documentation of competencies to enhance learners' preparedness. We also call for the use of experiential learning to develop and assess skills. This process may illuminate paths to developing a stronger theoretical foundation for cybersecurity. In our exploration of CBE, we focus on the use of pedagogical tools that have been largely relegated to extra-curricular activities but should be required in formal cybersecurity education. Specifically, the assessment of competencies through competitions and Competency Based Education (CBE) is explored with an intent to create a wider variety of assessable activities that report on the attainment of Knowledge, Skills, and Abilities (KSAs) related to cybersecurity. This paper assumes the use of the NICE (National Initiative for Cybersecurity Education) framework as a source of KSAs, which have been developed through joint work between the federal government and academic partners.

4. INTRODUCTION TO RED VS BLUE (RVB)

Red vs. Blue (RvB) is a cybersecurity competition where teams of five students, the blue team, are provided with a fake business' vulnerable network, usually consisting of a router and five machines running different operating systems, that they must secure. The students are also actively defending their systems against outside threats that the red team (attackers) inflict upon them. Through RvB, students can troubleshoot and practice their incident response skills in an active breach scenario.

RvB uses three main metrics to give points to participants: Secure Configuration Score (SCS), Service Uptime, and Injects. SCS measures specific security configurations that are implemented on the computer. It rewards points for things such as removing unauthorized users or turning on the firewall. Please see Appendix A for examples of security misconfigurations for a Windows machine and Appendix B for examples on a Linux machine. Since RvB simulates a real business environment, the competitors' main job is to keep the business' services, such as DNS and HTTP, up and running. Service uptime awards competitors points for multi-minute intervals in which services are still functional. Please see Appendix C for a table of all the scored services in the competition. Lastly, throughout the competition, as if defending a network from an active red team was not enough, students are given tasks, called injects, from the fake business manager. These injects are designed to allow students to apply their technical skills in a business setting, doing things such as submitting a report summarizing the findings on the threats they've encountered, or giving a presentation on cybersecurity safety guidelines that all employees should follow. Submitting injects within deadlines provides additional points.

5. RVB & TEACHING OF TECHNICAL SKILLS

Cybersecurity competitions are an efficient way of teaching and reinforcing students' technical skills, as students learn so much more through applying the knowledge they have accumulated in the classroom in a hands-on way and troubleshooting when coming across problems. RvB holistically assesses a student's skills, since the true test of whether a student understands technical concepts is how well they apply their understanding and create solutions or mitigations in simulated real-world situations.

Through competing in RvB, students can apply concepts such as setting up DNS, configuring a pfSense router, and editing the Windows Registry. The list of cyber tools students implement are seemingly endless as students use their initiative and creativity in developing solutions. A common situation competitors come across is finding that their computer is filled with hundreds of users, each with domain level privileges and default passwords. Having all these high-level privileges with default passwords poses a serious vulnerability to their network, since a threat actor can easily log in on any one of the hundreds of accounts and then gain domain-level access to the machine. Therefore, competitors must quickly research on the spot how to mass change user passwords to secure this vulnerability. Students then discover how to use commands such as *dsmod* and *dsquery* and how they are applied to real-world scenarios. This is one of countless examples of how competitors learn through the unexpected situations they encounter.

Additionally, students do not only learn from the technical problems that they are able to overcome during the competition. Their failures and shortcomings serve as a starting point for further research after the competition ends. After every competition, the development team and the red team perform a debrief, where they take turns explaining all the vulnerabilities on the machines, how they can be breached, and ways to patch them. Through these debrief sessions, competitors learn what they missed and how to fix them for next time. One student example of this is based on a situation with PAM, or Pluggable Authentication Module, which is used by Linux systems to authenticate a user to applications or services. PAM was misconfigured on this student's computer to allow any input for a password, meaning that a red teamer could login to a user with a random password. This student was aware that someone was logging onto their computer but was unsure how they were able to do so. Through the debrief, the student was told about PAM, and learned about a new Linux file and can now do their own research about how to secure it for future competitions.

Competing more than one time in RvB also poses a valuable way for students to learn. One past competitor stated, "most, if not all, of my technical skills I practiced during RvB. I would research how to avoid these vulnerabilities by looking up fixes to them after the event, making me more effective for the next [one]" (Student A, May 30, 2022). Cybersecurity competitions, such as this one, greatly motivate students to take the initiative to study for their own improvement. When a student is able to directly experience what it feels like to be breached, they are inspired and more compelled to do research on the different ways to prevent the breach in the future, as displayed through one past competitor describing their experience competing in RvB:

> Because RvB has a more hands-on approach than a traditional class, I was able to become more interested in cybersecurity. Additionally, because RvB has a competitive aspect, it was fun trying to secure as many services as I could and see how other teams were doing as well. (Student B, May 30, 2022)

Student competitors greatly enjoy and take great interest in Red vs Blue. This interest sparks a desire for wider and deeper learning based on the material they encounter during the competition.

The lessons and skills learned from competing in competitions, such as RvB, are more valuable than what can often be learned in a regular cyber class. A repeating competitor in RvB explains how RvB has been more valuable:

> Traditional classroom settings don't teach you anything you could have possibly learned in RvB. I can confidently say that everything that I learned about cybersecurity in the past year was due to activities I did outside the classroom. Going to class teaches you the theory of things, but no real hands-on experience. Not even emphasis courses go in as much detail as RvB does. (Student C, May 29, 2022)

Students become more confident in their own skills after seeing themselves put their knowledge to use. This leads to how students with hands-on experience are more equipped to deal with the technical demands of their future jobs. Hiring managers look for people with technical experience, and cybersecurity competitions are a way for students to demonstrate what they know and what they can do. This student disclosed how RvB has been a highlight in their interviews:

> RvB has been one of the many things I have talked about in interviews, and it always brings up more and more questions from interviewers. I've noticed that they would rather keep asking me questions about the business injects, what vulnerabilities I patched, and how I dealt with the red team instead of asking me to 'describe some of the courses you have taken. (Student C, May 29, 2022)

Many competitors who participate in RvB feel that they can learn much more by competing than taking a class.

6. RVB & TEACHING OF PROFESSIONAL SKILLS

RvB not only puts the technical skills of competitors to the test, but it also helps build critical professional skills including teamwork, communication, and collaboration. These are all qualities that are much more difficult to learn in a traditional classroom setting. RvB puts students in an environment that forces them to work together to become successful in the competition. Practicing teamwork begins even before the competition day arrives. Strong teams learn to strategize beforehand, setting up meetings to determine what their game plan is, deciding who owns which tasks on which computers, and sharing resources to study. Being in a team environment can therefore result in more successful students, as they are surrounded with others who are working towards the same goals.

During the competition, students are quick to learn that because there are so many tasks to juggle and factors to keep track of during the competition, cooperation is necessary to stay on top of the day's demands. For example, competitors are given multiple machines to secure, but it is up to their team to decide if one set person is going to be working on one machine or if they are going to rotate the machines around. A past competitor suggests the importance of this decision:

> An important lesson that I learned was how to delegate what each member of a team works on. I feel that this is a very good way to increase the overall efficiency of the team, as well as become more educated on the specific boxes each member is assigned. During the first couple of RvB events, many members weren't assigned a role and were just let loose on the environment. Because of this, it was harder to find who was working on what and which boxes were available to be worked on. (Student A, May 30, 2022)

Additionally, teams must decide on other factors, such as who will take on which injects and who will check the scoreboard to see which of their services were hit. Generally, teams decide on a leader, or team captain, whose job is to keep all their teammates on track and delegate tasks when necessary. However, every competitor takes on a leadership role in one way or another since teammates will help each other out when another teammate is unsure about how to approach a situation. The competitors know their own strengths and learn the strengths of their teammates, so when a problem does arise, they know who they can talk to and can communicate effectively.

The testimonials of past students stress how important having soft skills are when competing:

> Leadership and teamwork skills are essential to the competition. There are more boxes to fix than people. Also, many of the challenges can take time, while tasks pile on with the red team and injects. Competing in RvB is like trying to fix a sinking ship. If you panic and try to do things yourself, your team will fall very quickly. Having a calm, planned approach while aiding each other with our personal strengths leads to a smooth approach to problems. (Student B, May 29, 2022)

Another student comments on the large technical strain of RvB,:

> Multiple people will be working on the same machine throughout the competition, so it's important to document and communicate what is being done to each box and make it clear enough for everything to be understood. If this wasn't established properly, someone could accidentally undo all the work that had been done. (Student C, May 29, 2022)

7. THE DEVELOPERS PERSPECTIVE

Red vs. Blue is an entirely student-run and student-developed event, from its very inception to the execution of the event itself. On the development side, it all begins with a theme, then an entire environment is developed around it. Developers then configure various operating systems in virtual machines to be vulnerable. The boxes are designed to interact with each other to appropriately simulate a business environment. The vulnerabilities and possible mitigations are well documented, so that competitors can learn from after-action reports. In summation, throughout the development process, students create these boxes, integrate them with the theme, and configure them appropriately for business devices – although they are improperly configured with common cybersecurity flaws to force competitors to apply fixes.

The key skills that development bolsters are technical. When students develop the competition environment, they first must fully understand how to configure a business environment that depends on the systems that are being scored e.g., SSH, HTTP, MySQL, and FTP. This is because to design vulnerable systems in a way that is consistent and stable enough that they will be repairable, they must first be familiar with a fully functional and properly configured version of the system. One developer described the value they achieved from the program thusly, "I believe that the understanding that resulted from identifying misconfigurations and proper security practices for multiple services in RvB environments provided me with the perspectives necessary to truly understand the nuances of securing these services." (Student 2, June 1, 2022). Much of this technical experience, while condensed into a much shorter time frame than if they were to self-study, is entirely achievable on their own. However, learning to work under extreme pressure, as a team, and manage a massive project on a tight and immovable deadline are skills that do not come out of personal projects.

High-quality, functional, time management skills are sometimes missing among college students. Usually, it is a skill not learned until they are exposed to industry pressure and consequences. However, RvB completely simulates this experience in a comparably low stakes environment. Developers hone their time management skills when it comes to managing tasks assigned to them, such that development is completed on time and with enough lead time that iteration and growth can occur. One developer described their newly developed time management strategy as, "Just do the work. If you don't, you're done for." (Student 2, June 1, 2022). This developer felt this way because they were faced with the dependency that all the developers have on each other, as all the boxes are inter-reliant. A failure on one machine means potential failure on the other systems. A student described the process as "incremental work as a driver for good ideas" (Student 2, June 1, 2022). Since RvB is an ever-evolving project, the more time developers invest iterating on an idea and improving the system, the higher the product quality. Waiting until the last minute would still result in a finished product, but it would be dramatically less creative and inventive, which would sacrifice the novelty and whimsey of the competition.

8. MEASURING AND ASSESSING SKILLS IN CYBER COMPETITIONS

As aforementioned, cybersecurity is a continually growing and changing industry. Red vs. Blue provides the ability for education to adapt to change better in ways that other curricular or extracurricular program tools cannot. This is because of the hands-on nature of the competition and fact that the development team works tirelessly to keep the environment on par with industry standards and expectations completing multiple competition environments each year. Consequently, the development team stays up to date on the knowledge base and relevant current events. Moreover, as the development process is incredibly fluid and nuanced, the developers must be able to move between machines as necessary to complete the deliverable on time. One developer commented that, "I was initially terrified of being stuck in such a situation [where the answer is unknown] … I quickly learned that researching, troubleshooting, and filling in the gaps of my knowledge with Google searches was inherent to anything cyber related." (Student 2, June 1, 2022). This is the single most important skill that RvB teaches because it guarantees that the developers cannot be made obsolete in industry. Self-directed learning will continue to serve them for decades to come.

The challenge to measuring competencies in competitions for student players comes down to planning, design, and construction of the competition and its underlying challenges. Beginning with the end in mind is a key conceptual mantra for organizers. This concept is critical to constructing learning outcomes and crafting the challenges in a building block style regardless of the modality of the competition. (I.E. CTF, RED V. BLUE, or Forensic Scenario). Careful consideration needs to go into the design in the form of challenge difficulty, skill area, and overall cybersecurity discipline. Couple this with the overall "story" of the competition, and these compounding factors contribute to the time it takes to develop and produce a well-rounded, meaningful, and enjoyable experience for the student which includes the ability to measure their skill level. The final key component for an exercise such as this would be a measurement scale based on challenges with an overall matrix which would gauge a student's competency in one or more areas. This is supported by Pusey, Gondree, and Peterson (2016), in which they found that learning outcomes are taking a back seat to making competitions fun and meaningful for diverse groups of students across many modalities nationwide.

Other preliminary work on this topic has been characterized by Straub (2020), where quantitative measurements were taken from a broad population of students in the National Cyber League (NCL) to measure skills and how they gain those skills compared to the classroom. However, the NCL competition is static and easy to measure, but does not provide intangible skill measurements such as troubleshooting or dynamic incident response.

To date the expectation is that students learn cybersecurity fundamentals using traditional pedagogies in the classroom. Classroom learning then serves as a series of diagnostic assessments in which students gain insight into their strengths and weaknesses and plan their gameplay accordingly.

Cybersecurity competitions are often used as a forum to exercise the knowledge and skills attained in the classroom. This allows students to test and revise their skills as they adapt and learn through a series of competition experiences allowing competitions to serve as a formative assessment to refine knowledge, skills and abilities.

However, the next evolution of RvB is to craft the measurement of skills into a summative assessment focused on competition outcomes. The competition will not only serve as a summative assessment for fundamental skills but also a diagnostic assessment showing that students attained a degree of mastery with respect to knowledge, skills and abilities related to applied cybersecurity outcomes and are ready to engage in higher-level learning with respect to the application of cybersecurity practices in organizations. The need to assess learning in competitions stresses the importance of outcomes-based competitions mentioned in Pusey, Gondree, and Peterson (2016) and the need to design competition content to support the assessment of outcomes.

9. CONCLUSIONS & OPPORTUNITIES FOR FURTHER RESEARCH

This case clearly outlines the advantages of using competitions to gauge student's strengths learned in the classroom. Furthermore, it supports the link between curricula and practice, and the need for experiential learning. This case also gives educators different views to hone and improve upon the pedagogy of their program. This comes in the form of different instruction and assessment modalities and provides the opportunity to measure skills in different ways and track them over years or even careers.

Maintaining an inventory of competencies, such as the NICE Framework, and relating these to courses may offer enhanced ways to effectively measure students' knowledge, skills and abilities and lead to more enhanced capabilities in offering credit for courses in a curriculum set. Effective cybersecurity pathways from middle school to the end of a career necessitate the ability to understand skills and develop/award credit in a manner that is based on the ability to assess skills and not rely upon a sequence of courses. The assessment of experiential learning offers important contributions to assessing learning in a manner that leverages automation and limits time and expense in the assessment process.

Given these opportunities, we propose that further research on dynamic competitions, such as National and Regional Collegiate Cyber Defense Competitions, Dynamic CTFs, Red vs Blue competitions and others. A common leaderboard across these competitions will offer an ability to better understand how to assess learning and enhance the definition and measurement of these dynamic skills that are so critical to aspects of cybersecurity such as incident response, penetration testing, and real-time forensics.

There were several suggestions to leverage a cloud hosting platform for Red vs. Blue moving forward. A cloud hosting platform such as AWS, Azure, or Google would make the competition more accessible and efforts to develop cyber competitions that address both technical and financial constraints would open such competitions to everyone everywhere and would be an important contribution to the discipline.

10. REFERENCES

Educause (2022, June9). Competency-Based Education (CBE) Retrieved from: https://library.educause.edu/topics/teaching-and-learning/competency-based-education-cbe#:~:text=The%20competency%2Dbased%20education%20(CBE,to%20more%20efficient%20student%20outcomes.

Davenport, Thomas H., and M. Lynne Markus. "Rigor Vs. Relevance Revisited: Response to Benbasat and Zmud." *MIS quarterly* 23.1 (1999): 19–23. Web.

Petersen, Santos, Smith, Wetzel & Witte (2020). Workforce Framework for Cybersecurity NICE Framework), Retrieved from: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181r1.pdf.

Pike, R. (2022, April). Innovation in Education through Holistic Assessment. Paper presented at 50th Annual Meeting of Western Decision Sciences Institute: WDSI 2022, http://wdsinet.org/Annual_Meetings/2022_Proceedings/papers.php.

Pusey, P., Gondree, M., & Peterson, Z. (2016). The outcomes of cybersecurity competitions and implications for underrepresented populations. IEEE Security & Privacy, 14(6), 90-95.

Straub, J. (2020, June), Assessment of Cybersecurity Competition Teams as Experiential Education Exercises. Paper presented at 2020 ASEE Virtual Annual Conference Content Access, Virtual On line . 10.18260/1-2—34187

Appendix A
Environment Details for Ubuntu 14.04 Machine

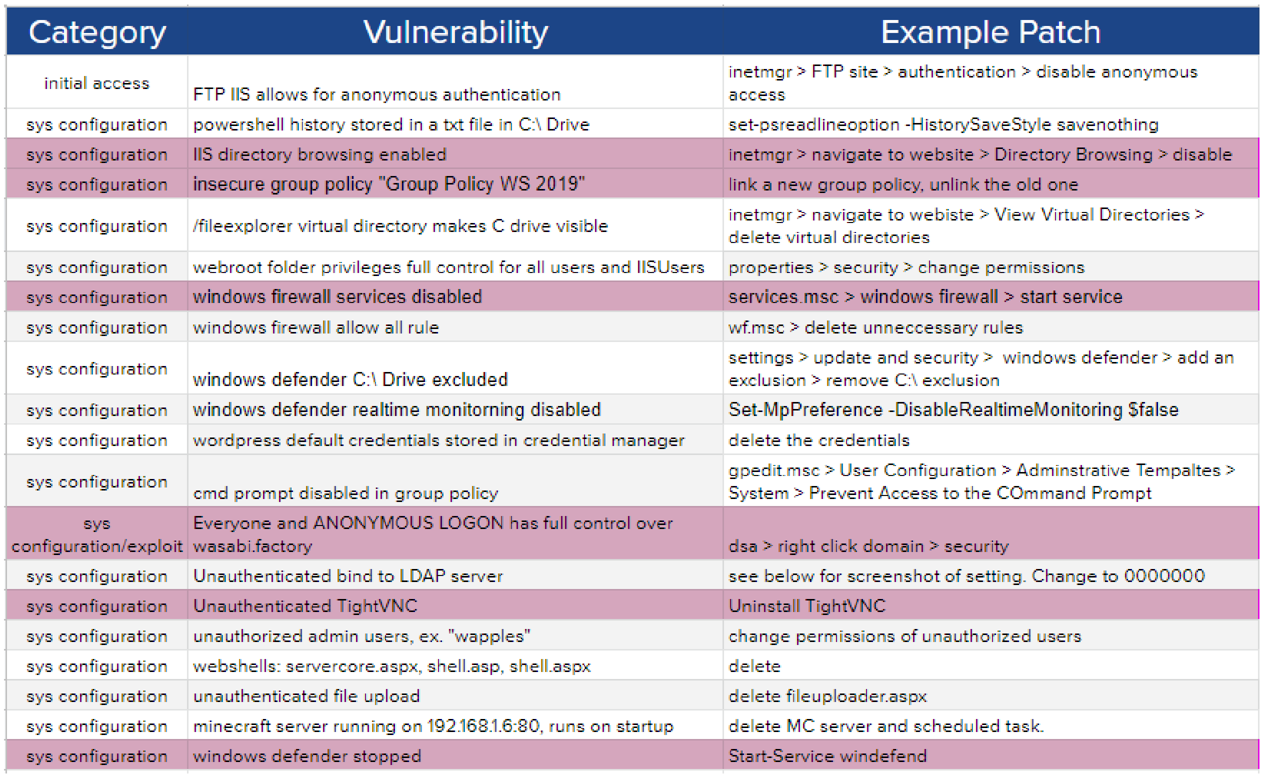
This appendix shows a table from a Red vs. Blue competition on the development side. The developer will document the security misconfigurations they add to a computer, similar to the table that is shown below for a Linux machine. The "Category" column categorizes the vulnerability as a service, which means the setting relates to a scored service like DNS or FTP, or as a system, which regards the computer itself. The "Vulnerability" column is to explain what misconfiguration was added, and the "Example Patch" column gives a possible solution to the insecure setting. Teams are awarded 100 points for each of the misconfigurations they fix. Additionally, the more of these vulnerabilities competitors fix, the harder it is for the red team to infiltrate their systems and take down their scored services, resulting in a higher number of points for service uptime. Competitors do not have access to this table during the competition.

Category	Vulnerability	Example Patch
Service	root user can login to database with default creds from any host dev user has all privs over all databases	implement proper UAC
Service	blobmaster user has reused credentials and all privileges over all databases and can login remotely	delete the user
Service	dropbear on port 22 and sshd on port 2222	make sure to edit correct ssh conf
System	www-data has bash	remove it from /etc/passwd
System	www-data in sudo group and can run sudo w/o password	remove it from /etc/passwd
System	www-data has password admin	change password or lock accounts
Service	webshells in webroot webshell.php shell.php	remove them or disable php functions
System	root password resuse	change password or lock accounts
System	/etc/passwd and /etc/shadow are 777	edit permissions
System	entire /root directory is 777	
System	users wasabi, widesabi, widestsabi, wasmolbi, and walolbi are all allowed	leave them
System	users blobmaster, factoryadmin, factoryworker, factorymanager, wasabli, and waspraisebe are all malicious users	delete them
System	all users have same creds	change creds
System	suid binaries on /bin/cp /usr/bin/find /bin/sh	remove them
System	user darksabi exists and is in root group with password darksabi and UID 420	remove it
System	run binary	remove it
Service	PermitRootLogin yes	PermitRootLogin no
Service	nginx running on port 5432	stop the service
Service	webshell.php on port 5432	
System	users walolbi, wasabi, widesabi, widestsabi, wasmolbi, wasadmin have root privileges with NOPASSWD	remove the users from /etc/sudoers (visudo)
System	PAM allows no authentication needed	pam-auth-update --force

Appendix B

Environment Details for Windows Server 2016

This appendix shows the misconfigurations applied on a Windows machine for a Red vs. Blue competition.

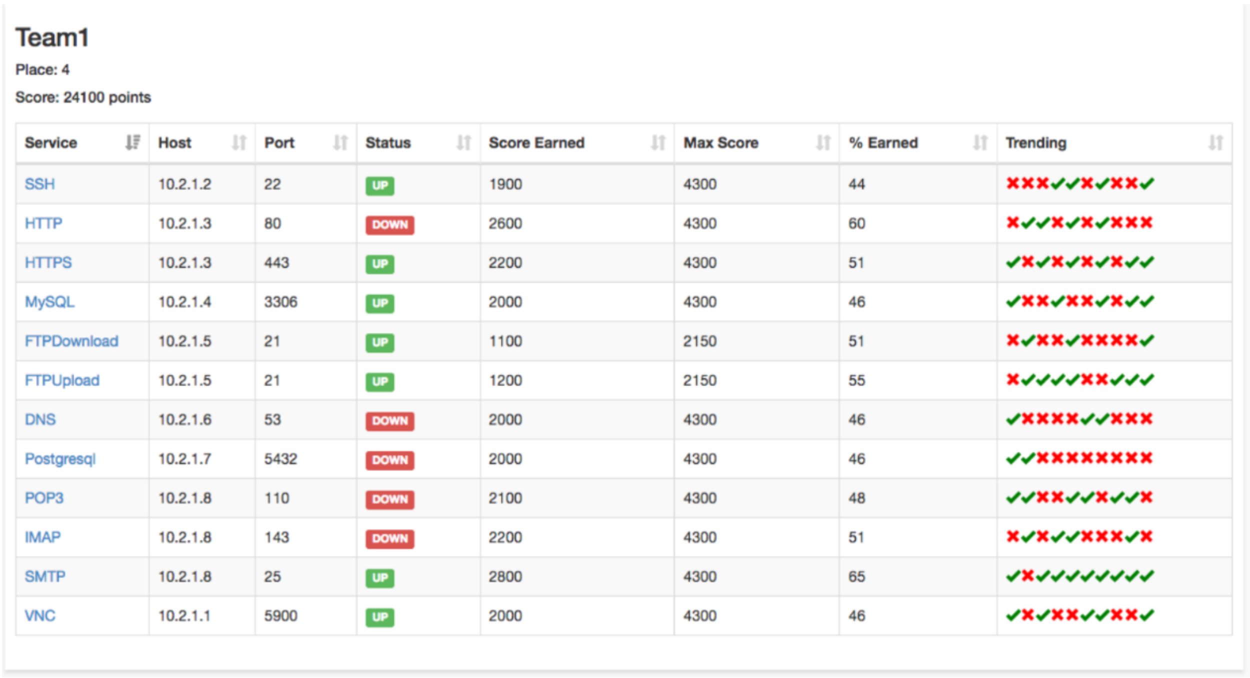


Category	Vulnerability	Example Patch
initial access	FTP IIS allows for anonymous authentication	inetmgr > FTP site > authentication > disable anonymous access
sys configuration	powershell history stored in a txt file in C:\ Drive	set-psreadlineoption -HistorySaveStyle savenothing
sys configuration	IIS directory browsing enabled	inetmgr > navigate to website > Directory Browsing > disable
sys configuration	insecure group policy "Group Policy WS 2019"	link a new group policy, unlink the old one
sys configuration	/fileexplorer virtual directory makes C drive visible	inetmgr > navigate to webiste > View Virtual Directories > delete virtual directories
sys configuration	webroot folder privileges full control for all users and IISUsers	properties > security > change permissions
sys configuration	windows firewall services disabled	services.msc > windows firewall > start service
sys configuration	windows firewall allow all rule	wf.msc > delete unneccessary rules
sys configuration	windows defender C:\ Drive excluded	settings > update and security > windows defender > add an exclusion > remove C:\ exclusion
sys configuration	windows defender realtime monitorning disabled	Set-MpPreference -DisableRealtimeMonitoring $false
sys configuration	wordpress default credentials stored in credential manager	delete the credentials
sys configuration	cmd prompt disabled in group policy	gpedit.msc > User Configuration > Adminstrative Tempaltes > System > Prevent Access to the COmmand Prompt
sys configuration/exploit	Everyone and ANONYMOUS LOGON has full control over wasabi.factory	dsa > right click domain > security
sys configuration	Unauthenticated bind to LDAP server	see below for screenshot of setting. Change to 0000000
sys configuration	Unauthenticated TightVNC	Uninstall TightVNC
sys configuration	unauthorized admin users, ex. "wapples"	change permissions of unauthorized users
sys configuration	webshells: servercore.aspx, shell.asp, shell.aspx	delete
sys configuration	unauthenticated file upload	delete fileuploader.aspx
sys configuration	minecraft server running on 192.168.1.6:80, runs on startup	delete MC server and scheduled task.
sys configuration	windows defender stopped	Start-Service windefend

Appendix C
Table of Scored Services and their Status

This appendix shows the list of scored services that each team has to secure during the competition. The scoring engine checks every three minutes whether the service is up or down. If the service is up, the team is awarded 100 points. If it is not, then they get 0 points. For example, the scoring engine sees that HTTP is up, so the team gets 100 points. The next three minutes, the scoring engine sees that HTTP is down, so the team gets zero points. Then the next three minutes, the scoring engine sees that HTTP is back up, so the team gets 100 points and now has 200 points in total.

The "Status" column shows if the service is currently up or down, and the "Trending" column tracks every three minutes whether the service was up or down. For example, in the table below, SSH is currently up, so the last character in the "Trending" category for SSH is a check mark. The red cross to the left of the check mark tells us that SSH was down 3 minutes ago.



Team1

Place: 4

Score: 24100 points

Service	Host	Port	Status	Score Earned	Max Score	% Earned	Trending
SSH	10.2.1.2	22	UP	1900	4300	44	✗✗✗✓✓✗✓✗✗✓
HTTP	10.2.1.3	80	DOWN	2600	4300	60	✗✓✓✗✓✗✓✗✗✗
HTTPS	10.2.1.3	443	UP	2200	4300	51	✓✗✓✗✓✗✓✗✓✓
MySQL	10.2.1.4	3306	UP	2000	4300	46	✓✗✗✓✗✗✓✗✓✓
FTPDownload	10.2.1.5	21	UP	1100	2150	51	✗✓✗✗✓✗✗✗✗✓
FTPUpload	10.2.1.5	21	UP	1200	2150	55	✗✓✓✓✓✗✗✓✓✓
DNS	10.2.1.6	53	DOWN	2000	4300	46	✓✗✗✗✗✓✓✗✗✗
Postgresql	10.2.1.7	5432	DOWN	2000	4300	46	✓✓✗✗✗✗✗✗✗✗
POP3	10.2.1.8	110	DOWN	2100	4300	48	✓✓✗✗✓✓✗✓✓✗
IMAP	10.2.1.8	143	DOWN	2200	4300	51	✗✓✗✓✓✗✗✗✓✗
SMTP	10.2.1.8	25	UP	2800	4300	65	✓✗✓✓✓✓✓✓✓✓
VNC	10.2.1.1	5900	UP	2000	4300	46	✓✗✓✗✗✓✓✗✗✓

IoT Security Vulnerabilities Analysis by Reverse Engineering: A Face-recognition IoT Application-based Lab Exercises

Sam Elfrink
selfrink3s@semo.edu

Mario Alberto Garcia
mgarcia@semo.edu

Xuesong Zhang
xzhang@semo.edu

Zhouzhou Li
zli2@semo.edu

Department of Computer Science
Southeast Missouri State University
Cape Girardeau, MO, US

Qiuyu Han
2003138@hlju.edu.cn
Heilongjiang University
Harbin, Heilongjiang, CN

Abstract

The rapid growth of the Internet users and the proliferation of IoT devices in recent years has created a significant need for vulnerability detection and mitigation in these devices and their applications. Exposing computer science and cybersecurity students to these skills can help them strengthen their competencies in the industry. One approach that can be used to achieve this objective is reverse engineering, which involves gaining a thorough understanding of the relationship between the individual components of an IoT application. This paper presents lab exercises that teach students the concepts and practical techniques of reverse engineering for the purpose of detecting and mitigating vulnerabilities in IoT devices. The lab exercises are based on a real facial recognition web application hosted on a small IoT device, and they use both manual exploration and automated tools to provide students with a systematic and comprehensive understanding of reverse engineering. These well-designed, hands-on labs can meet the practical needs of cybersecurity education and inspire heuristic learning on difficult cybersecurity topics such as reverse engineering.

Keywords: Cybersecurity, Reverse Engineering, Internet of Things, Artificial Intelligence, Face Recognition, Re-engineering.

1. INTRODUCTION

In recent years, the number of users with access to the Internet has increased dramatically. An estimated one million new users join the Internet every day. And six billion people are projected to be connected to the Internet by the end of 2022, a 1 billion increase from the 5 billion people in 2020 (Morgan, 2020). Internet of Things (IoT) are also becoming prevalent. In 2015, regarding the number of connected devices by 2020, the low estimate was 18 billion, the most bullish forecast stated 50 billion devices (Lueth, 2015). Reputable companies such as Cisco, Intel, IDC, Gartner, etc., provide similar and higher estimates recently.

This substantial growth of Internet users and Internet capable devices has increased the risk of cyberattacks. Cybercrime costs are estimated over six trillion USD globally in 2021, and that number is expected to grow 15 percent each year, reaching 10.5 trillion USD in 2025 (Morgan, 2020). Because of this increase in both users and attacks, the need for security vulnerability analysis has never been more vital, especially for web applications. According to a global threat analysis report produced by Radware, the average number of blocked malicious web application requests increased by 88% from 2020 to 2021, with 75 percent of those attacks performed via broken access control and injection (Radware, 2022).

The massive increase in Internet access has also led to large increases in the quantity and diversity of Internet capable devices. The term "Internet of Things" (IoT) has been used to describe this phenomenon. Shwartz, Mathov, Bohadana, Elovici, & Oren (2018) define IoT as "a network of smart electronic devices with Internet connectivity." Nowadays a home containing a variation of IoT devices (also known as smart devices) that provide a variety of functions to the consumer is common. The introduction of these devices also introduces new security concerns and vulnerabilities.

As cybercrime and security vulnerabilities increase, Cybersecurity and Computer Science professionals need the skills to mitigate threats to software applications. They must be proactive in detecting and patching vulnerabilities before malicious actors can exploit the system.

