Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5545

EasyGuard: A Gamified App for Generating Strong and Memorable Passwords

Hugo L. Romão [**Universidade Federal de Roraima** | ***hugo.romao@ufrr.br***]
Marcelo H. O. Henklain [**Universidade Federal de Roraima** | ***marcelo.henklain@ufrr.br***]
Felipe L. Lobo [**Universidade Federal de Roraima** | ***felipe.lobo@ufrr.br***]
Eduardo L. Feitosa [**Universidade Federal do Amazonas** | ***efeitosa@icomp.ufam.edu.br***]

Departamento de Ciência da Computação, Universidade Federal de Roraima, Av. Capitão Ene Garcez, 2413, Bairro Aeroporto, Boa Vista, Roraima, 69304-000, Brasil.

Received: 14 February 2025 • **Accepted:** 14 June 2025 • **Published:** 17 March 2026

Abstract. Although the use of online services has increased substantially over the past decade, the strength of user-created passwords has remained at concerning levels. This study aimed to develop and evaluate the efficiency of a gamified application in promoting the behavior of designing strong passwords. Two rounds of experiments were conducted, each lasting nine days. In the first experiment ($n = 10$), we evaluated the passwords generated based on user inputs compared to random passwords. Our findings showed that our app generated passwords with an improvement of 68.43 percentage points in the memorization test, 4.87 p.p. in the typing test, and 60.38 p.p. in the combined memorization and typing test. In the second experiment ($n = 15$), we incorporated a dictionary-based password generation policy into the evaluation and applied an automated tool for data collection. User input-based passwords outperformed random ones by 87.26 p.p. in the memorization test, 2.75 p.p. in the typing test, and 85.92 p.p. in the combined test. Meanwhile, dictionary-based passwords showed improvements of 54.32 p.p., 1.69 p.p., and 69.70 p.p., respectively. Our approach proved promising in promoting strong and memorable passwords. Nonetheless, EasyGuard requires further development and should be further investigated in future studies.

Keywords: Cybersecurity, Human factor, Password

1 Introduction

The use of the Internet has become increasingly pervasive [Feldmann *et al*., 2021; Wells *et al*., 2023] and is accompanied by cybersecurity challenges [Chigada and Madzinga, 2021]. One such challenge is user behavior when designing passwords. Passwords are the primary mechanism for authentication and data protection, and they will remain so indefinitely [Bošnjak and Brumen, 2019]. However, they are one of the most neglected components in the cybersecurity field, even in the context of awareness campaigns about the risks of cyberattacks [Bošnjak *et al*., 2018; Carvalho *et al*., 2017; Ji *et al*., 2017].

This study expands upon the findings of Romão *et al*. [2024], incorporating additional participants, a novel password generation algorithm, and a web-based tool designed to automate the administration of forms and tests to facilitate the app evaluation. In this study, our objective was to develop and evaluate the efficiency of a gamified app designed to promote the behavior of "designing strong and memorable passwords". The app was tested by evaluating the passwords it generated under its two available policies, focusing on how easy they were to type and remember, both in comparison to each other and to randomly generated passwords. We investigated whether the behavior of designing strong passwords is acquired or improved after using the app. We also evaluated the app's usability.

This work distinguishes itself from other research in the literature primarily by focusing not only on developing technology but also on achieving behavioral change. Moreover, we evaluated the extent to which the technology effectively promoted this change. In this study, we employed two password-generation strategies through an application that incorporates gamification to encourage the creation of strong passwords.

This study is organized as follows: first, we address the research context and its motivations. In Section 2, we present the main concepts and theories that guided the development of the intervention and the discussion of the findings. In Section 3, we review the available scientific literature to identify the gaps that allowed us to formulate our research problem. In Section 4, we describe the methodology of the study, reporting the advances made from our previous work, and detailing the characteristics of the developed technology, the evaluation instruments, and the procedure for conducting the experimental evaluation. In Section 5, we present the results, discussion, and limitations of the experiments conducted. Finally, in Section 6, we suggest possible directions for future research and conclude the study with a summary of our answer to the research problem.

2 Background

2.1 Promoting secure behaviors from an educational perspective

Behavior is a term that encompasses psychological phenomena such as cognition, feelings, and attitudes. It is defined

in Behavior Analysis as a system of interactions between the antecedent environment, an organism's actions, and the subsequent environment [Skinner, 1981; Kienen *et al*., 2022]. Designing strong passwords is a behavior whose antecedent environment includes stimuli such as "knowledge about what makes a password strong", actions such as "identifying characteristics of a strong password" and "writing a password", and a subsequent environment centered on "reducing the chances of data loss and cyberattacks". Analyzing behaviors is useful for developing educational resources because it clarifies which actions should be encouraged, under what conditions they should occur, and what outcomes they should produce so that the resulting behavior is considered correct and complete [Kienen *et al*., 2022].

The app we created serves as a pedagogical resource because it aims to develop behavior [Cianca *et al*., 2020], focusing on the behavior of designing strong passwords. We anticipated that those who did not know this behavior would become able to demonstrate it after exposure to our app, with that change indicating that learning had occurred [Kienen *et al*., 2022]. Learning emerges from the selection of a person's actions within a given context, based on the consequences that follow in the subsequent environment [Moreira and Medeiros, 2018]. Such consequences are called reinforcers and are characterized by their effects of making an action stronger and more likely to reoccur [Skinner, 1981; Moreira and Medeiros, 2018]. The temporal proximity between action and consequence is critical for learning, and also the repetition of the interaction between context, action and reinforcer [Skinner, 1981]. In practice, consistently delivering immediate consequences to multiple people can be challenging. Therefore, digital and gamified educational resources can help educators, since digital solutions can be scalable and gamified ones tend to boost motivation.

2.2 Gamification

Gamifying means inserting game elements into non-game contexts [Azoubel and Pergher, 2017]. Clearly defined objectives, immediate feedback, progressively increasing difficulty, and challenging activities are gamification techniques that proved useful for teaching [Bai *et al*., 2020]. One reason for this effectiveness is that gamification uses multiple reinforcement strategies to select and maintain behaviors that we aim to promote or develop [Groening and Binnewies, 2019]. For instance, with a pleasant and user-friendly interface, the behavior of interacting with the gamified system is encouraged. Indicating points earned and challenges overcome, increases the chances of learning the actions that produced these consequences.

In this sense, rather than merely raising awareness (as in knowing something), we aim to create conditions that increase the likelihood of designing strong and memorable passwords. From this perspective, in addition to gamification, it was helpful for the generated password to be easy to memorize.

2.3 Memory interpreted as behaviors of remembering and forgetting

Memory can be interpreted as a behavioral phenomenon that involves the behaviors of remembering and forgetting [Skinner, 1981]. Remembering consists of a person's ability to be durably modified (that is, learn) and, from that, to interact with the environment in a new and consistent way [Skinner, 1981; Arantes *et al*., 2012].

Our app facilitates the behavior of remembering by requesting the insertion of words and numbers that hold meaning for the user (considering our first password creation policy) or by allowing him/her to select a set of words with meaning (second policy). In a general analysis, as an in-depth examination of the issue is beyond the scope of this article, these inputs have meaning because they can be members of an equivalent stimulus class (when, although distinct, they share the same cultural/simbolic meaning), becoming more memorable as the size of the class they belonged to increases [Haydu *et al*., 2009]. They could also be stimuli linked by a well-practiced behavioral chain (like a song, where singing one verse increases the likelihood of recalling the next), which also enhances memorability [Arantes *et al*., 2012]. Finally, it remains necessary to clarify what constitutes a strong password.

2.4 Creation of strong passwords

Many studies evaluated what constitutes a strong password. Ji *et al*. [2017] examined various password-cracking algorithms in real databases and found that passwords with more complex structures, a wide variety of symbols, and no easily identifiable user data were more difficult to break. Bošnjak and Brumen [2019] further observed that simply having a very long password already made it stronger. According to Glory *et al*. [2019], using longer passwords was easier for users.

The well-established technique in the literature for evaluating these different policies, measuring a password's strength, is to calculate its entropy using the following equation: $Password\_Length \times log2(Character\_Set\_Size)$. Strong passwords are those whose entropy exceeds 60 bits [Glory *et al*., 2019].

3 Related works

We conducted a brief literature review to identify the existing knowledge gap from which this study could be proposed. Švábenský *et al*. [2020] found, in a review on cybersecurity education, that the teaching of how to create strong passwords was seldom addressed. Carvalho *et al*. [2017], in turn, observed that 40% of respondents in cybersecurity surveys did not master strategies for creating secure passwords. It was also verified by Abdrabou *et al*. [2021] that creating strong passwords, compared to weak ones, required a high cognitive cost for users, discouraging the adoption of secure passwords.

In this context, one should ask what can be done to help users create strong passwords. To address this problem, Glory *et al*. [2019] developed a password-generation algorithm that uses keywords provided by the user and produces secure,

memorable passwords that surpass both the entropy and usability of passwords created by conventional password generators. This algorithm protects the user against brute-force attacks by generating long passwords with varied characters, and it defends against dictionary attacks by employing more than one word, transforming text characters into special characters, and inserting numbers. Although promising, that study did not develop an application for real users, nor did it assess whether the generated passwords were indeed memorable; moreover, no strategy was devised to encourage people to use this algorithm.

