

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, Hellingongjiang 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

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 editorcppj@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

Jonathan 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 autocompile 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.

10 Minutes 43 Seconds

CLICK HERE WHEN FINISHED

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 "NA" if you think the code is fine. After writing the answer, click on the button below.

19 Minutes 45 Seconds

CLICK HERE WHEN FINISHED

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.

16 Minutes 50 Seconds

CLICK HERE WHEN FINISHED

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 step for reducing cybersecurity risks.

Click on the image at the bottom right corner of the screen to launch the chatbot. Interact as much as you want with the chatbot before the timer expires. When ready, click on the button below.

14 Minutes 30 Seconds

CLICK HERE WHEN FINISHED

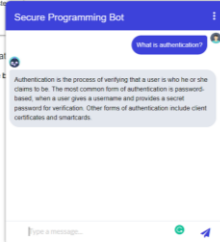


Figure 2. Chatbot widget

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 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 Figure 5, 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.

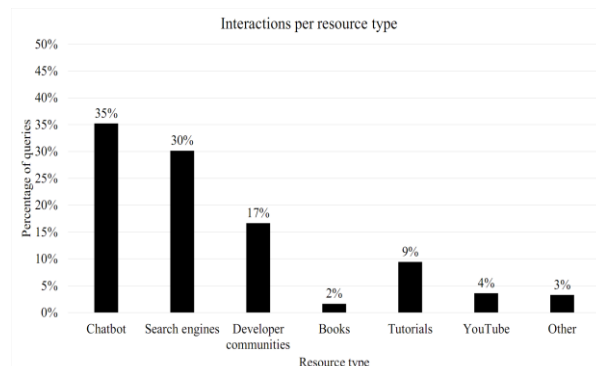


Figure 4. Interactions per resource type

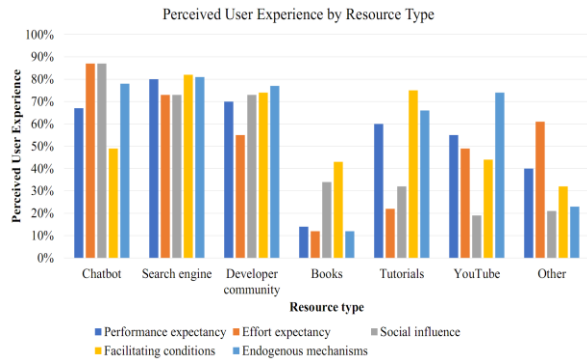


Figure 5. Perceived User Experience by Resource Type

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 scores completeness and security raise the most concerns about the adoption of GPT-3 as a tool for teaching students how to develop secure software.

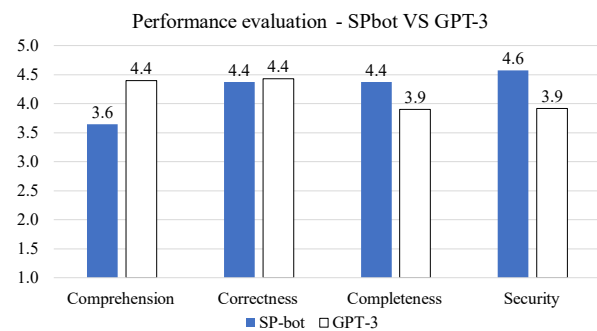


Figure 6. Performance of SPbot and GPT-3

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 Information Management (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.
- Introducing 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., & Wählisch, 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.