As students graduate and enter the industry, vulnerability detection and mitigation skills will be invaluable to corporations, as patching vulnerabilities before they are exploited can save corporations millions of dollars as well as their reputation as a reliable company. While some software engineers can be tasked with creating new systems, many are tasked with maintaining, testing, reusing, and integrating existing systems (Canfora & Penta, 2007). New hires probably will be asked to work on systems that are already established, thus the ability to adapt and understand code that is not their own and test it for vulnerabilities is an important skill to possess. In order to accomplish this, they must be able to identify and comprehend the different components in a system and understand the relationship between them. However, most curricula only focus on designing software applications from scratch without regarding the common situation in the industry.

There are a variety of tools and methods that can be used for software vulnerability detection, such as automated dynamic scanning (tool-based), static analysis (tool-based), and Reverse Engineering. While tools can be useful, their ease of use and quick results can lead to a lack of a holistic understanding of how a piece of software operates. Thus, courses that only utilize tools can run the risk of focusing on teaching how to use the tools instead of how to understand the underlying security and software principles.

Reverse Engineering can be defined as "the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction" (Chikofsky & Cross, 1990). It has a vast number of goals, such as managing system complexity, creating alternate views, recovering lost understanding of a system, and existing software maintenance. It can also be used to audit the security and vulnerabilities of an application (this is a unique step in Secure Software Development, so it is not equivalent to software maintenance).

From an offensive perspective, in many cases attackers or pen-tester have limited or incomplete access to the application's entire framework. For instance, an attacker or pen-tester may be able to view the front-end source code of an application but will not easily be aware of the backend code. However, through the process of Reverse Engineering, the attacker or pen-tester can gain an understanding of how the entire application operates.

There are three basic approaches on how to Reverse Engineer a system. The first is white box analysis, which is a static approach that does not require running the application (Ali, 2005). This approach is often used when all the systems components are easily available. The second approach is black box analysis, which uses inputs to check the behavior of the application/program. This approach is used when the user has less access to the backend of the system and must use inputs to determine the hidden components of the system. Finally, gray-box analysis utilizes both white box and black box in unison to evaluate the system, using each approach on various parts of the system.

Real-world software systems are in constant need of modification and improvement due to new user requirements, modified business models, and changing legislation (Canfora, Penta, & Cerulo, 2011). When applied to IoT-based web application security, Reverse Engineering can be used to develop a detailed understanding of how the application is constructed and how it operates. An IoT device hosting a web application can be broken down into 5 simplified layers: the physical, link, network, transport, and web app layers (Fig. 1). To have a complete understanding of a system's vulnerabilities, each layer of the application must be considered. Each layer has different attack surfaces and capabilities, and together their specific vulnerabilities can be combined to form a complete picture of the application's potential security concerns.

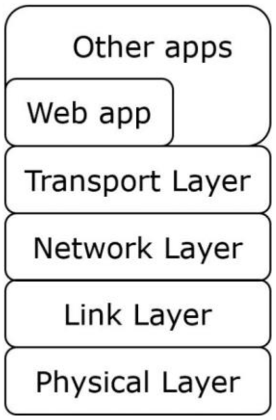


Fig. 1. Layer Diagram of Web Application

There are several challenges to effectively creating a teaching curriculum for Reverse Engineering an IoT-based web application. First, the application and device must be at an appropriate level of complexity to match the abilities and knowledge of the students. Because Reverse Engineering often requires making inferences about the structure of an application and complicated analysis of the system, many applications are too complex to suit a curriculum for beginners. In addition, the IoT device used in the curriculum must be affordable and easily accessible so that faculty members and students can use them effectively. Lastly, the testing environment must be easily assembled so that entry into the exercises is not too difficult.

The purpose of this paper is to provide a curriculum that students can utilize to gain hands-on experience with Reverse Engineering on a real-world IoT-based web application that integrates a face-recognition Artificial Intelligence (AI) model. This is accomplished by providing general background information on the topics discussed to promote a better understanding, followed by 5 hands-on labs that identify a variety of vulnerabilities of a facial recognition web application hosted by the ESP32-CAM IoT device. The vulnerabilities encompass multiple layers of the application and are discovered and mitigated through the process of Reverse Engineering and re-engineering. Thus, giving students the opportunity to practice these vital industry skills on a real application.

The remainder of this paper is organized as follows. In the 'Literature Review' section, we review the current courseware or labs designed for teaching Reverse Engineering. 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 'Vulnerabilities Found by Reverse Engineering' section, the specific vulnerabilities identified in the course will be discussed followed by an outline of the step-by-step processes taken to assess each vulnerability that was identified. A summary of what areas can be improved and expanded is provided in the 'Future Work' section, and the 'Conclusion' section provides a general summary of the paper's contents and findings.

2. LITERATURE REVIEW

The concept of Reverse Engineering system began to get traction in the beginning of the 1990s and a variety of methods and approaches have been proposed as systems have shifted to web-based user interfaces (Müller, Jahnke, Smith, Storey, Tilley, & Wong, ,2000). Early papers provide a helpful summary of the history of Reverse Engineering and the different methods and approaches involved. However, they only provide conceptual information and do not provide any sort of tutorial to learn Reverse Engineering.

Shwartz et al. (2018) provides a detailed methodology and tutorial for Reverse Engineering and security vulnerability evaluation of what are considered "full stack OS devices" that contain a modern operating system such as Linux. These devices divide execution into kernel mode and user mode. However, this method does not address partial stack OS devices that have specially designed real-time operating systems, or devices that execute compiled instructions directly with no operating system. It provides a narrow focus of Reverse Engineering from a physical access perspective, dealing with device's firmware memory images rather than the applications that are hosted by the devices as well.

Ali (2015) gives a detailed argument for the importance of Reverse Engineering for undergraduate software engineering students. While Reverse Engineering can be difficult and time consuming, the process itself is very informative and gives students the ability to understand a system more rapidly and effectively. In addition, it points out that traditionally students are often given the task of designing and implementing new systems in the classroom. While this is a valuable skill, it is also crucial that they be able to understand code written by other programmers and improve upon existing software. Often in industry, companies already have legacy software and are interested in improving it rather than creating an entirely new product. Thus, Reverse Engineering skills are tremendously important. However, the paper merely emphasizes the importance of Reverse Engineering, and does not provide tutorials for students to use to achieve a better understanding of these concepts.

Bellettini et al. outlines the usage of an automatic tool for creating UML (Unified Modeling Language) models for web applications called *WebUml* (Bellettini, Marchetto, & Trentini, 2004.) It describes how the tool can be used for Reverse Engineering by utilizing the rich information that the extracted UML models provide. It provides the source code for a simple XML node construction application that can be used to test the effectiveness of *WebUml*. Thus, while this provides a simple use case for learning to use the tool for Reverse Engineering, the scope of the paper is limited to producing UML diagrams.

Taylor & Collberg (2016) also proposed a tool for teaching Reverse Engineering. It notes that the current instruction materials regarding Reverse Engineering code have a lack of easy tools for students to use. It also notes that students take a substantial amount of time to configure and set up an environment that has the tools necessary to practice Reverse Engineering. The solution provided is an automated code obfuscator tool called Tigress that is combined with a web application. The application allows the instructor to generate custom target programs, which can then be obfuscated with a set level of complexity. The application then creates virtual machines that have the necessary Reverse Engineering tools selected by the instructor. The paper provides two example C programs that can be used, one that checks the current time and prints the variable, and an additional one that adds a password check to the first. A group of students were given this exercise and then polled at the end. While most students reported that the difficulty of solving the challenge was hard, a good percentage also indicated that it was easy, indicating that the student's previous experience could be a large factor. In addition, most students were able to finish the assignment in a reasonable time, indicating that the proposed application was effective in creating a comfortable environment to practice Reverse Engineering skills.

This paper provides an in-depth framework for students to follow to get hands on experience of Reverse Engineering through the purposes of vulnerability detection and mitigation that to our best knowledge has not been provided by existing works. The application that is examined is hosted by a IoT device with a compact operating system, and the vulnerabilities addressed are not just at the physical layers but encompass multiple layers and incorporate analysis of the applications code and the devices firmware. While automated tools can be helpful aids in Reverse Engineering, relying on them too much keeps students from understanding the process in depth. This paper provides a mixture of automated tools and manual exploration to provide students with a more systematic approach to Reverse Engineering.

3. BACKGROUND

As technology has advanced in recent years, the availability and usefulness of IoT devices has increased dramatically. While smart devices such as Amazon Alexa or smart appliances are popular, IoT devices can also be used for facial recognition purposes as well. Facial recognition is of particular interest because it can be used for a variety of purposes. For instance, IoT devices with facial recognition capabilities can be used for security purposes where authorized individual's faces are scanned into the system and all other faces considered hostile. In addition, it could be used for recreational home use, such as identify what individuals are in a home to adjust the experience accordingly.

The ESP32-CAM (Wikipedia contributors, 2022) IoT device (Fig. 2) integrates a member of the most popular IoT SoCs, the ESP series (Li, Ren, Chou, Liu, & McAllister, 2022), which offers a good introduction to the concept of IoT device-based face recognition by integrating an AI Model with the IoT device. The ESP32-CAM interfaces with the computer via micro-USB connection. It can be purchased for $5 to $10 per device depending on the vendor. The inexpensive cost of the device and the ease of deployment makes it a suitable option for introductory exploration into the concepts of IoT device-based face recognition.

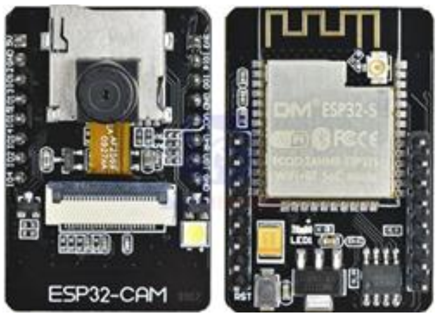


Fig. 2. ESP32-CAM IoT Device

The device contains a dual-core 32-bit ESP32-S CPU, 520 KB of SRAM, and 4M of PSRAM (Fig. 2). Furthermore, it has an 802.11b/g/n Wi-F0 BT/BLE System on Chip (SoC) module and a camera that supports OV2640 and OV7670, giving it both Wi-Fi and image capability.

There are a variety of pre-prepared programs that can be easily implemented on the device, including a face recognition web platform (Li, Chou, & McAllister, 2022). The web application provides the user with the ability to register faces to the application and then recognize registered users when they come into the view of the camera. The device sends a continuous stream of video footage back to the application, acting as a live feed security camera as well.

The application also allows the user to modify a variety of settings for the camera display, such as saturation, brightness, and the frame size of the stream. The program is also designed to log the facial details of recognized individuals. The logs are transported via USB connection and are displayed in the serial monitor, recording the ID of individuals that have come into the view of the device's camera.

The benefits of the web platform for facial recognition are apparent, as the user can easily access the application on any Internet capable device and can access it from any location if they have access to the network and knowledge of the web application's IP address. In addition to being deployed on an existing local network, the device's code can be easily modified to create a unique Wi-Fi AP that has its own SSID and password credentials.

However, the ability for the device to connect the facial recognition and streaming capabilities to a web application presents some unique challenges for security as well. This paper aims to address these security vulnerabilities and demonstrate how students can learn essential security principles by evaluating and mitigating the vulnerabilities that are exposed. Although it may appear simple because of the ease of deployment, the facial recognition web application offers significant depth. Fig. 3 is a model created by Microsoft's STRIDE application that shows the data flow and trust boundaries during the device's operation.

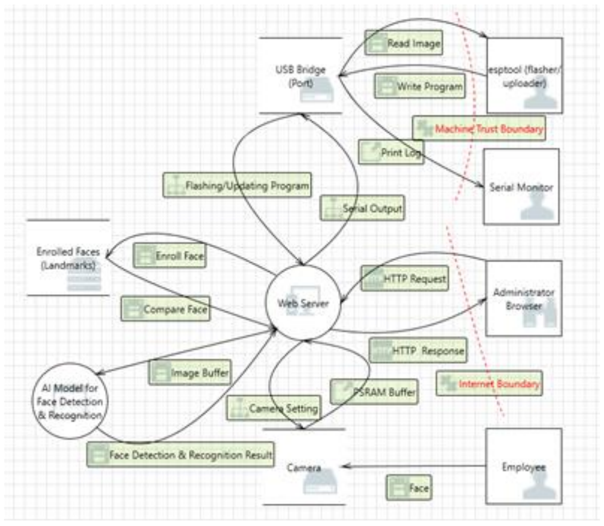
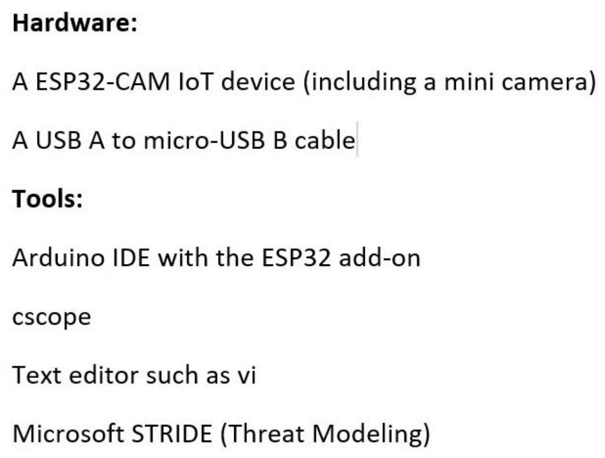


Fig. 3. STRIDE (Microsoft) Data Flow Model for the Face-Recognition Web Application

In addition to the devices, there are a variety of common items and tools that will need to be assembled to reproduce the methods and concepts introduced in this paper. Fig. 4 outlines required hardware and tools in detail. ESP32-CAM is the target IoT device hosting a face-recognition Web App. Its camera can take photo of the user and send the user's face image to an AI model for analysis. Then the AI model can conduct face detection and face recognition (if the user registered before). The USB cable is used to connect the IoT device to the computer, where the Arduino IDE (Fezari & Al, 2018) is installed to prepare, compile, and upload the face-recognition application to the device. Also, the computer is the output peripheral of the device, where the device log can be printed out. cscope is the tool used to build the code database including the App's source code (4 files) and thousands library files. cscope must work with a text editor to fully function. Finally, the Micro STRIDE tool will be used to do threat modeling (Fig. 3).


Hardware:

A ESP32-CAM IoT device (including a mini camera)

A USB A to micro-USB B cable

Tools:

Arduino IDE with the ESP32 add-on

cscope

Text editor such as vi

Microsoft STRIDE (Threat Modeling)

Fig. 4. Curriculum Hardware and Software Requirements

4. VULNERABILITIES FOUND BY REVERSE ENGINEERING

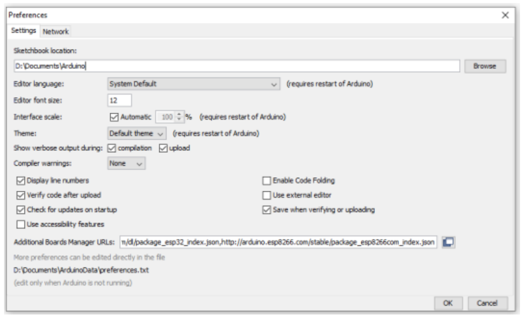
Through Reverse Engineering, five categories of vulnerabilities were found hidden in ESP32-CAM IoT devices when they are programmed with the example face recognition application. To set up a proper lab environment to host the web application, follow the instructions detailed in the attached "Lab Environment Setup" document. The following sections provide detailed steps for the vulnerabilities that were detected and mitigated with Reverse Engineering and re-engineering.

Figure out the hard-coded credential

This lab demonstrates how physical access to the UART/USB port on the IoT device can be exploited by an attacker to gain access to the hardcoded credentials. Students will need to Reverse Engineer the application upload process (from a development host to the IoT device) to figure out the tools for them to fetch the binary image from the IoT device (in reverse direction). Then, the students will need to Reverse Engineer the binary image to figure out where the hardcoded credential was saved. The Arduino IDE is simply a wrapper that provides a graphical user interface for ease of use, but at the core it calls a code compiler and uploader to upload the binary executable to the device. The attacker can upload code to a similar device with a verbose log output to understand what tools the program is using. Through Reverse Engineering, the program tools used for flashing to the device can be discovered. Then, further exploration can be done to determine the parameters of the tool to discover how it can read the flash as well. Then with the correct parameters for the flash reader, contents of the flash can be copied to a file and examined for hard-coded credentials and sensitive information.

The SSID and password are hard coded into the application's code. If physical access to the device is gained, these sensitive credentials could be compromised through analysis of the binary code. We know that the binary code was cross compiled in the Arduino IDE and flashed to the device via USB port. The same uploading tool could be used to extract the binary code from the device, thus reversing the direction and exploiting the hard-coded credential vulnerability. Here are the steps:

1. In the Arduino IDE, navigate to File -> Preferences, and make sure that verbose output is checked for compilation and uploading (Fig. 5).


Preferences

Settings | Network

Sketchbook location: D:\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ✓ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ✓ compilation ✓ upload

Compiler warnings: None

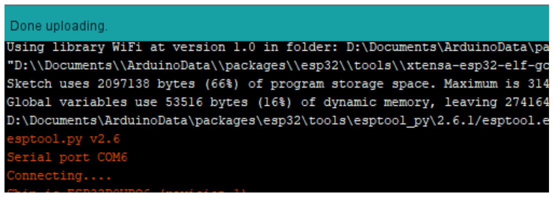
✓ Display line numbers
✓ Verify code after upload
✓ Check for updates on startup
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
✓ Save when verifying or uploading

Additional Boards Manager URLs: n/dl/package_esp32_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file
D:\Documents\ArduinoData\preferences.txt
(edit only when Arduino is not running)

Fig. 5. Arduino IDE Preference Settings

2. Compile and upload the program, look at the verbose log to see what underlying tools are called to upload the compiled program to the IoT device and where they are installed. From the output, we can see that "D:\Documents\ArduinoData\packages\esp32\tools\esptool_py\2.6.1/esptool.exe" is the uploading tool (Fig. 6).



```
Done uploading.
Using library WiFi at version 1.0 in folder: D:\Documents\ArduinoData\pa
"D:\\Documents\\ArduinoData\\packages\\esp32\\tools\\xtensa-esp32-elf-gc
Sketch uses 2097138 bytes (66%) of program storage space. Maximum is 314
Global variables use 53516 bytes (16%) of dynamic memory, leaving 274164
D:\Documents\ArduinoData\packages\esp32\tools\esptool_py\2.6.1/esptool.e
esptool.py v2.6
Serial port COM6
Connecting....
```

Fig. 6. Verbose Output for Uploading

3. Locate "esptool.exe" and the underlying esptool.py (Fig. 7).

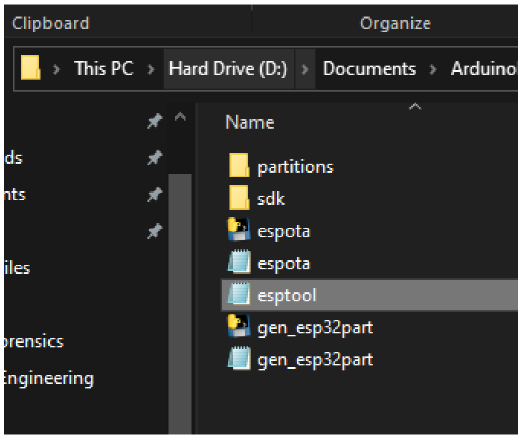
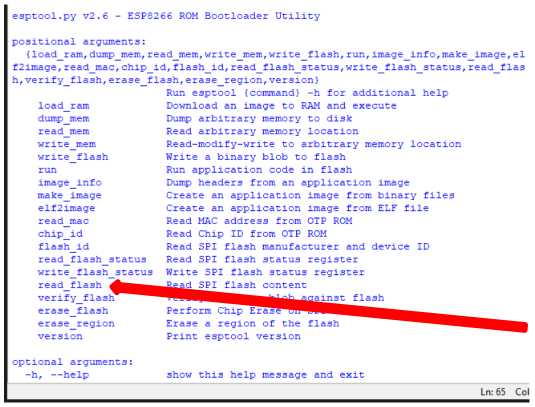


Fig. 7. Esptool.py File Location

4. Run the esptool.py to see the options and get more information. The write_flash option was used to upload the compiled program to the IoT device. To download the active image from the device (in a reverse direction), note the "read_flash" parameter (Fig. 8).

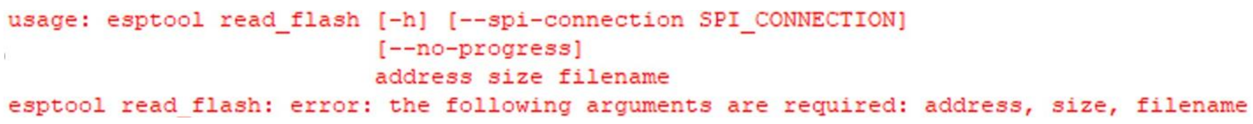


```
esptool.py v2.6 - ESP8266 ROM Bootloader Utility

positional arguments:
  {load_ram,dump_mem,read_mem,write_mem,write_flash,run,image_info,make_image,el
f2image,read_mac,chip_id,flash_id,read_flash_status,write_flash_status,read_flas
h,verify_flash,erase_flash,erase_region,version}
                        Run esptool {command} -h for additional help
    load_ram            Download an image to RAM and execute
    dump_mem            Dump arbitrary memory to disk
    read_mem            Read arbitrary memory location
    write_mem           Read-modify-write to arbitrary memory location
    write_flash         Write a binary blob to flash
    run                 Run application code in flash
    image_info          Dump headers from an application image
    make_image          Create an application image from binary files
    elf2image           Create an application image from ELF file
    read_mac            Read MAC address from OTP ROM
    chip_id             Read Chip ID from OTP ROM
    flash_id            Read SPI flash manufacturer and device ID
    read_flash_status   Read SPI flash status register
    write_flash_status  Write SPI flash status register
    read_flash          Read SPI flash content
    verify_flash        [illegible] blob against flash
    erase_flash         Perform Chip Erase [illegible]
    erase_region        Erase a region of the flash
    version             Print esptool version

optional arguments:
  -h, --help            show this help message and exit
```

Fig. 8. Esptool.py Parameters

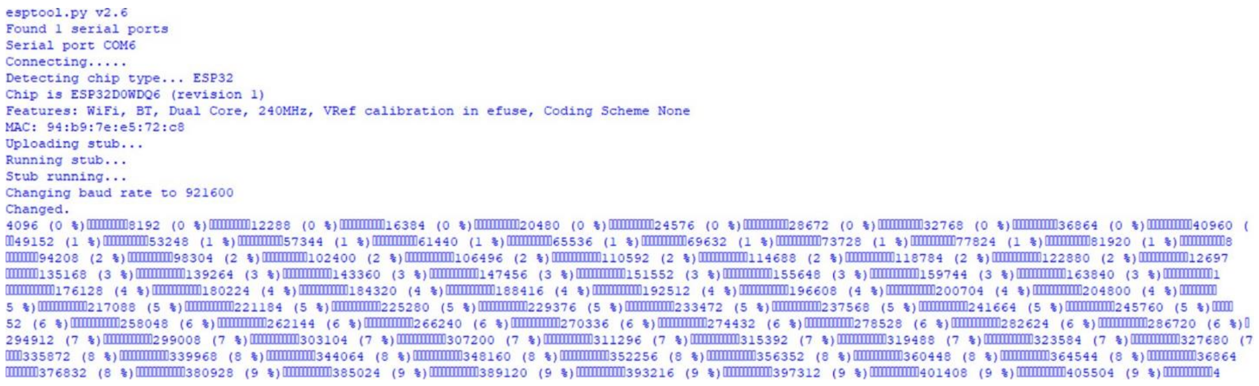
5. Run the "esptool.py" (Fig. 9) again with the "read_flash" parameter to see the sub-options. 'address', 'size', and 'filename' are mandatory sub-options.

6. Enter in the parameters "-b 921600 read_flash 0 0x400000 targetImg". The '-b' option will set the USB communication's baud rate to 921600, which is the maximum speed. '0' is the starting address of the image download. '0x400000' is the image size – ESP32-CAM has a 4MB flash. And the "targetImg" option specifies the image output filename (Fig. 10).



```
usage: esptool read_flash [-h] [--spi-connection SPI_CONNECTION]
                          [--no-progress]
                          address size filename
esptool read_flash: error: the following arguments are required: address, size, filename
```

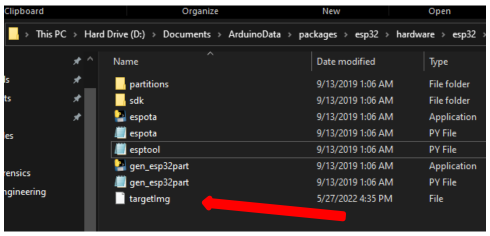
Fig. 9. Esptool.py read_flash Parameters



```
esptool.py v2.6
Found 1 serial ports
Serial port COM6
Connecting.....
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
MAC: 94:b9:7e:e5:72:c8
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
4096 (0 %)▯▯▯▯▯▯▯▯8192 (0 %)▯▯▯▯▯▯▯▯12288 (0 %)▯▯▯▯▯▯▯▯16384 (0 %)▯▯▯▯▯▯▯▯20480 (0 %)▯▯▯▯▯▯▯▯24576 (0 %)▯▯▯▯▯▯▯▯28672 (0 %)▯▯▯▯▯▯▯▯32768 (0 %)▯▯▯▯▯▯▯▯36864 (0 %)▯▯▯▯▯▯▯▯40960 (
▯49152 (1 %)▯▯▯▯▯▯▯▯53248 (1 %)▯▯▯▯▯▯▯▯57344 (1 %)▯▯▯▯▯▯▯▯61440 (1 %)▯▯▯▯▯▯▯▯65536 (1 %)▯▯▯▯▯▯▯▯69632 (1 %)▯▯▯▯▯▯▯▯73728 (1 %)▯▯▯▯▯▯▯▯77824 (1 %)▯▯▯▯▯▯▯▯81920 (1 %)▯▯▯▯▯▯▯▯8
▯▯▯▯▯▯94208 (2 %)▯▯▯▯▯▯▯▯98304 (2 %)▯▯▯▯▯▯▯▯102400 (2 %)▯▯▯▯▯▯▯▯106496 (2 %)▯▯▯▯▯▯▯▯110592 (2 %)▯▯▯▯▯▯▯▯114688 (2 %)▯▯▯▯▯▯▯▯118784 (2 %)▯▯▯▯▯▯▯▯122880 (2 %)▯▯▯▯▯▯▯▯12697
▯▯▯▯▯▯135168 (3 %)▯▯▯▯▯▯▯▯139264 (3 %)▯▯▯▯▯▯▯▯143360 (3 %)▯▯▯▯▯▯▯▯147456 (3 %)▯▯▯▯▯▯▯▯151552 (3 %)▯▯▯▯▯▯▯▯155648 (3 %)▯▯▯▯▯▯▯▯159744 (3 %)▯▯▯▯▯▯▯▯163840 (3 %)▯▯▯▯▯▯▯▯1
▯▯▯▯▯▯▯▯176128 (4 %)▯▯▯▯▯▯▯▯180224 (4 %)▯▯▯▯▯▯▯▯184320 (4 %)▯▯▯▯▯▯▯▯188416 (4 %)▯▯▯▯▯▯▯▯192512 (4 %)▯▯▯▯▯▯▯▯196608 (4 %)▯▯▯▯▯▯▯▯200704 (4 %)▯▯▯▯▯▯▯▯204800 (4 %)▯▯▯▯▯▯▯▯
5 %)▯▯▯▯▯▯▯▯217088 (5 %)▯▯▯▯▯▯▯▯221184 (5 %)▯▯▯▯▯▯▯▯225280 (5 %)▯▯▯▯▯▯▯▯229376 (5 %)▯▯▯▯▯▯▯▯233472 (5 %)▯▯▯▯▯▯▯▯237568 (5 %)▯▯▯▯▯▯▯▯241664 (5 %)▯▯▯▯▯▯▯▯245760 (5 %)▯▯▯▯
52 (6 %)▯▯▯▯▯▯▯▯258048 (6 %)▯▯▯▯▯▯▯▯262144 (6 %)▯▯▯▯▯▯▯▯266240 (6 %)▯▯▯▯▯▯▯▯270336 (6 %)▯▯▯▯▯▯▯▯274432 (6 %)▯▯▯▯▯▯▯▯278528 (6 %)▯▯▯▯▯▯▯▯282624 (6 %)▯▯▯▯▯▯▯▯286720 (6 %)▯
294912 (7 %)▯▯▯▯▯▯▯▯299008 (7 %)▯▯▯▯▯▯▯▯303104 (7 %)▯▯▯▯▯▯▯▯307200 (7 %)▯▯▯▯▯▯▯▯311296 (7 %)▯▯▯▯▯▯▯▯315392 (7 %)▯▯▯▯▯▯▯▯319488 (7 %)▯▯▯▯▯▯▯▯323584 (7 %)▯▯▯▯▯▯▯▯327680 (7
▯▯▯335872 (8 %)▯▯▯▯▯▯▯▯339968 (8 %)▯▯▯▯▯▯▯▯344064 (8 %)▯▯▯▯▯▯▯▯348160 (8 %)▯▯▯▯▯▯▯▯352256 (8 %)▯▯▯▯▯▯▯▯356352 (8 %)▯▯▯▯▯▯▯▯360448 (8 %)▯▯▯▯▯▯▯▯364544 (8 %)▯▯▯▯▯▯▯▯36864
▯▯▯▯▯▯376832 (8 %)▯▯▯▯▯▯▯▯380928 (9 %)▯▯▯▯▯▯▯▯385024 (9 %)▯▯▯▯▯▯▯▯389120 (9 %)▯▯▯▯▯▯▯▯393216 (9 %)▯▯▯▯▯▯▯▯397312 (9 %)▯▯▯▯▯▯▯▯401408 (9 %)▯▯▯▯▯▯▯▯405504 (9 %)▯▯▯▯▯▯▯▯4
```

Fig. 10. Esptool.py Running Results

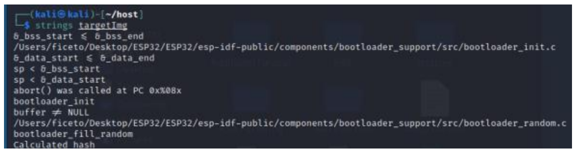
7. Once it has finished running, we can now navigate back to the folder containing "esptool.py" and should see the "targetImg" file that was generated (Fig. 11).


Clipboard
Organize
New
Open
This PC › Hard Drive (D:) › Documents › ArduinoData › packages › esp32 › hardware › esp32

Name	Date modified	Type
partitions	9/13/2019 1:06 AM	File folder
sdk	9/13/2019 1:06 AM	File folder
espota	9/13/2019 1:06 AM	Application
espota	9/13/2019 1:06 AM	PY File
esptool	9/13/2019 1:06 AM	PY File
gen_esp32part	9/13/2019 1:06 AM	Application
gen_esp32part	9/13/2019 1:06 AM	PY File
targetImg	5/27/2022 4:35 PM	File

Fig. 11. Downloaded ESP32-CAM Image

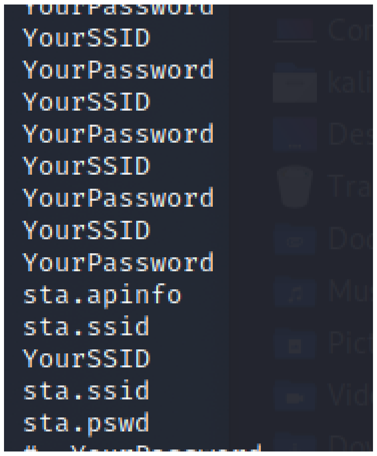
8. Take the targetImg file and upload it to a Kali Linux VM. Using a terminal, navigate to the folder that contains the file and type the command "strings targetImg" to derive all readable strings from the binary image (Fig. 12).



```
┌──(kali㉿kali)-[~/host]
└─$ strings targetImg
&_bss_start ≤ &_bss_end
/Users/ficeto/Desktop/ESP32/ESP32/esp-idf-public/components/bootloader_support/src/bootloader_init.c
&_data_start ≤ &_data_end
sp < &_bss_start
sp < &_data_start
abort() was called at PC 0x%08x
bootloader_init
buffer ≠ NULL
/Users/ficeto/Desktop/ESP32/ESP32/esp-idf-public/components/bootloader_support/src/bootloader_random.c
bootloader_fill_random
Calculated hash
```

Fig. 12. Commands to Derive Strings from a Binary Image

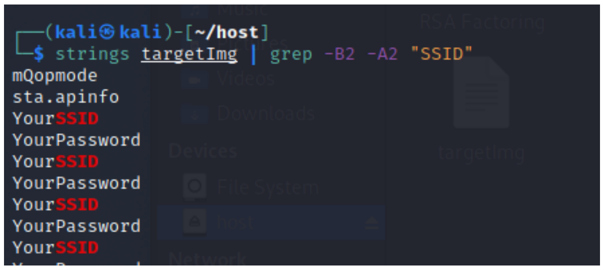
9. So many strings are derived from the "targetImg" file (Fig. 13).



```
YourPassword
YourSSID
YourPassword
YourSSID
YourPassword
YourSSID
YourPassword
YourSSID
YourPassword
sta.apinfo
sta.ssid
YourSSID
sta.ssid
sta.pswd
```

Fig. 13. Embedded Strings in targetImg

10. The SSID can be scanned by a cell phone. The idea is if we can find the SSID string embedded in targetImg, we probably will be able to find its password around it. The grep command along with the context searching options (2 lines 'B'efore and 2 lines 'A'fter the keyword 'SSID') can be used to search for strings around the SSID string (Fig. 14). This will



```
┌──(kali㉿kali)-[~/host]
└─$ strings targetImg | grep -B2 -A2 "SSID"
mQopmode
sta.apinfo
YourSSID
YourPassword
YourSSID
YourPassword
YourSSID
YourPassword
YourSSID
```

Fig. 14. Narrow down the Password Searching by "grep"

11. Thus, by Reverse Engineering the compiler and uploader that are called by the Arduino IDE, secrets hardcoded on the device can be extracted.