Bonk *et al*. [2021] found that policies emphasizing the use of long passwords, if they are easy to type, favor the creation of strong passwords, a strategy recommended by Han *et al*. [2021] and Mukherjee *et al*. [2023] as secure and lower in cognitive cost. Wu *et al*. [2022] suggested that long passwords composed of five words, each with three to five characters, are secure, although they were not found to be memorable. These studies imply the need for security policies that offer high usability and produce memorable, easy-to-type passwords.

Paudel and Al-Ameen [2024] investigated how to motivate users to create strong passwords. They assessed the impact of persuasion techniques applied before the password creation process, aiming to motivate users to reflect on the consequences of using weak passwords. The strategies included visual materials and statistical data, emphasizing the risks associated with weak passwords. The results demonstrated an increase in user motivation regarding the behavior of creating strong passwords. Our app uses gamified techniques to foster motivation.

At the same time, recent studies have begun investigating the use of password managers. Tian [2024] investigated the factors influencing online service users' adoption of password managers. Their data suggest that social influence, cognitive cost, and perceived risk directly impact trust in password managers. In this context, we believe that learning new methods for creating passwords, such as the algorithms implemented in our app, may involve a lower cognitive cost.

Chatzoglou *et al*. [2024] evaluated the security of 24 modern password managers. The analysis revealed that 75% of desktop password managers and 83% of browser plugin-based managers temporarily store passwords in memory without encryption, exposing them to potential risks. The study highlights that modern password managers remain vulnerable to data breaches and cyberattacks despite implementing security measures.

Considering these findings, we decided to refine the algorithm from Glory *et al*. [2019] to develop and test an app that generates strong and memorable passwords based on user input, with the creation and use of strong passwords supported by gamification — a context that may be especially effective for young users [Farias *et al*., 2019]. We also included a second password generation algorithm based on the five-word password policy of Wu *et al*. [2022]. Table 1 compared the attributes of the app we developed with those of the leading password-generation solutions on the industry. As the table shows, EasyGuard offers new features to users when compared to what we found in the literature and the cybersecurity industry.

In summary, our app lowers the cognitive load of creating strong passwords by generating them, handling necessary validations for the user, and offering cybersecurity tips, while also motivating users to create better passwords through one of the app's two available policies.

4 Method

The evaluation of our app was divided into two distinct experiments conducted in a laboratory setting. Based on the results of the first experiment, we refined the app by adding new features and tools to facilitate data collection. In the following, we detail the two experiments. This research was approved by the Research Ethics Committee, and all participants invited to this study were required to provide informed consent prior to their participation.

4.1 Experiment 1

4.1.1 Computational solution

We developed a gamified password generator, called EasyGuard, which creates strong passwords by adhering to the policy of generating long passwords with varied characters. Although the app was designed to be intuitive and suitable for adults of all ages, the evaluation was conducted with participants aged 20 to 26 years, as this group was readily available for testing. Participants were asked to employ meaningful words or phrases and digits so that the password would be memorable, emphasizing that obvious user information should be avoided. To reduce the issue with obvious user information, we checked for the use of common words in passwords to avoid them. Finally, we created a gamified environment to encourage the use of the app and the creation of strong passwords.

EasyGuard was developed as a Progressive Web Application (PWA) using NextJS, a framework built on top of the React library for interface development [Neutkens and Vercel, 2016; Walke and Facebook, 2013]. This approach allowed us to simultaneously deploy the project as a web application and as an Android app, maintaining typical native mobile application features such as offline functionality and direct installation on devices. Additionally, it enabled app distribution through platforms like the Google Play Store.

We implemented the entire project using TypeScript, a typed extension of JavaScript that improved security through static typing, simplified maintenance, and reduced potential runtime errors [Hejlsberg and Microsoft, 2012]. We also adopted the Model-View-ViewModel (MVVM) architectural pattern, widely used in modern mobile application development. This architectural pattern promoted a clear separation between the user interface components and the layers responsible for business logic and data access. In addition, adopting the MVVM pattern facilitated the implementation of automated tests, including unit and integration tests. These tests ensured that password generation procedures remained compliant with defined security criteria, even in the event of future code modifications. Screen examples can be seen in Figure 1, and the application code is available on GitHub at (`https://github.com/hugoromao/easyguard`).

Table 1. Attribute comparison between EasyGuard and leading industry password-generation tools.

Application	Password persistence	Generates memorable passwords	Total number of password-generation algorithms	Uses user input in generation	Gamification	Cloud dependent
1Password	✓	✓	2	✗	✗	✓
Bitdefender	✓	✗	1	✗	✗	✓
Bitwarden	✓	✓	2	✗	✗	✓
LastPass	✓	✓	2	✗	✗	✓
EasyGuard	✗	✓	2	✓	✓	✗

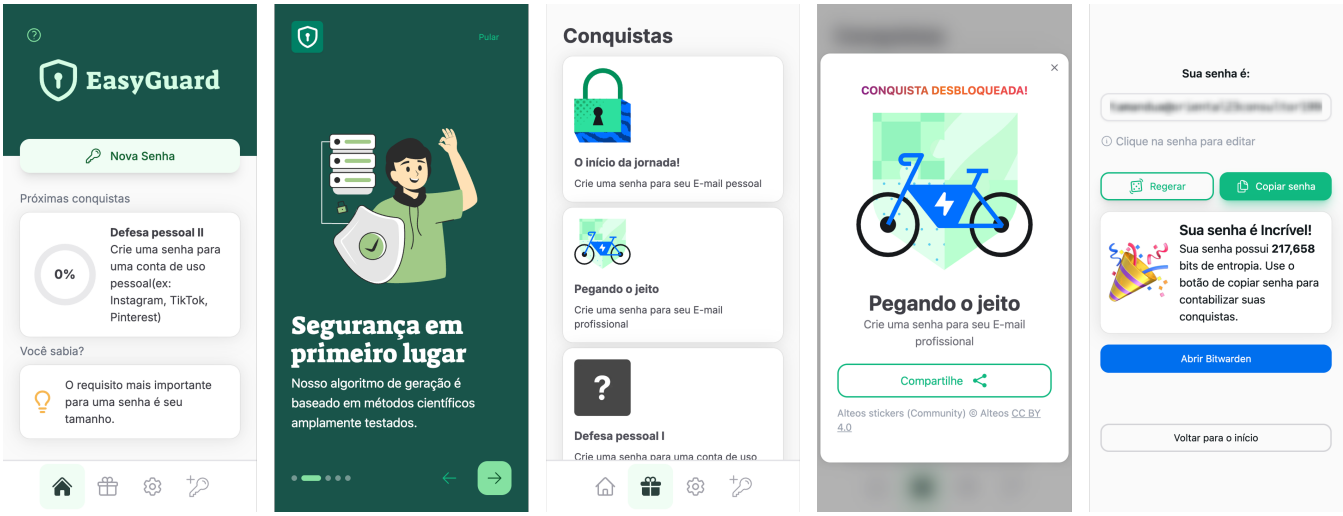

Figure 1. Example Screens from the EasyGuard App.

4.1.2 Password generation algorithm - The user-input policy

Our algorithm (see Algorithm 1) first asked the user to enter at least four words and two numbers, though more could be added. The words needed to be meaningful and drawn from diverse contexts. The app interface suggested that the words form a story (e.g., "brave, crimson, grape, made a trip") so the password could be memorized without using obvious user-related terms. Next, building on the work of Glory *et al*. [2019], the algorithm checked whether the inputs met the following criteria: (1) they could not be shorter than three characters or repeated; (2) they could not form keyboard layout patterns (e.g., QWERTY); (3) they could not appear in our list of common Portuguese words.

The algorithm then concatenated the words provided, forming a phrase. In a random decision, the words could be separated by random special symbols or one of the numbers provided by the user. Consequently, the same sequence of words could generate different passwords, thereby increasing their strength, as suggested by Han *et al*. [2021] and Wu *et al*. [2022]. In the final stage, we applied the LeetSpeak technique to one type of character, chosen randomly (e.g., an example with the letter "a": the password brave#crimson.grape@made-a-trip became br4ve#crimson.gr4pe@m4de-4-trip). Thus, even if part of the words were compromised, the search space for the correct password remains large.

The algorithm then performed additional validations, once again innovating in relation to Glory *et al*. [2019]: (1) It calculated the generated password's entropy, with a minimum acceptable value of 60 bits; (2) it checked whether the password contained at least 16 characters, a requirement that increased guessing difficulty [Wu *et al*., 2022; Mukherjee *et al*., 2023]; (3) it used two online security evaluation tools. On the Have I Been Pwned website [Troy Hunt, 2013], which includes data breaches from Brazil, we verified whether the generated password had already been compromised. Next, we evaluated the password's strength on The Password Meter, commonly used in the literature [Yıldırım and Mackie, 2019; Glory *et al*., 2019]. Both tools are open-source projects, and communication with them was carried out through automated requests to their web interfaces, which was the available means of access at the time of app development. Currently, in the case of Have I Been Pwned, an official API is available.

If the generated password did not satisfy any of these criteria, the app asked the user to provide additional words as input or to replace the current information with larger and more complex words or digits. It is important to emphasize that the entire password-generation process takes place on the user's device, and nothing is permanently stored, either by the app or by the third-party tools we use. Furthermore, the connection between the password generator and the online tools was encrypted via the HTTPS protocol. This entire process is summarized in Figure 2.

4.1.3 Gamified environment

The gamification employed in the application focused on creating: (1) an interactive tutorial to reinforce behaviors related to interacting with the app; (2) achievements, (3) badges, and (4) feedback on password strength to reinforce the behavior of designing strong passwords. In the following, we

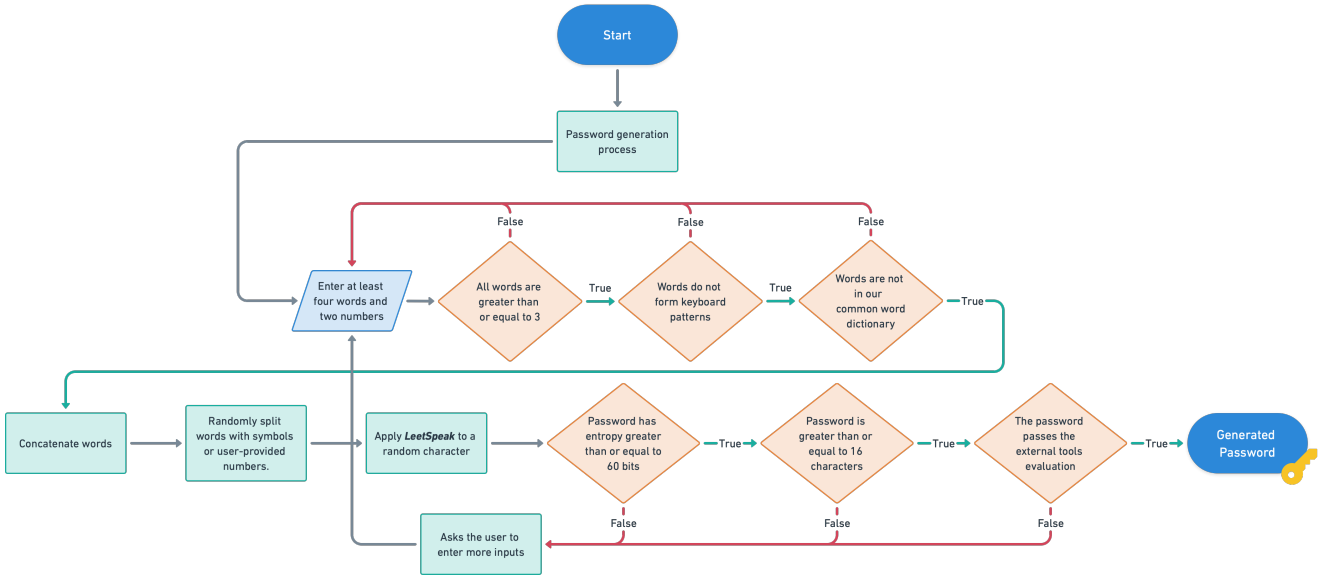

Figure 2. Flowchart of the password generation process based on user input.

Algorithm 1 Password Generation Algorithm

Require: Four words and two numbers significant to the user.

$words \leftarrow [buriti, crimson, brave, took\ entrance\ exam]$
$numbers \leftarrow [43, 92]$, $specialChars \leftarrow [-, @, *, =, ., +, ;, /, (,), !]$
$commonWordsList \leftarrow list\ of\ common\ words$, $keyboardPatterns \leftarrow list\ of\ keyboard\ patterns$
$uniqueWords \leftarrow []$
for $i \leftarrow 1, words.size$ **do**
    if $uniqueWords.includes(words[i])$ **then**
        Error: "Word " + $words[i]$ + " is repeated."
    end if
    if $words[i].size < 3$ **then**
        Error: "Word " + $words[i]$ + " is too short."
    end if
    if $keyboardPatterns.includes(words[i])$ **then**
        Error: "Word " + $words[i]$ + " includes a keyboard pattern."
    end if
    if $commonWordsList.includes(words[i])$ **then**
        Error: "Word " + $words[i]$ + " is too common."
    end if
    $uniqueWords.add(words[i])$
end for
$password \leftarrow words.concat(getRandomItem(specialChars, numbers))$
$characters \leftarrow password.splitCharacters()$, $randomChar \leftarrow getRandomItem(characters)$
$password \leftarrow password.replace(randomChar, getRandom(specialChars, numbers))$
if $password.size * log(countCharacterTypes(password), 2) < 60$ **then**
    Error: "Password entropy too low."
end if
if $password.size < 16$ **then**
    Error: "Password too short."
end if
if $externalEvaluation(password) = False$ **then**
    Error: "Password failed security evaluation."
end if
return password

describe how these elements operate. In this study, we state that the "app reinforces" in the sense that our expectation is that the programmed consequences will have reinforcing value for users. However, we recognize that asserting that a consequence functions as a reinforcer depends on empirical verification. For this reason, the app must be tested.

(1) Interactive Tutorial. It introduces the app, its features, and how to use it upon first access, but it can be revisited if the user needs it. We assessed that a newly installed app may be aversive to the user since they still have to learn how to use it [Sidman, 1995]. Therefore, this tutorial indicates the expected behaviors and reinforces them when they are performed. **(2) Achievements**. We adopted the design proposed by Groening and Binnewies [2019], for whom achievement is informative feedback on the user's success in a task, reinforcing the behaviors that must be learned. We used two types of achievements: those related (i) to password usage and (ii) to daily app usage, totaling 10 achievements. Upon completion, the user receives a badge as a reward. To earn achievements, the user must, on the achievements page, mark the achievement as completed or, in the case of those involving daily usage, create passwords with the app for as many consecutive days as possible. Encouraging password creation is important for learning this behavior and for replacing any weak passwords that the user may have. **(3) Real-Time Feedback**. After providing the inputs, the user can copy the generated password or edit it. To prevent users from editing the password in a way that makes it less secure, a feedback message appears and changes according to the password's entropy. If the entropy falls below 60 bits, the copy-password function is disabled, and the feedback message warns that the password is weak.

4.1.4 Participants

Ten adults participated, with a mean age of 22.45 years ($\pm$1.72), ranging from 21 to 26. P1, P6, P7, P8, P9, and P10 were Computing students, P2 studied Visual Arts, and P5 studied Architecture. P3 was an IT professional, specialized in interfaces, and P4 was a business administrator. We found that 30% of this sample never updated their passwords, 30% sometimes shared them, 70% never checked whether their passwords were found in data breaches, 30% always stored their passwords insecurely (e.g., in a text file), and 40% did so occasionally; 70% used the same password regularly or always, and only 20% considered themselves to always use strong passwords. In addition to these participants, we enlisted the help of a psychology student in the pilot study to refine the data collection procedure. All participants signed an Informed Consent Form (ICF).

4.1.5 Instruments

Except for the Usability Test, the instruments described below were developed for this research. They have not yet undergone psychometric studies.

Learning Test (Pre and Post-test). At the beginning of the study and again at the end, we asked each participant to perform the following task: "Create a password that you consider strong and suitable for everyday use." This allowed us to examine potential changes in password length, variety of characters used, and entropy.

Cognitive Cost Tests. (1) Memory Test: Two random 16-character passwords were generated for each participant, who had three minutes to study them. At the end of this period, as a distraction, each participant was shown two movie trailers totaling five minutes. They were then asked to type the passwords they had studied. After that activity, the participant generated two passwords using EasyGuard, and the same procedure was repeated with new trailers. We then analyzed which type of password yielded a higher percentage of correct entries. This percentage was calculated based on the Levenshtein distance, a metric indicating the minimum number of modifications required to transform one word into another [Levenshtein, 1966]. In our study, we compared the passwords generated by the participants with the inputs provided after the distraction periods. Passwords with fewer errors had a lower Levenshtein distance, resulting in a higher percentage of correct entries. We applied this metric in both the typing test and the combined test. (2) Typing Test: We generated a new password with each approach and asked participants to type them, one at a time, as quickly as possible, repeating the process five times. This test evaluated the frequency of typing errors for meaningful versus random passwords. (3) Combined Memory and Typing Test: We generated a new password with each approach and asked participants to study it for three minutes and then type it from memory, one at a time, as quickly as possible, repeating the process five times. This test evaluated the frequency of typing errors when participants relied solely on memory.

Engagement Tests. (1) Post-Study System Usability Questionnaire (PSSUQ): This instrument is widely adopted in the literature [Vlachogianni and Tselios, 2023]. We used the 16-item reduced version of the Post-Study System Usability Questionnaire (PSSUQ) found in Azlan and Junaini [2023], but had not yet undertaken a cross-cultural adaptation, , which is also a limitation of our study. Due to an error, Item 15 — "This system has all the functions and capabilities I expect it to have" — was not administered, which represents a limitation regarding this particular data point. Therefore, we administered 15 items from the PSSUQ, which originally contains 19 items as described by Lewis *et al*. [1990]. This instrument employed a seven-point Likert scale ("1 = Strongly disagree" to "7 = Strongly agree"), and its score was obtained by averaging its item data points. A higher score reflected a stronger agreement with respect to usability. Participants completed the PSSUQ at the end of the tasks carried out in the laboratory and, optionally, after the ninth day of app testing. (2) Password Generator Usage Log Protocol: This protocol was manually filled in by the user and allowed us to record the total number of created passwords and the dates on which participants accessed the app during the nine days of use outside the laboratory.

4.1.6 Data collection and analysis procedure

We conducted a pilot study with a psychology student to refine the experiment protocols and identify potential issues with the assessment tools. The pilot study comprised two phases over nine days. In the first phase, the participant completed the pre-

test and, with the app installed on his/her phone, sequentially performed the memory, typing, and usability tests. In the second phase, the participant was instructed to use the app for nine consecutive days and completed the post-test at the end of this period.