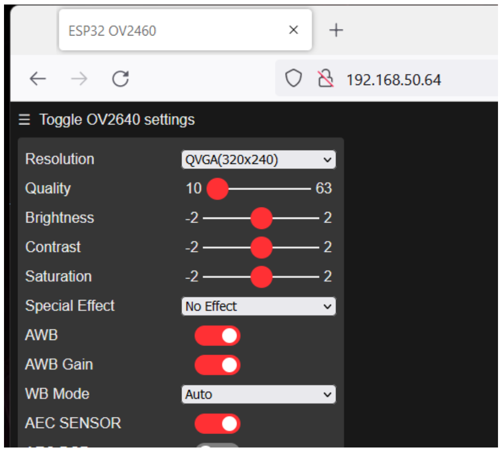
Trigger the application corruption by a buffer overflow

Front-end code of a Web application is usually the start point for attackers to perform Reverse Engineering against the application. Testing a variety of inputs can help discover vulnerabilities. Through the process of black box Reverse Engineering, a buffer overflow vulnerability in the framesize input variable can be detected. White box Reverse Engineering can then be used to inspect the code and find the area that pertains to the handling of the control variables of the application.

In addition, discovering this vulnerability demonstrates the importance of manual exploration, as an automated scan for vulnerabilities would not have necessarily discovered this vulnerability.

Here are the steps taken to discover the vulnerability:

1. Manually explore the web application. Note that the applications control panel offers a variety of user preferences that can be adjusted such as resolution, quality, brightness, etc. (Fig. 15).



```
Resolution       QVGA(320x240)
Quality          10 ● 63
Brightness       -2 ● 2
Contrast         -2 ● 2
Saturation       -2 ● 2
Special Effect   No Effect
AWB              ●
AWB Gain         ●
WB Mode          Auto
AEC SENSOR       ●
```

Fig. 15. Web Application Control Panel

2. Click the "Resolution" drop-down menu and select the largest setting. Then click "Start Stream". You should see that the stream window size is now much larger than the default (Fig. 16).

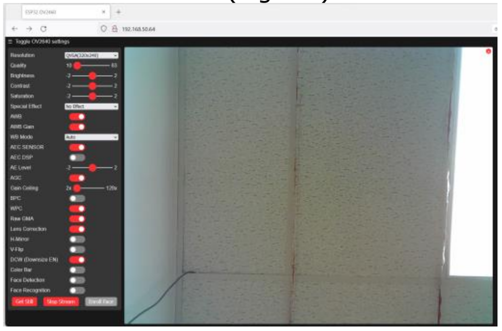
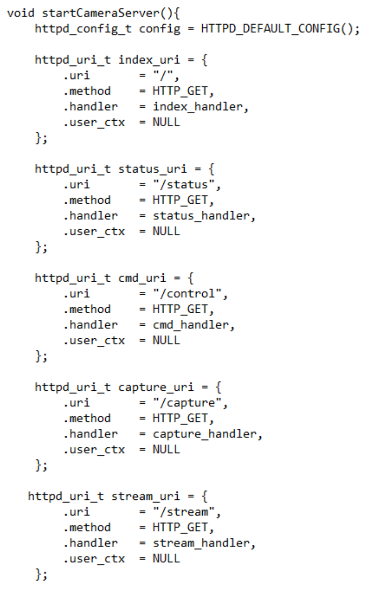


Fig. 16. Web Application Large Frame size

3. Now that we know that users can send control requests to the backend code that modifies aspects of the application, an analysis of the backend code can be performed to look for vulnerabilities.

4. By a quick look through the code, we can see that the "app_httpd.cpp" file contains all the handlers for the application: "stream_handler()," "cmd_handler()," "status_handler(),", "capture_handler()", and "index_handler()."

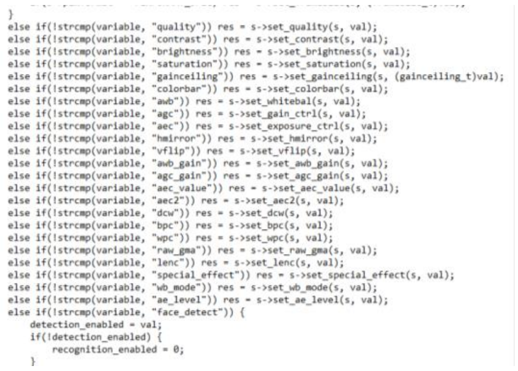
5. The handlers are initialized by the function "startCameraServer()". Fig. 17 shows the code fragment of the startCameraServer() function.



```
void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    httpd_uri_t index_uri = {
        .uri       = "/",
        .method    = HTTP_GET,
        .handler   = index_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t status_uri = {
        .uri       = "/status",
        .method    = HTTP_GET,
        .handler   = status_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t cmd_uri = {
        .uri       = "/control",
        .method    = HTTP_GET,
        .handler   = cmd_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t capture_uri = {
        .uri       = "/capture",
        .method    = HTTP_GET,
        .handler   = capture_handler,
        .user_ctx  = NULL
    };

   httpd_uri_t stream_uri = {
        .uri       = "/stream",
        .method    = HTTP_GET,
        .handler   = stream_handler,
        .user_ctx  = NULL
    };
```

Fig. 17. Code fragment of "startCameraSever()" Function

6. From the names of the handlers, the "cmd_handler()" function should control the user commands on the control panel. Inspecting the code inside of the handler confirms this as it contains all the different control panel variables (Fig. 18).



```
}
else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
else if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
else if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
else if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s, (gainceiling_t)val);
else if(!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
else if(!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
else if(!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
else if(!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
else if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
else if(!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
else if(!strcmp(variable, "awb_gain")) res = s->set_awb_gain(s, val);
else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
else if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
else if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
else if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
else if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
else if(!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
else if(!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
else if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
else if(!strcmp(variable, "special_effect")) res = s->set_special_effect(s, val);
else if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
else if(!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
else if(!strcmp(variable, "face_detect")) {
    detection_enabled = val;
    if(!detection_enabled) {
        recognition_enabled = 0;
    }
```

Fig. 18. Code Fragment of "cmd_handler()" Function

7. By looking at the code, we can see there is no input validation for the "framesize" variable in the cmd_handler() function. It simply uses the value of the "val" parameter that is passed from the user request. So, what if the user tries setting the framesize to a negative number?

```
int val = atoi(value);
sensor_t * s = esp_camera_sensor_get();
int res = 0;

if(!strcmp(variable, "framesize")) {
        if(s->pixformat == PIXFORMAT_JPEG) {
            res = s->set_framesize(s, (framesize_t)val);
        }
    }
```

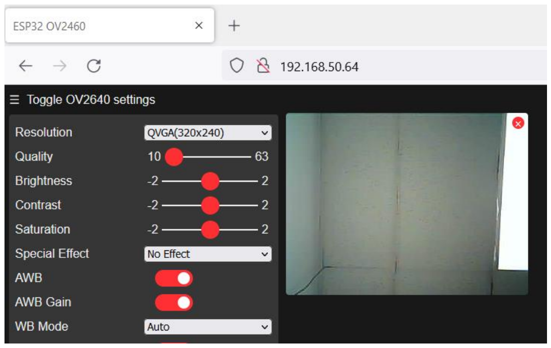
8. From this block of code, the "/control" URL via a HTTP_GET request can be used to trigger the "cmd_handler()" function (Fig. 19).

```
httpd_uri_t cmd_uri = {
            .uri       = "/control",
            .method    = HTTP_GET,
            .handler   = cmd_handler,
            .user_ctx  = NULL
    };
```

Fig. 19. Code Fragment of "startCameraServer()" Function

9. With this information, we can conclude that we can bypass the drop-down menu of the application and enter in the request directly into the webpage's search box by combining the designation for the command handler, the control variable, and the control value. Thus, the attack vector http://{IP}/control?var=framesize&val=x where the "IP" is replaced with the IP address of the App and the "x" can be replaced with any value. This will allow for values outside of the drop-down box parameters to be input into the system.

10. Go to the application and click "start stream" and verify it is working properly. Note the size of the stream window (Fig. 20).



Resolution QVGA(320x240)
Quality 10 63
Brightness -2 2
Contrast -2 2
Saturation -2 2
Special Effect No Effect
AWB
AWB Gain
WB Mode Auto

Fig. 20. Web Application Stream

11. Enter the attack vector with "x" being set to the value 10. A blank page will load, click back to the application. You should see that the window is bigger, indicating that request was successful in modifying the application's frame size (Fig. 21). Now a variety of strange inputs can be tried to test the application for bugs.

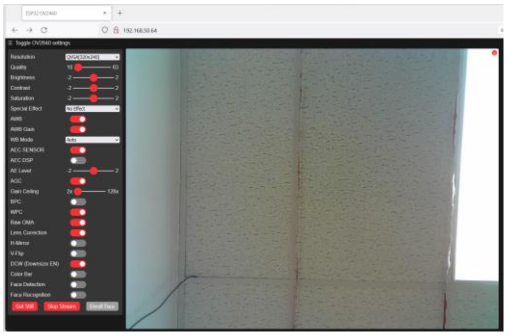
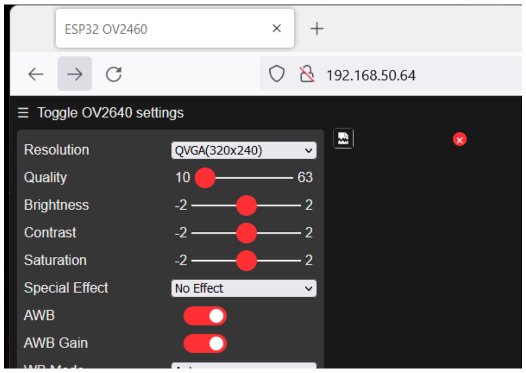


Fig. 21. Web Stream Enlarged

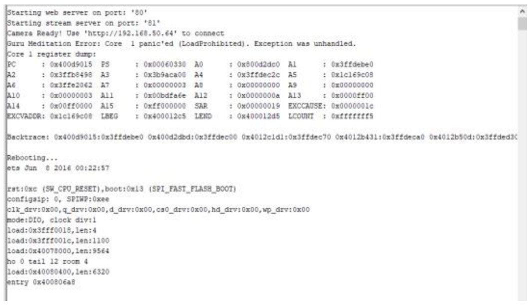
12. Try a negative value for the parameter. The application fails to display a window at all, revealing a successful attack to the system (Fig. 22).



Resolution QVGA(320x240)
Quality 10 63
Brightness -2 2
Contrast -2 2
Saturation -2 2
Special Effect No Effect
AWB
AWB Gain

Fig. 22. Web Stream Broken

13. Try a ridiculous big value for the parameter. You should see that this request causes the stream to be unresponsive and timeout. A look at the serial monitor logs reveals that it has caused an exception and the system is continually rebooting (Fig. 23).



```
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.50.64' to connect
Guru Meditation Error: Core  1 panic'ed (LoadProhibited). Exception was unhandled.
Core 1 register dump:
PC      : 0x400d9015  PS      : 0x00060330  A0      : 0x800d2dc0  A1      : 0x3ffdebe0
A2      : 0x3ffb8498  A3      : 0x3b9aca00  A4      : 0x3ffdec2c  A5      : 0x1c169c08
A6      : 0x3ffe2062  A7      : 0x00000003  A8      : 0x00000000  A9      : 0x00000000
A10     : 0x00000003  A11     : 0x00bdfa6e  A12     : 0x0000000a  A13     : 0x0000ff00
A14     : 0x00ff0000  A15     : 0xff000000  SAR     : 0x00000019  EXCCAUSE: 0x0000001c
EXCVADDR: 0x1c169c08  LBEG    : 0x400012c5  LEND    : 0x400012d5  LCOUNT  : 0xfffffff5

Backtrace: 0x400d9015:0x3ffdebe0 0x400d2dbd:0x3ffdec00 0x4012c1d1:0x3ffdec70 0x4012b431:0x3ffdeca0 0x4012b50d:0x3ffded3

Rebooting...
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1100
load:0x40078000,len:9564
ho 0 tail 12 room 4
load:0x40080400,len:6320
entry 0x400806a8
```

Fig. 23. Serial Monitor Error Log

14. Try a smaller value that is still larger than 10. You should see that the stream is broken and full of unintelligible bars (Fig. 24). In this case since 12 appears to be just slightly over the given threshold, the overflow resulted in just the stream data being corrupted and did not crash the entire application.

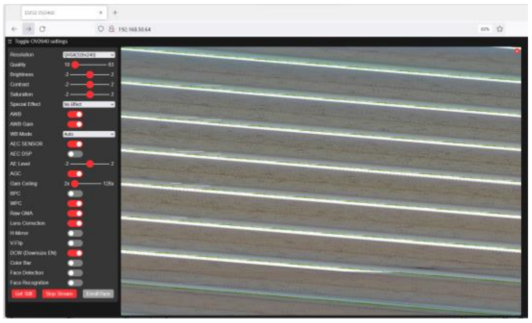


Fig. 24. Frame size Buffer Overflow

15. Try to enter in a decimal value next, such as 5.5. The display should simply round the number and display the appropriate rounded value (Fig. 25).

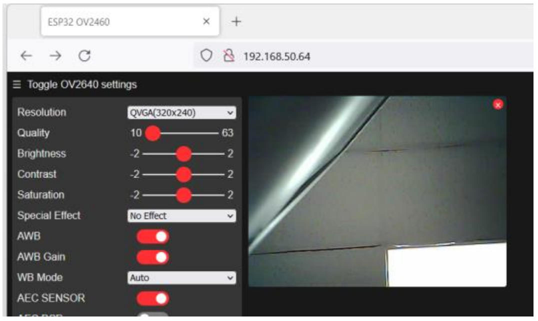
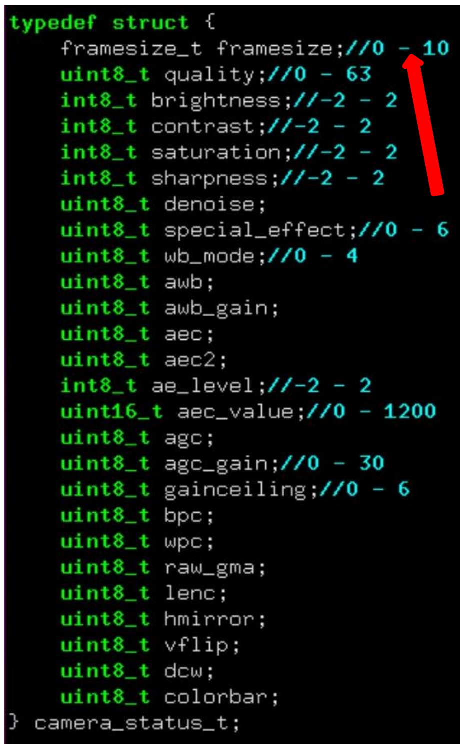


Fig. 25. Decimal Value for Frame size

16. Thus, the proper parameters for the framesize variable can be inferred: do not include negative numbers or numbers greater than 12. However, we do not yet know how large the numbers can be.



```
typedef struct {
    framesize_t framesize;//0 - 10
    uint8_t quality;//0 - 63
    int8_t brightness;//-2 - 2
    int8_t contrast;//-2 - 2
    int8_t saturation;//-2 - 2
    int8_t sharpness;//-2 - 2
    uint8_t denoise;
    uint8_t special_effect;//0 - 6
    uint8_t wb_mode;//0 - 4
    uint8_t awb;
    uint8_t awb_gain;
    uint8_t aec;
    uint8_t aec2;
    int8_t ae_level;//-2 - 2
    uint16_t aec_value;//0 - 1200
    uint8_t agc;
    uint8_t agc_gain;//0 - 30
    uint8_t gainceiling;//0 - 6
    uint8_t bpc;
    uint8_t wpc;
    uint8_t raw_gma;
    uint8_t lenc;
    uint8_t hmirror;
    uint8_t vflip;
    uint8_t dcw;
    uint8_t colorbar;
} camera_status_t;
```

Fig. 26. cscope Findings

17. We can use the 'cscope' tool to collect library code installed for Arduino IDE and ESP32 add-on. Along with the 4 source files, we can build a code database, where we can use vi to search for all the occurrences of an interesting variable through the code database. Then, we can get more information about that variable. For example, by searching for the keyword 'framesize', we can find a struct, where the valid values per that variable are shown in the comments (Fig. 26).

18. Thus, the valid inputs for the framesize parameter are between 0-10. Negative numbers cause a display failure and values greater than 10 cause the stream data to be corrupted due to a buffer overflow.

5. FUTURE WORK

While this paper identifies a variety of vulnerabilities in the face-recognition Web App for the ESP32-CAM IoT device, further exploration of the code and additional testing could reveal additional vulnerabilities that were not identified in this paper. E.g., we will cover "vulnerability detected by auto scan: X-frame options header not set", "AI model vulnerability", and "3rd party library vulnerability" lab exercises in another paper. In addition, the "Data Corruption" and "AI Model Manipulation" vulnerabilities were not patched as the solutions are beyond the scope of this paper. Future work could include an in-depth analysis of the AI model used for facial recognition and detection to prevent the model from accepting manipulated user input such as printed photos and sunglasses. Furthermore, tests could be performed on the AI model and devices to determine what made some of the photos cause the system to crash. Also, we plan to add a multimedia supplement walking through the exercises.

6. CONCLUSIONS

It has become increasingly crucial for Cybersecurity and CS students to possess the skills needed to understand the different components of a software application and how they relate to one another in the system. The process of Reverse Engineering can provide students with these skills. This paper provides an IoT-based framework to accomplish this goal, developing various labs to enhance students' competency on vulnerability analysis by Reverse Engineering.

7. REFERENCES

Ali, M. (2005). Why teach Reverse Engineering. In ACM SIGSOFT Software Engineering Notes (pp. 1–4). ACM Digital Library. https://doi.org/10.1145/1082983.1083004

Bellettini, C., Marchetto, A., & Trentini, A. (2004). WebUml: Reverse Engineering of web applications. In SAC '04: Proceedings of the 2004 ACM symposium on Applied computing (pp. 1662-1669). ACM Digital Library. https://doi.org/10.1145/967900.968231

Canfora, G., & Penta, M. (2007). New Frontiers of Reverse Engineering. In Future of Software Engineering, FOSE 2007 (pp. 326-341). IEEE Xplore. https://doi.org/10.1109/FOSE.2007.15

Canfora, G., Penta, M., & Cerulo, L. (2011). Achievements and challenges in software Reverse Engineering. In Communications of the ACM (pp. 142-151). ACM Digital Library. https://doi.org/10.1145/1924421.1924451

Chikofsky, E., & Cross J. (1990). Reverse engineering and design recovery: a taxonomy. In IEEE Software (pp. 13-17). IEEE Xplore. https://doi.org/10.1109/52.43044

Fezari, M. & Al Dahoud, A. (2018). Integrated Development Environment "IDE" For Arduino. Research Gate. https://www.researchgate.net/publication/328615543_Integrated_Development_Environment_IDE_For_Arduino

Li, Z., Chou, E., & McAllister, C. (2022). An IoT Based New Platform for Teaching Web Application Security. CYBERSECURITY PEDAGOGY & PRACTICE JOURNAL.

Li, Z., Ren, H., Chou, E., Liu, X., & McAllister, C. D. (2022). Retrieving Forensically Sound Evidence from the ESP Series of IoT Devices. IEEE Internet of Things Journal.

Lueth, K. L. (2015). IoT market analysis: Sizing the opportunity. IoT Analytic Report. March.

Morgan, S. (2020, Nov. 13). Cybercrime To Cost the World $10.5 Trillion Annually by 2025. In Cybercrime Magazine. https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

Müller, H., Jahnke, J., Smith, D., Storey, M., Tilley, S., & Wong, K. (2000). Reverse engineering: a roadmap. In ICSE '00 (47-60). ACM Digital Library. https://doi.org/10.1145/336512.336526

Patel, R., Coenen, F., Martin, R., & Archer, L. (2007). Reverse Engineering of Web Applications: A Technical Review.

Potter, B. (2009). Microsoft SDL threat modelling tool. Network Security, 2009(1), 15-18.

Radware (2022). Global Threat Analysis Report. https://www.radware.com/getattachment/3d26f50b-f2a7-4ffa-9a84-1b5a598a0b27/2021-2022-Global-Threat-Analysis-Report_2022-FINAL-V2.pdf.aspx

Shwartz, O., Mathov, Y., Bohadana, M., Elovici, Y., & Oren, Y. (2018). Reverse Engineering IoT Devices: Effective Techniques and Methods. In IEEE Internet of Things Journal (pp. 4965-4976). IEEE Xplore. https://doi.org/10.1109/JIOT.2018.2875240

Taylor, C., & Collberg, C. (2016). A tool for teaching Reverse Engineering. In 2016 USENIX Workshop on Advances in Security Education, ASE 2016. USENIX. https://www.usenix.org/conference/ase16/workshop-program/presentation/taylor

Wikipedia contributors. (2022, July 5). ESP32. In Wikipedia, The Free Encyclopedia. Retrieved 19:04, July 15, 2022, from https://en.wikipedia.org/w/index.php?title=ESP32&oldid=1096637291

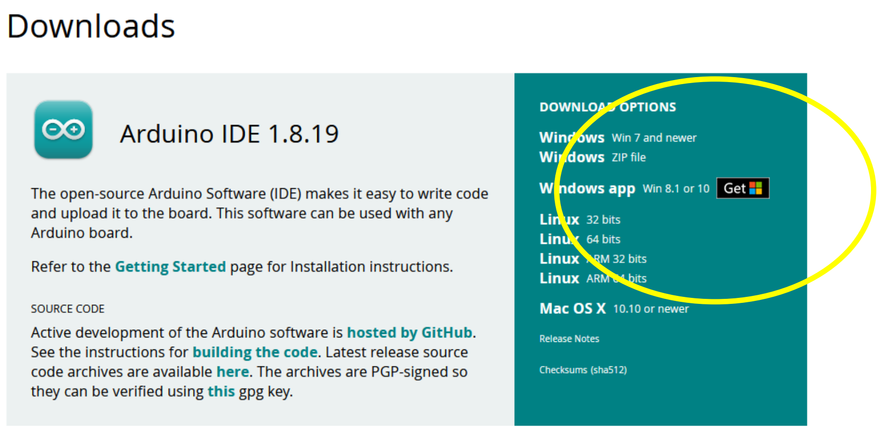
Appendix: Set up the lab environment

It is assumed that cybersecurity students participating in the course understand how to set-up a Kali Linux Virtual Machine using tools such as Virtual Box. The screenshots taken for this tutorial are on a Windows machine, but the entire curriculum could be performed on a Kali Linux VM.

1. **Download the latest Arduino IDE**

    1. Go to https://www.arduino.cc/en/software

    2. Select the appropriate download option for your operating system. For the purposes of this paper and for simplicity, we recommend starting with the Windows operating system



Downloads

Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

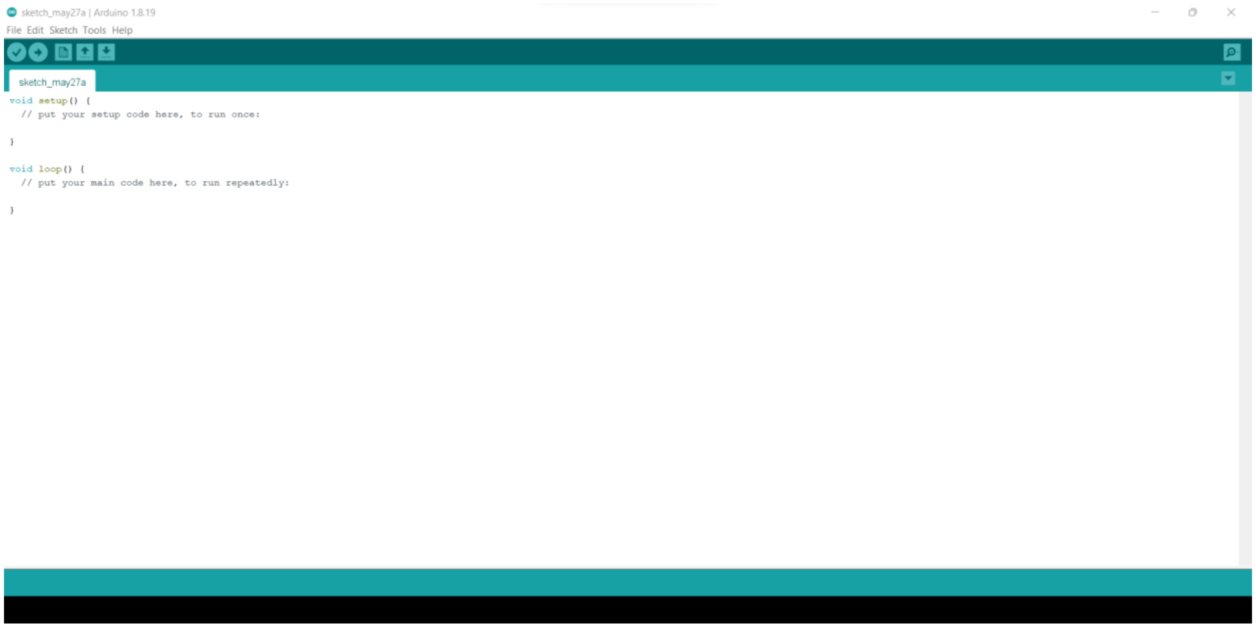
DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer
Release Notes
Checksums (sha512)

    3. Once the installer has been downloaded, follow the installation instructions

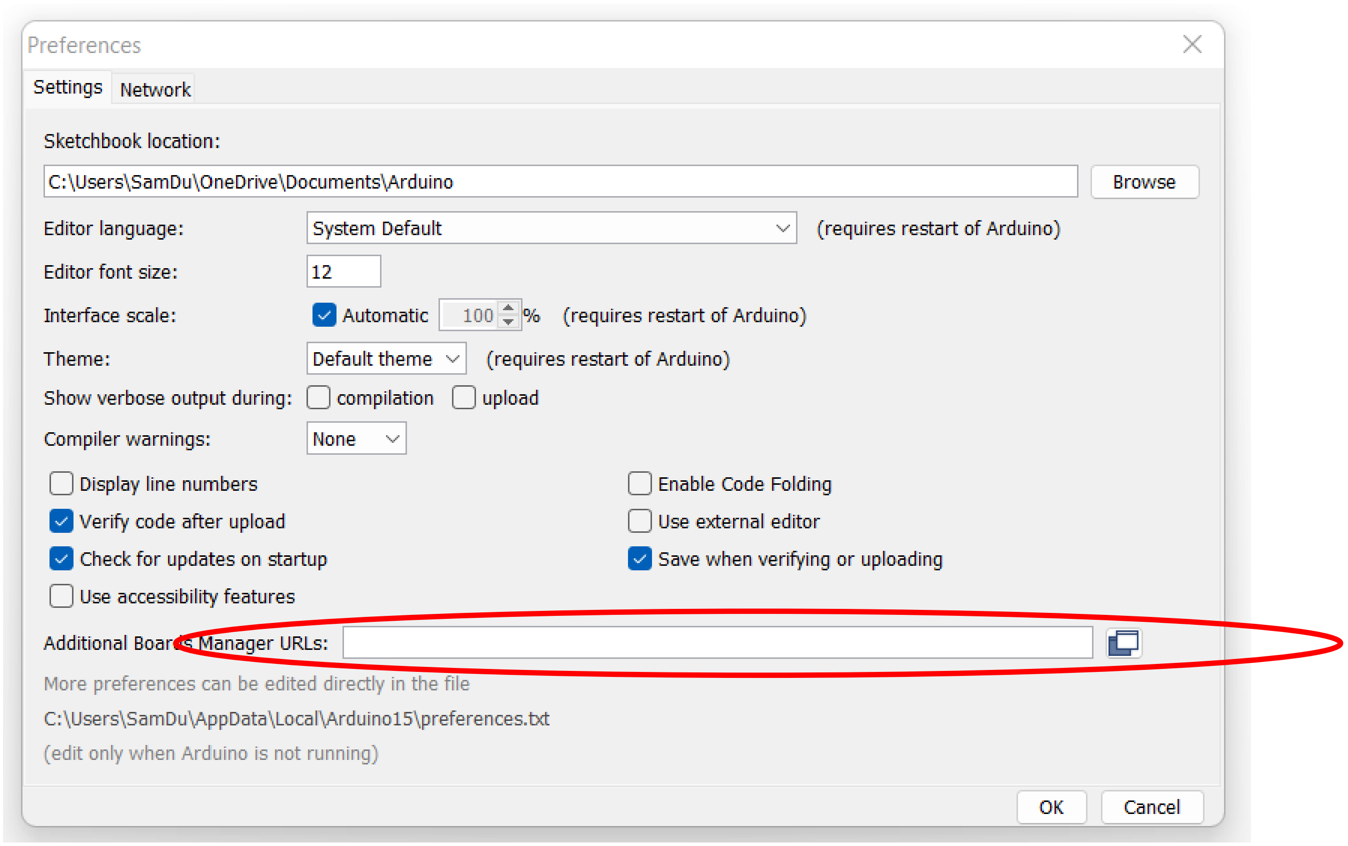
    4. Open the Arduino IDE via the start menu or desktop shortcut

It should look like this:



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

5. Navigate to File->Preferences->AdditionalBoardsManagerURLS:



Sketchbook location: C:\Users\SamDu\OneDrive\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

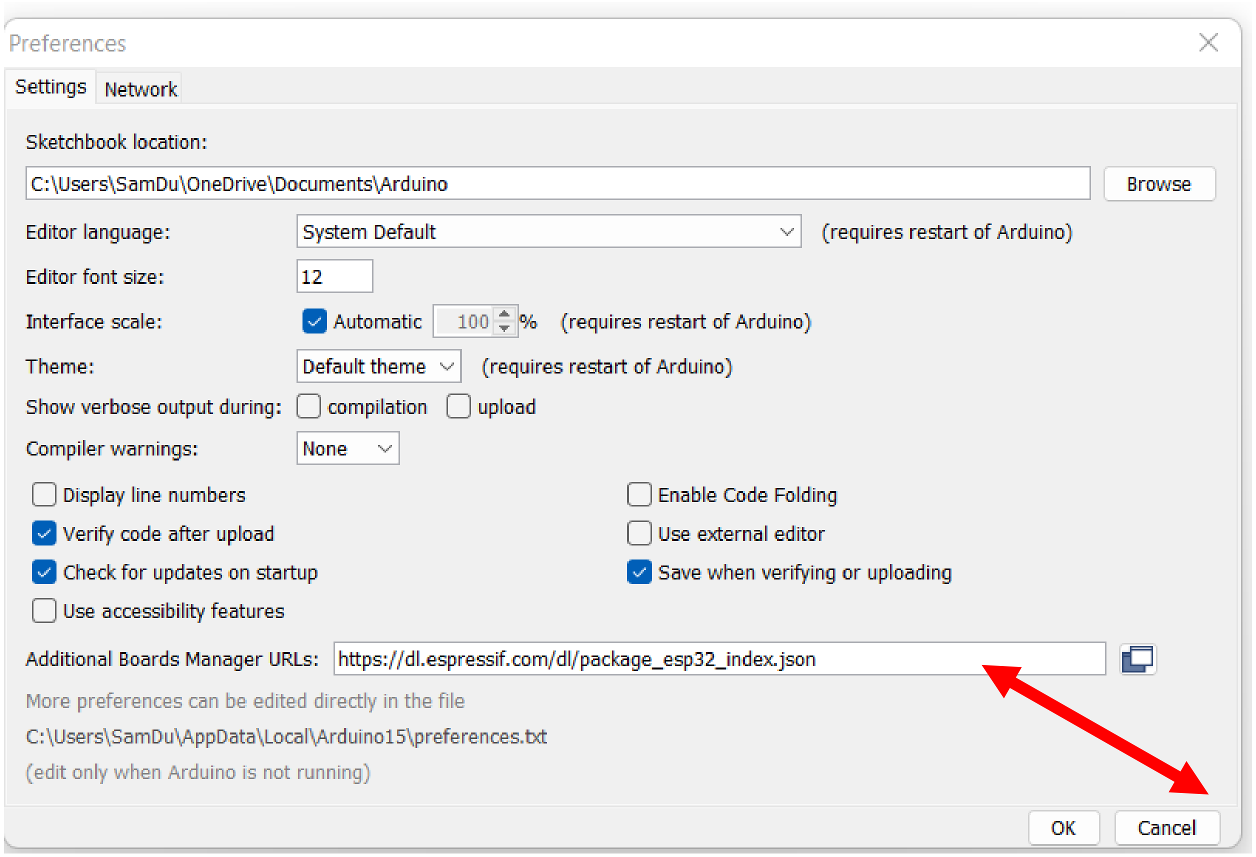
[ ] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Check for updates on startup
[✓] Save when verifying or uploading
[ ] Use accessibility features