Following the pilot study, we implemented two adjustments to the main experiment: (1) we developed a combined test that integrated the memory and typing tasks. During the standalone typing test, we observed that participants first had to look at the password displayed on-screen and only then typed it, and this preliminary glance artificially inflated the measured typing time; and (2) we reduced the memorization time from five to three minutes, making the test more sensitive to variations among password-generation methods. These changes increased the sensitivity of the results, allowing us to capture the effects of the proposed intervention more accurately. The following describes the procedures performed in the main experiment.

We divided the experiment into two stages. In the first stage, carried out in a laboratory, participants signed the Informed Consent Form, took the pre-test, received the link to access EasyGuard, and, after the interactive tutorial, began the memory, typing, and combined tests. We collected data from these tests through an external web form. Additionally, in the memory and combined tests, we manually recorded the time using a stopwatch. We subsequently refined this protocol in Experiment 2, in which we used a form fully integrated into the app, enabling greater precision and efficiency in data collection. At the beginning of each test, we asked participants to create passwords using EasyGuard and a Bitwarden-based 16-length random password generation algorithm, which was selected because it is a robust open-source password management solution [8bit Solutions, 2016], and its password generation algorithm reflects the conventional strategy of fully random passwords. At the end of the tests, participants were asked to complete the PSSUQ. Finally, we asked them to try, at home, to earn the EasyGuard achievements over nine days, registering their use of the tool with our Log Protocol.

In the second stage, we collected the app usage logs from the participants, administered the post-test, and asked whether they wanted to change any of their PSSUQ responses. We adopted a qualitative-quantitative approach, primarily using descriptive statistics. We calculated the Wilcoxon signed-rank test, which is non-parametric, to compare pre and post-test results and to examine performance differences based on the use of the different EasyGuard password generation algorithms.

4.2 Experiment 2

We refined our app based on the results obtained during the first data collection. The improvements included the addition of a new password-generation policy, based on Wu *et al*. [2022], grounded in a database of Portuguese words, the development of an automated tool to assist in data collection, and the refinement of the typing and combined tests. These advancements are discussed in detail in the following subsections. Notably, this experiment tested the user-input policy against our dictionary policy and both against the fully random passwords.

4.2.1 Password generation algorithm - The dictionary policy

The new generation algorithm creates passwords without requiring user inputs by randomly selecting words from a dictionary composed of 50,000 terms in Brazilian Portuguese. To construct this dictionary, we utilized the Léxico do Português Brasileiro database, available at https://lexicodoportugues.com, which provides a psycholinguistic corpus of Brazilian Portuguese. From this database, we selected the 50,000 most commonly used words to form the dictionary used in the application. The chosen words are sufficiently diverse to remain difficult to guess, even if the attacker knows the dictionary. Our dictionary size results in 2,603,645,869,790,625,010,000 (approx. 2^{71}) possible combinations of unique five-word passwords. The dictionary used in our application is available at https://bit.ly/3WUVKW3.

The algorithm selects five distinct words from the dictionary and then randomly chooses a special character to serve as a connector between the words that will form the password. After this combination, the same validation criteria applied to the first algorithm are used. The password must have a minimum entropy of 60 bits, contain at least 19 characters, and pass the evaluations from the previously mentioned online platforms. If any of these requirements are not met, the process of word selection and special character assignment is repeated until all criteria are satisfactorily met.

The user has the freedom to run the algorithm as many times as desired until finding a set of words he/she consider suitable for their password. In the end, the generated password can be edited, and with each modification, the minimum requirements are re-evaluated. If the edited password does not meet the requirements, the copy-password function is automatically disabled, and a warning message is displayed to the user.

4.2.2 Participants

Fifteen adults participated, with a mean age of 23.00 years ($\pm$1.32), ranging from 20 to 26 years. All participants were Computing students, except for P1, who was an Architecture student. We observed that 6.7% of the participants never updated their passwords, 53.3% occasionally shared them, 33.3% never checked whether their passwords had been exposed in data breaches, 33.3% regularly stored their passwords insecurely (e.g., in a text file), and 33.3% did so occasionally. Additionally, 46.7% regularly or always reused the same password, and only 20.0% believed they always used strong passwords. As in the first experiment, all participants signed an Informed Consent Form (ICF).

4.2.3 Instruments

In the second experiment, we revised the Cognitive Cost Tests to accommodate the inclusion of a new password generation algorithm.

Cognitive Cost Tests. We made a few specific modifications to the cognitive cost tests applied in the first experiment. First, we reduced the distraction time to three minutes in all tests to make them less exhausting, particularly because we added new steps to evaluate the dictionary-based password

generation policy together with the policies from the previous experiment. In addition, we added more recent trailers to decrease the likelihood that participants had already watched them. Another change compared to the first experiment is the incorporation of the instruments directly into the application. Users can access them through a digital form located in the settings. This approach simplified the data collection process by reducing researcher intervention, ensuring that all participants received the same information, and enabling data to be automatically sent to a database. Additionally, this solution allowed for the collection of additional metrics during the tests. In the typing and combined tests, the typing times between passwords were recorded, along with the total number of times that participants used the backspace function on the keyboard. These improvements enabled a more detailed and accurate assessment of usability aspects related to the algorithms developed for the application.

4.2.4 Data collection and analysis procedure

Similar to the first experiment, the study was divided into two stages. However, in the first stage, conducted in a laboratory, we asked participants to install EasyGuard on their mobile phones and complete the tool's integrated form, which simulated the sequence of the pre-test and cognitive cost tests, much like the previous experiment. In other words, unlike the first experiment, in which we required the assistance of an external form, in the second experiment we collected data solely through the participant's smartphone, without the need to exit the app. At the end of this stage, they filled out the PSSUQ, and for nine days, they sought to meet the goals available in the application, recording their use of the tool in a separate form.

The second stage of the study followed the same procedure as the first experiment. In Experiment 2, given our small sample and the lack of normality, we employed the *non-parametric* Kruskal–Wallis H test, which provides an overall rank-based comparison among the three algorithms, and, when that omnibus test was significant, we carried out Holm-adjusted Dunn post-hoc rank-sum tests to identify which algorithm pairs differed.

5 Results and Discussion

In this section, we present our findings from the two experiments. Based on the results, we formulated our response to the research problem.

5.1 Experiment 1

5.1.1 Results of the memory, typing, and combined tests

Figures 3A, 3B, and 3C show the percentage of correct responses in the memory, typing, and combined tests for the 10 participants. Figure 3D shows the average percentage of correct responses for each test, according to the application used. The results of the memory test indicated a higher percentage of correct responses with EasyGuard ($M = 92.81 \pm 9.79$) compared to fully random passwords ($M = 24.38 \pm 30.61$), with a large effect size ($z = 2.666, p = 0.009, r_{pb} = 1$). This finding supports that passwords generated from user inputs seems more memorable. The only exception we found was P2, whose performance was similar for both types of passwords, possibly due to using some memorization technique. We highlight that participants P5 through P10 reported forgetting the random passwords after the distraction period, and their low percentage of correct entries suggests inattention. This observation highlights the importance of investigating how participants organize their strategies for memorizing passwords in future studies. Additionally, it would be beneficial to conduct the test one day or more after password creation to better reflect real-world conditions.

In the typing test, we evaluated the cost associated with using passwords generated by both tools, given that passwords that are more difficult to type are more costly [Bonk *et al*., 2021]. We found a smaller, non-significant difference ($z = 1.478, p = 0.153, r_{pb} = 0.527$) between the two approaches, with a 92% accuracy rate (± 4.44) using EasyGuard and 87.13% (± 9.88) with randomly generated approach. Although the passwords generated by our app tended to involve lower typing costs, as they were composed of terms familiar to the user [Shay *et al*., 2014; Bonk *et al*., 2021], the fact that we did not control typing time or the total number of corrections made while typing, and allowed participants to consult the generated passwords, made it more difficult to detect a difference. These were aspects that needed improvement, which we attempted to address by including the combined test. With it, we found data favorable to EasyGuard. The mean accuracy rate was 92.38% (± 4.05) for our app and 32% (± 38.28) for random passwords, representing a significant difference with a large effect size ($z = 8.803, p = 0.006, r_{pb} = 1$). Once again, participants P5 through P10 reported greater difficulty in recalling the randomly generated passwords, explaining that they sometimes forgot the password after watching the trailers or confused it with previously created ones. Meanwhile, participants P1 through P4 showed little difference between the approaches. Therefore, expanding the sample and refining the tests are necessary for future studies. Next, we examined how participants evaluated the app's usability, an important indicator of whether it tends to be used in a real-world scenario.

5.1.2 Usability test results

Figure 4 displays the average usability scores per participant, based on the PSSUQ. We observed a high overall PSSUQ score, with a mean value of 6.71 (± 0.21). The lowest mean score was 6.33, suggesting that EasyGuard was perceived as easy to use. Nevertheless, we believed that we still need to improve EasyGuard because, during the testing sessions, questions arose about how the tool worked (e.g., how to earn achievements, and how many words and numbers could be entered). According to Sauro and Lewis [2012], this was an indication that the application's features need further refinement. Next, we examined how participants used the app outside the laboratory.

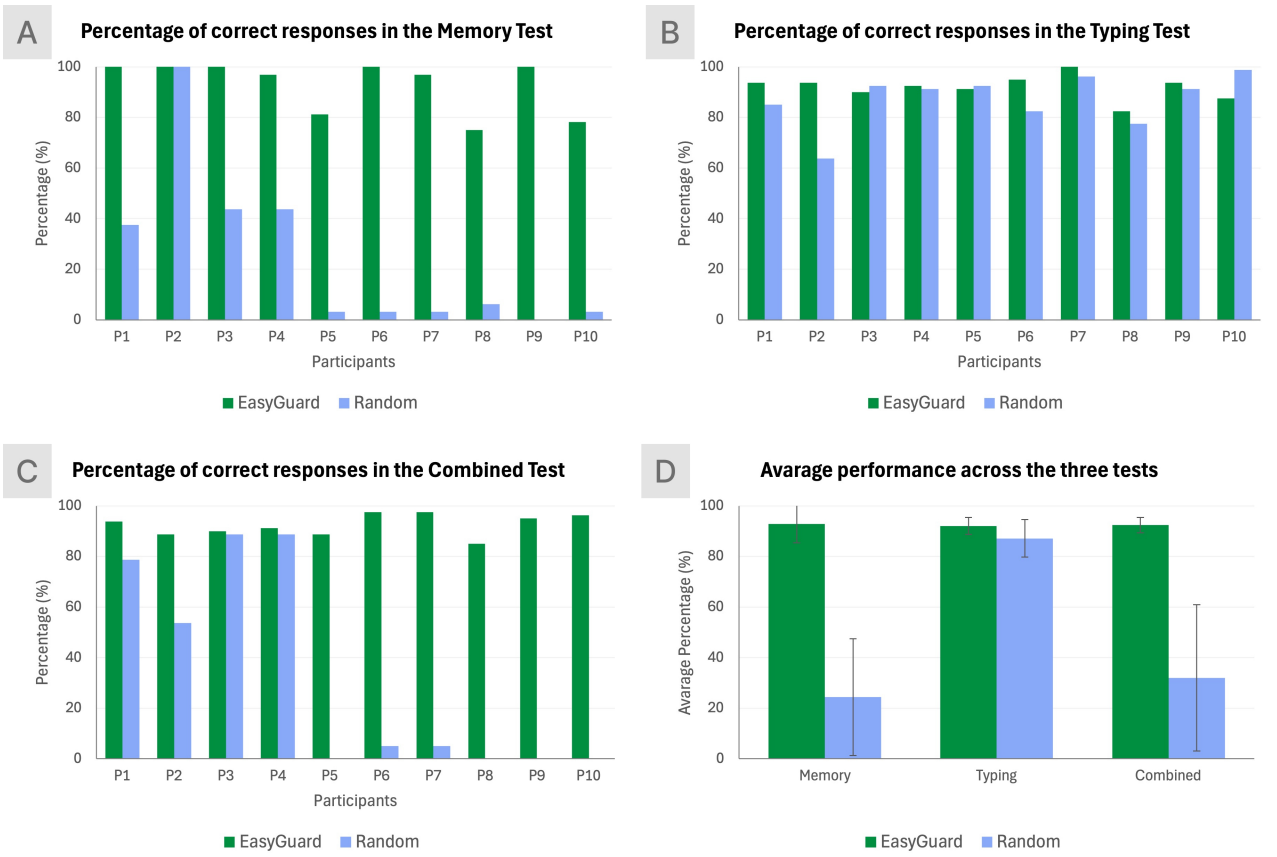

Figure 3. Percentages of correct answers in the memory (Figure 3A), typing (Figure 3B), and combined (Figure 3C) tests, and the average percentage of correct answers across all tests (Figure 3D).

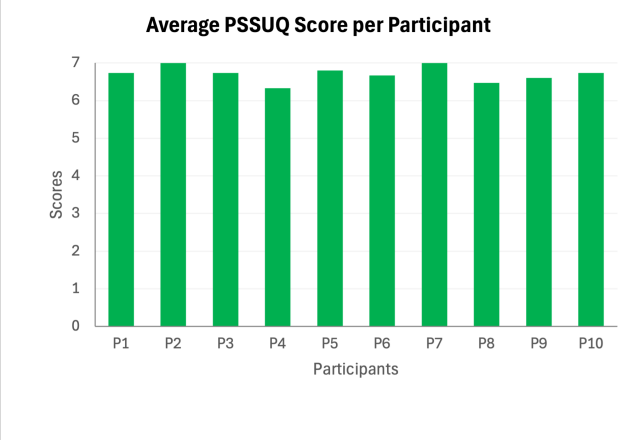

Figure 4. Average score on the PSSUQ for each participant.

5.1.3 Results of the usage log protocol

During the app usage period, participants were instructed to record the number of passwords created and the achievements reached each day. The frequency of app use was a direct measure of engagement in the behavior of "designing strong passwords".

On average, participants used the app for 1.60 days (± 1.28). P6 was the one who used the app the most, while P5 created the most passwords (9) and reached the greatest number of achievements (9). Conversely, P7 and P9 did not use the app outside the laboratory. The achievements seemed to act as reinforcers only for P2, P3, P5, and P6, as they created the largest number of passwords. Groening and Binnewies [2019] observed similar results, since one effect of using achievements was an increase in persistence. However, upon examining usage by P1 and P10, we noted that they gave up right after the first day. As expected from theory, a stimulus does not necessarily have the same value for different individuals, so it may reinforce one person's behavior but not another one [Moreira and Medeiros, 2018]. This leads us to further studies because, in the face of a high-cost behavior, such as creating strong passwords, whose reinforcing or punitive consequences are delayed over time, we must continue to investigate how to strengthen the creation of strong passwords by refining the gamification in our tool. Finally, we needed to verify whether using EasyGuard led to any degree of learning.

5.1.4 Learning test results (Pre and Post-Test)

Before the intervention, the passwords created had an average length of 10.90 characters (± 2.21), 3.70 groups of different characters (± 0.64), and 68.06 bits of entropy (± 16.02). After using the app, the average password length increased to 15.40 characters (± 4.84), showing a statistically significant difference with a large effect size ($z = 2.366$, $p = 0.022$, $r_{pb} = 1$). While the variability of character groups was maintained, entropy increased to 96.82 bits (± 32.21), also representing a statistically significant difference with a large effect size ($z = 2.521$, $p = 0.014$, $r_{pb} = 1$).

5.2 Experiment 2

5.2.1 Results of the memory, typing, and combined tests

Figures 5A, 5B, and 5C present the percentage of correct responses in the memory, typing, and combined tests for the 15 participants. Figure 5D presents the average accuracy in each test according to the algorithm used. In the second experiment, we incorporated a new password generation policy based on dictionary usage, along with metrics for the average typing time (in seconds) and the average number of corrections performed by each participant in the typing and combined tests. A detailed presentation of the data from the second experiment is shown in Table 2.

The results of the memory test indicated a higher percentage of correct responses for passwords generated from user-input policy ($M = 90.38 \pm 14.93$) compared to those generated using the dictionary policy ($M = 57.44 \pm 25.36$) and completely random passwords ($M = 3.13 \pm 4.42$). Upon conducting the Kruskal-Wallis test, we identified a statistically significant difference with a large effect size ($\chi^2 = 34.215, df = 2, p < 0.001, \eta^2 = 0.767$). Subsequent Dunn's post-hoc tests revealed differences among all compared groups: user-input policy versus dictionary policy ($z = 2.150, p_{\text{holm}} = 0.032, r_{pb} = 0.684$) and versus random passwords ($z = 5.786, p_{\text{holm}} < 0.001, r_{pb} = 1$), as well as dictionary policy versus random passwords ($z = 3.636, p_{\text{holm}} < 0.001, r_{pb} = 1$).

In the typing test, we evaluated the cost associated with using passwords generated by the different tested algorithms. We found a difference that was not statistically significant ($\chi^2 = 5.015, df = 2, p = 0.081, \eta^2 = 0.072$) among the three approaches regarding the average accuracy: 98.58% (± 1.87) for passwords generated from user-input policy, 97.53% (± 2.89) for passwords generated from the dictionary policy, and 95.83% (± 3.37) for completely random passwords. However, when analyzing the average typing time and the number of corrections, passwords generated from the user-input policy demonstrated the lowest cost, with an average typing time of 10.07 seconds (± 4.28) and 1.39 corrections (± 1.18). Next, the passwords generated using the dictionary policy recorded an average typing time of 16.15 seconds (± 6.70) and 3.04 corrections (± 2.34). Finally, the completely random passwords incurred the highest cost, with an average typing time of 22.85 seconds (± 18.80) and 1.55 corrections (± 1.37). We found a statistically significant difference between the typing times ($\chi^2 = 8.845, df = 2, p = 0.012, \eta^2 = 0.163$). Dunn's post hoc tests showed that these differences hold only between the user-input policy when compared to the dictionary ($z = -2.433, p_{\text{holm}} = 0.030, r_{pb} = 0.573$) and random ($z = -2.697, p_{\text{holm}} = 0.021, r_{pb} = 0.520$) policies. We did not find any statistically sound difference concerning the number of corrections ($\chi^2 = 5.089, df = 2, p = 0.079, \eta^2 = 0.074$).