Additional Boards Manager URLs:

More preferences can be edited directly in the file
C:\Users\SamDu\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

6. Paste this URL into the indicated textbox and click "ok":

https://dl.espressif.com/dl/package_esp32_index.json



Sketchbook location: C:\Users\SamDu\OneDrive\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

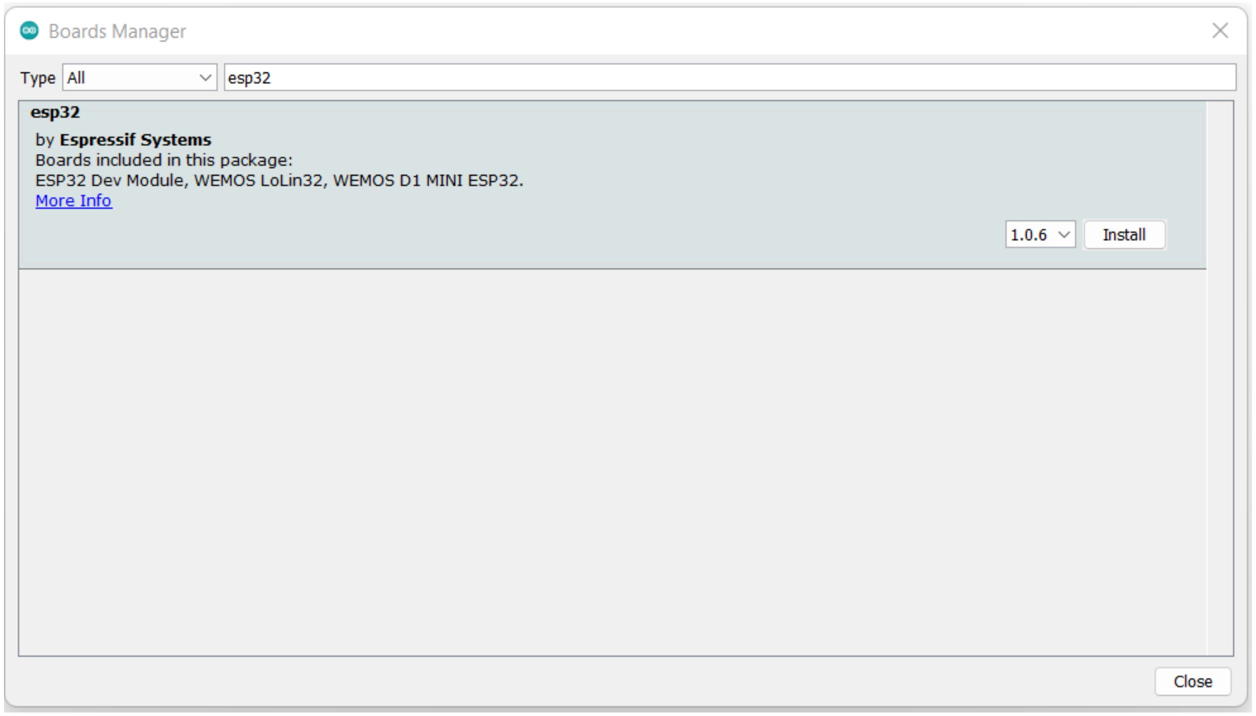
Compiler warnings: None

[ ] Display line numbers
[✓] Verify code after upload
[✓] Check for updates on startup
[ ] Use accessibility features
[ ] Enable Code Folding
[ ] Use external editor
[✓] Save when verifying or uploading

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json

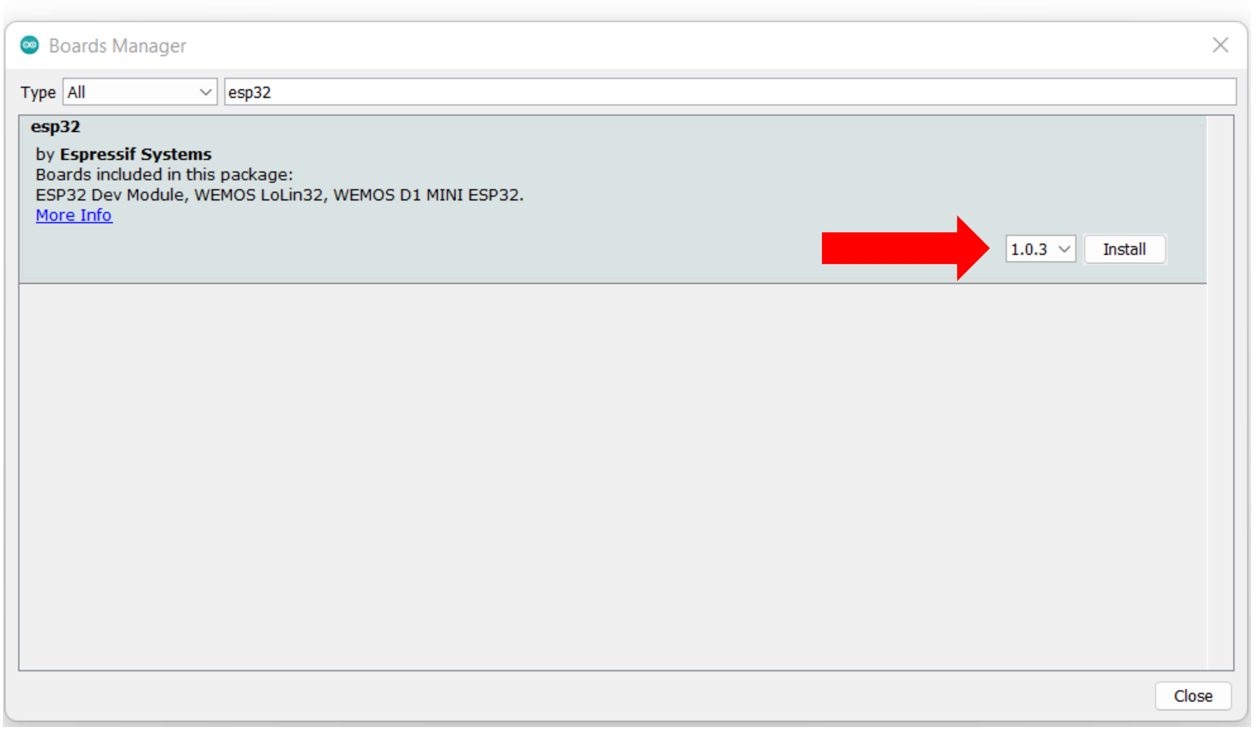
More preferences can be edited directly in the file
C:\Users\SamDu\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

7. Navigate to tools->Board->Boards Manager-> and type "esp32" in the search bar. If the additional package was added in the previous step, you should see esp32 as an option:

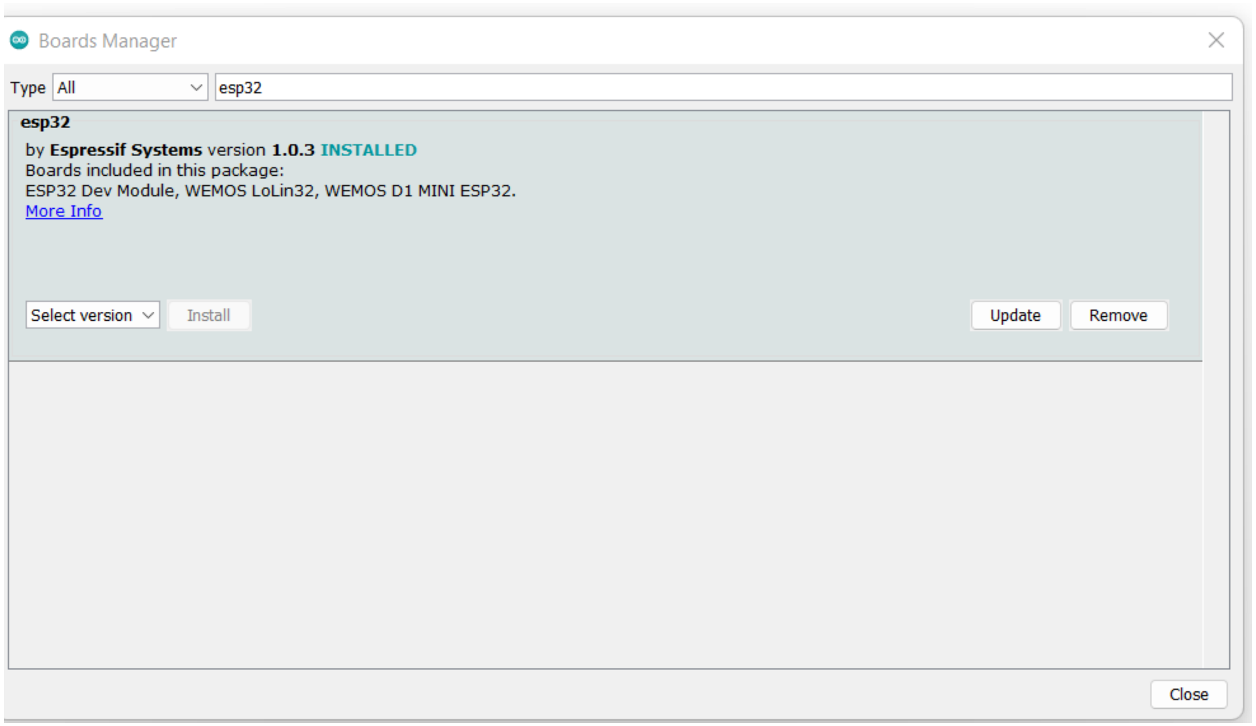


esp32
by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

8. In the right corner, select the 1.0.3 option and click install

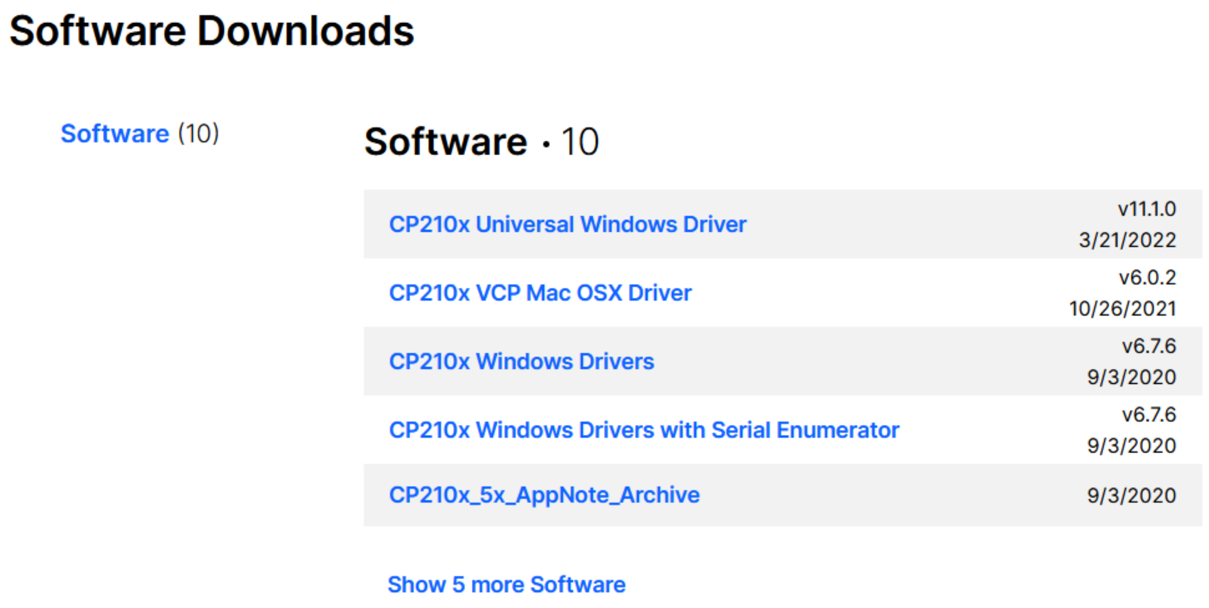

Boards Manager
Type All
esp32
esp32
by Espressif Systems
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info
1.0.3
Install
Close

9. Wait for the add-on to download. When it is finished you should see that it is installed:


Boards Manager
Type All
esp32
esp32
by Espressif Systems version 1.0.3 INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info
Select version
Install
Update
Remove
Close

10. Plug in the device to your computer via a USB cable. The cable must support data transportation, not just power.

11. Navigate to Tools->Port. You should see a selected COM port available.
12. If "Port" is grayed out, then you will need to install the proper UART driver for the device. Download the driver available at this link: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers
    1. Select the version compatible with your OS:



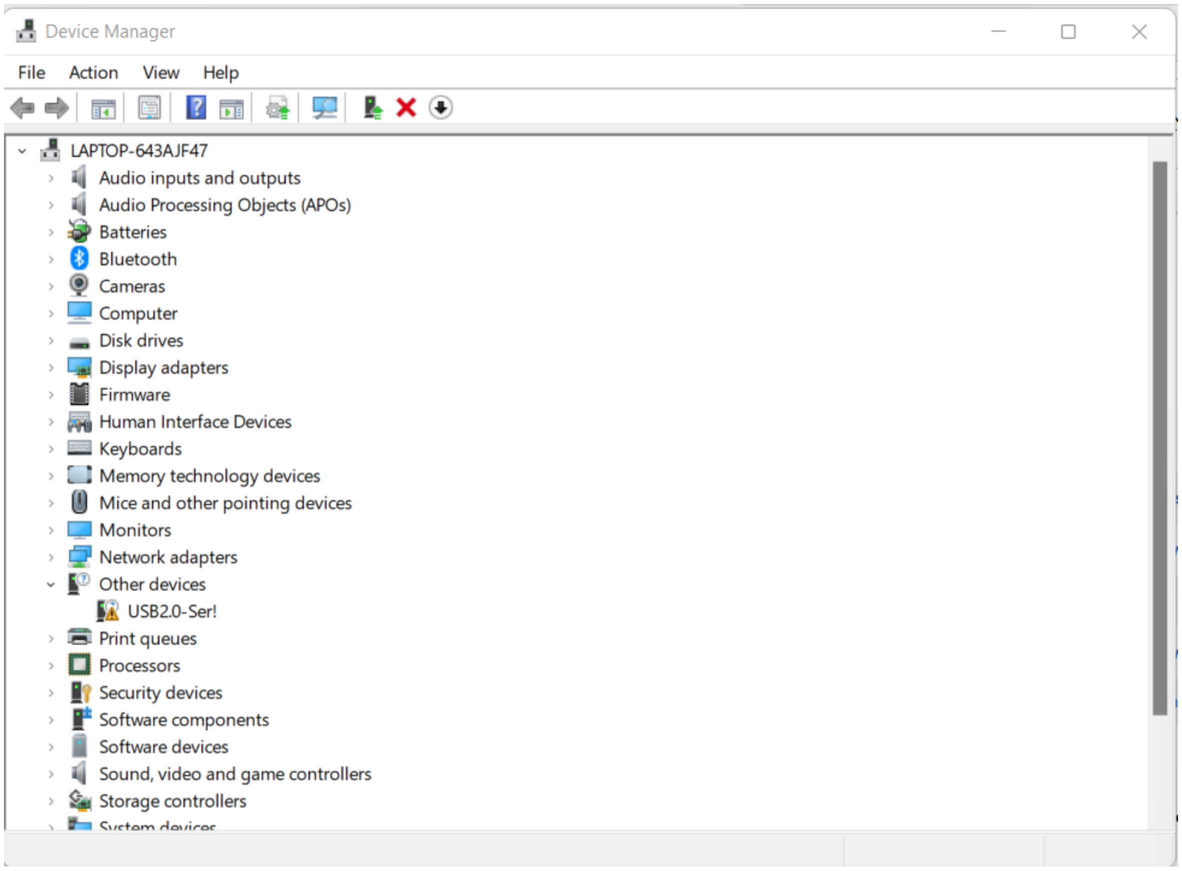
Software Downloads

Software (10)

Software · 10

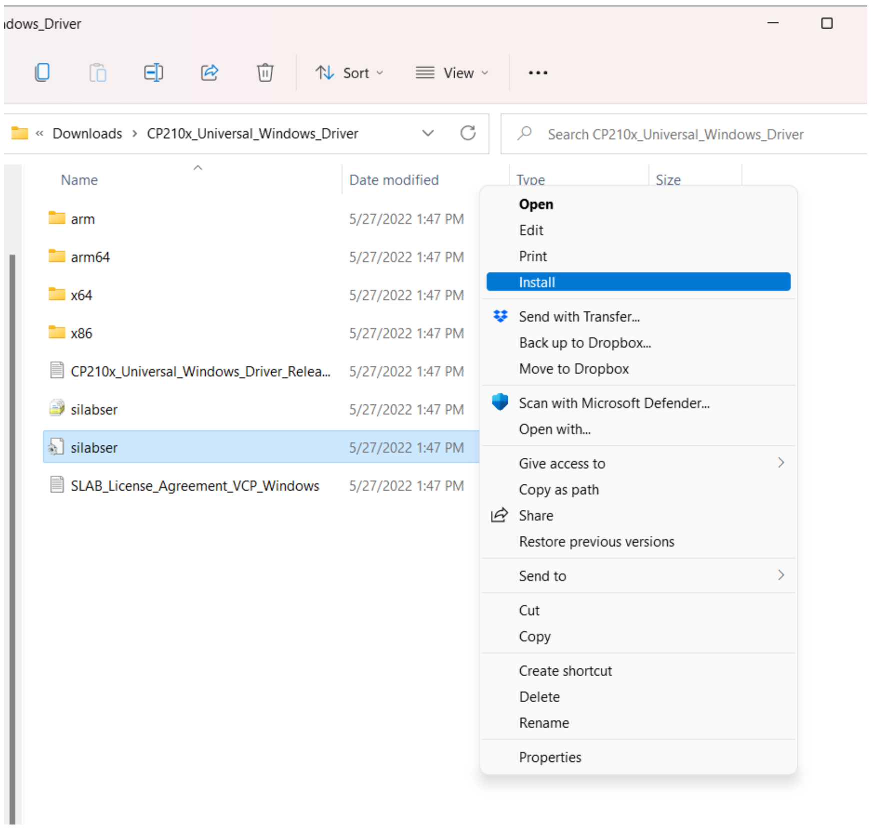
CP210x Universal Windows Driver	v11.1.0 3/21/2022
CP210x VCP Mac OSX Driver	v6.0.2 10/26/2021
CP210x Windows Drivers	v6.7.6 9/3/2020
CP210x Windows Drivers with Serial Enumerator	v6.7.6 9/3/2020
CP210x_5x_AppNote_Archive	9/3/2020

    2. Wait for the folder to download and extract the contents
    3. Go to your start menu and type in device manager. Locate the device in the "Other devices" section. This indicates that the device is not being recognized by the proper driver

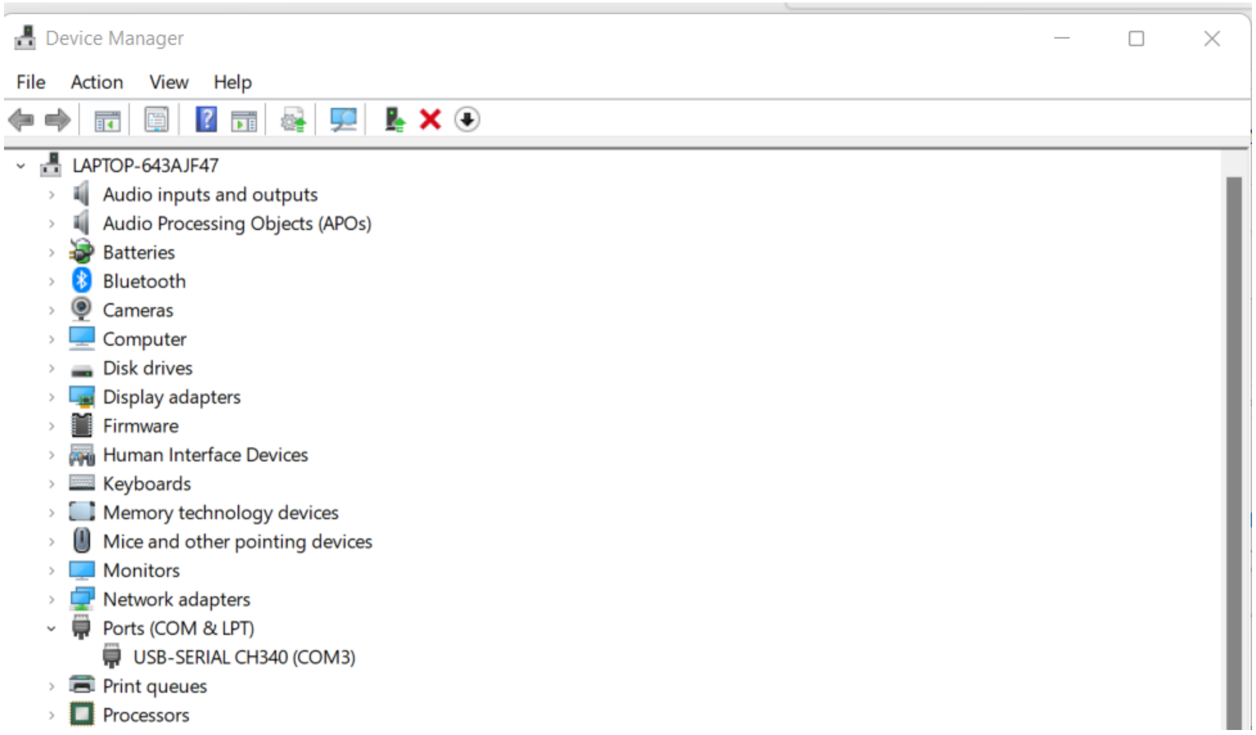


LAPTOP-643AJF47
- Audio inputs and outputs
- Audio Processing Objects (APOs)
- Batteries
- Bluetooth
- Cameras
- Computer
- Disk drives
- Display adapters
- Firmware
- Human Interface Devices
- Keyboards
- Memory technology devices
- Mice and other pointing devices
- Monitors
- Network adapters
- Other devices
  - USB2.0-Ser!
- Print queues
- Processors
- Security devices
- Software components
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices

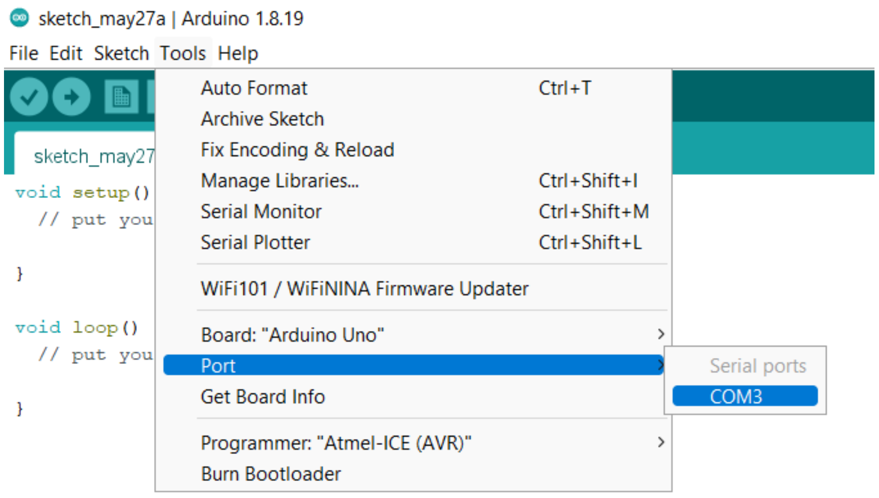
4. Navigate to the extracted driver folder, right click on the “silabser.inf” file and click install



5. Wait for the install to complete, then re-open device manager. You should see the device listed under “Ports (COM & LPT)”

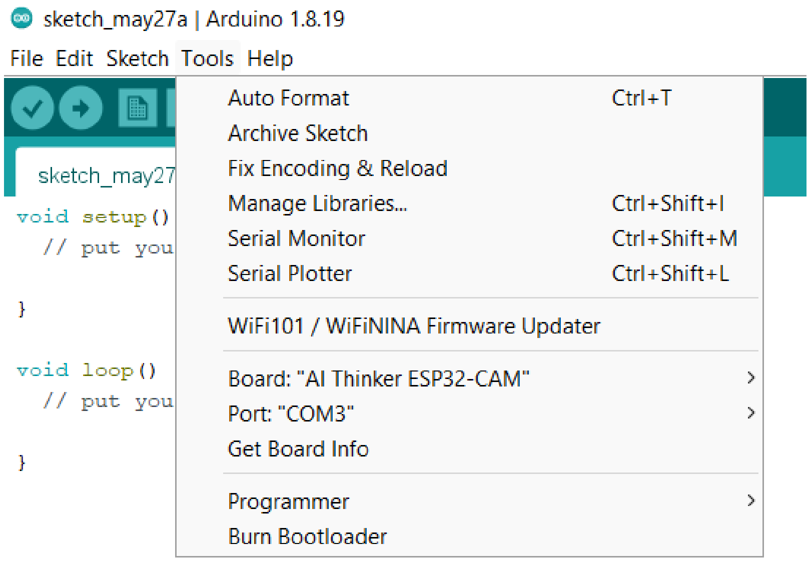


13. Once the proper driver is installed, re-open the Arduino IDE. Navigate to Tools -> Ports. Select the COM option available if it is not already selected.


sketch_may27a | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "Atmel-ICE (AVR)"
Burn Bootloader
Serial ports
COM3

```
sketch_may27
void setup() {
  // put you

}

void loop() {
  // put you

}
```

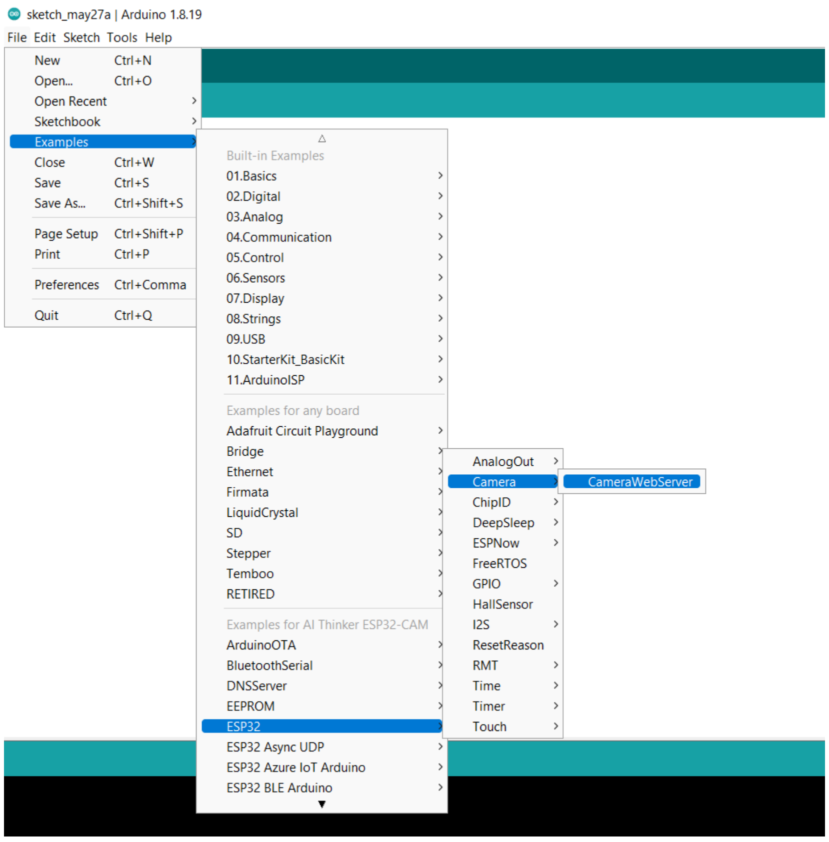
14. Then go to Tools -> Board -> ESP32 Arduino -> AI Thinker ESP32-CAM. The result should look like this (the COM port may be different):


sketch_may27a | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "AI Thinker ESP32-CAM"
Port: "COM3"
Get Board Info
Programmer
Burn Bootloader

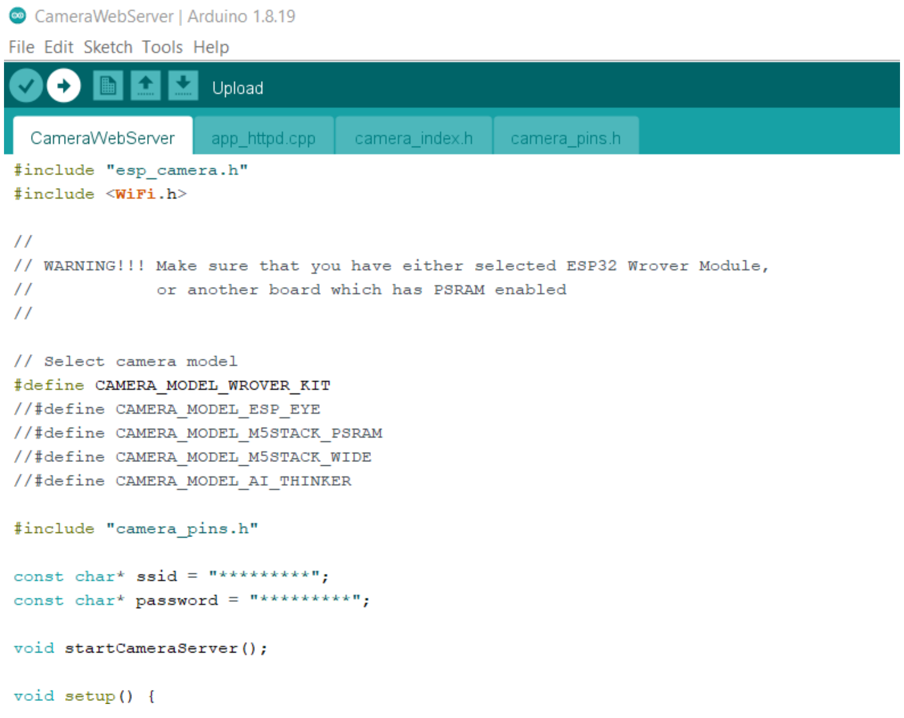
```
sketch_may27
void setup() {
  // put you

}

void loop() {
  // put you

}
```

2. Find CameraWebServer Application and Flash to ESP32-CAM Device:

a. The Arduino IDE with the esp32 package installed comes with several example programs. Navigate to File -> Examples -> ESP32 -> Camera -> CameraWebServer.