Finally, the combined test results demonstrated favorable outcomes for EasyGuard. Passwords generated from user inputs achieved an average accuracy of 90.75% (± 20.23), with an average typing time of 8.43 seconds (± 4.18) and an average of 1.09 corrections (± 1.02). In comparison, the passwords generated from the dictionary performed worse, with an average accuracy of 74.54% (± 30.12), an average typing time of 15.16 seconds (± 7.67), and an average of 2.61 corrections (± 2.01). Completely random passwords resulted in the worst performance, with an average accuracy of 4.83% (± 5.94), an average typing time of 18.24 seconds (± 19.96), and an average of 1.25 corrections (± 1.24). The accuracy difference was statistically significant, with a large effect size ($\chi^2 = 32.265, df = 2, p < 0.001, \eta^2 = 0.721$). In the post hoc tests, we found that this difference occurred between the user-input policy versus random passwords ($z = 5.460, p_{\text{holm}} < 0.001, r_{pb} = 1$) and between the dictionary policy versus random passwords ($z = 4.088, p_{\text{holm}} < 0.001, r_{pb} = 1$). We also found a statistically sound difference between the typing times ($\chi^2 = 6.259, df = 2, p = 0.044, \eta^2 = 0.101$). Post hoc tests indicated that this difference holds only when comparing the user-input policy and the dictionary ($z = -2.399, p_{\text{holm}} = 0.049, r_{pb} = 0.578$). There was a statistically significant difference among the groups considering the corrections ($\chi^2 = 6.782, df = 2, p = 0.034, \eta^2 = 0.114$), but the post hoc tests, which have less statistical power, showed no significant differences. The difference between the user-input policy and the dictionary ($z = -2.338, p_{\text{holm}} < 0.058, r_{pb} = 0.502$), and the dictionary and the random password policy ($z = 2.163, p_{\text{holm}} < 0.061, r_{pb} = 0.453$) were marginally significant.

Our results suggested that both tested policies are more beneficial for users than random passwords. Considering the present findings, the user-input policy showed the best performance. Next, we analyzed how participants evaluated the application's usability.

5.2.2 Usability test results

Figure 6 displays the average usability scores per participant on PSSUQ. The results indicated a high overall PSSUQ score across all participants, with a mean value of 6.29(± 0.76). However, the minimum overall mean score observed was 4.93. We believe that some participants faced difficulties understanding certain functionalities of the app, such as the achievement system and the suggestion to use it in conjunction with a password manager. Therefore, we considered that enhancing the information in the interactive tutorial might improve users' understanding of the security features and operation of EasyGuard. Next, we examined the app's use outside the laboratory.

5.2.3 Results of the usage log protocol

In the second experiment, participants used the application for an average of 1.07 days (± 1.29). Participants P10 and P15 used the application for the longest duration (four consecutive days), while P1 and P4 created the highest number of passwords (9). P1 also achieved the highest number of accomplishments (9). On the other hand, participants P3, P5, P9, P11, P12, P13, and P14 did not use the application at any time after the laboratory experiment. The achievements appeared to reinforce behavior only for P1, P4, P7, P10, and P15, as these participants created passwords. Analysis of the usage by P2, P6, and P8 revealed that they discontinued use after the first day. These results indicate that, as in the first experiment, the gamification strategies did not exert a

Table 2. Data from memory, typing, and combined tests applied in the second experiment.

MEMORY TEST									
Participants	**EG - User-input**			**EG - Dictionary**			**Random**		
	Avg. score	**Avg. time**	**Avg.corrections**	**Avg. score**	**Avg. time**	**Avg. corrections**	**Avg. score**	**Avg. time**	**Avg. corrections**
P1	100.00%	N/A	N/A	76.31%	N/A	N/A	3.13%	N/A	N/A
P2	87.50%	N/A	N/A	29.28%	N/A	N/A	6.25%	N/A	N/A
P3	100.00%	N/A	N/A	45.00%	N/A	N/A	6.25%	N/A	N/A
P4	84.38%	N/A	N/A	35.23%	N/A	N/A	3.13%	N/A	N/A
P5	100.00%	N/A	N/A	29.91%	N/A	N/A	3.13%	N/A	N/A
P6	100.00%	N/A	N/A	62.86%	N/A	N/A	3.13%	N/A	N/A
P7	96.88%	N/A	N/A	23.94%	N/A	N/A	3.13%	N/A	N/A
P8	100.00%	N/A	N/A	89.29%	N/A	N/A	0.00%	N/A	N/A
P9	100.00%	N/A	N/A	97.83%	N/A	N/A	0.00%	N/A	N/A
P10	46.88%	N/A	N/A	100.00%	N/A	N/A	15.63%	N/A	N/A
P11	84.38%	N/A	N/A	45.31%	N/A	N/A	0.00%	N/A	N/A
P12	75.00%	N/A	N/A	60.85%	N/A	N/A	0.00%	N/A	N/A
P13	100.00%	N/A	N/A	50.98%	N/A	N/A	-3.13%	N/A	N/A
P14	56.25%	N/A	N/A	17.86%	N/A	N/A	0.00%	N/A	N/A
P15	90.63%	N/A	N/A	70.14%	N/A	N/A	6.25%	N/A	N/A
AVG	90.38%	N/A	N/A	57.44%	N/A	N/A	3.13%	N/A	N/A
STD	14.93%	N/A	N/A	25.36%	N/A	N/A	4.42%	N/A	N/A
TYPING TEST									
Participants	**EG - User-input**			**EG - Dictionary**			**Random**		
	Avg. score	**Avg. time**	**Avg. corrections**	**Avg. score**	**Avg. time**	**Avg. corrections**	**Avg. score**	**Avg. time**	**Avg. corrections**
P1	98.75%	8.4	0	96.82%	31.6	0	88.75%	22.6	0
P2	98.75%	7.8	2.6	100.00%	9.2	3.2	93.75%	16	0.8
P3	98.75%	7.8	1.6	98.75%	12.2	2.4	93.75%	14.2	1.4
P4	100.00%	11.8	1	98.33%	7.6	1.2	92.50%	22	3.8
P5	100.00%	5	0.4	98.18%	24.8	2.4	100.00%	65.6	1.8
P6	100.00%	6	0.4	100.00%	13.2	0.6	98.75%	63.2	2.2
P7	97.50%	8.2	0	91.20%	22.4	4.2	92.50%	11.8	1
P8	100.00%	5.6	0.8	100.00%	10	2.2	100.00%	14.8	0.6
P9	100.00%	7.4	1.2	98.05%	13.2	1.6	93.75%	13.2	0.6
P10	100.00%	10.4	2.2	100.00%	18	6	95.00%	46.4	0
P11	100.00%	16	3.2	92.86%	20.8	4.8	97.50%	7	3
P12	95.00%	6.4	0	97.27%	20.8	6.8	98.75%	2.6	0
P13	97.50%	18.2	2	92.34%	12.6	1.6	98.75%	13.8	0.6
P14	98.75%	15.8	4	99.11%	17.8	8	100.00%	17.6	3.2
P15	93.75%	16.2	1.4	100.00%	8	0.6	93.75%	12	4.2
AVG	98.58%	10.07	1.39	97.53%	16.15	3.04	95.83%	22.85	1.55
STD	1.87%	4.28	1.18	2.89%	6.7	2.34	3.37%	18.8	1.37
COMBINED TEST									
Participants	**EG - User-input**			**EG - Dictionary**			**Random**		
	Avg. score	**Avg. time**	**Avg. corrections**	**Avg. score**	**Avg. time**	**Avg. corrections**	**Avg. score**	**Avg. time**	**Avg. corrections**
P1	81.25%	8.4	0	69.58%	31.6	0	0.00%	22.6	0
P2	100.00%	7.8	2.6	41.46%	9.2	3.2	6.25%	16	0.8
P3	98.75%	7.8	1.6	97.39%	12.2	2.4	0.00%	14.2	1.4
P4	25.00%	11.8	1	20.75%	7.6	1.2	0.00%	22	3.8
P5	62.50%	5	0.4	100.00%	24.8	2.4	0.00%	65.6	1.8
P6	93.75%	6	0.4	100.00%	13.2	0.6	12.50%	63.2	2.2
P7	100.00%	8.2	0	46.00%	22.4	4.2	6.25%	11.8	1
P8	100.00%	5.6	0.8	100.00%	10	2.2	7.50%	14.8	0.6
P9	100.00%	7.4	1.2	98.95%	13.2	1.6	12.50%	13.2	0.6
P10	100.00%	10.4	2.2	54.76%	18	6	0.00%	0	0
P11	100.00%	16	3.2	78.75%	20.8	4.8	18.75%	7	3
P12	100.00%	6.4	0	90.24%	20.8	6.8	2.50%	2.6	0
P13	100.00%	18.2	2	99.41%	12.6	1.6	6.25%	13.8	0.6
P14	100.00%	3.2	0.8	20.75%	3.6	1.2	0.00%	6	3
P15	100.00%	4.2	0.2	100.00%	7.4	1	0.00%	0.8	0
AVG	90.75%	8.43	1.09	74.54%	15.16	2.61	4.83%	18.24	1.25
STD	20.23%	4.18	1.02	30.12%	7.67	2.01	5.94%	19.96	1.24

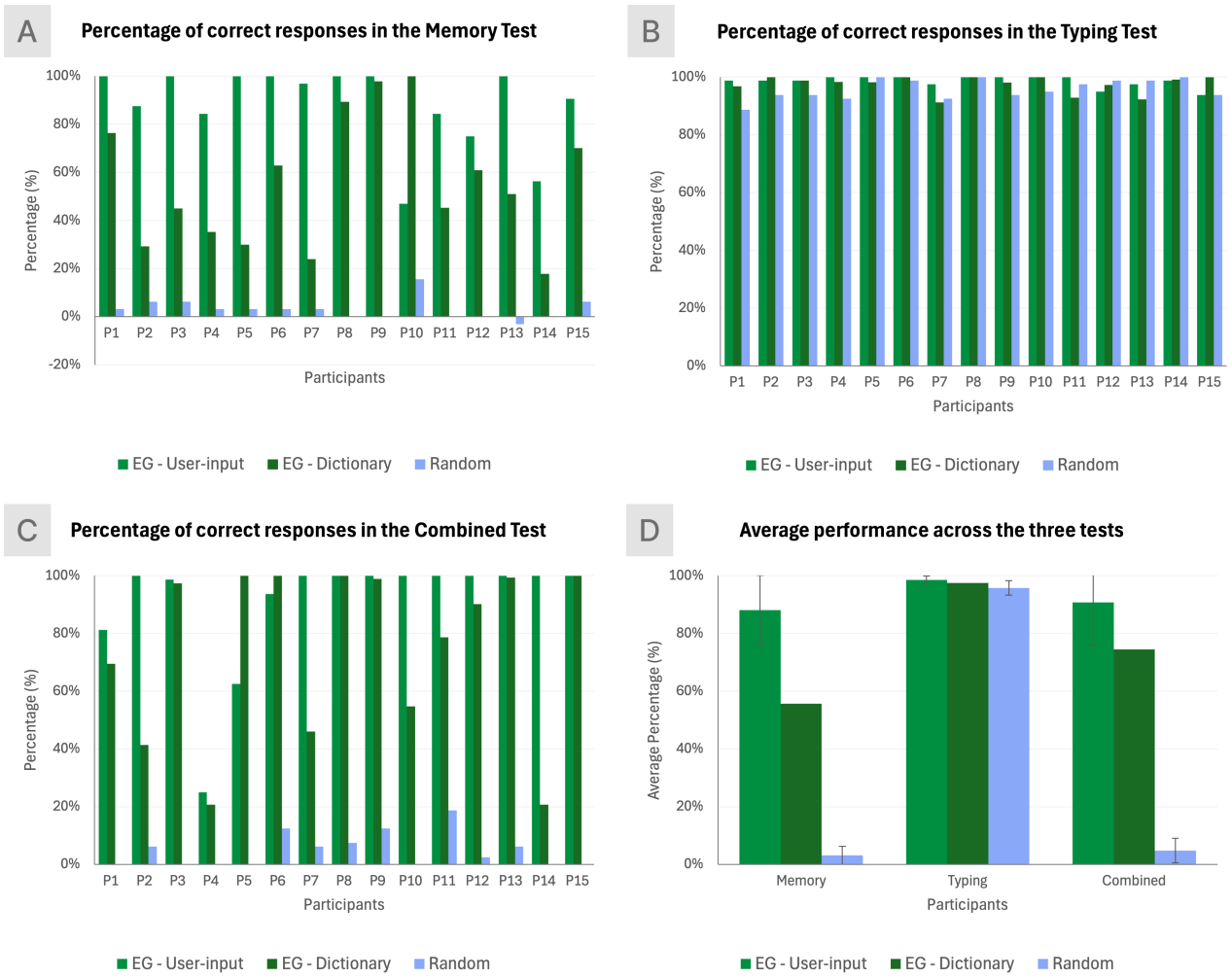

Figure 5. Percentages of correct answers in the memory (Figure 5A), typing (Figure 5B), and combined (Figure 5C) tests, and the average percentage of correct answers across all tests (Figure 5D).

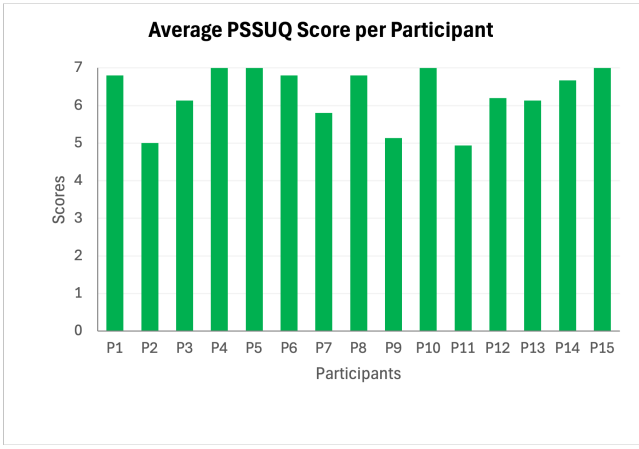

Figure 6. Average score on the PSSUQ for each participant.

motivating effect for the majority of the sample. Finally, we examined whether the use of EasyGuard led to any degree of learning.

5.2.4 Learning test results (Pre and Post-Test)

We verified that before the intervention, the created passwords had an average of 12.20 characters (± 3.76), four different character groups (± 0), and 79.97 bits of entropy (± 24.67). After using the app, we found an average of 16.53 characters (± 5.10) per password, with a statistically significant difference and a large effect size ($z = 2.706$, $p = 0.008$, $r_{pb} = 0.885$). We observed that the variability of characters remained constant, but the entropy increased to 107.96 bits (± 33.75), which was also a statistically significant difference with a large effect size ($z = 2.667$, $p = 0.009$, $r_{pb} = 0.872$).

5.3 General discussion

In summary, the results from both experiments suggested that passwords generated by the algorithms implemented in EasyGuard are more memorable than the completely random approach, and they also offer greater usability, in terms of ease of typing. These results corroborate the hypothesis of Glory *et al*. [2019] that passwords generated from user inputs are more memorable, and of Wu *et al*. [2022] that passwords generated from a dictionary policy are easier to deal with than usual random passwords we usually find in commercial software.

We need to highlight that, in addition to the user input policy, in our experiments passwords generated from dictionaries also proved promising, achieving higher accuracy rates compared to completely random passwords. However, according to Wu *et al*. [2022], long passwords composed of five words may exhibit low long-term retention. Nonetheless, we believe that the use of common words in our dictionary may enhance the memorability of this policy. Therefore, we need studies that evaluate the long-term usability of this policy with a larger number of participants.

Finally, regarding the learning test, the results showed a behavioral change in how users create passwords after being exposed to the app, suggesting that the behavior of designing strong passwords improved. Thus, we gathered preliminary evidence that the EasyGuard app can be an effective pedagogical tool [Kienen *et al*., 2022].

5.4 Limitations

This study has limitations that should be considered in future research. Regarding the algorithm, our approach favored protection against brute-force and dictionary attacks, but it remains vulnerable to attacks such as shoulder surfing. It was not possible to determine whether the developed app temporarily stores passwords in memory as plain text. Future research should include a memory analysis during app execution to ensure that no sensitive data is retained in memory. Additionally, we need to evaluate EasyGuard using instruments with better psychometric evidence. For instance, the PSSUQ was not specifically adapted to Brazilian culture. Future studies should also deal with the need to collect data from a larger, more age-diverse sample with longer app usage to obtain more robust data on its efficiency.

Furthermore, we believe that enhancing the gamification features — as well as adding new ones, such as daily rewards and a scoring system — could be tested to increase participant engagement. Similarly, integrating EasyGuard dynamically into password creation forms in browsers through a widget might facilitate access and encourage its adoption. Finally, it is necessary to test this app against other existing password generation apps, including at least one that aims to generate memorable passwords, such as Lingpass [Balayogi and Kuppusamy, 2024].

6 Conclusion

The objective of this study was to develop and evaluate the efficiency of a gamified app to promote the behavior of designing strong passwords. We created the app to serve as a reinforcing environment that reduces the effort involved in creating strong passwords. We found that the use of meaningful inputs or long passwords generated from dictionaries, coupled with gamification elements, proved promising in enhancing this behavior. The results suggest that passwords generated by EasyGuard are more memorable and involve lower cognitive costs compared to the usual approach of generating fully random passwords. In future work, we will broaden our sample to include participants from a wider range of ages, backgrounds, and usage contexts to enhance the generalization of our findings. We also plan to incorporate additional scientifically tested password generation algorithms that provide less burdensome strategies for users. Furthermore, we will refine the app's gamified elements to boost adherence and identify new reinforcers for strong-password behavior. Finally, we intend to explore techniques for strengthening passwords against additional attack vectors.

This study, although exploratory, produced promising results supporting the use of EasyGuard in scientific contexts. We innovated primarily by advancing the algorithm for password creation based on user inputs and creating a Brazilian dictionary that supports a second promising policy of strong password creation. Furthermore, we developed a technological artifact that implemented both algorithms. We also tested

our app and its algorithm's claims of being better than random passwords in a realistic scenario, guided by solid psychological theory. Finally, we developed resources to make our data collection scalable and made our app available for scientific use and scrutiny, since EasyGuard requires further development and investigation by future studies.

Declarations

Authors' Contributions

MHOH and FLL contributed to the conception of this study. HLR developed the app and performed the experiments. MHOH, FLL, and ELF acted as reviewers, providing revisions and supervision. HLR is the main contributor and writer of this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Finance Code 001. This work was partially supported by Amazonas State Research Support Foundation - FAPEAM - through the POSGRAD project 2024/2025.

Availability of data and materials

The instruments generated and/or analyzed during the current study are available upon request to the first author.

AI Use Statement. In this manuscript, OpenAI ChatGPT and Copilot in Microsoft Word was used to assist in translating scientific text into English. All final editorial decisions and content validations were made solely by the authors, who retain full responsibility for the accuracy and integrity of the work.

References

8bit Solutions (2016). Bitwarden open-source password manager. Available at:`https://bitwarden.com`. Accessed on: April 26, 2025.

Abdrabou, Y., Abdelrahman, Y., Khamis, M., and Alt, F. (2021). Think harder! investigating the effect of password strength on cognitive load during password creation. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3411763.3451636.