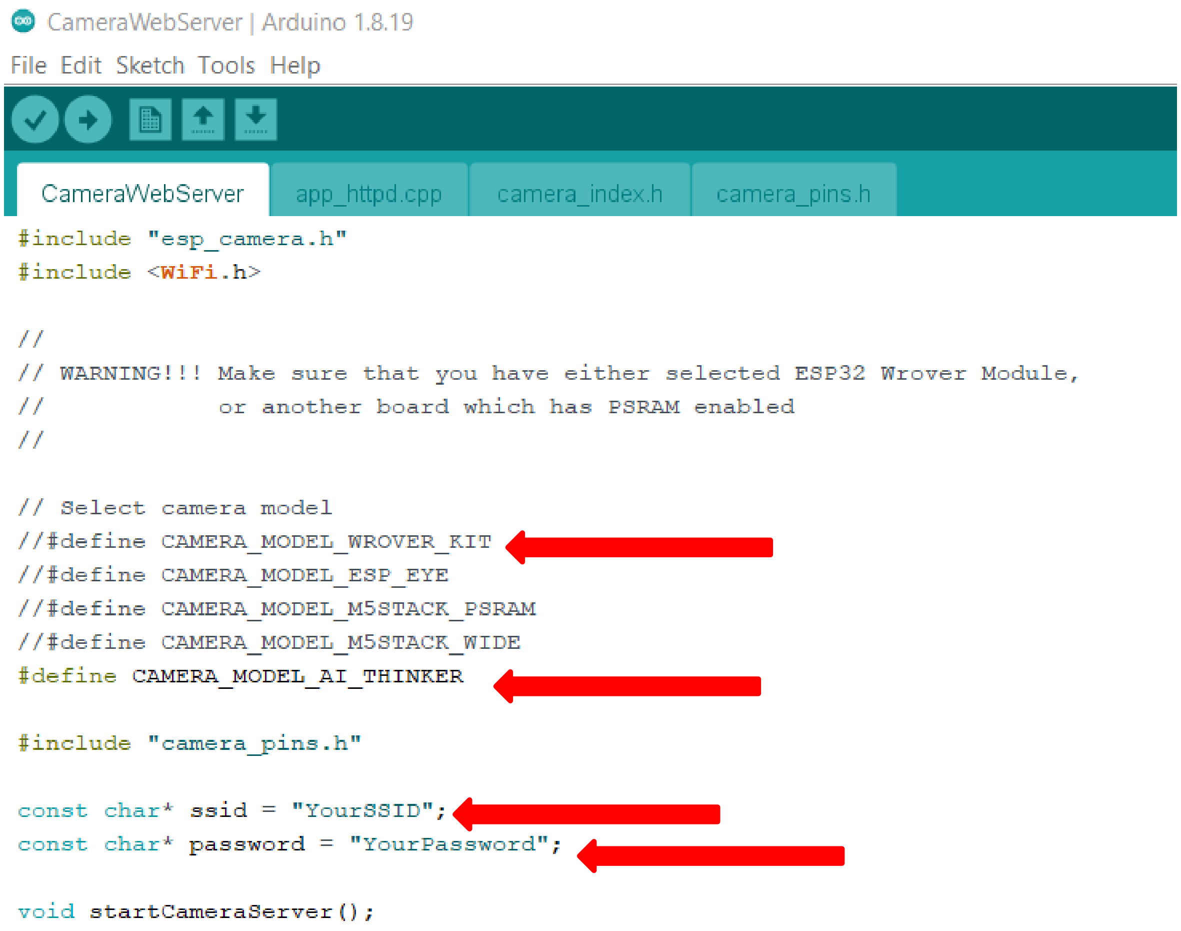


b. Load the program, it should look like this:



```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
//            or another board which has PSRAM enabled
//

// Select camera model
#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
//#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "*********";
const char* password = "*********";

void startCameraServer();

void setup() {
```

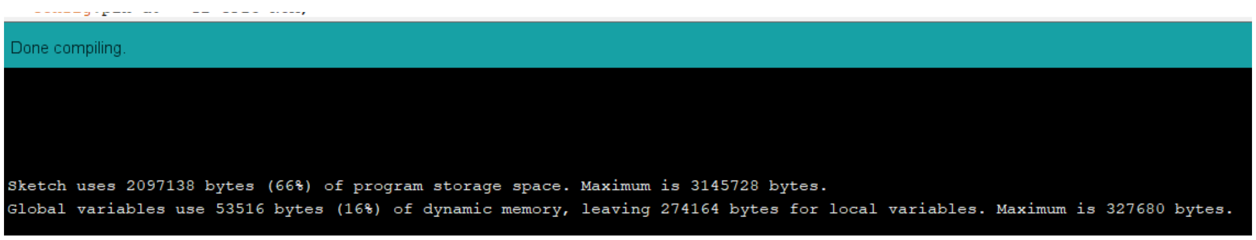
c. Comment out the ” define CAMERA_MODEL_WROVER_KIT” line, and uncomment the “#define CAMERA_MODEL_AI_THINKER”. Replace the "ssid" and "password" variables with your choice of ssid and password. Your code should now look something like this:



```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
//            or another board which has PSRAM enabled
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "YourSSID";
const char* password = "YourPassword";

void startCameraServer();
```

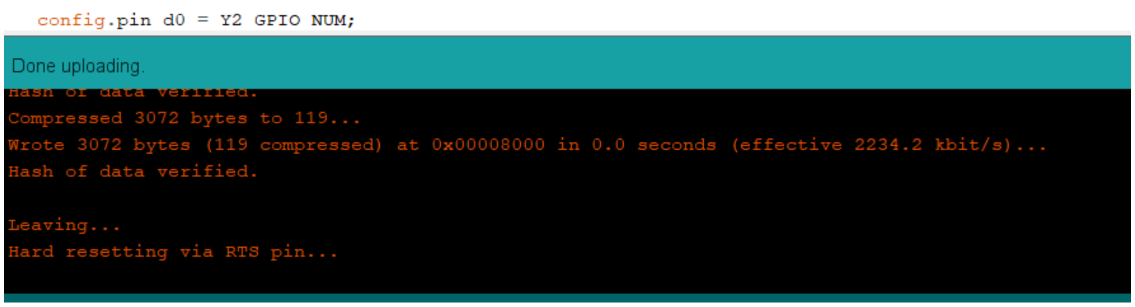
d. Click File -> Save to save the modifications. You may be prompted to save the sketch in a folder of your choosing.

e. Click the check mark in the top left to compile the code. If it compiles successfully, the output at the bottom of the IDE should look something like this:



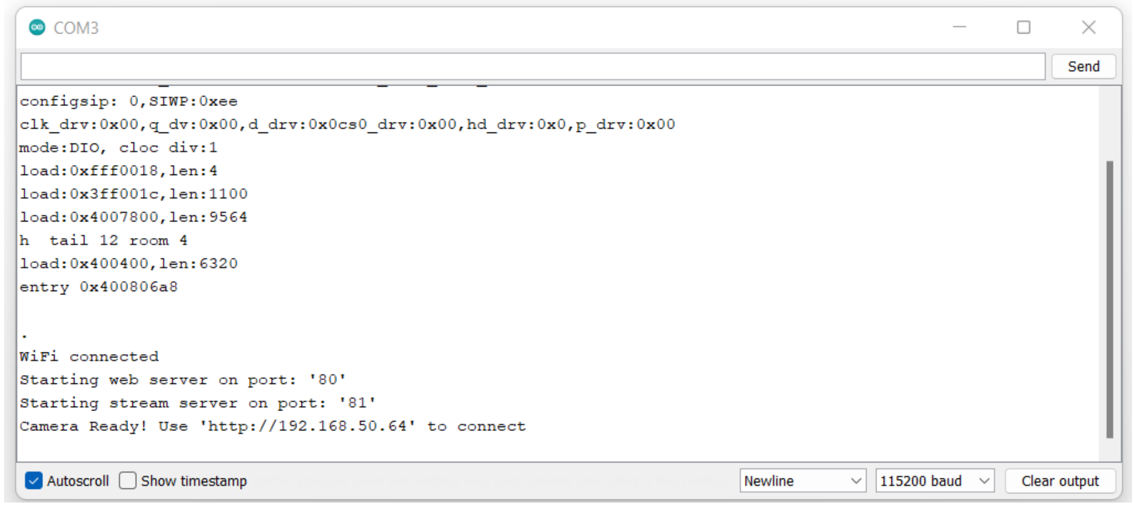
```
Sketch uses 2097138 bytes (66%) of program storage space. Maximum is 3145728 bytes.
Global variables use 53516 bytes (16%) of dynamic memory, leaving 274164 bytes for local variables. Maximum is 327680 bytes.
```

f. Next, click the arrow just to the right of the check mark to upload the code to the Device. Once again, make sure the proper COM port is selected and the proper device type is selected as well. If the code is uploaded properly, you should see something like this at the bottom of the IDE:



```
  config.pin d0 = Y2 GPIO NUM;
Done uploading.
Hash of data verified.
Compressed 3072 bytes to 119...
Wrote 3072 bytes (119 compressed) at 0x00008000 in 0.0 seconds (effective 2234.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

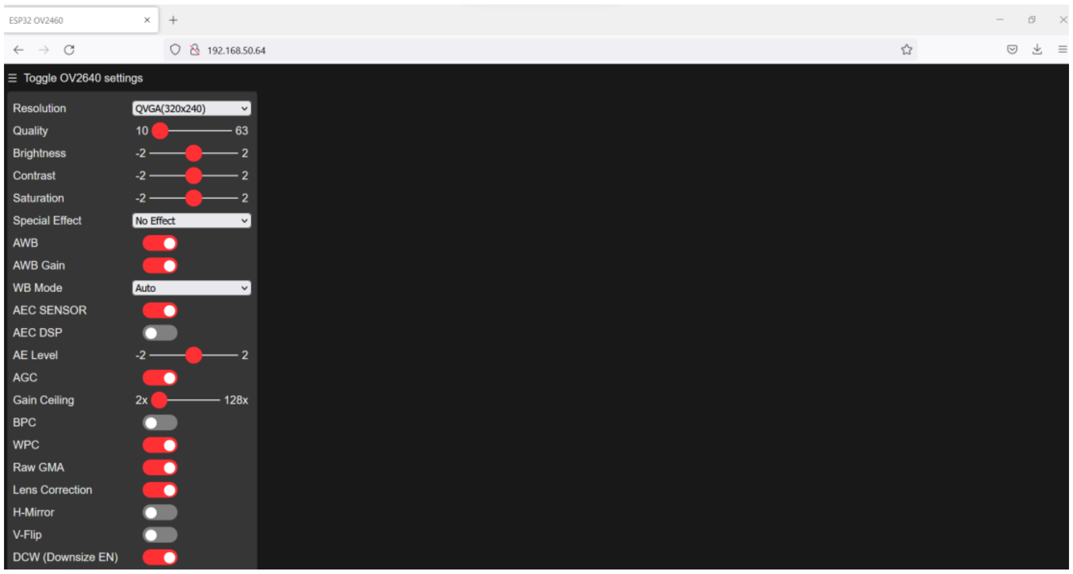
g. To access the IP address information for the web application, you must use the serial monitor. Click Tools -> Serial Monitor and set the baud rate to 115200 baud. Then click the RST button on the device. You should see something like this as an output:



```
configsip: 0,SIWP:0xee
clk_drv:0x00,q_dv:0x00,d_drv:0x0cs0_drv:0x00,hd_drv:0x0,p_drv:0x00
mode:DIO, cloc div:1
load:0xfff0018,len:4
load:0x3ff001c,len:1100
load:0x4007800,len:9564
h  tail 12 room 4
load:0x400400,len:6320
entry 0x400806a8

.
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.50.64' to connect
```

Autoscroll [✓] Show timestamp [ ] Newline 115200 baud Clear output

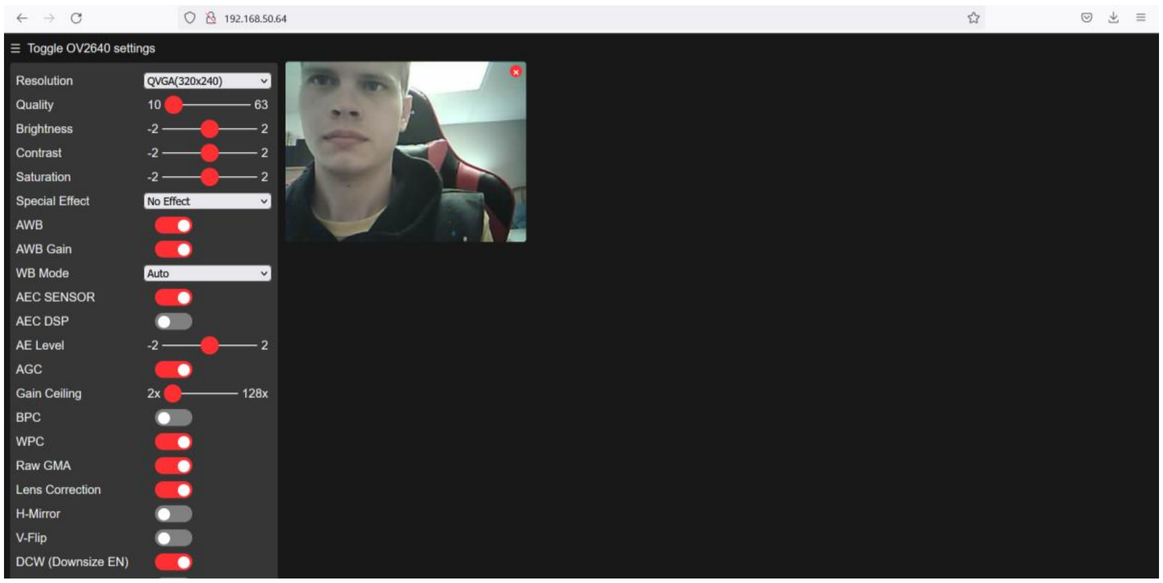
h. Finally, use a device connected to the same Wi-Fi network that matches the SSID and password in the code. Open a web browser and type in the address IP displayed in the serial monitor. You should be directed to a webpage like this:



☰ Toggle OV2640 settings

Resolution	QVGA(320x240)
Quality	10 ● 63
Brightness	-2 ● 2
Contrast	-2 ● 2
Saturation	-2 ● 2
Special Effect	No Effect
AWB	●
AWB Gain	●
WB Mode	Auto
AEC SENSOR	●
AEC DSP	○
AE Level	-2 ● 2
AGC	●
Gain Ceiling	2x ● 128x
BPC	○
WPC	●
Raw GMA	●
Lens Correction	●
H-Mirror	○
V-Flip	○
DCW (Downsize EN)	●

i. Toggle on the “Face Detection” and “Face Recognition” features and click “Start Stream”. A video stream of the device’s camera should appear. If this works correctly, you are ready to start the reverse and re-engineering process on the application!

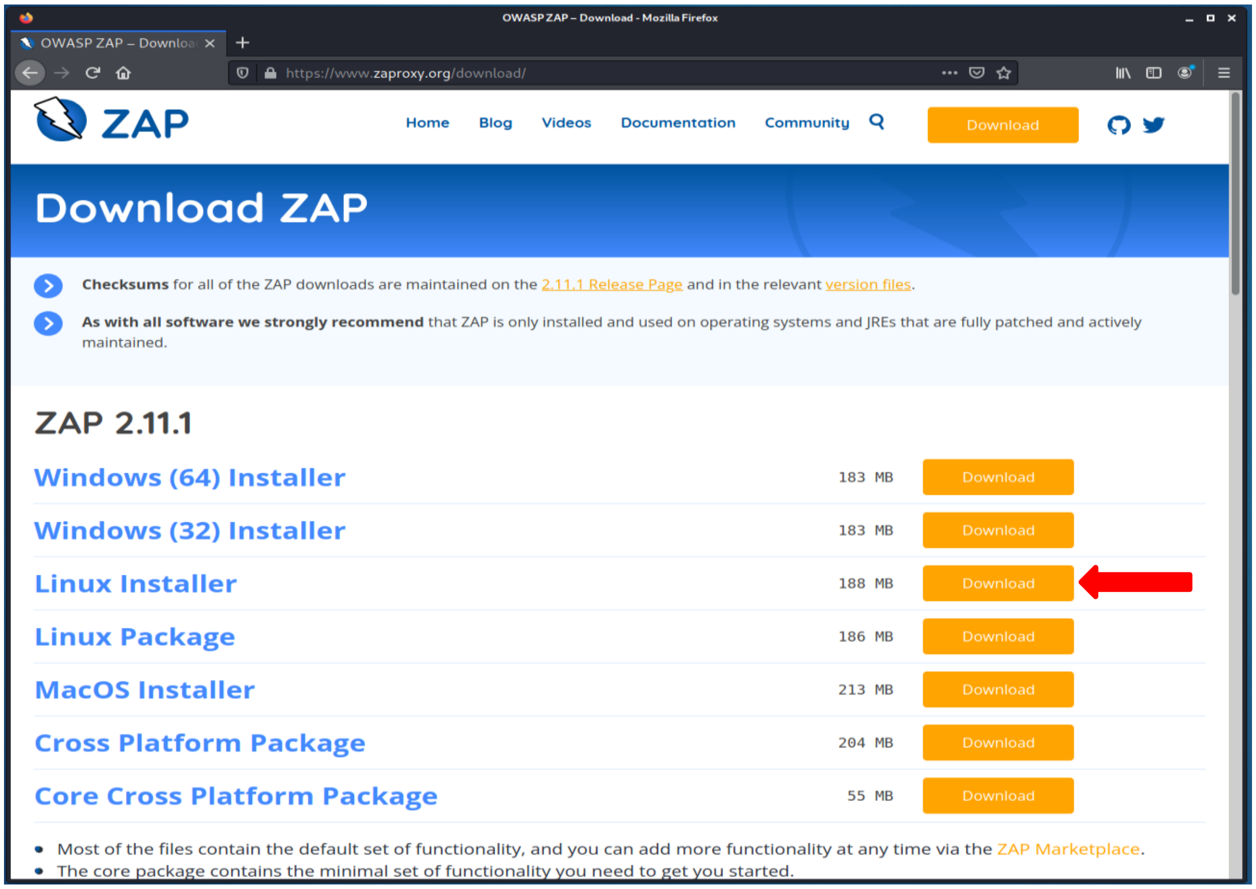


☰ Toggle OV2640 settings

Resolution	QVGA(320x240)
Quality	10 ● 63
Brightness	-2 ● 2
Contrast	-2 ● 2
Saturation	-2 ● 2
Special Effect	No Effect
AWB	●
AWB Gain	●
WB Mode	Auto
AEC SENSOR	●
AEC DSP	○
AE Level	-2 ● 2
AGC	●
Gain Ceiling	2x ● 128x
BPC	○
WPC	●
Raw GMA	●
Lens Correction	●
H-Mirror	○
V-Flip	○
DCW (Downsize EN)	●

3. Install OWASP ZAP on Kali Linux VM

1. Start your Kali Linux VM. Visit this webpage: https://www.zaproxy.org/download/. Click the “Linux Installer” download option:



OWASP ZAP – Download - Mozilla Firefox

https://www.zaproxy.org/download/

Download ZAP

Checksums for all of the ZAP downloads are maintained on the 2.11.1 Release Page and in the relevant version files.

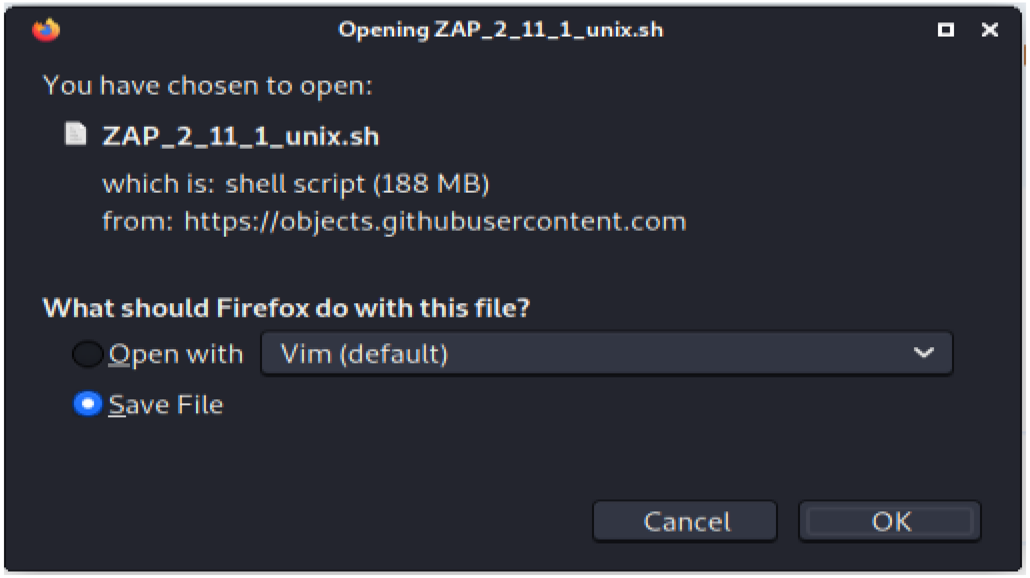
As with all software we strongly recommend that ZAP is only installed and used on operating systems and JREs that are fully patched and actively maintained.

ZAP 2.11.1

Windows (64) Installer	183 MB	Download
Windows (32) Installer	183 MB	Download
Linux Installer	188 MB	Download
Linux Package	186 MB	Download
MacOS Installer	213 MB	Download
Cross Platform Package	204 MB	Download
Core Cross Platform Package	55 MB	Download

- Most of the files contain the default set of functionality, and you can add more functionality at any time via the ZAP Marketplace.
- The core package contains the minimal set of functionality you need to get you started.

2. Save the file when prompted and click "OK"



Opening ZAP_2_11_1_unix.sh

You have chosen to open:

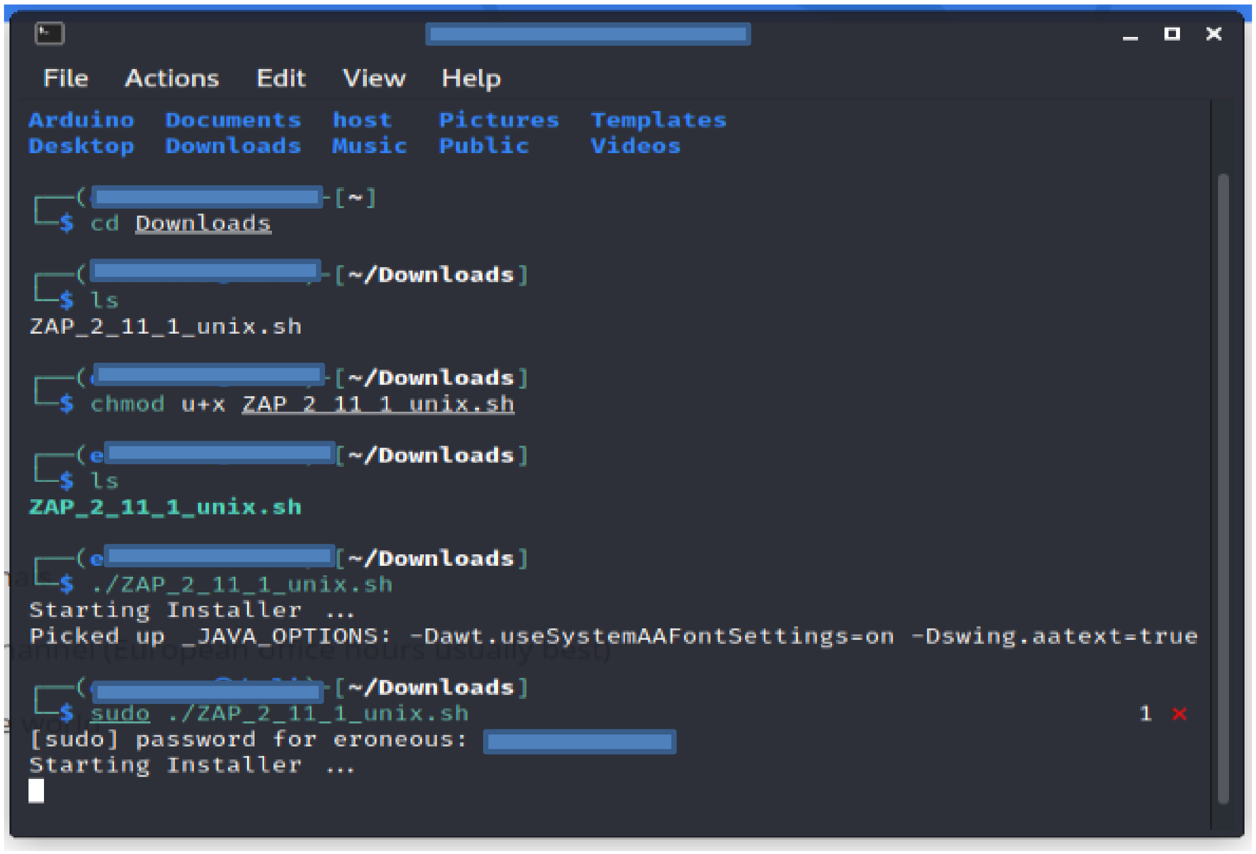
ZAP_2_11_1_unix.sh

which is: shell script (188 MB)
from: https://objects.githubusercontent.com

What should Firefox do with this file?

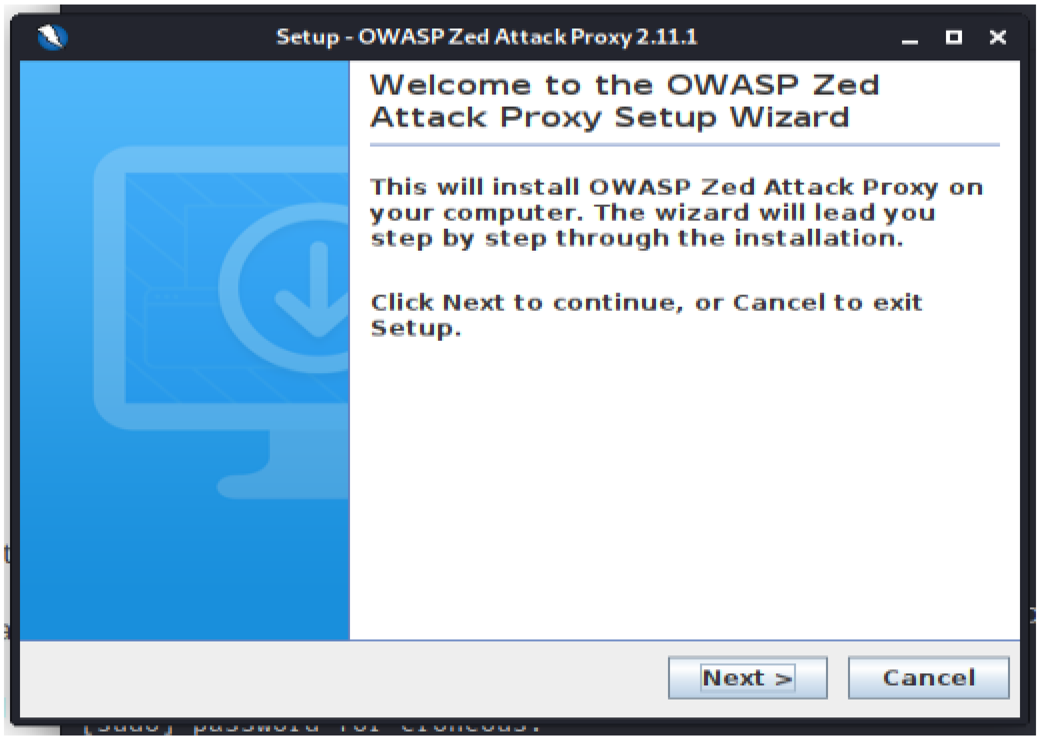
Open with Vim (default)
[✓ Save File]

3. Open a terminal and navigate your downloads folder. Verify the "ZAP_2_11_1_unix.sh" file is there with he "ls" command. Enter "chmod u+x ZAP_2_11-1_unix.sh" to change the file to an executable. Then run the command "sudo ./ZAP_2_11_1_unix.sh" to run the installer:



```
Arduino   Documents   host    Pictures   Templates
Desktop   Downloads   Music   Public     Videos

┌──([redacted])-[~]
└─$ cd Downloads

┌──([redacted])-[~/Downloads]
└─$ ls
ZAP_2_11_1_unix.sh

┌──([redacted])-[~/Downloads]
└─$ chmod u+x ZAP_2_11_1_unix.sh

┌──(e[redacted])-[~/Downloads]
└─$ ls
ZAP_2_11_1_unix.sh

┌──(e[redacted])-[~/Downloads]
└─$ ./ZAP_2_11_1_unix.sh
Starting Installer ...
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