Arantes, A. K. L., Mello, E. L., and Domeniconi, C. (2012). *Memória*. Guanabara Koogan, Rio de Janeiro. Book.

Azlan, Z. H. Z. and Junaini, S. N. (2023). Erudite survivor: Usability testing of a gamification-based mobile app for disaster awareness among children. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 31(3):290298. Acesso em: 27 dez. 2024. DOI: 10.37934/araset.31.3.290298.

Azoubel, M. S. and Pergher, N. K. (2017). Levantamento sobre a utilização de jogos na análise do comportamento aplicada. *Perspectivas em Análise do Comportamento*, 8(2):215–225. DOI: 10.18761/PAC.2016.014.

Bai, S., Hew, K. F., and Huang, B. (2020). Does gamification improve student learning outcome? evidence from a meta-analysis and synthesis of qualitative data in educational contexts. *Educational Research Review*, 30:100322. DOI: 10.1016/j.edurev.2020.100322.

Balayogi, G. and Kuppusamy, K. (2024). Lingpass: An approach for multilingual passphrase generation by integrating english and tamil. *Internet Technology Letters*, page e580. DOI: 10.1002/itl2.580.

Bonk, C., Parish, Z., Thorpe, J., and Salehi-Abari, A. (2021). Long passphrases: Potentials and limits. In *2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–7. DOI: 10.1109/PST52912.2021.9647800.

Bošnjak, L. and Brumen, B. (2019). Rejecting the death of passwords: Advice for the future. *Computer Science and Information Systems*, 16(1):313–332. DOI: 10.1007/s10207-019-00429-y.

Bošnjak, L., Sreš, J., and Brumen, B. (2018). Brute-force and dictionary attack on hashed real-world passwords. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1161–1166. DOI: 10.23919/MIPRO.2018.8400211.

Carvalho, E., Reis, T., and Alves, F. (2017). Ensino de noções básicas de segurança da informação nas escolas brasileiras. In *Anais do XXIII Workshop de Informática na Escola*, pages 765–774, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/cbie.wie.2017.765.

Chatzoglou, E., Kampourakis, V., Tsiatsikas, Z., Karopoulos, G., and Kambourakis, G. (2024). Keep your memory dump shut: Unveiling data leaks in password managers. In Pitropakis, N., Katsikas, S., Furnell, S., and Markantonakis, K., editors, *ICT Systems Security and Privacy Protection*, pages 61–75, Cham. Springer Nature Switzerland. DOI: 10.1007/978-3-031-65175-5_5.

Chigada, J. and Madzinga, R. (2021). Cyberattacks and threats during covid-19: A systematic literature review. *South African Journal of Information Management*, 23(1):11. DOI: 10.4102/sajim.v23i1.1277.

Cianca, B. C., Panosso, M. G., and Kienen, N. (2020). Programação de condições para desenvolvimento de comportamentos: Caracterização da produção científica brasileira de 1998-2017. *Perspectivas em Análise do Comportamento*, 11(2):114–136. DOI: 10.18761/PAC.2020.v11.n2.01.

Farias, F., Medeiros, N., Rocha, S., Medeiros, D., Nóbrega, E., Burlamaqui, A., and Madeira, C. (2019). Self protect: Um jogo para auxílio no ensino de conceitos relacionados a segurança na internet para crianças e adolescentes. In *Anais do XXV Workshop de Informática na Escola*, pages 246–255, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/cbie.wie.2019.246.

Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J.,

Vallina-Rodriguez, N., Hohlfeld, O., and Smaragdakis, G. (2021). A year in lockdown: how the waves of covid-19 impact internet traffic. *Commun. ACM*, 64(7):101–108. DOI: 10.1145/3465212.

Glory, F. Z., Ul Aftab, A., Tremblay-Savard, O., and Mohammed, N. (2019). Strong password generation based on user inputs. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0416–0423. DOI: 10.1109/IEMCON.2019.8936178.

Groening, C. and Binnewies, C. (2019). "achievement unlocked!" - the impact of digital achievements as a gamification element on motivation and performance. *Computers in Human Behavior*, 97:151–166. DOI: 10.1016/j.chb.2019.02.026.

Han, W., Xu, M., Zhang, J., Wang, C., Zhang, K., and Wang, X. S. (2021). Transpcfg: Transferring the grammars from short passwords to guess long passwords effectively. *IEEE Transactions on Information Forensics and Security*, 16:451–465. DOI: 10.1109/TIFS.2020.3003696.

Haydu, V. B., Omote, L. C. F., Vicente, P., Ággio, N. M., and De Paula, J. B. C. (2009). Efeitos do tamanho da classe na manutenção de relações de equivalência em um delineamento intragrupo. *Interação em Psicologia*, 13:179–193. DOI: 10.1590/S0102-79722008000200009.

Hejlsberg, A. and Microsoft (2012). Typescript. Available at:`https://www.typescriptlang.org`. Accessed on: April 26, 2025.

Ji, S., Yang, S., Hu, X., Han, W., Li, Z., and Beyah, R. (2017). Zero-sum password cracking game: A large-scale empirical study on the crackability, correlation, and security of passwords. *IEEE Transactions on Dependable and Secure Computing*, 14(5):550–564. DOI: 10.1109/TDSC.2015.2481884.

Kienen, N., Panosso, M. G., Nery, A. G. S., Waku, I., and Carmo, J. d. S. (2022). Contextualização sobre a programação de condições para desenvolvimento de comportamentos (pcdc): Uma experiência brasileira. *Perspectivas em Análise do Comportamento*, 12(2):360–390. DOI: doi.org/10.18761/PAC.2021.jul110.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710. Available at:`https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf`.

Lewis, J. R., Rieman, J. A., and Wharton, C. (1990). Integrated office software benchmarks: A case study. In *Proceedings of the CHI '90 Conference on Human Factors in Computing Systems*. ACM. Available at:`https://www.researchgate.net/publication/221053907_Integrated_office_software_benchmarks_A_case_study`.

Moreira, B. M. and Medeiros, A. C. (2018). *Princípios básicos de análise do comportamento*. Artmed, Porto Alegre. Book.

Mukherjee, A., Murali, K., Jha, S. K., Ganguly, N., Chatterjee, R., and Mondal, M. (2023). Mascara: Systematically generating memorable and secure passphrases. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '23, page 524–538, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3579856.3582839.

Neutkens, T. and Vercel (2016). Next.js. Available at: `https://nextjs.org`. Accessed on: April 26, 2025.

Paudel, R. and Al-Ameen, M. N. (2024). Priming through persuasion: Towards secure password behavior. *Proc. ACM Hum.-Comput. Interact.*, 8(CSCW1). DOI: 10.1145/3637387.

Romão, H., Henklain, M., Lobo, F., and Feitosa, E. (2024). Construção e teste de app gamificado gerador de senhas fortes e memoráveis: Um estudo exploratório em cibersegurança. In *Anais Estendidos do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 256–269, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/sbseg$_e$*stendido*.2024.243316.

Sauro, J. and Lewis, R. J. (2012). *Quantifying the user experience: Practical statistics for user research*. Elsevier, Waltham. Book.

Shay, R., Komanduri, S., Durity, A. L., Huh, P. S., Mazurek, M. L., Segreti, S. M., Ur, B., Bauer, L., Christin, N., and Cranor, L. F. (2014). Can long passwords be secure and usable? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 2927–2936, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2556288.2557377.

Sidman, M. (1995). *Coerção e suas implicações*. Editorial Psy, Campinas. Book.

Skinner, B. F. (1981). *Ciência e comportamento humano*. Martins Fontes, São Paulo. Book.

Tian, X. (2024). Unraveling the dynamics of password manager adoption: a deeper dive into critical factors. *Information and Computer Security*, ahead-of-print(ahead-of-print). Published on 16 July 2024. DOI: 10.1108/ICS-09-2023-0156.

Troy Hunt, Charlotte Hunt, S. J. S. (2013). Have i been pwned. Available at: `https://haveibeenpwned.com`. Accessed on: April 26, 2025.

Vlachogianni, P. and Tselios, N. (2023). Perceived usability evaluation of educational technology using the post-study system usability questionnaire (pssuq): A systematic review. *Sustainability*, 15(17). DOI: 10.3390/su151712954.

Švábenský, V., Vykopal, J., and Čeleda, P. (2020). What are cybersecurity education papers about? a systematic literature review of sigcse and iticse conferences. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 2–8, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3328778.3366816.

Walke, J. and Facebook (2013). Reactjs. Available at: `https://react.dev`. Accessed on: April 26, 2025.

Wells, J., Scheibein, F., Pais, L., dos Santos, N. R., Dalluege, C.-A., Czakert, J. P., and Berger, R. (2023). A systematic review of the impact of remote working referenced to the concept of work–life flow on physical and psychological health. *Workplace Health & Safety*, 71(11):507–521. PMID: 37387511. DOI: 10.1177/21650799231176397.

Wu, X., Munyendo, C. W., Cosic, E., Flynn, G. A., Legault, O., and Aviv, A. J. (2022). User perceptions of five-word passwords. In *Proceedings of the 38th Annual Com-*

puter Security Applications Conference, ACSAC '22, page 605–618, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3564625.3567981.

Yıldırım, M. and Mackie, I. (2019). Encouraging users to improve password security and memorability. *International Journal of Information Security*, 18(6):741–759. DOI: 10.1007/s10207-019-00429-y.