┌──([redacted])-[~/Downloads]
└─$ sudo ./ZAP_2_11_1_unix.sh                                              1 ×
[sudo] password for eroneous: [redacted]
Starting Installer ...
█
```

4. A dialog box will pop up, click next and agree to the terms of agreement:

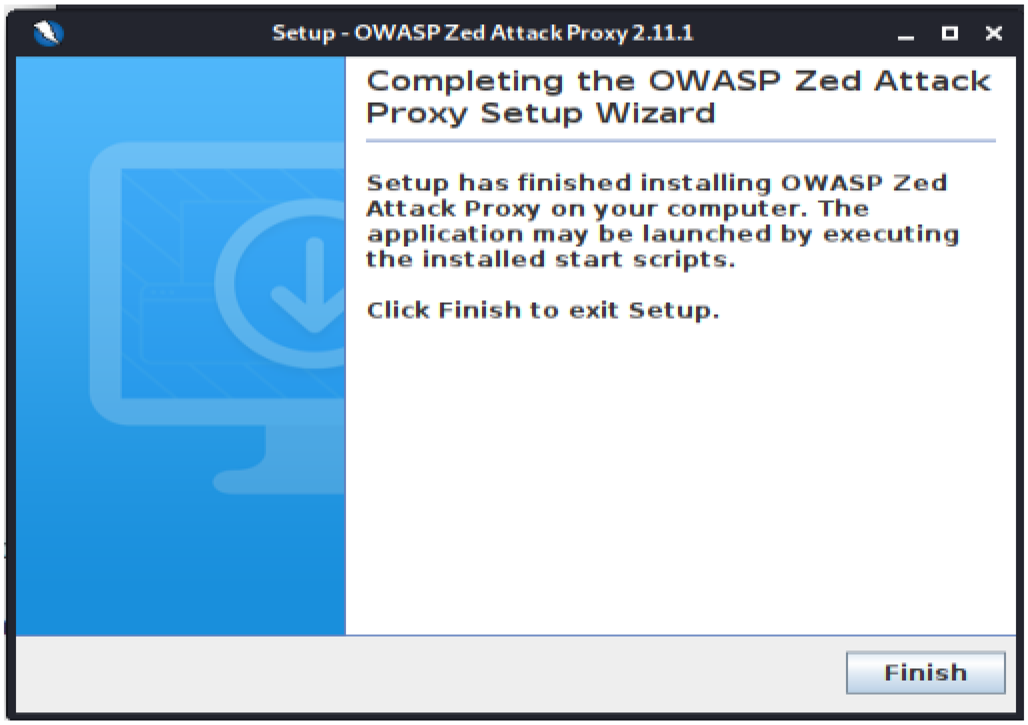


Welcome to the OWASP Zed Attack Proxy Setup Wizard

This will install OWASP Zed Attack Proxy on your computer. The wizard will lead you step by step through the installation.

Click Next to continue, or Cancel to exit Setup.

5. Click "Finish" to finish the installation:

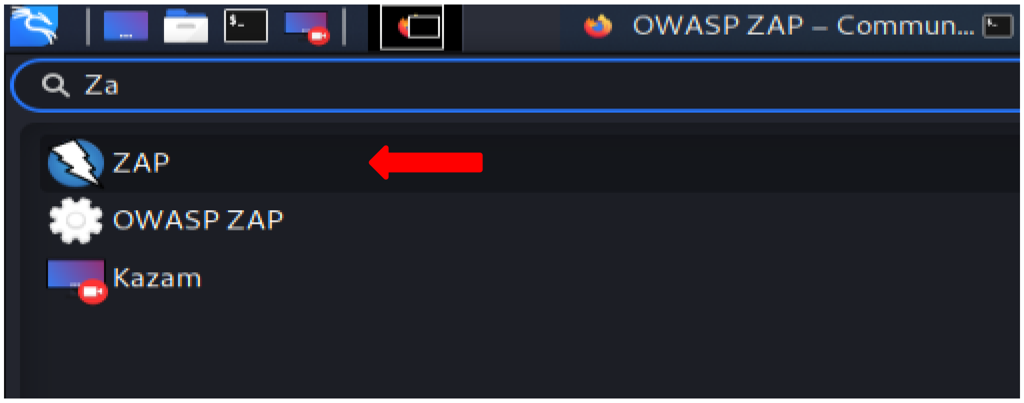


Completing the OWASP Zed Attack Proxy Setup Wizard

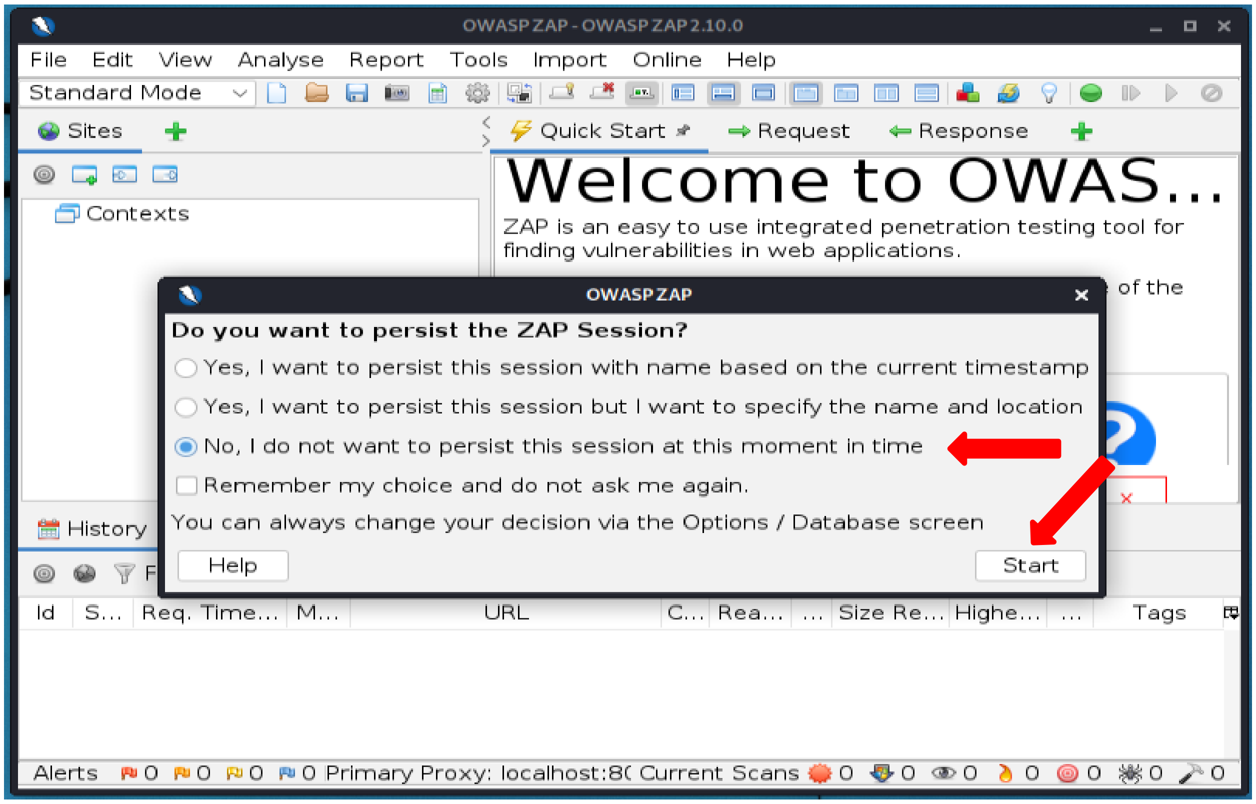
Setup has finished installing OWASP Zed Attack Proxy on your computer. The application may be launched by executing the installed start scripts.

Click Finish to exit Setup.

6. Click the menu in the top left corner and type Zap. You should see ZAP installed on your machine. Click the application to run it:



7. Select no for the persistent ZAP session and click start:


OWASP ZAP - OWASP ZAP 2.10.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Contexts
Quick Start
Request
Response
History
Alerts 0 0 0 0 Primary Proxy: localhost:8(Current Scans 0 0 0 0 0 0 0

Welcome to OWAS...

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

OWASP ZAP

Do you want to persist the ZAP Session?

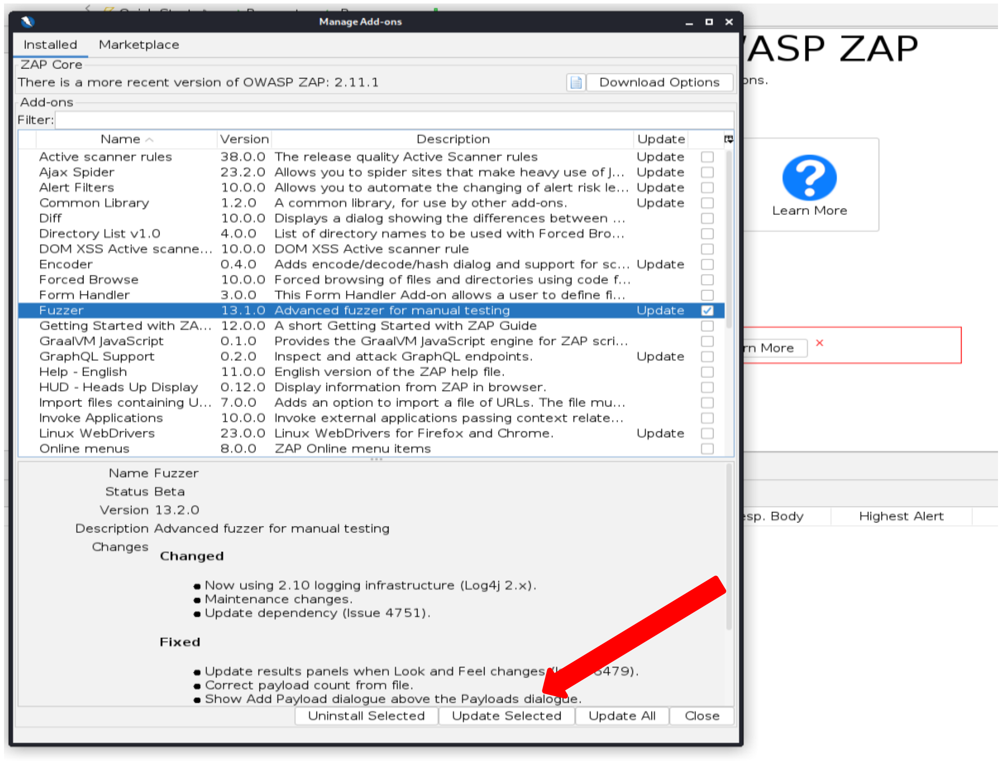
- ( ) Yes, I want to persist this session with name based on the current timestamp
- ( ) Yes, I want to persist this session but I want to specify the name and location
- (●) No, I do not want to persist this session at this moment in time
- ☐ Remember my choice and do not ask me again.

You can always change your decision via the Options / Database screen

Help | Start

Id	S...	Req. Time...	M...	URL	C...	Rea...	...	Size Re...	Highe...	...	Tags

8. A manager add on box may pop up. If so, select the "Fuzzer" option and then click "Update Selected":



ZAP Core

There is a more recent version of OWASP ZAP: 2.11.1 | Download Options

Add-ons

Filter:

Name	Version	Description	Update	
Active scanner rules	38.0.0	The release quality Active Scanner rules	Update	☐
Ajax Spider	23.2.0	Allows you to spider sites that make heavy use of J...	Update	☐
Alert Filters	10.0.0	Allows you to automate the changing of alert risk le...	Update	☐
Common Library	1.2.0	A common library, for use by other add-ons.	Update	☐
Diff	10.0.0	Displays a dialog showing the differences between ...		☐
Directory List v1.0	4.0.0	List of directory names to be used with Forced Bro...		☐
DOM XSS Active scanne...	10.0.0	DOM XSS Active scanner rule		☐
Encoder	0.4.0	Adds encode/decode/hash dialog and support for sc...	Update	☐
Forced Browse	10.0.0	Forced browsing of files and directories using code f...		☐
Form Handler	3.0.0	This Form Handler Add-on allows a user to define fi...		☐
Fuzzer	13.1.0	Advanced fuzzer for manual testing	Update	✓
Getting Started with ZA...	12.0.0	A short Getting Started with ZAP Guide		☐
GraalVM JavaScript	0.1.0	Provides the GraalVM JavaScript engine for ZAP scri...		☐
GraphQL Support	0.2.0	Inspect and attack GraphQL endpoints.	Update	☐
Help - English	11.0.0	English version of the ZAP help file.		☐
HUD - Heads Up Display	0.12.0	Display information from ZAP in browser.		☐
Import files containing U...	7.0.0	Adds an option to import a file of URLs. The file mu...		☐
Invoke Applications	10.0.0	Invoke external applications passing context relate...		☐
Linux WebDrivers	23.0.0	Linux WebDrivers for Firefox and Chrome.	Update	☐
Online menus	8.0.0	ZAP Online menu items		☐

Name Fuzzer
Status Beta
Version 13.2.0
Description Advanced fuzzer for manual testing
Changes

Changed

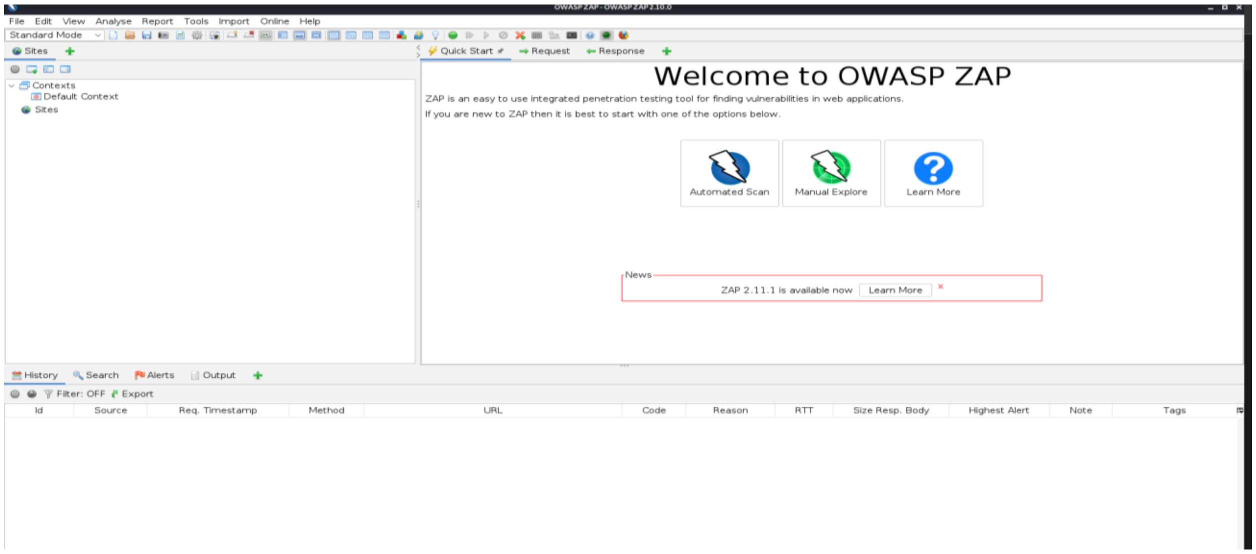
- Now using 2.10 logging infrastructure (Log4j 2.x).
- Maintenance changes.
- Update dependency (Issue 4751).

Fixed

- Update results panels when Look and Feel changes (I 6479).
- Correct payload count from file.
- Show Add Payload dialogue above the Payloads dialogue.

Uninstall Selected | Update Selected | Update All | Close

9. Now the OWASP ZAP tool should be ready to use, and it should look something like this:



Welcome to OWASP ZAP

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

If you are new to ZAP then it is best to start with one of the options below.

ZAP 2.11.1 is available now Learn More

Id	Source	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags

10. With this installed, students should be read to follow the detailed instructions outline in the paper to practices the vulnerability detection via reverse engineering and re-engineering.

Recommendations for Developing More Usable and Effective Hands-on Cybersecurity Education Materials Based on Critical Evaluation Criteria

Ahmed Ibrahim
aibrahim@pitt.edu
University of Pittsburgh
Pittsburgh, PA

Vitaly Ford
fordv@arcadia.edu
Arcadia University
Glenside, PA

Abstract

Effective cybersecurity education requires offering hands-on exercises in addition to lecture-based learning. The study provides insights into the challenges of cybersecurity hands-on education and offers a pathway for developing better cybersecurity educational materials. The fast-paced nature of the cybersecurity field makes it difficult for educators to keep up and create realistic exercises. Hence, there is a need for a common framework that would enable educators to produce more usable and effective cybersecurity hands-on educational resources, avoiding the reinvention of the proverbial wheel. In this work, we developed criteria to categorize and evaluate existing cybersecurity education resources based on their technical and educational characteristics. We analyze and evaluate four existing cybersecurity resources and provide critical remarks on their usability and effectiveness. Finally, we propose recommendations for developing new cybersecurity labs or exercises as well as designing cybersecurity platforms.

Keywords: cybersecurity, hands-on education, usability, effectiveness, evaluation, criteria.

1. INTRODUCTION

Providing effective cybersecurity education is challenging since it requires a lot of hands-on exercises, not just lecture-based learning. Additionally, developing students' competitive skills in cybersecurity puts educators in a position of creating (and often re-creating) a large variety of hands-on materials to keep up with the fast pace of this ever-growing field, which takes time and requires significant computational resources.

Many of the existing hands-on cybersecurity education resources exemplify high-quality learning materials and tools. However, there is no unified structure nor defined order allowing educators to use them seamlessly and coherently in their courses. Having a cybersecurity education framework for such resources would make it simpler to find and use what is needed in specific courses as well as determine the support structure required for efficiently and successfully deploying those resources in the classroom. As a result, we believe that educators should unite their efforts to design usable and effective cybersecurity education materials based on robust standardized criteria.

Developing and deploying real-life cybersecurity scenarios in academia take more time than regular assignments/projects for other types of courses. As educators design their assignments and labs, one element they should ponder about is how to minimize frustration and maximize active learning. For instance, a considerable amount of time is dedicated to exercise setup and troubleshooting rather than spending time on actual learning. An educator may spend 30 minutes setting up a 5-minute attack demonstration. At the same time, if a lab requires a lot of setup or is somewhat vague, it may deter students from learning because it does not work as it should or does not work at all. Additionally, the fast-paced cybersecurity and technological landscape makes it infeasible for educators to keep up and develop realistic exercises for their students as it often requires them to revamp all exercises every semester. As a result, hands-on cybersecurity education is behind a lot of other STEM education areas.

In this study, we investigate known available resources/environments, assess them, share our experience based on their educational and technical pros/cons, provide recommendations for everyone, and then share our vision on what criteria need to be there for exercises and platforms to be effective. The primary goal of this project is to make cybersecurity exercises consistent and better. Critiquing the projects is part of our analysis that is not meant to downplay the importance and usefulness of the projects but is rather meant to help the reader understand the existing challenges with such projects.

2. OBJECTIVE AND METHODOLOGY

Our objective is to develop a framework for producing more *usable* and *effective* cybersecurity hands-on educational resources. In the context of this work, *usability* is mainly related to technical characteristics (e.g., customizable labs and lab access method) and *effectiveness* is mainly related to educational characteristics (e.g., clear instructional materials, learning objectives, and progress tracking).

To achieve our objective, we developed criteria (Appendix A) to categorize and evaluate (determining the instructional value) existing cybersecurity education resources. At the beginning of the criteria development process, we compiled a list of questions to ask before considering a new resource:

- Does the education resource provide enough support such that the instructor finds it easy to use in their class?
- Does it clearly state learning objectives?
- Does it provide supplemental material (e.g., network map)?
- Does it contain relevant content?
- Does it come with an instructor's manual?
- Does it have a grading rubric?
- Does it include an instructor answer key?
- Is it simple to deploy, run, and administrate exercises?
- Is it modular, allowing for mix and match / plug and play / not-sequential exercise completion? Does it allow instructors with different special areas to find what they need? Is each exercise/lab an independent unit and it does not necessarily depend on finishing the one before it? Does the project allow users to put the available exercises/labs in the order they need them to be rather than enforcing a particular order for using the labs?
- Does it challenge students to complete the exercise or merely hand-holds them to follow the step-by-step instructions?
- Is there a way to assess the student's learning?
- Does it provide the instructors with assessment tools/measures?

Based on these questions, we identified specific criteria metrics described in section 3. We then conducted a critical review of each of the resources' usability and effectiveness characteristics based on our experience and following the developed criteria. We used the collective knowledge from our experience to design the pathway for more usable and effective materials.

The rest of this paper is outlined as follows. In section 3, we list the developed evaluation criteria. Section 4 critically evaluates several existing environments according to the developed evaluation criteria. Section 5 discusses some of the other known cybersecurity projects that are not included in this study. Section 6 introduces the proposed recommendations. Section 7 presents the conclusion and the future direction of cybersecurity education resource development.

3. EVALUATION CRITERIA

For evaluation to be useful, it must be based on well-developed criteria. To accomplish that, we first introduce the following two categories: usability and effectiveness. Under each category, we list several criteria that are either quantitative or have a simple answer (e.g., yes/no). Each criterion will have a number, a label, possible answers, and a description. Due to the difficulty of objectively measuring qualitative criteria without introducing an opinionated bias, we minimized the number of qualitative criteria. However, more qualitative criteria will be introduced in a later work after polling the broader community as will be mentioned in the Future Directions section.

Usability Criteria
In this subsection, we list four criteria developed under the usability category: (C1) type of labs, (C2) customization possibility, (C3) access method (in the form of 3 sub-criterion: C3-A, C3-B, and C3-C), and (C4) level of support.

Number: C1
Label: Type of labs
Possible Answers: Stand-alone / Connected machines / Both
Description: This criterion lists the type of labs available on a provided platform. The "Stand-alone" type describes the exercises/labs in the platform that can be executed on a single operating system without any need to connect to another OS using network communication. The lab may be done on a virtual machine or may represent a set of instructions that a student can perform on their own computer. The "Connected-machines" type implies that labs require the involvement of at least two computer/virtual machines and a computer network. The machines required can be either hosted on the cloud and available online or downloadable virtual machines with setup instructions. The "Both" type implies that the project offers "Stand-alone" and "Connected-machines" labs.

Number: C2
Label: Customizable Labs
Possible Answers: Yes / No
Description: This criterion lists whether a project enables instructors to customize the lab environment according to the instructor's needs or not. In the case of "Stand-alone" labs, this may be adding, editing, or removing services (either provided by the project or the instructor) in any given system. In the case of "Connected-machines" labs, this may be adding, editing, or removing virtual machines (either provided by the project or the instructor) on the network.

Number: C3-A
Label: Cloud-based (the whole education resource is accessible through a web browser)
Possible Answers: Yes / No
Description: This criterion lists whether a project enables instructors to access and use the education resource through a web browser or not. "Yes" means using the resource does not require downloading any virtual machines or software by instructors or students. "No" means the project requires some sort of downloading, configuring, or installing software that is dependent on some specified requirements (e.g., hardware or operating system).

Number: C3-B
Label: Lab access (for cloud-based projects)
Possible Answers: SSH Only / Web Interface Only / Both
Description: This criterion lists whether the labs are accessible via SSH (terminal), a web browser, or both.

Number: C3-C
Label: Setup guidelines (for downloadable material)
Possible Answers: Yes / No
Description: This criterion lists whether projects providing the downloadable material include directions and guidelines pertaining to the setup process.

Number: C4
Label: Level of support
Possible Answers: Institutional / Individual / None
Description: This criterion lists the level of support provided to instructors and students by the project. An "institutional" support implies that the project has some form of a ticketing/helpdesk system to report problems, ask questions, and get support (e.g., in case of network failure, environment not being available, or account problems). An "individual" support implies that the project is supported by a single individual via email or a form fill-out. None implies that there is no clear way for instructors and students to ask questions or get support.

Effectiveness Criteria
In this subsection, we list eight criteria developed under the effectiveness category: (C5) instructor's manual availability, (C6) student instructions availability, (C7) includes learning objectives, (C8) mapping to frameworks, (C9) limitations, (C10) progress tracking, (C11) time tracking, and (C12) accessibility level.

Number: C5
Label: Instructor's Manual Availability
Possible Answers: Yes / No / Partial
Description: This criterion identifies if a project provides an instructor's manual to help instructors understand and prepare for the material (i.e., exercises or labs). "Yes" means that all exercises have instructor's manuals. "No" means that none of the materials have instructor's manuals. "Partial" means that some, but not all, of the material has instructor's manuals or that instructor's manuals are partially incomplete.

Number: C6
Label: Student Instructions Availability
Possible Answers: Yes / No / Partial
Description: This criterion lists if a project provides instructions for students on how to walk through the material. The instructions can be either step-by-step or general guidelines on how to complete the labs. "Yes" means that all labs have student instructions. "No" means that none of the materials have student instructions. "Partial" means that some, but not all, of the materials have student instructions.

Number: C7
Label: Includes Learning Objectives
Possible Answers: Yes / No / Partial
Description: This criteria lists if the project materials include clear learning objectives. "Yes" means that all materials have learning objectives. "No" means that none of the materials have learning objectives. "Partial" means that some, but not all, of the material have learning objectives.

Number: C8
Label: Mapping to Frameworks
Possible Answers: NICE KSAs / CAE KUs / Both / None
Description: This criterion lists the cybersecurity educational frameworks which the project uses for mapping its materials. At the time of writing this article, the NICE KSAs (Petersen et al., 2020) and CAE KUs (CAE Documents Library, n.d.) are the two widely adopted cybersecurity education frameworks.

Number: C9
Label: Limitations
Possible Answers: Resources / Time / Both / None
Description: This criterion identifies the project's limitations. The "Resources" limitation can be a limit on the number of students running an exercise at any given time, a capacity per account/course (e.g., a lab may state that no more than 10 users can have access at the same time or only 16 machines are available for provisioning), or the necessity to have students download and install/import one or more virtual machines. The "Time" limitation can be a limit on the period of usage (e.g., resources are only available for two days).

Number: C10
Label: Progress Tracking
Possible Answers: Yes / No
Description: This criterion identifies projects that track students' progress on the assignment and allows instructors to view it.

Number: C11
Label: Time Tracking
Possible Answers: System / Lab / Both / None
Description: This criterion lists projects that track students' time and allow instructors to view it. The "System" time tracking implies that the project tracks the total time students have spent on the platform. The "Lab" time tracking implies that the project tracks the time students have spent on a specific lab.

Number: C12
Label: Accessibility Level
Possible Answers: Nationwide / Limited / Paid
Description: This criterion identifies the accessibility level of a project. The "Nationwide" level means it is accessible (free of charge) to any educational institution in the United States. The "Limited" level means it is accessible (free of charge) to a certain population (e.g., only Virginia State institutions). The "Paid" level means it is accessible for anyone who pays a fee (varying by the material provider).

4. APPLYING THE CRITERIA THROUGH CRITICAL EVALUATION

One of the struggles that educators face is developing labs and exercises that would not hand-hold students but instead would promote discovery and self-learning and open the opportunity to make mistakes without affecting grades. Cybersecurity is a field where technical knowledge is closely interleaved with theoretical foundations. Thus, it is challenging to determine a balance between how much information is enough and how much information is too little or too much for students to complete an exercise and facilitate learning.

In this section, we analyze and evaluate four existing cybersecurity resources (based on the evaluation criteria defined in the previous section) providing critical remarks on their usability and effectiveness. We realize that it is challenging to measure the materials' value, hence, we follow up with a discussion and recommendations section about educational and technical pros/cons according to our experience. In Appendix A, we include a table that shows which criteria are included in each of the four major hands-on educational resources evaluated in this work.

DETERLab

DETERLab (DETER Project, n.d.) (Mirkovic & Benzel, 2012) is a cluster environment focusing on allowing researchers and instructors to deploy cybersecurity experiments with custom network configurations to investigate cyber attacks and defenses. DETERLab aims to provide an active learning scalable platform, a large number of computing resources, exercise setup automation, as well as access to reusable and modular experiments. In 2012, DETERLab was used by over 47 universities and colleges and had more than 400 general-purpose computing nodes (Mirkovic & Benzel, 2012). As of December 2016, DETERLab users have created 192 projects for their classes and DETERLab has served 13,000 students (DETERLab, n.d.). We decided to evaluate DETERLab's usability and effectiveness in educational settings, providing practical recommendations for improvement, because we believe that the platform has the potential and clear direction if it is made more adoptable.

Personal Experience
The first major struggle we faced was related to a confusing, outdated user interface (UI). It was challenging to navigate the platform, find where and how to start, and figure out how to add experiments to our project/class. Tabs on the homepage were not consistent. We did not see an "Experiments" tab until we figured out how to add our first experiment -- and only then did a new tab called "Experiments" appear on the instructor's page. It was difficult to find an answer to our questions using the site's Wiki since the search feature was not giving the correct results. Eventually, we had to use an external search engine to actually look things up on the Wiki. Overall, the Wiki did not seem to be written with the end-user in mind.

Labs are accessible using nested SSH connections to connect to the remote hosts which makes navigation between the hosts on the network confusing. The upper right corner of the DETERLab website showed the number of available PCs out of 691 PCs in total. The number of PCs freely available for deployment has been very low on a daily basis during the spring of 2020 (under 100) which poses a serious scalability challenge. During the spring of 2022 we found out that the total number of available PCs went down from 691 to 360 due to the retirement of the Berkley DETERLab site administrator (Figure 1).

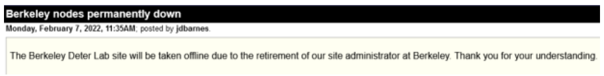

Berkeley nodes permanently down
Monday, February 7, 2022, 11:35AM; posted by jdbarnes.
The Berkeley Deter Lab site will be taken offline due to the retirement of our site administrator at Berkeley. Thank you for your understanding.

Figure 1 DETERLab Berkeley nodes are down

The DETERLab project allows for lab customization (DETERLab's custom NS syntax) but there is a significant learning curve associated with the customization process. The DETERLab project has a simple FAQ page, a wiki, and a ticketing system. Based on their publicly available ticket information (DETERLab Ticket System, n.d.), some tickets are addressed within days while others may require follow-ups and take months.

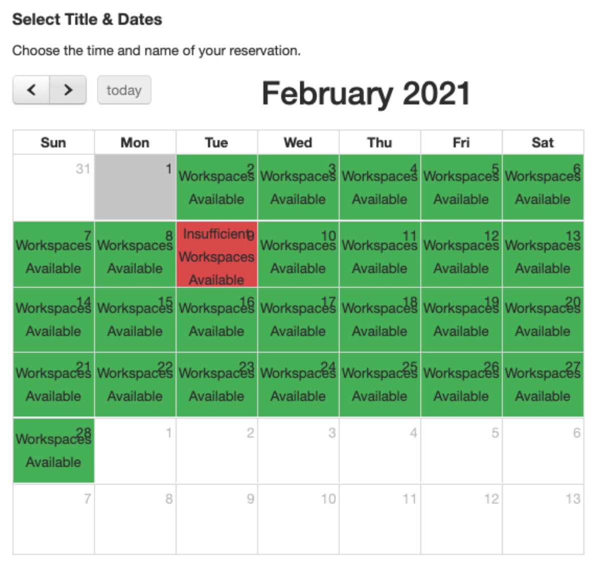
More information about the DETERLab experience, the sign-up process, steps required, etc. can be found here (Ibrahim & Ford, 2021).

Recommendations
We recommend the DETERLab project to restructure and redo the Wiki, make the website usable and user-friendly, make homework and teacher manuals consistent, and provide training videos/classes for instructors. In addition, we think that offering an online training module (e.g., tutorial videos) to use DETERLab for all new users and publishing user reviews provides transparency and makes the material reliable, hence improving the material over time. And for those interested in using DETERLab we recommend expecting to put in significant effort and spend a considerable amount of time familiarizing yourself with how things work.

NICE Challenge
The NICE Challenge Project (NICE Challenge, n.d.) allows educators to use real-world virtualized business environments to teach cybersecurity. It contains over 100 different labs including defense, offense, server administration, configuration, setup, auditing, logs, malware, and other topics. At the time of writing this work, the NICE Challenge served more than 550 institutions and 1,000 faculty with 3,000 virtual machines/day and more than 150,000 total workspaces deployed. All NICE challenges are mapped against the NICE Framework's KSAs (Petersen et al., 2020) and CAE KUs (CAE Documents Library, n.d.). The project is available at no cost to educational institutions, provides training for new educators, has a support portal, requires only a web browser, and has a ticketing system to provide feedback and request support. The NICE Challenge Project evaluated according to the Usability and Effectiveness Criteria can be found in Appendix A.

The NICE Challenge project has a strategy guide that only includes a short explanation of the objectives and does not include any instructions on how to perform the tasks or reach the goal. From an instructional perspective, it does not provide the instructors with any useful information to help them be prepared for the challenge. The NICE Challenge project provides students with a narrative-driven scenario, a workspace, and a set of technical objectives and/or a written deliverable, but it does not tell students how to complete the challenge and reach the technical objectives and/or a written deliverable.

The NICE Challenge project does not allow challenges to be available immediately. Instructors (a.k.a. curators) have to reserve pods for their students. Each reservation is limited to two consecutive days. And, each instructor has a limited number of seats to use for the reservations. Reservations must be requested by the instructor at least one day prior to the beginning of the required reservation day. In some cases, some days are not available due to insufficient workstation availability as shown in Figure 2.


Select Title & Dates

Choose the time and name of your reservation.

today

February 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2 Workspaces Available	3 Workspaces Available	4 Workspaces Available	5 Workspaces Available	6 Workspaces Available
7 Workspaces Available	8 Workspaces Available	9 Insufficient Workspaces Available	10 Workspaces Available	11 Workspaces Available	12 Workspaces Available	13 Workspaces Available
14 Workspaces Available	15 Workspaces Available	16 Workspaces Available	17 Workspaces Available	18 Workspaces Available	19 Workspaces Available	20 Workspaces Available
21 Workspaces Available	22 Workspaces Available	23 Workspaces Available	24 Workspaces Available	25 Workspaces Available	26 Workspaces Available	27 Workspaces Available
28 Workspaces Available	1	2	3	4	5	6
7	8	9	10	11	12	13

Figure 2 NICE Challenge workspace availability

The NICE Challenge project has an FAQ section and a helpdesk. Based on our experience, they respond very quickly (within 24 hours) to new tickets submitted through the helpdesk site.

The NICE Challenge project does not track the students' progress per challenge. However, it divides challenges into multiple checkpoints and it has automatic triggers to know whether a student was successful to reach the end goal for each checkpoint or not. It does not track the students' actions from when they start working on the checkpoint until they encounter an issue or complete the challenge. The NICE Challenge project provides information about the amount of time it took the student to complete each challenge checkpoint (Figure 3).

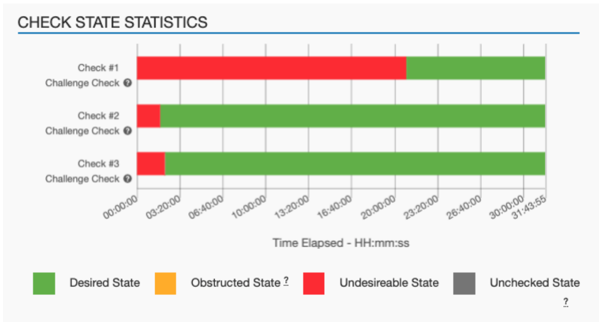


Figure 3 NICE Challenge student statistics

Personal Experience

The account application process is straightforward. The application is available on the main webpage and requires the instructor's name, EDU email, and their plan to use the challenges. When we applied to use the NICE Challenge platform, our application got reviewed and approved within a couple of days. It was simple to find out how to create a reservation. Once a reservation was active, it was easy to begin a lab and try it out.

Before using the challenges in the classroom, we had to go through the labs ourselves as there are no instructor's manuals for any lab. Most of the time, we experienced roadblocks and we had to reach out to the support team for clarifications. The support team promptly responded and our issues were resolved in a timely manner, even during weekends. After our confusion was resolved, it was straightforward to adopt the labs and provide hints and instructions to our students. We used it in the classroom and to host a cybersecurity awareness month competition. Students who participated in the competition expressed that they especially enjoyed the real-world, scenario-based experience.

There are many advantages to using the NICE Challenge project. It has a large variety of labs that helps educators find the right fit on a specific topic for their students to experience the practical application of theoretical knowledge that is often covered in cybersecurity courses. The labs are simple to deploy, provide an engaging scenario for students while they are waiting for the virtual environment deployment, and can be used by all students at the same time since the reservation is already made. A demo challenge is available that helps students understand how the platform operates and how to submit a challenge. In addition, the instructor can access students' deployments (GUI in the browser) and look at their machines which is especially beneficial when troubleshooting or providing support to students. It can be used for developing skills within a course and to train/prepare students for attack and defense competitions. Instructors can also submit a "Challenge/Feature Request" through the project's ticketing system.

The NICE Challenge project also has some limitations that instructors should be aware of. Instructors cannot add students to a reservation if it has already started. A student cannot deploy more than one challenge at a time. The lab reservations are limited to two consecutive days, and the instructors have a limited number of seat credits available to use for reservations. An instructor's manual is not available, making it challenging for instructors to support their students when they encounter difficulties. Also, some challenges may be harder than they seem to be and some may not be working correctly. Thus, instructors must go through the challenges and ensure the "checks" (objectives) work correctly before assigning any of the challenges to their students. Finally, if an instructor has a teaching assistant (TA) added to the portal as an overseer to grade submissions, the TA must be the one who creates the reservations in order to be able to view student submissions. The instructor cannot allow an overseer (e.g., TA) to view submissions made for reservations that were created by the instructor.

Recommendations
We hope that in the future the NICE Challenge would allow educators to reserve available pods for more than just 2 consecutive days. Also, it would be beneficial to have a mechanism to generate a custom environment. A repository of challenge instructions would help educators in adopting the challenges and assisting students when they get stuck. Consequently, hints would be a useful feature to add. Instructors should have the option to allow TAs to view any reservation created at any point as well as submissions for reservations created before the TA was assigned to the class.

SEED Labs
The SEED project (SEED Labs, n.d.) started in 2002 by Kevin Du and has been growing since then. The SEED project's objective is to develop hands-on laboratory exercises (called SEED labs) for computer and information security education and help instructors adopt these labs in their curricula. As of 2021, the project has been funded by a total of 1.3 million dollars from NSF, and is now used by over a thousand educational institutes worldwide.

The SEED project consists of four main elements: Labs, Books, Lectures, and Workshops which cover topics such as computer and information security, cryptography, software security, network security, web security, operating system security, and mobile app security.

The *labs* are hosted as downloadable virtual machines that instructors and students would deploy themselves on their local computers. Lately, there has been an effort to make virtual machines available on cloud platforms (e.g., AWS, Google Cloud). However, the instructors would be responsible for any associated fees to host and operate the virtual machines or they would need to join a cloud-based Educate Program (e.g., AWS Educate) providing free credits for students and instructors. Additionally, instructors can receive lab manuals via email after providing evidence that they are the instructors of the course where the labs are going to be incorporated.

The *books* cover computer/Internet security topics and include problem sets associated with the hands-on labs allowing students to practice and learn both theoretical and practical cybersecurity paradigms. Slides and solution manuals are freely available in an electronic format for instructors upon request.

The *lectures* are recorded on the Udemy platform as two separate courses, namely Computer Security and Internet Security, and can be accessed for a fee. As of the time of writing this article, the SEED website provides a Udemy coupon to receive discounted access to the recorded lectures.

The *workshops* provide training to instructors who are interested in using SEED labs in their courses. They have been offered annually every summer since 2015, free of charge to accepted instructors. The SEED labs project according to the Usability and Effectiveness Criteria can be found in Appendix A.

Personal Experience
Downloading the SEED virtual machines and adopting the labs is simple and available for anyone to use under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The project provides detailed setup instructions on how to get started, download, import, and configure the SEED virtual machine. It is possible to customize the labs but instructors would need to do it on their own without the involvement of anyone from the SEED Project team. The lab manuals are made with the end-user in mind, assuming zero previous knowledge. The labs' documentation is very

illustrative, including screenshots and code snippets describing every configuration step in detail, a troubleshooting section, and a guide on related topics. Additionally, we found it beneficial that each lab includes suggested supervised and unsupervised times that students should typically spend doing the lab.

SEED labs (version 2.0) provide 9 software security, 10 network security, 3 web security, 2 hardware security, 8 cryptography, and 2 mobile security labs. Some of the labs (e.g., web security) can be done using only one copy of the SEED virtual machine, whereas other labs require cloning the VM to one or two additional copies in order to go through the lab. The instructors should ensure beforehand that students' hardware supports virtualization and has enough hard disk space (at least 10 GB per VM) and RAM (at least 2 GB of RAM available per VM). In regards to support, we received timely answers to our questions from the PI of the SEED project.

Recommendations
The SEED labs are very stable and are being used by many institutions all over the world. One major topic we recommend everyone investigate is finding a simple way to host the SEED VM on the cloud to minimize the overhead of requiring one VM per student. For instance, the SQL injection lab can be done by hosting a centralized version of the VM and all students need to only use the browser to do the lab rather than having all students install and download the VM to do that lab. Lately, the SEED project started providing instructions on how to deploy the VMs in commercial cloud systems.

Another important topic is creating a web-portal (forum) for instructors and users to connect and share knowledge. Since the SEED project is individually supported, developing an online forum/blog can be helpful to connect all those who use the SEED labs. Instructors who adopt the SEED labs may need to restructure and clarify the submission guidelines and deliverables that students need to follow as well as include a grading rubric, if necessary. Additionally, we believe that mapping the labs to the major cybersecurity education frameworks would be of high benefit to the instructors.

EDU Range
EDURange (Boesen et al., 2014) is an NSF-funded project "providing hands-on exercises, a student-staffed help-desk, and webinars". Initially, EDURange was a cloud-based platform hosted on AWS. But currently, EDURange is no longer a cloud-based project and the code is provided for instructors to host it on their own cloud or servers. The exercises' goal is to allow faculty with little prior background to teach security and increase the number of schools teaching cybersecurity concepts. The gamified exercises (called *scenarios*) are open-source and available on GitHub. Based on their project's website, there are 8 scenarios currently available on the platform, namely: getting started, file wrangler, SSH inception, total recon, strace, ELF infection, treasure hunt, and metasploitable. These scenarios go over learning about the basics of using the Linux command line, permission loopholes in Linux, nested SSH, executable file examination using strace, SQL and XSS injections on a web app, nmap scanning, files and directories in Linux, basics of metasploit on a widely used image of Metasploitable2 (Linux-based), and infected binaries.

Personal Experience
When we started investigating how to use the EDURange project, we found that a server is required to host the scenarios but there are no server specifications provided. Thus, we created an Ubuntu 20.04 Virtual Machine with 6 GB RAM, 2 CPU cores, and 16 GB of storage on an ESXi virtualization server. To deploy the environment on the server we used the commands in the EDURange GitHub repo which went well with a few hiccups. After our first failed deployment attempt, we contacted EDURange support about our deployment process and they updated the repo with commands that worked the second time we attempted the deployment process. Afterward, we were able to create one administrative account and two student accounts on the platform for testing. However, we encountered issues with the lack of hardware specifications. The first scenario we chose to deploy was Metasploitable which constantly failed to deploy. After tracing the terminal output, we found that the scenario was not progressing. Given that we knew that Metasploitable requires a large storage space, we checked the remaining storage space to find out that we ran out of space on the server. We had to extend the disk storage from 16 GB to 40 GB and then redeploy the scenario.

It is of major importance to set up the EDURange *.env* file correctly from the first time. That file will include the hosting server's hostname or IP address which will be used in every scenario to be deployed. When we set up the *.env* file incorrectly during our first deployment, there was no available option to update the values after the server was started. To resolve the error resulting from a misconfiguration in the *.env* file, we had to completely destroy the first deployment and start a totally new one with the correct configuration.

Another issue we encountered is that student accounts cannot access a scenario if the scenario was created prior to the student registrations. Hence, students must already be in the system before the scenario is assigned to the students' group and later deployed. Otherwise, the students would see an error message if they try to join that scenario even after rebooting the server.

An additional challenge we encountered is that a scenario may not deploy correctly if the name includes a space in it. Some scenarios (e.g., Elf, strace, and WebFu) did not deploy at all or were deployed with errors at the time of writing this work.

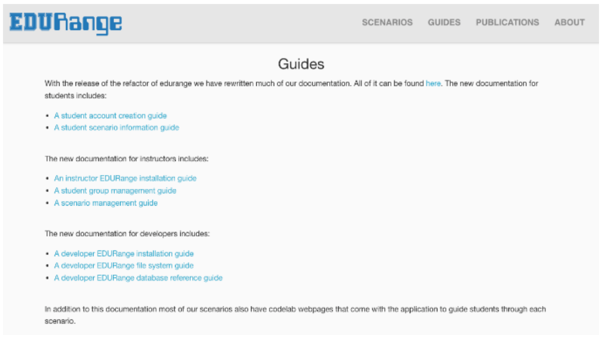
Almost every scenario has some inconsistency in it. For example, in the File Wrangler scenario, the task numbering in the student guide does not align with the actual numbering of the questions that the students should answer (e.g., the guide for task #4 will cover question #4 and #5, then task #5 will cover question #6) which can be confusing. In addition, all the steps in the task guide were numbered as "1" instead of having a sequential numbering. We experienced many inconsistent or wrong formatting in different scenarios. Also, student guidelines are not complete and do not map properly to the task(s) required to finish a scenario.

Another example of an inconsistency is that question #6 in the Getting Started scenario asks for six file names of image files. However, there is only one textbox to enter the answers. The question mentions entering each filename separately. We found this confusing and it would have been better if the environment included an unambiguous way to accept the answers.

In addition, the order of the deployed scenarios displayed in a table format (which includes the buttons to start/stop/destroy a scenario) on the administrative dashboard keeps changing in real-time, making it difficult for instructors to be sure they click the correct button. It has happened a couple of times that when we meant to click "stop" for a particular scenario, it actually stopped another scenario.

In an attempt to offer the SSH Inception scenario to students, we developed our own set of questions in an exercise using our Learning Management System. In the exercise instructions, we included the local IP addresses that students need to use according to the SSH Inception scenario. When we destroyed and created a new SSH Inception scenario for a new group of students, the IP addresses in the new scenario were different from the first one. This means that we had to take new screenshots and update the IP addresses we used in our exercise to match the newly deployed scenario. Instructors should expect that every new deployment of an exercise will have a different set of IP addresses associated with the scenario.

Recommendations
The first recommendation is to have a clear starting point for how to set up the EDURange server since the "Guides" tab on the EDURange website (visited on 04/05/2023) points to outdated instructions. For example, when clicking on the "An instructor EDURange installation guide" link (in Figure 4), it navigates to a deprecated GitHub repo (Figure 5). Also, there are two websites that host the EDURange information, guides, etc. (edurange.org and edurange.github.io), and the latter is an outdated version.



Guides

With the release of the refactor of edurange we have rewritten much of our documentation. All of it can be found here. The new documentation for students includes:

- A student account creation guide
- A student scenario information guide

The new documentation for instructors includes:

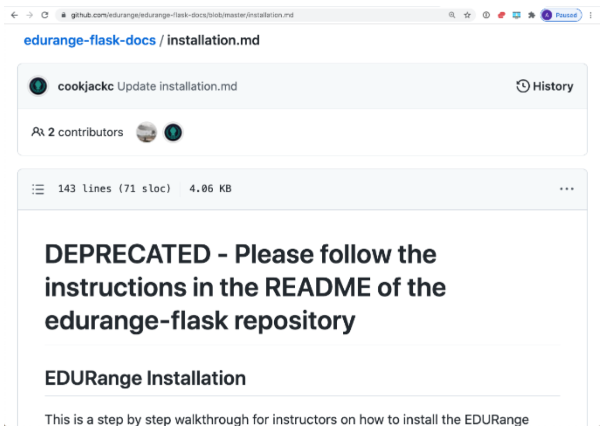
- An instructor EDURange installation guide
- A student group management guide
- A scenario management guide

The new documentation for developers includes:

- A developer EDURange installation guide
- A developer EDURange file system guide
- A developer EDURange database reference guide

In addition to this documentation most of our scenarios also have codelab webpages that come with the application to guide students through each scenario.

Figure 4 EDURange guides


github.com/edurange/edurange-flask-docs/blob/master/installation.md
Paused
History

edurange-flask-docs / installation.md

cookjackc Update installation.md

2 contributors

143 lines (71 sloc) 4.06 KB

DEPRECATED - Please follow the instructions in the README of the edurange-flask repository

EDURange Installation

This is a step by step walkthrough for instructors on how to install the EDURange

Figure 5 EDURange deprecated repository

After navigating the EDURange's main GitHub, we found the correct repository which is the one we used for setting up and running the EDURange server. We highly recommend pointing to that GitHub repository directly from the EDURange website to eliminate any confusion or trouble on the instructors' end.

We also recommend adding a "reset scenario" button for the whole class as well as separately for every student to address cases when students accidentally make irreversible changes to the environment preventing further progression through the assignment.

For those interested in using the EDURange project, keep in mind that you need to plan your hardware requirements according to the number of scenarios you will deploy and the number of students you will engage. If EDURange developers include example estimates for hardware requirements needed to support students in classrooms, that can help instructors plan accordingly.

There are two ways to create student accounts. Admins can create a "Group" which generates a "Registration Code" to share with students. Each student then has to create their own account with that Registration code. Note that if a student registers after the admin builds a scenario to the "Group", the student will not be able to access the scenario (an error will be shown when the student tries to open the scenario). Therefore, the instructors need to ensure that all students are registered before assigning a scenario to the "Group".

Alternatively, the admin can pre-populate the Group with a number of Temporary Members. This option gives the admin a standardized usernames and random passwords list which the admin can pass on to their students. However, if students use these usernames and passwords, they cannot change their usernames to their real names which can become a challenge during grading. There should be a way to import/export group members from an Excel/CSV file. The export function could provide the details of students' performance in the labs. The import function can allow the automated creation of user accounts, using students' real names/email addresses.

When an administrator creates a scenario, the only way to know which group it was created for is to go to the "Command History" tab inside the scenario and look for the player names and then find out the group they belong to from the administrator dashboard. For the users of EDURange, when creating a scenario, we recommend typing the GROUP NAME as part of the Scenario name because there's no easy way to know which group a scenario belongs to.

When building a scenario, the terminal will show information about the time elapsed for some scenarios (e.g., Metasploitable), but the web interface will not display any information about the progress of building a scenario. We recommend displaying information about the progress of building a scenario on the instructor dashboard. Also, indicating the disk space used by a scenario is of vital benefit in managing resources. Thus, we recommend adding the disk space used by each scenario on the instructor dashboard (e.g., Metasploitable requires about 7GB of disk space). The information about the required storage for all scenarios would allow the instructors to ensure that they have allocated enough disk space for the machine. Thus, it would also be useful to see how much disk space is left in total on the instructor dashboard. We recommend that the platform does not allow the instructor to create a scenario unless there is enough disk space available, displaying an appropriate notification message.

The dashboard could also benefit from an option for the instructors, notifying them when there is an update of the platform available on GitHub. As of now, there is no way to check for updates on new scenarios or incomplete scenarios.

It would be beneficial to provide an instructor Answer Key. Currently, instructors can find the correct answers to the questions by creating a test student account, going through the scenario tasks, and finding the answers themselves. Or, the test student account can be used to type any kind of an answer; then the instructor can go to the admin dashboard to see the correct answers for the test student user account trials. A separate answer key will also eliminate the highly unlikely event of a student running an EDURange server and finding the embedded answers themselves.

The answers to all scenarios are publicly available on the GitHub repository (meaning that students could potentially be able to find it), for example, the source for the SSH Inception scenario can be found here:
https://github.com/edurange/edurange-flask/blob/master/scenarios/prod/ssh_inception/questions.yml

The EDURange playground should allow instructors to specify the IP addresses range they need to use when deploying a scenario. If that IP range is not already in use by another scenario, the IP range should be accepted to be deployed. Otherwise, instructors should be notified that the IP range they chose is currently in use and should be able to change their choice.

5. OTHER PROJECTS NOT INCLUDED IN THIS STUDY

In this section we list other projects we hoped to assess but could not do so for different reasons.

GENI CyberPaths (Mountrouidou, 2019) included a variety of network-related exercises, such as DoS attacks, covert storage channels, and intrusion detection systems. However, the project is no longer maintained and most of the exercises are not possible to run. The SecKnitKit project (Siraj, Ghafoor, Tower, & Haynes, 2014), funded by the NSF, offers a VirtualBox standalone virtual machine to cover four different security areas: Network, Software Engineering, Operating Systems, and Database Management. It is important to note that this project is of a smaller size than the other projects included in this study. At the same time, the topics that the exercises cover could be introduced in non-security-related courses. The project is no longer maintained and is accessible for downloading from the CLARK platform (Taylor, Kaza, & Zaleppa, 2021).

There are some state-sponsored cyber ranges established in the US like the Michigan Cyber Range, Florida Cyber Range, and Virginia Cyber Range (Priyadarshini, 2018). For instance, the Virginia Cyber Range is a cloud-hosted infrastructure with hands-on cybersecurity labs, modules, and courseware repository that maps to the NICE Framework KSAs (Petersen, Santos, Smith, Wetzel, & Witte, 2020) and CAE KUs (CAE Documents Library, n.d.). The materials are freely available for Virginia State high schools and colleges that meet eligibility criteria. The same material is also available nationwide but under the name of the US Cyber Range. The US Cyber Range has a pricing model that is dependent upon the class enrollment and the number of months that the students plan to use the cyber range for.

There are several other paid platforms similar to the US CyberRange that we have not included here. Additionally, there are cyber ranges that have a limited availability scope, such as Cyber.org Range (n.d.) which is only accessible for K-12 schools.

6. RECOMMENDATIONS

Hands-on cybersecurity learning depends on two main elements: *effective exercises* and *usable platforms* hosting such exercises. In this section, we propose recommendations for developing new cybersecurity labs or exercises (following an evidence-based learning approach) as well as designing cybersecurity platforms.

According to our experience, we noticed that some students would follow all the steps in an exercise but would not be able to put all the exercise pieces together, therefore they would not fully understand the purpose of what they were doing. They would complete the exercises and pass the class but it would become a waste of time as no effective knowledge transfer had occurred. Thus, the exercise development process (as recommended in the following Exercise Development subsection) is key in making sure that knowledge transfer is effective for students. Additionally, without doubt, the usability characteristics of cybersecurity platforms (as recommended in the following Platform Development subsection) directly impact the use of effective exercises.

Exercise Development Recommendations
At the beginning of the exercise development, educators should clearly define what they want students to learn (learning objectives) so that they can evaluate it based on known frameworks such as the CAE Knowledge Units (KUs). Exercises should state what students need to know (prerequisite knowledge) and what they should be able to do using that knowledge (e.g., actionable outcomes). Educators should incorporate reflection questions at different stages (checkpoints) of the exercise to verify that students have grasped the individual KUs correctly. An additional benefit of such an approach is that other educators would be able to quickly understand which KUs are covered by the exercise, thus facilitating the continued development of new exercises and labs that address missing KUs of the existing resources.

We believe that each exercise should include the following sections:

- Learning objectives
- A mapping of learning objectives to NICE KSAs and/or CAE KUs
- Prerequisite knowledge
- Network map (or other necessary diagrams pertaining to the exercise)
- Glossary of major terms
- Complete and clear setup directions (for instructors), if applicable
- Scenario-based guided directions for students
- A comprehensive walk-through directions, such as instructional videos (for instructors)

- Hints and references to helpful resources at the end of every stage of the exercise that students could look into before proceeding (a system of checkpoints rather than step-through instructions)
- Submission guidelines (for students)
- An exercise answer key (for instructors)
- A sample evaluation rubric/methodology (for instructors)
- Knowledge base where students can comment, engage in discussions, and ask questions (e.g., Piazza)
- Optional: for extra complex projects, an FAQ section could be beneficial
- Optional: a final challenge scenario (e.g., a capstone project - for students)

Platform Development Recommendations
Based on the platforms evaluated in this work, we developed usability and feature recommendations pertaining to cybersecurity education platforms. The essential platform usability requirements include having a user-friendly intuitive UI/UX design, informative text about progress, and easy-to-access support pages.

Our recommendations for the platform features are:
- A "getting started" demo exercise for students and instructors.
- Setup guidelines (including required resources) to deploy the platform if self-hosted by instructors.
- A process (with examples) for instructors to publish exercises on the platform.
- An ability to add new students to the exercise at any time (some platforms do not allow adding new students if the exercise has been deployed).
- Students' time tracking spent on working on exercises.
- Students' exercise progress tracking.
- An ability to rate, review, and provide feedback for each exercise on the platform.
- An ability to add co-instructors or TAs to assignments.
- A list of platform limitations (e.g., number of students working at the same time, a list of required computing resources, etc.).
- Support mechanisms for students and instructors.
- FAQ for students and instructors.

Additionally, cybersecurity platform developers should keep in mind that some educators are new to this field (especially at the K-12 level) and need easy access to material and instructions. We also believe that funding agencies (e.g., NSF, NSA) should provide opportunities for projects with promising initiatives (e.g., NICE Challenge, US Cyber Range) to give gifts/funds to schools that want to start using these environments for a predefined period of time (like a year) and require these newcomers to provide effective feedback on how to enhance the project.

7. CONCLUSION AND FUTURE DIRECTION

Cybersecurity resources have come to a state of spaghetti code: unstructured and difficult to maintain. In this work, we developed evaluation criteria for cybersecurity exercises and platforms and used them to evaluate existing cybersecurity education resources, determining the instructional value for each of them. We shared our personal experiences using the platforms and provided recommendations to developers and users. These recommendations are not meant to contain an exhaustive list of best practices but rather be a starting point for cybersecurity educators to use and improve upon. Finally, we listed recommendations for exercises and platform development which can enhance the existing cybersecurity posture in education.

Our next steps in this research include developing a survey to poll the community (NICE, NSA CAEs, SIGCSE, etc.) about the evaluation criteria, their feedback on their experiences, and recommendations. In addition, we plan to ask them what resources they use and what approaches they follow to share their experiments, if any. We will compile the results of our future work in the form of a publicly available white paper to the community. Additionally, we plan to work with the community at large to improve the existing cybersecurity posture in education.

8. REFERENCES

Boesen, S., Weiss, R. S., Sullivan, J. F., Locasto, M. E., Mache, J., & Nilsen, E. (2014, August). EDURange: Meeting the Pedagogical Challenges of Student Participation in Cybertraining Environments. In *CSET*.

CAE Documents Library. (n.d.). Documents library. Retrieved May 8, 2023, from https://public.cyber.mil/ncae-c/documents-library

Cyber.org Range. (n.d.). Home. Retrieved May 8, 2023, from https://cyber.org/range

DETERLab. (n.d.). Home. Retrieved May 8, 2023, from https://www.isi.deterlab.net

DETERLab Ticket System. (n.d.). New ticket (login required). Retrieved May 8, 2023, from https://trac.deterlab.net/newticket

DETER Project. (n.d.). Home. Retrieved May 8, 2023, from https://deter-project.org

Ibrahim, A., & Ford, V. (2021). Observations, Evaluations, and Recommendations for DETERLab from an Educational Perspective. *Journal of Cybersecurity Education, Research and Practice*, *2021*(1).

Mirkovic, J., & Benzel, T. (2012). Teaching cybersecurity with DeterLab. *IEEE Security & Privacy*, *10*(1), 73-76.

Mountrouidou, X. (2019). CyberPaths. *Journal of Computing Sciences in Colleges*, *34*(3), 16-16.

NICE Challenge. (n.d.). Home. Retrieved May 8, 2023, from https://nice-challenge.com

Petersen, R., Santos, D., Smith, M. C., Wetzel, K. A., & Witte, G. (2020). Workforce Framework for Cybersecurity (NICE Framework) NIST Special Publication 800-181 Revision 1. *National Institute of Standards and Technology*.

Priyadarshini, I. (2018). *Features and architecture of the modern cyber range: a qualitative analysis and survey*. The University of Delaware.

SEED Labs. (n.d.). Home. Retrieved May 8, 2023, from https://seedsecuritylabs.org

Siraj, A., Ghafoor, S., Tower, J., & Haynes, A. (2014, June). Empowering faculty to embed security topics into computer science courses. In *Proceedings of the 2014 Conference on Innovation & Technology in computer science education* (pp. 99-104).

Taylor, B., Kaza, S., & Zaleppa, P. A. (2021). CLARK: A Design Science Research Project for Building and Sharing High-Quality Cybersecurity Curricula. *IEEE Security & Privacy*, *19*(5), 72-76.

APPENDIX A

Resource Name	**Usability Criteria**					
	Type of labs	Customizable Labs	Cloud-based	Lab access	Setup guidelines	Level of support
	C1	C2	C3-A	C3-B	C3-C	C4
DETERLab	Both	Yes	Yes	SSH only	N/A	Institutional
NICE Challenge	Both	No	Yes	Web Interface Only	N/A	Institutional
SEED Labs	Both	Yes	No	N/A	Yes	Individual
EDURange	Both	Yes	Yes	SSH	Yes	Individual

Resource Name	**Effectiveness Criteria**							
	Instructor's Manual Available	Student Instructions Available	Includes Learning Objectives	Mapping to Frameworks	Limitations	Progress Tracking	Time Tracking	Accessibility Level
	C5	C6	C7	C8	C9	C10	C11	C12
DETERLab	Partial	No	No	No	Resources	No	No	Nationwide
NICE Challenge	No	No	No	NICE TKSAs + CAE KUs	Both	No	Lab	Nationwide
SEED Labs	Yes	Yes	Yes	No	Resources	No	No	Nationwide
EDURange	No	Yes	Yes	No	Resources	Yes	No	Nationwide

Utilizing Discord-based Projects to Reinforce Cybersecurity Concepts

Marc Waldman
marc.waldman@manhattan.edu

Patricia Sheridan
patricia.sheridan@manhattan.edu

Computer Information Systems and Business Analytics Department
O'Malley School of Business
Manhattan College
Riverdale, NY 10471, USA

Abstract

This paper describes two Discord-based undergraduate student projects that were created to reinforce cybersecurity concepts covered in a data privacy course and an introduction to cloud computing course. The first project, created for the data privacy course, is used to provide students with a practical and "hands-on" project to utilize role-based access control (RBAC). This course was developed primarily for business students who do not necessarily have the information technology background possessed by computer information systems or computer science students. The second project, created for an introduction to cloud computing course, is used to illustrate the basic function of a botnet and incorporates a few other cybersecurity topics. We believe these projects represent just the tip of the iceberg of how Discord can be utilized as a source of course projects. Additional Discord-based project ideas are described at the end of the paper.

Keywords: information systems education, cybersecurity projects, discord, data privacy, role-based access control (RBAC), bot programming, botnets

1. INTRODUCTION

Discord is a free voice, video, and text communication platform that was initially designed to facilitate communication between video game players (Discord, 2023). Since its launch in 2015, the platform has also been widely adopted by non-gamers seeking to create shared-interest virtual communities. Since the COVID-19 pandemic, it has also been used as a tool in the delivery of online courses - being one of the few easy-to-use free platforms that combines text-based chat, direct messaging, screen sharing, and both voice and video conferencing for multiple concurrent users.

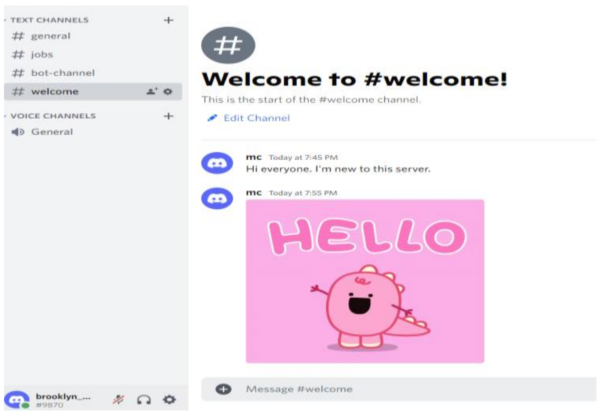
Much has been written about utilizing Discord's sharing and communication features as a tool to facilitate online education (Grimbsy, 2019; Wiles & Simmons, 2022). However, the Discord ecosystem (software, community, content) can also be used as a source of course projects. We describe two cybersecurity-related Discord-based student projects that we created for a data privacy course and an introduction to cloud computing course. Both courses are offered in our Computer Information Systems (CIS) Department, which is housed within the School of Business. All project-related materials are available for download (https://github.com/marcwaldman/discord_rbac_botnet).

2. DISCORD

The Discord platform allows users to navigate in and out of virtual communities. Each community is referred to as a "server" or a "guild". Each server consists of one or more multimedia-rich discussion "channels" that allow participants to talk, in real-time, using voice, text chat, or video conferencing.

Channels are voice or text-based "virtual rooms" set aside for the specified type of communication. Voice channels also support video and screen sharing. Multiple named channels can exist within a single server and can optionally be grouped under an umbrella name called a "category". For example, a Discord server for individuals interested in computer programming might have a "Java" and a "Python" category, each containing a group of channels related to topics within the particular programming language.

Figure 1 shows a very basic Discord client user screen. Individual text and voice channels can be selected from the left-hand side listing. In this case, the "welcome" channel has been selected. The two categories appearing on this screen are "TEXT CHANNELS" and "VOICE CHANNELS". The chat area is located to the right of the channel listing - messages from users that have posted to this channel will appear here. Older messages can be read by scrolling up; newer messages appear at the bottom of the channel. This channel contains only two messages. Both text and multimedia (emojis, pictures, and video clips) can be posted to text channels.



Welcome to #welcome!
This is the start of the #welcome channel.

mc Today at 7:45 PM
Hi everyone. I'm new to this server.

mc Today at 7:55 PM

Figure 2: Discord Client User Screen

The Discord software, referred to as the Discord client, is available for all major desktop and phone-based operating systems. In addition, one can access Discord via a web-based front-end. Note that the Discord platform itself is centralized and hosted on Discord company hardware. Users and developers simply interact with the platform via the client software or its application programmer interface (API).

The Discord client provides a relatively intuitive graphical user interface that allows one to join, create, or administer servers. No programming or system administration experience is needed. With the click of a button, a server can be created and pre-populated with default text and voice channels. All channels can be changed or removed by the server's creator.

In addition to creating servers, one can join other pre-existing servers. Servers can be configured to allow anyone to join or require an invite to join. An invite is essentially a server-specific URL generated by Discord. These options are controlled by the server's creator, who is, by default, considered the server's administrator.

A Discord-provided search engine allows individuals to find servers catering to specific interests. Web-accessible third-party search services and listings are also available.

Permissions and Roles
Popular Discord servers may have many thousands of users. Administering and policing members in channels and other Discord objects is facilitated by permissions and roles. Permissions are assigned to grant access to Discord objects (e.g. a text channel) or to allow specific actions such as the ability to delete any post within a channel. Roles can be thought of as a named alias for a group of permissions. Anyone who joins a server can be assigned to one or more roles. The default role is the "everybody" role - permissions granted to this role apply to all users of the server. Permissions or roles can be assigned to users by the server's administrator.

Certain permissions, such as removing a message posted by another user, are typically reserved solely for the administrator. However, the administrator can assign these permissions, or a role incorporating these permissions, to other users - allowing the moderation workload to be shared among special "deputized" members.

Why Discord?
Several Discord competitors exist, including the well-known Slack and Microsoft Teams applications. When evaluating the alternatives, we were looking for an application that had the following:
- Wide adoption and utilization.
- Freely available for all operating systems.

- Allowed the creation of "virtual communities" that kept a persistent record of public text-based messages that could later be viewed by community members who were not present at the time the messages were posted.
- Allowed the creation of "bots" -programs that interact with the communications platform.
- Provided the ability to grant application-specific permissions or roles to different users of the system.

While the free-to-use tier of several of Discord's competitors, including Slack, do satisfy these requirements, Discord was chosen due to its perceived ease of use, popularity among college students (D'Agostino, 2023; Sukhanova, 2023), and generous limits on its free services.

The two projects described in this paper do not require any Discord-specific features. Therefore, the projects can be adapted to any platform offering the last three features outlined above.

Although originally developed for use by gamers, Discord has been widely (well over 100 million users) utilized and adopted as a platform to host virtual communities in areas as diverse as music appreciation and cryptocurrencies (Browning, 2021). There exists a rich support community for this platform as evidenced by the large collection of quality resources that can be readily accessed on YouTube or otherwise found via a search engine.

The authors, during the Spring 2022 and Spring 2023 semesters, utilized Discord-based student projects in two courses offered in our CIS department. The courses and the associated projects are described below.

3. DATA PRIVACY COURSE

This is a CIS elective course that is open to all students in the School of Business. Due to the importance of this subject matter, CIS majors are strongly encouraged to enroll. The course has a law-oriented focus, examining data privacy laws, policies, and issues related to data protection. There are no course prerequisites, but junior or senior standing is expected.

As the material covered in this course will likely be of interest to students in other majors, no formal prerequisites were added. While this lack of prerequisites expanded the audience for the course, it also limited the type of computer-based projects that could be assigned. Projects involving computer programming, system administration, or database development were not an option. Nevertheless, applied "hands-on" projects can greatly enhance student learning and show how in-course concepts can be applied in the "real world".

The important concepts of "roles" and "permissions" are covered in the course. These concepts are relatively easy to understand and are found in many aspects of life (society, school, work, etc.). However, it is only when one tries to apply these concepts to non-trivial problems does one realize the complexity of the underlying implementation issues (identity management, conflicting permissions, role revocation, etc.). Students majoring in information technology-related fields (Computer Science, Business Analytics, Information Systems, etc.) will likely be able to apply these concepts to implementable projects in areas such as database management or system administration. Fewer opportunities to practically apply these concepts exist for those with non-information technology backgrounds. A Discord-based project was developed to help fill this gap.

Role-Based Access Control

One of the reasons we chose the Discord platform was its ability to allow server owners to establish and grant roles to specific users. This provided us with the opportunity to create a student project that incorporated the concept of Role-Based Access Control (RBAC), a topic that is covered in the course. As defined in (Ferraiolo et al., 1995), "the central notion of Role-Based Access Control (RBAC) is that users do not have discretionary access to enterprise objects. Instead, access permissions are administratively associated with roles, and users are administratively made members of appropriate roles. Users can be made members of roles as determined by their responsibilities and qualifications and can be easily reassigned from one role to another without modifying the underlying access structure."

In more simplistic terms, a role can be thought of as an alias for a set of permissions, and each role can be assigned to, or revoked from, one or more users. Roles can be combined and built into hierarchies, forming new roles that provide incremental addition of new permissions. For example, within a company, everyone may be assigned the role of employee. The employee role has a limited set of access permissions to the company's Human Resources (HR) system. The manager role incorporates the employee role and includes additional access permissions for the HR system.

Multiple roles can also be assigned to a single user. Within RBAC systems there exist mechanisms that allow, possibly conflicting, permissions within different roles to be disambiguated and then applied. Discord, which supports the assignment and combination of multiple roles, has a rich set of permissions that can be applied to various objects including communication channels, messages, and users. The full set of permissions and disambiguation rules are described on the Discord Developer website (Discord Permissions, 2023).

Preparation and Project
Before being assigned the project, the students were required to complete a PowerPoint-based tutorial/exercise that led them through Discord setup and usage. The tutorial also covered the basics of server creation, permissions, and roles. In a subsequent class, the following project was assigned:

1) For this project you will create a Discord Server for a physical therapy office so that staff members and patients can exchange general health and wellness advice. Participants may include the office manager, receptionist, patients, etc.

    - The group wants to share nutrition tips, recipes, and ideas for healthy eating. They also want to recommend fitness workouts and outdoor trails to each other and suggest local shops and restaurants that cater to health-conscious individuals.

    - Some members of the office staff want to be able to monitor the server, delete inappropriate content or inadvertent disclosures of personal health information, and remove any members if necessary.

2) Create a Discord server for this purpose and submit a writeup that includes the following:

    - List of at least 6 roles that will be utilized by your server.

    - List of at least 6 channels that will be available to your members. Briefly describe the purpose of each channel. Some may be private – but specify which ones and which roles can access them.

    - List of at least three categories and which channels they will contain.

    - List of permissions assigned to each role. Briefly explain the reason for assigning such permissions.

    - Email the URL for your server to the instructor and add the instructor as an administrator.

The physical therapy scenario can be replaced by a similar one – doctor's office, law office, day-care center, etc. Another alternative is to offer students the opportunity to choose one of many scenarios or create their own scenario. Section 6 describes student feedback and survey results.

4. INTRODUCTION TO CLOUD COMPUTING COURSE

This is an upper sophomore/junior level course that is meant to introduce cloud-based applications and programming to students. It is required for all CIS majors. The only prerequisite is our one-term introduction to programming course which covers the Python programming language.

Over the past few years, communications platforms such as Microsoft Teams, Slack, and Discord have increasingly been adopted by organizations of all sizes. Students in our program will very likely utilize one or more of these platforms during an internship, part-time job, or as a member of a student club. These systems are cloud-based and allow the creation of programs, called bots, that can interact with users of the platform.

Discord API
Discord allows developers to create programs, usually referred to as bots, that can interact independently (don't need constant human monitoring) with the platform. Bots typically provide utility functionality such as generating a nightly server activity report, monitoring a chat stream for inadmissible language, or replying to simple questions posed by users. Bots that receive user-generated text and then reply, in real-time, with a meaningful text response, are frequently referred to as chatbots (Adamopoulou & Moussiades, 2020).

Discord, Slack, and Teams all provide an API that can be used to create bots. One does not need a programming background to add a bot to a server. Many types of pre-built bots exist, and they can be easily added by a server's administrator. Some of the most popular bots are those that automatically perform routine operations such as greeting new users or reposting content from social media (Twitter, Instagram, etc.) to a specific channel. These bots are typically created by third parties and are not necessarily vetted or pre-approved by the Discord company. When a server administrator requests a bot be added to their server, the Discord server will reveal which permissions the bot is requesting. The administrator must use their discretion to decide if the bot should be allowed to run with the requested permissions.

The Discord API is based on REST (Fielding & Taylor, 2002) and WebSockets (Melnikov & Fette, 2011). However, Discord does not recommend using the API directly, as it is "complex and fairly unforgiving" (Discord API, 2023). Instead, bot developers typically utilize code libraries that provide an easier-to-use class and/or function-based interface to the API. The Discord company does not provide code libraries to utilize the API. Nevertheless, several well-supported, third-party, free, and open-source libraries are available for many programming languages. These code libraries abstract away many of the low-level details of the underlying communication between the user's program (the bot) and the server. Discord does provide a list of code libraries (Discord Code, 2023) that it has vetted for valid API rate-limit implementations. These limits, which place a cap on the number of API requests a bot can make per second, will likely not be of concern to students as the quotas are generous and are essentially meant to prevent abuse or service overloads by bots with a large user base.

Project Description

This course contains a cybersecurity component that provides a high-level overview of some of the cloud-relevant secure computing concepts outlined in the IS2020 Competency Model (Leidig & Salmela, 2020). To help reinforce and apply this material, a cybersecurity-related bot project was chosen instead of a chatbot. This project was inspired by a class lesson/discussion on cybercrime and botnets.

A botnet is a network of compromised computers, called "bots", that are under the remote control of a human operator who is referred to as the "botmaster" (Feily et al., 2009). So as not to confuse the reader with the two uses of the term "bot" (innocuous program interfacing with the Discord server vs. a compromised computer), the term "zombie" (Cooke et al., 2005) will be used to refer to the compromised computer that is part of a botnet.

There exists a wealth of literature on botnet history, uses, architecture, defense, and detection (Bailey et al., 2009; Cooke et al., 2005; Feily et al., 2009; Vormayr et al., 2017). Therefore, only the project-relevant aspects of botnets will be discussed in this paper.

A botnet can be thought of as a form of distributed computing. The botmaster issues commands to all participating computers (zombies) via some command and control (C&C) mechanism. The actual C&C mechanism depends on the level of sophistication of the botnet but can be as simple as an internet relay chat (IRC) channel or social media account (Pantic & Husain, 2015) or as complex as a peer-to-peer network hidden via the TOR anonymizing network (Casenove & Miraglia, 2014). Although this description of a botnet may make it appear to be rather benign, the term botnet has a strong association with cybercrime and other malicious activity.

Probably the most publicized and well-known use of a botnet is for distributed denial of service (DDOS) attacks against websites or other Internet-based services (Hoque et al., 2015). These attacks inundate a set of IP addresses, belonging to a targeted organization, with network traffic in an attempt to prevent or slow legitimate access to the services provided by the target. DDOS attacks have been utilized for things as trivial as eliminating an opponent from an online game to serious crimes such as online blackmail and critical service disruption. Other uses of botnets include sending spam and crypto-currency mining (Sood & Enbody, 2013).

A victim computer (zombie) almost always joins a botnet involuntarily. This usually occurs when the user of the victim computer is tricked into executing some malicious code via a phishing e-mail, instant message hyperlink, or other form of infection. Once infected with the malicious code the zombie typically contacts the botmaster via the C&C mechanism or other prearranged communication channel, and then awaits commands from the botmaster.

For this project, each student designed a Discord bot that functioned as a zombie – accepting commands issued by the botmaster (the student entering commands into Discord). Upon startup, the zombie first connects to Discord. This connection allows the zombie to contact the botmaster – Discord acts as the C&C mechanism. On initial connection to Discord, the bot sends its "ID" (a small randomly generated sequence of

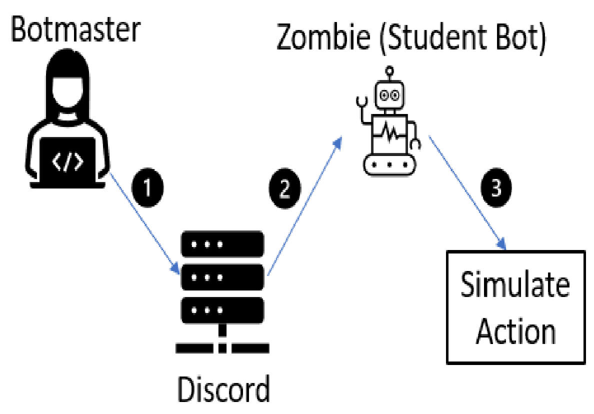


Figure 3: Botmaster to Zombie Communication

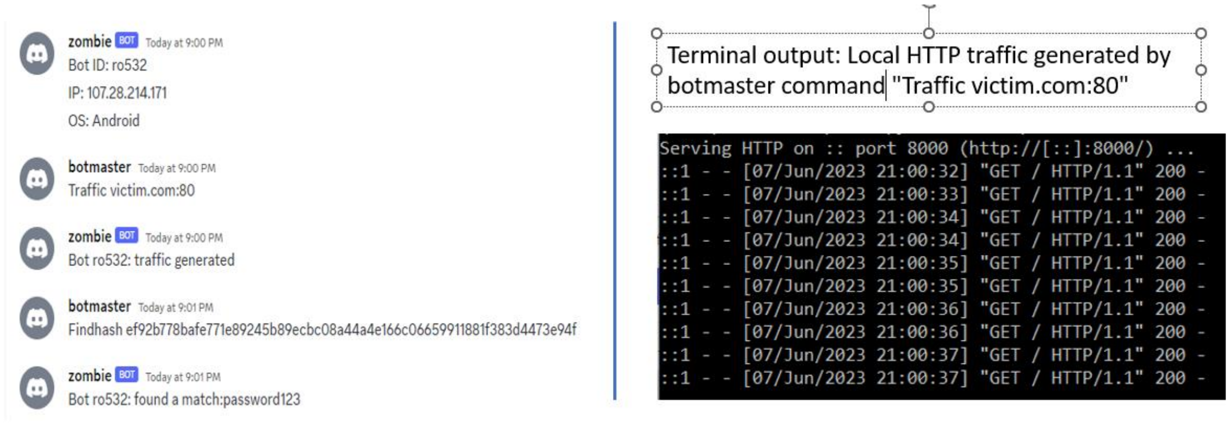
characters and numbers), an IP address, and operating system name (e.g. "Windows 10", "Android", etc.) to the botmaster. Both the IP address and operating system name are randomly generated – they are sent to Discord to illustrate the type of information collected by botmasters upon initial infection of the zombie. The zombie then awaits botmaster commands from the botmaster.

Figure 2 illustrates the botmaster to zombie communication process. The student (botmaster) issues commands. Each command is forwarded by Discord to the zombie (student-created bot), which is listening for botmaster commands in a specific channel (step 2). The zombie parses the command and then simulates the corresponding action (step 3).

Some of the botnet commands the students were required to simulate are listed in Table 1 (all commands can be found in the project documentation). Each command has one or more required arguments that are specified along with the command – just like the specification of arguments to shell commands. Arguments are separated from the command and each other by one or more spaces. For example, the botnet command "Traffic victim.com:80" instructs the zombie to simulate the sending of traffic to victim.com on port 80 (as part of a DDOS attack). The student-implemented zombie does not send network traffic (spam, etc.) to externally targeted services. Instead, to mimic the generation of malicious traffic, HTTP requests are sent to a local (on the student's computer) running instance of Python's bundled HTTP server. By default, all requests received by the HTTP server are displayed in the HTTP server's terminal window – allowing the student to view the zombie-initiated requests as they are generated.

A sample botmaster and zombie interaction is shown in Figure 3. Upon startup, the zombie sends its randomly generated bot ID, IP Address, and operating system to the botmaster via Discord. The botmaster requests the zombie to send network traffic to victim.com on port 80. This is simulated by sending requests to a locally running HTTP server. Requests received by the HTTP server are displayed in a terminal window (right-hand side of Figure 3). A true DDOS attack would utilize many such zombies to inundate the targeted host with network traffic. The Findhash command, also shown on the left-hand side of Figure 3, asks the bot to search its locally cached list of commonly used passwords to "crack" (find a match for) the supplied SHA256 hash. In this case, the zombie found a match on the string "password123". Real botnets would likely utilize more advanced forms of hash cracking/matching (Dev, 2013).

Based on student programming and computer security background, the project can be expanded to include more sophisticated zombie behavior, coordination, and simulation of botnet



zombie BOT Today at 9:00 PM
Bot ID: ro532
IP: 107.28.214.171
OS: Android

botmaster Today at 9:00 PM
Traffic victim.com:80

zombie BOT Today at 9:00 PM
Bot ro532: traffic generated

botmaster Today at 9:01 PM
Findhash ef92b778bafe771e89245b89ecbc08a44a4e166c06659911881f383d4473e94f

zombie BOT Today at 9:01 PM
Bot ro532: found a match:password123

Terminal output: Local HTTP traffic generated by botmaster command "Traffic victim.com:80"

```
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
::1 - - [07/Jun/2023 21:00:32] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:33] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:34] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:34] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:35] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:35] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:36] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:36] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:37] "GET / HTTP/1.1" 200 -
::1 - - [07/Jun/2023 21:00:37] "GET / HTTP/1.1" 200 -
```

Figure 4:Sample botmaster and zombie interaction via Discord

Command	Argument(s)	Description
Update	URL	Simulate updating the zombie executable with the one specified by the URL.
Traffic	IP_Address:port or Hostname:port	Simulate a DDOS attack against the specified IP_Address (or hostname) and associated port.
Findhash	Cryptographic_Hash	Simulate the behavior of a zombie being assigned to password cracking or cryptographic hash matching.
Mail	Email_type, Email_address	Simulate a zombie sending phishing email of the specified type (banking, payment_request, password_reset, etc.) to the user with the specified Email_address.

Table 1: Sample Botnet Commands

commands. Possible enhancements, inspired by the botnet literature, include encrypted botmaster and zombie communication, using the TOR anonymizing network for botnet communication, channel or server hopping based on prearranged functions, and generation of "cover" traffic to hinder C&C mechanism discovery.

Although this project utilized only one zombie it is certainly possible to have multiple bots (zombies) join a Discord server, allowing for project additions such as those described in the previous paragraph.

Preparation
One course session, an hour and fifteen minutes in length, was devoted to introducing the topics and sample code needed for Discord bot creation. Before the start of this session, the students were required to complete, as a homework assignment, readings and exercises that introduced Discord. This is the same intro material that was utilized for the Data Privacy course project.

It is best to access the Discord API by utilizing one of the freely available, open-source, rate-limit vetted code libraries. For the Python programming language, several such libraries exist. By far the most popular (as measured by GitHub "stars") is discord.py (Rapptz, Danny, 2015/2023).

Figure 4 shows the code for a very basic bot that utilizes discord.py. This bot, once connected to a Discord server, will display a "Bot logged in as" message in the terminal window. The bot will then receive messages that users send to the Discord server. If the bot receives, via Discord, the string "$hello", it will respond to the message's author with a "Bot welcomes" greeting. Note that each

```
import discord

TOKEN="Copy Your Bot Token Here"

client = discord.Client()

@client.event
async def on_ready():                    # executed when bot logs into Discord
    print("Bot logged in as ",client.user)

@client.event
async def on_message(message):           # executed when a message is received
   if message.author == client.user:     # don't respond to bot's own messages
      return

   if message.content=="$hello":         # welcome user who sent a $hello message
      await message.channel.send("Bot welcomes "+message.author.name)

client.run(TOKEN)
```

Figure 4: Slightly modified version of the basic bot code that appears on the discord.py documentation webpage (Rapptz, Danny, 2015/2023).

message sent to the bot is automatically accompanied by associated metadata, allowing the bot to incorporate the message author's username.

As part of a series of in-class exercises, students were asked to make incremental changes to this code to expand the bot functionality. Each exercise required the student to modify and then test the code via Discord server interaction. The exercises culminated in the creation of a bot that performed basic calculations (addition, subtraction, etc.) at the request of a user. The botnet project was assigned after the preparation materials and associated exercises were completed.

5. RELATED WORK

This section provides a literature-based survey of the methods (assignments, exercises, or projects) utilized to introduce or reinforce the topics of access control and botnets.

Courses requiring software development skills, such as software engineering or operating systems design, typically approach access control projects from an implementation angle – requiring students to either audit or write the code that implements a particular access control mechanism (Luburic et al., 2019; Nieh & Vaill, 2005; Petullo et al., 2016). A similar approach has been taken by database management and design courses (Yang, 2009). In this case, access control features (users, policies, roles, etc.) are implemented or audited using the database management system's tools.

Lab-based access control exercises are frequently found in system administration and cybersecurity courses (Du et al., 2010; Hay et al., 2008; Hu & Wang, 2008). These lab exercises require the student to implement, via operating system tools or features, various forms of access control on system resources such as networks or external storage. The use of these tools may be reinforced by team-based capture-the-flag competitions (Eagle & Clark, 2004) in which students attempt to either breach or defend computing resources (Chothia & Novakovic, 2015; Mirkovic & Peterson, 2014).

Two freely-available software-based tools have been developed to allow an individual to create, visualize and query RBAC (Wang et al., 2015) and MLS (Wang et al., 2014) access control policies. The student uses a simple point-and-click interface to create, interact with, and modify a graph-based visualization of the policy. The software can be used to verify policy constraints.

A less formal approach to access control training can be found in CyberCIEGE, a video game intended to support education and training in computer and network security (Cone et al., 2007). The game employs resource management and simulation to illustrate information assurance concepts for education and training (Thompson & Irvine, 2014). Access control is only one of the topics covered by the game (*CyberCIEGE Scenario Listing*, 2023).

Botnet projects and lab exercises run the gamut from passive to active. Lab exercises on the more passive end (Dias et al., 2017; Hay et al., 2008) focus on watching a "harmless" botnet (which doesn't contain exploit code) in action by utilizing packet sniffers and network visualization tools. Due to the distributed nature of botnets, the lab exercises are conducted on virtual machines and bots are prevented from spreading to the external network. In these lab exercises the student plays the role of the botmaster issuing commands to the zombies via an IRC server that is distributed as part of the virtual machine images. Another variation of this lab exercise replaces the "harmless" botnet code with "real" botnet code (Hannah & Gianvecchio, 2015).

The second type of botnet lab exercises and projects focuses on modifying the botnet codebase to introduce new features such as additional infection methods or new client information capture (DeCusatis et al., 2021; Vergos, 2011).

6. ADDITIONAL DISCORD-RELATED PROJECT IDEAS

We believe that Discord's popularity, availability, and content make it a rich source for student projects. Since it is fundamentally a communications platform, chatbots are the most obvious type of project that can be assigned (Cerezo et al., 2019; Raglianti et al., 2022).

The desktop Discord client caches, on disk, much information about a user's session (Iqbal et al., 2021). This is done to minimize the amount of bandwidth utilized during a Discord session. This cached information can form the basis of digital forensics-based projects.

Discord's popularity among the Non-Fungible Token (NFT) and cryptocurrency community (Sharma et al., 2022) may provide a good source of data to analyze what might be involved in launching and maintaining a successful NFT project. This community is frequently the target of scammers and cybercriminals (TRM Labs, 2022)– this malicious activity generates Discord-based text and links that can also be extracted and analyzed by students.

7. STUDENT FEEDBACK AND DISCUSSION

We have been utilizing Discord-based projects since the Spring 2022 term. Overall, about half the students report having previously used Discord. This number is slightly higher for students taking the cloud computing course. Nevertheless, none of the students reported any difficulty completing the exercises/tutorial that introduced Discord and server creation.

The Data Privacy project was initially assigned as a group project to be completed during one in-class session. However, based on post-project student feedback, it was converted to a single student project and split over two class sessions. This gave the students more time to think about the project and they did not have to worry about delegating tasks to different members of the group. After these changes were implemented, most students strongly agreed that the project improved their understanding of how role-based permissions are an important part of an organization's privacy protection program.

Cloud Computing course students were given one week to complete the project. Each student had to complete the project on their own - this was not a team-based project. Students were encouraged to contact the instructor if they experienced any code errors that prevented them from making progress on the project. Only about a quarter of the students contacted the instructor with code-related error issues. Most of these errors had to do with the improper use of the Python hashlib module. This module provides an interface to several secure hash and message digest algorithms. Based on this experience, and student feedback, additional course time will be devoted to proper use of this module. In a post-project survey slightly under 75% of the students reported that they liked working on the project – using words such as "fun", "enjoyed" and "great" in their response. Slightly under 80% of the students agreed with the statement that the project helped them improve their understanding of the project-relevant computer security topics.

All project-related materials are available for download (https://github.com/marcwaldman/discord_rbac_botnet). The scenarios utilized in, and the components of, both projects can be tailored based on student background.

8. REFERENCES

Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II 16*, 373–383. https://doi.org/10.1007/978-3-030-49186-4_31

Bailey, M., Cooke, E., Jahanian, F., Xu, Y., & Karir, M. (2009). A survey of botnet technology and defenses. *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, 299–304. https://doi.org/10.1109/CATCH.2009.40

Browning, K. (2021, December 29). How Discord, Born From an Obscure Game, Became a Social Hub for Young People. *The New York Times*. https://www.nytimes.com/2021/12/29/business/discord-server-social-media.html

Casenove, M., & Miraglia, A. (2014). Botnet over Tor: The illusion of hiding. *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, 273–282. https://doi.org/10.1109/CYCON.2014.6916408

Cerezo, J., Kubelka, J., Robbes, R., & Bergel, A. (2019). Building an Expert Recommender Chatbot. *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*, 59–63. https://doi.org/10.1109/BotSE.2019.00022

Chothia, T., & Novakovic, C. (2015). An Offline Capture The {Flag-Style} Virtual Machine and an Assessment of Its Value for Cybersecurity Education. *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*. https://www.usenix.org/system/files/conference/3gse15/3gse15-chothia.pdf

Cone, B. D., Irvine, C. E., Thompson, M. F., & Nguyen, T. D. (2007). A video game for cyber security training and awareness. *Computers and Security*, *26*(1), 63–72. https://doi.org/10.1016/j.cose.2006.10.005

Cooke, E., Jahanian, F., & McPherson, D. (2005). The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. *Usenix SRUTI*, *5*. https://www.usenix.org/legacy/event/sruti05/tech/full_papers/cooke/cooke_html/

CyberCIEGE Scenario Listing. (2023). https://nps.edu/web/c3o/scenarios

D'Agostino, S. (2023, April 26). *Discord for Leaking Military Files—And Exam Questions*. Inside Higher Ed. https://www.insidehighered.com/news/tech-innovation/digital-teaching-learning/2023/04/26/discord-leaking-military-files-and-exam

DeCusatis, C. M., Bavaro, J., Cannistraci, T., Griffin, B., Jenkins, J., & Ronan, M. (2021). Red-blue team exercises for cybersecurity training during a pandemic. *11th IEEE Annual Computing and Communication Workshop and Conference, CCWC 2021, Las Vegas, NV, USA, January 27-30, 2021*, 1055–1060. https://doi.org/10.1109/CCWC51732.2021.9376016

Dev, J. A. (2013). Usage of botnets for high speed md5 hash cracking. *Third International Conference on Innovative Computing Technology (INTECH 2013)*, 314–320. https://doi.org/10.1109/INTECH.2013.6653658

Dias, J. P., Pinto, J. P., & Cruz, J. M. (2017). A Hands-on Approach on Botnets for Behavior Exploration. In M. Ramachandran, V. M. Muñoz, V. Kantere, G. B. Wills, R. J. Walters, & V. Chang (Eds.), *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security, IoTBDS 2017, Porto, Portugal, April 24-26, 2017* (pp. 463–469). SciTePress. https://doi.org/10.5220/0006392404630469

Discord. (2023, March 8). *About Discord*. About Discord. https://discord.com/company

Discord API. (2023, April 23). *Discord Gateways API*. Discord Developer Portal. https://discord.com/developers/docs/topics/gateway

Discord Code. (2023, May 3). *Discord Vetted Code Libraries*. Discord Developer Portal. https://discord.com/developers/docs/topics/community-resources#libraries

Discord Permissions. (2023, April 24). *Discord Permissions Documentation*. Discord Developer Portal. https://discord.com/developers/docs/topics/permissions

Du, W., Jayaraman, K., & Gaubatz, N. B. (2010). Enhancing security education with hands-on laboratory exercises. *Proceedings of the 5th Annual Symposium on Information Assurance (ASIA'10)*, 56–61. https://www.academia.edu/download/74371140/ASIA10Proceedings.pdf#page=65

Eagle, C., & Clark, J. L. (2004). Capture-the-flag: Learning computer security under fire. *Proceedings from the Sixth Workshop on Education in Computer Security (WECS6). Monterey, CA*. https://apps.dtic.mil/sti/citations/tr/ADA435319

Feily, M., Shahrestani, A., & Ramadass, S. (2009). A survey of botnet and botnet detection. *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, 268–273. https://doi.org/10.1109/SECURWARE.2009.48

Ferraiolo, D., Cugini, J., & Kuhn, D. R. (1995). Role-Based Access Control (RBAC): Features and Motivations. *Proceedings of 11th Annual Computer Security Application Conference*, 241–248. https://csrc.nist.gov/pubs/conference/1995/12/15/rolebased-access-control-rbac-features-and-motivat/final

Fielding, R. T., & Taylor, R. N. (2002). Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology (TOIT)*, *2*(2), 115–150. https://doi.org/10.1145/514183.514185

Grimbsy, G. (2019). Using Discord To Foster A Learning Community. *Innovations in Teaching & Learning Conference Proceedings*, *11*, Hall-Hall. https://doi.org/10.13021/itlcp.2019.2520

Hannah, K., & Gianvecchio, S. (2015). Zeuslite: A tool for botnet analysis in the classroom. *Journal of Computing Sciences in Colleges*, *30*(3), 109–116. https://doi.org/10.1145/2714576.2714579

Hay, B., Dodge, R., & Nance, K. L. (2008). Using Virtualization to Create and Deploy Computer Security Lab Exercises. In S. Jajodia, P. Samarati, & S. Cimato (Eds.), *Proceedings of The IFIP TC-11 23rd International Information Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008, September 7-10, 2008, Milano, Italy* (Vol. 278, pp. 621–635). Springer. https://doi.org/10.1007/978-0-387-09699-5_40

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2015). Botnet in DDoS attacks: Trends and challenges. *IEEE Communications Surveys & Tutorials*, *17*(4), 2242–2270. https://doi.org/10.1109/COMST.2015.2457491

Hu, D., & Wang, Y. (2008). Teaching computer security using xen in a virtual environment. *2008 International Conference on Information Security and Assurance (Isa 2008)*, 389–392. https://doi.org/10.1109/ISA.2008.18

Iqbal, F., Motyliński, M., & MacDermott, Á. (2021). Discord Server Forensics: Analysis and Extraction of Digital Evidence. *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1–8. https://doi.org/10.1109/NTMS49979.2021.9432654

Leidig, P., & Salmela, H. (2020). *IS2020 A Competency Model for Undergraduate Programs in Information Systems: The Joint ACM/AIS IS2020 Task Force*. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/is2020.pdf

Luburic, N., Sladic, G., Slivka, J., & Milosavljevic, B. (2019). A Framework for Teaching Security Design Analysis Using Case Studies and the Hybrid Flipped Classroom. *ACM Trans. Comput. Educ.*, *19*(3), 21:1-21:19. https://doi.org/10.1145/3289238

Melnikov, A., & Fette, I. (2011). *The WebSocket Protocol* (Request for Comments RFC 6455). Internet Engineering Task Force. https://doi.org/10.17487/RFC6455

Mirkovic, J., & Peterson, P. (2014). Class Capture-the-Flag Exercises. In Z. N. J. Peterson (Ed.), *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, 3GSE '14, San Diego, CA, USA, August 18, 2014*. USENIX Association. https://www.usenix.org/conference/3gse14/summit-program/presentation/mirkovic

Nieh, J., & Vaill, C. (2005). Experiences teaching operating systems using virtual platforms and linux. In W. P. Dann, T. L. Naps, P. T. Tymann, & D. Baldwin (Eds.), *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2005, St. Louis, Missouri, USA, February 23-27, 2005* (pp. 520–524). ACM. https://doi.org/10.1145/1047344.1047508

Pantic, N., & Husain, M. I. (2015). Covert botnet command and control using twitter. *Proceedings of the 31st Annual Computer Security Applications Conference*, 171–180. https://doi.org/10.1145/2818000.2818047

Petullo, W. M., Moses, K., Klimkowski, B., Hand, R., & Olson, K. (2016). The Use of Cyber-Defense Exercises in Undergraduate Computing Education. In M. A. Gondree & Z. N. J. Peterson (Eds.), *2016 USENIX Workshop on Advances in Security Education (ASE 16), Austin, TX, USA, August 9, 2016*. USENIX Association. https://www.usenix.org/conference/ase16/workshop-program/presentation/petullo

Raglianti, M., Nagy, C., Minelli, R., & Lanza, M. (2022, May 16). Using Discord Conversations as Program Comprehension Aid. *30th International Conference on Program Comprehension (ICPC '22)*. ACM International Conference on Program Comprehension (ICPC), Virtual Event, USA. https://doi.org/10.1145/3524610.3528388

Rapptz, Danny. (2023). *Discord.py* [Python]. https://github.com/Rapptz/discord.py (Original work published 2015)

Sharma, T., Zhou, Z., Huang, Y., & Wang, Y. (2022). *"It's A Blessing and A Curse": Unpacking Creators' Practices with Non-Fungible Tokens (NFTs) and Their Communities* (arXiv:2201.13233). arXiv. https://doi.org/10.48550/arXiv.2201.13233

Sood, A. K., & Enbody, R. J. (2013). Crimeware-as-a-service—A survey of commoditized crimeware in the underground market. *International Journal of Critical Infrastructure Protection*, *6*(1), 28–38. https://doi.org/10.1016/j.ijcip.2013.01.002

Sukhanova, K. (2023, July 23). *The Latest Discord Statistics & Trends for 2023*. https://techreport.com/statistics/discord-statistics/

Thompson, M. F., & Irvine, C. E. (2014). CyberCIEGE Scenario Design and Implementation. In Z. N. J. Peterson (Ed.), *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, 3GSE '14, San Diego, CA, USA, August 18, 2014*. USENIX Association. https://www.usenix.org/conference/3gse14/summit-program/presentation/thompson

TRM Labs. (2022, July 25). *Analysis of Recent NFT Discord Hacks Shows Some Attacks Are Connected | TRM Insights*. https://www.trmlabs.com/post/trms-analysis-of-recent-surge-in-discord-hacks-shows-some-attacks-are-connected

Vergos, D. (2011). *Botnet lab creation with open source tools and usefulness of such a tool for researchers* [Rochester Institute of Technology]. https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1144&context=theses

Vormayr, G., Zseby, T., & Fabini, J. (2017). Botnet communication patterns. *IEEE Communications Surveys & Tutorials*, *19*(4), 2768–2796. https://doi.org/10.1109/COMST.2017.2749442

Wang, M., Carr, S., Mayo, J., Shene, C.-K., & Wang, C. (2014). MLSvisual: A visualization tool for teaching access control using multi-level security. In Å. Cajander, M. Daniels, T. Clear, & A. Pears (Eds.), *Innovation and Technology in Computer Science Education Conference 2014, ITiCSE '14, Uppsala, Sweden, June 23-25, 2014* (pp. 93–98). ACM. https://doi.org/10.1145/2591708.2591730

Wang, M., Mayo, J., Shene, C.-K., Lake, T., Carr, S., & Wang, C. (2015). RBACvisual: A Visualization Tool for Teaching Access Control using Role-based Access Control. In V. Dagiene, C. Schulte, & T. Jevsikova (Eds.), *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCS 2015, Vilnius, Lithuania, July 4-8, 2015* (pp. 141–146). ACM. https://doi.org/10.1145/2729094.2742627

Wiles, A. M., & Simmons, S. L. (2022). Establishment of an Engaged and Active Learning Community in the Biology Classroom and Lab with Discord. *Journal of Microbiology & Biology Education*, *23*(1), e00334-21. https://doi.org/10.1128/jmbe.00334-21

Yang, L. (2009). Teaching database security and auditing. In S. Fitzgerald, M. Guzdial, G. Lewandowski, & S. A. Wolfman (Eds.), *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2009, Chattanooga, TN, USA, March 4-7, 2009* (pp. 241–245). ACM. https://doi.org/10.1145/1508865.1508954